Representation Learning for Texts and Graphs

A Unified Perspective on
Efficiency, Multimodality, and Adaptability

Lukas Paul Achatius Galke
aus
Wuppertal

Dissertation
zur Erlangung des akademischen Grades
Doktor der Ingenieurwissenschaften
(Dr.-Ing.)
der Technischen Fakultät
der Christian-Albrechts-Universität zu Kiel
eingereicht im Jahr 2022

Kiel Computer Science Series (KCSS) 2023/1 dated 2023-04-03

ISSN 2193-6781 (print version)
ISSN 2194-6639 (electronic version)

Electronic version, updates, errata available via https://www.informatik.uni-kiel.de/kcss

The author can be contacted via http://lpag.de

Published by the Department of Computer Science, Kiel University

Knowledge Discovery (defunct)

Please cite as:

▻ Lukas Galke. *Representation Learning for Texts and Graphs* Number 2023/1 in Kiel Computer Science Series. Department of Computer Science, 2023. Dissertation, Faculty of Engineering, Kiel University.

```
@book{galke2023dissertation,
  author    = {Lukas Galke},
  title     = {Representation Learning for Texts and Graphs},
  publisher = {Department of Computer Science, Kiel University},
  year      = {2023},
  number    = {2023/1},
  doi       = {10.21941/kcss/2023/1},
  series    = {Kiel Computer Science Series},
  note      = {Dissertation, Faculty of Engineering,
               Kiel University.}
}
```

About this Series

The Kiel Computer Science Series (KCSS) covers dissertations, habilitation theses, lecture notes, textbooks, surveys, collections, handbooks, etc. written at the Department of Computer Science at Kiel University. It was initiated in 2011 to support authors in the dissemination of their work in electronic and printed form, without restricting their rights to their work. The series provides a unified appearance and aims at high-quality typography. The KCSS is an open access series; all series titles are electronically available free of charge at the department's website. In addition, authors are encouraged to make printed copies available at a reasonable price, typically with a print-on-demand service.

Please visit `http://www.informatik.uni-kiel.de/kcss` for more information, for instructions how to publish in the KCSS, and for access to all existing publications.

1. Gutachter:	Prof. Dr. Ansgar Scherp Universität Ulm
2. Gutachter:	Prof. Dr. Peer Kröger Christian-Albrechts-Unversität zu Kiel

Datum der mündlichen Prüfung: 23. August, 2022

Zusammenfassung

Durch die Fortschritte im Bereich des maschinellen Lernens wird die maschinelle Sprachverarbeitung immer mächtiger. Auch im Bereich der maschinellen Verarbeitung von graphstrukturierten Daten gibt es vielversprechende Fortschritte. Allerdings wird es zugleich immer schwieriger, den Bedarf an GPU-Speicher und Rechenkraft für Sprachmodelle und Graphen zu handhaben. Insbesondere für kleine Unternehmen und Forschungslabore wird es dadurch zunehmend schwierig, an den Entwicklungen teilzuhaben. Das Ziel dieser Arbeit ist es, effiziente Methoden für das maschinelle Lernen von Text- und Graphrepräsentationen zu finden, die sich zudem kontinuierlich auf neue Daten anpassen lassen. Diese Arbeit ist zwischen maschineller Sprachverarbeitung und der maschinellen Verarbeitung von graphstrukturierten Daten angesiedelt und behandelt ausgewählte Verbindungen.

Erstens werden Wortmatrizen als ein effizientes Modell für Text eingeführt, das die Wortreihenfolge berücksichtigt. Nach selbstüberwachtem Training wird ein Satz durch Matrixmultiplikation der Wortmatrizen repräsentiert. Diese Repräsentation kann dann für weitere Aufgaben verwendet werden. Experimente mit zehn linguistischen Aufgaben sowie elf überwachten und fünf unüberwachten Aufgaben zeigen, dass Wortvektoren und -matrizen über komplementäre Stärken verfügen und dass ein gemeinsam trainiertes hybrides Modell besser funktioniert als die beiden einzelnen Komponenten.

Zweitens wird BERT als ein weitverbreitetes Modell für die maschinelle Sprachverarbeitung in ein auf Wortmatrizen basierendes Modell destilliert. Die Wortmatrizen werden insofern erweitert, als dass eine Rückwärtskomponente und eine Strategie zum Umgang mit Satzpaaren hinzugefügt wird. Die Ergebnisse auf dem GLUE Benchmark zeigen, dass dieses Modell ähnlich gut funktioniert wie andere kontextualisierte Sprachmodelle, aber mit einem geringeren Zeit- und Speicherbedarf auskommt.

Drittens werden drei verschiedene Modelltypen für die Textklassifikation verglichen: solche, die auf der Eingabe eines Bag-of-Words, einer Sequenz oder eines Graphen basieren. Die Experimente auf fünf Datensätzen zeigen, dass ein breites mehrschichtiges Perzeptron auf einer Bag-of-Words Repräsentation kompetitiv zu kürzlich erschienenen graphbasierten Ansätzen ist. Große Sprachmodelle schneiden zwar am besten ab, benötigen jedoch deutlich mehr Rechenkraft.

Viertens wird die Verbindung zwischen Texten und Graphen im Kontext von dokumentenbasierten Empfehlungssystemen für Zitationen und Themen untersucht. Experimente auf sechs Datensätzen zeigen, dass das Hinzufügen des Titels als Seiteninformation die Leistung von Autoencoder-Modellen verbessert. Die Ergebnisse werden unter verschiedenen Bedingungen bestätigt: der Anzahl der zu empfehlenden Elemente und der Anzahl der bereits vorhandenen Elemente pro Dokument. Schlussendlich ist die Bedeutung von gemeinsam auftretenden Zitationen oder Themen essenziell für die Wahl der Eingabemodalitäten und eines angemessenen Modells.

Fünftens wird ein generisches Rahmenwerk für das lebenslange Lernen auf Graphen vorgestellt, in denen neue Knoten, Kanten und Klassen nach und nach hinzugefügt werden. Basierend auf Text- und Graphdaten sollen Knoten klassifiziert und neue Klassen erkannt werden. Wir experimentieren mit fünf repräsentativen neuronalen Netzen für Graphen und drei Datensätzen aus wissenschaftlichen Publikationen: zwei Zitationsgraphen und einem Kollaborationsgraph. Die Ergebnisse zeigen, dass es durch das Wiederverwenden von vorherigen Modellparametern in einem inkrementellen Trainingsverfahren möglich wird, nur eine kleine Historie zu verwenden, ohne viel Genauigkeit gegenüber vollständiger Historie einzubüßen. Des Weiteren ist eine gewichtete Kostenfunktion essenziell, um der unbalancierten Klassenverteilung beim Erkennen von neuen Klassen entgegenzuwirken.

Zusammengefasst eröffnet diese Arbeit neue Möglichkeiten für effiziente Methoden zum Lernen von Text- und Graphrepräsentationen. Sie zeigt, wie Empfehlungssysteme textuelle Seiteninformationen ausnutzen können und legt den Grundstein für lebenslanges Lernen auf Graphen, in denen die Knoten mit textuellen Attributen versehen sind.

Abstract

Fueled by deep learning, natural language processing is becoming increasingly influential. Meanwhile, graph representation learning shows how to process graph data effectively. However, the size of language models and large, evolving graphs becomes increasingly challenging. The immense computing power and GPU memory requirements make it difficult for small companies and research labs to participate. This thesis aims at finding efficient text and graph representation learning methods that continually adapt to new data. This thesis is situated between natural language processing and graph representation learning and investigates selected connections.

First, we introduce matrix embeddings as an efficient text representation sensitive to word order. After self-supervised pretraining, the matrix product acts as sentence encoding for downstream tasks. Experiments with ten linguistic probing tasks, 11 supervised, and five unsupervised downstream tasks reveal that vector and matrix embeddings have complementary strengths and that a jointly trained hybrid model outperforms both.

Second, a popular pretrained language model, BERT, is distilled into matrix embeddings. To this end, we extend matrix embeddings with a bidirectional component and equip them with a strategy to encode sentence pairs. The results on the GLUE benchmark show that these models are competitive with other recent contextualized language models while being more efficient in time and space.

Third, we compare three model types for text classification: bag-of-words, sequence-, and graph-based models. Experiments on five datasets show that, surprisingly, a wide multilayer perceptron on top of a bag-of-words representation is competitive with recent graph-based approaches, questioning the necessity of graphs synthesized from the text. Pretrained Transformer-based sequence models perform best but come with high computational costs.

Fourth, we investigate the connection between text and graph data in document-based recommender systems for citations and subject labels. Experiments on six datasets show that the title as side information improves the performance of autoencoder models. We confirm this result under different experimental conditions: the number of all possible items and the fraction of already-present items per document. We find that the meaning of item co-occurrence is crucial for the choice of input modalities and an appropriate model.

Fifth, we introduce a generic framework for lifelong learning on evolving graphs in which new nodes, edges, and classes appear over time. The task is to classify nodes and detect new classes based on textual and graph information. We experiment with five representative graph neural network models and three datasets based on scholarly articles: two citation graphs and one collaboration graph. The results show that by reusing previous parameters in incremental training, it is possible to employ smaller history sizes with only a slight decrease in accuracy compared to training with complete history. Moreover, weighting the binary cross-entropy loss function is crucial to mitigate the problem of class imbalance when detecting newly emerging classes.

This work opens up new opportunities for efficient text and graph representation learning. It shows how recommender systems can exploit textual side information and lays the foundation for lifelong and open-world learning in evolving graphs with text-attributed nodes.

Acknowledgements

First and foremost, I thank my principal PhD advisor, **Ansgar Scherp** for steadily supporting me throughout my PhD studies and for guiding me to become the researcher that I am today. Thank you, Ansgar, for supporting me throughout all the strange times. I learned a lot from you.

Of course, I also thank my committee members: **Peer Kröger**, **Olaf Landsiedel**, and **Willi Hasselbring** for taking the time to engage with my research.

I thank my colleagues and collaborators that sprinkled my PhD studies with inspirational interactions: **Iacopo Vagliano** and **Florian Mai**, **Anne Lauscher**, **Tetyana Melnychuk**, **Eva Seidlmayer**, and **Marcel Hoffmann**. Thank you.

I thank the Bachelor's and Master's students, with whom I had the opportunity to work: **Gavin Lüdemann**, **Steffen Trog**, **Gunnar Gerstenkorn**, **Benedikt Franke**, **Tobias Zielke**, **Isabelle Cuber**, **Christoph Meyer**, **Henrik Ferdinand Nölscher**, **Angelina Sonderecker**, **Andor Diera**, **Bao Xin Lin**, **Bhakti Khera**, **Tim Meuser**, and **Tushar Singhal**. Thank you.

I thank my parents **Sven and Lioba Galke** for for encouraging me to do something that I love and for spurring my interest in computer science, machine learning, and neural networks. Thank you.

I thank **Melanie Poech**. Only through your steady love and understanding, I had the strength to endure all peculiarities throughout my PhD journey. Thank you. And thank you to **Max and Gertrude Poech** for enduring me during the hardest deadline push of my life (until now). Thank you.

Finally, I thank everyone that made the journey through my PhD more enjoyable and who was not mentioned before, friends and lab mates, **Till Blume**, **Lars Rohwedder**, **Falk Böschen**, **Martin Töpfer**. Thank you.

Contents

1 Introduction **1**
- 1.1 Advances and Challenges in Text and Graph Representation Learning 2
- 1.2 A Unified Perspective on Representation Learning for Texts and Graphs 5
- 1.3 Outline and Contributions 9

2 Background **13**
- 2.1 Representation Learning 13
  - 2.1.1 Multilayer Perceptron 14
  - 2.1.2 Autoencoder 15
  - 2.1.3 Training of Neural Networks 15
- 2.2 Text Representation Learning 18
  - 2.2.1 Tokenization and the Vocabulary 19
  - 2.2.2 One-Hot Encoding 20
  - 2.2.3 Bag-of-Words Representation 20
  - 2.2.4 Word Embeddings 21
  - 2.2.5 Transformers and Language Models 23
- 2.3 Graph Representation Learning 24
  - 2.3.1 Types of Graphs 24
  - 2.3.2 Graph Neural Networks 26
  - 2.3.3 Graph Approaches to NLP 27
- 2.4 Learning Paradigms 28
- 2.5 Summary 29

3 Word Matrices for Text Representation Learning **31**
- 3.1 Related Prior Work 32
- 3.2 Problem Formulation 33
- 3.3 Methods 34
  - 3.3.1 Continuous Bag-of-Words 34
  - 3.3.2 Continual Multiplication of Words 34
  - 3.3.3 CMOW/CBOW-Hybrid 35
  - 3.3.4 Training Objective 36
  - 3.3.5 Initialization 37

3.4 Datasets . . . 38
3.4.1 Dataset for Pretraining . . . 39
3.4.2 Datasets for Linguistic Probing Tasks . . . 39
3.4.3 Datasets for Supervised and Unsupervised Downstream Tasks 40
3.5 Experiments . . . 42
3.5.1 Linguistic Probing Tasks . . . 43
3.5.2 Supervised Downstream Tasks . . . 43
3.5.3 Unsupervised Downstream Tasks . . . 45
3.6 Discussion . . . 45
3.7 Summary . . . 47

4 Cross-Architecture Distillation with Word Matrices 49
4.1 Related Prior Work . . . 51
4.2 Problem Formulation . . . 53
4.3 Methods . . . 55
4.3.1 Extending Matrix Embedding Models . . . 55
4.3.2 Cross-Architecture Distillation . . . 58
4.3.3 Two-Sequence Encoding with Matrix Embeddings . . . 58
4.4 Datasets . . . 60
4.4.1 Dataset for Pretraining . . . 60
4.4.2 Datasets for Downstream Tasks . . . 60
4.5 Experiments . . . 62
4.5.1 DiffCat Encoding versus Joint Encoding . . . 64
4.5.2 Bidirectional versus unidirectional CMOW/CBOW-Hybrid . 64
4.5.3 General Distillation versus Task-specific Distillation . . . 65
4.5.4 Comparing Bidirectional CMOW/CBOW-Hybrid to the Literature . . . 65
4.5.5 Comparison of Parameter Count and Runtime Performance . 66
4.6 Discussion . . . 67
4.7 Summary . . . 69

5 Wide Multilayer Perceptrons for Text Classification 71
5.1 Related Prior Work . . . 73
5.1.1 Bag-of-Word-based Models . . . 73
5.1.2 Graph-based Models . . . 74
5.1.3 Sequence-based Models . . . 75
5.1.4 Summary . . . 76

5.2 Problem Formulation . . . 77
5.3 Methods . . . 77
5.3.1 Bag-of-Words-based Text Classification . . . 78
5.3.2 Graph-based Text Classification . . . 78
5.3.3 Sequence-based Text Classification . . . 79
5.4 Datasets . . . 79
5.5 Experiments . . . 80
5.5.1 Classification Accuracy . . . 82
5.5.2 Efficiency . . . 82
5.6 Discussion . . . 84
5.7 Summary . . . 87

6 Multimodal Autoencoders for Document-based Recommendations 89
6.1 Related Prior Work . . . 93
6.1.1 Autoencoders as Recommendation Engines . . . 93
6.1.2 Research Paper and Citation Recommendation . . . 94
6.1.3 Subject Label Recommendation . . . 95
6.1.4 Summary . . . 96
6.2 Problem Formulation . . . 97
6.2.1 Scenarios and Common Framework . . . 97
6.2.2 Formal Problem Statement . . . 98
6.3 Methods . . . 100
6.3.1 Singular Value Decomposition . . . 100
6.3.2 Item Co-Occurrence . . . 101
6.3.3 Multilayer Perceptrons . . . 101
6.3.4 Undercomplete Autoencoders . . . 102
6.3.5 Denoising Autoencoders . . . 103
6.3.6 Variational Autoencoders . . . 103
6.3.7 Adversarial Autoencoder . . . 104
6.3.8 Conditioning Autoencoder on Side Information . . . 105
6.4 Datasets . . . 106
6.4.1 Datasets for Citation Recommendation . . . 107
6.4.2 Datasets for Subject Labels Recommendation . . . 109
6.4.3 Availability of Side Information . . . 113
6.4.4 Chronological Train-Test Splits . . . 114
6.4.5 Evaluation Measures . . . 115
6.5 Experiments . . . 115

6.5.1 Experiments under Varying Total Number of Items 116
6.5.2 Experiments under Varying Number of Items per Document 119
6.6 Discussion 122
6.6.1 Key Results 122
6.6.2 Meanings of Item Co-occurrence 124
6.6.3 Discussion of the Citation Recommendation Task 125
6.6.4 Discussion of Subject Label Recommendation Task 127
6.6.5 Threats to Validity 129
6.6.6 Practical Impact 129
6.7 Summary 130

7 Lifelong Learning on Evolving Graphs 131
7.1 Related Prior Work 136
7.1.1 Lifelong Learning 136
7.1.2 Graph Neural Networks 138
7.1.3 Unseen Class Detection and Out-of-Distribution Detection . . 139
7.1.4 Summary 140
7.2 Problem Formulation 140
7.3 Methods 142
7.3.1 Incremental Training for Lifelong Graph Learning 142
7.3.2 Unseen Class Detection 144
7.3.3 Measure of k-Neighborhood Time Differences 145
7.3.4 Base Graph Neural Network Models 148
7.4 Datasets 149
7.4.1 Static Graph Datasets 150
7.4.2 Evolving Graph Datasets 151
7.5 Experiments 155
7.5.1 Transductive versus Inductive Learning 155
7.5.2 Incrementally-trained vs Once-trained Models 159
7.5.3 Lifelong Learning on Graphs 160
7.5.4 Lifelong Learning with Limited Labeled Data 164
7.5.5 Self-Detection of Unseen Classes 166
7.6 Discussion 171
7.7 Summary 172

8 Conclusion **175**
8.1 Summary of Contributions . . . 175
8.2 General Discussion . . . 176
8.3 Future Work . . . 179
8.4 Summary . . . 180

A Reproducibility and Published Resources **181**
A.1 Reproducibility of Relevant Literature . . . 181
A.2 Reproducibility of Our Experiments . . . 182
A.3 Published Resources . . . 183

B Supplementary Material: Word Matrices **185**
B.1 Comparison of Training Objectives . . . 185
B.2 Comparison of Initialization Strategies . . . 185

C Supplementary Material: Cross-Architecture Distillation **189**
C.1 Hyperparameters for Distillation . . . 189
C.2 Extended Results . . . 189

D Supplementary Material: Text Classification **191**
D.1 Practical Guidelines for Designing a WideMLP . . . 191
D.2 Connection between BoW-MLP and TextGCN . . . 192
D.3 Equivalence of Micro-F1 and Accuracy in Multiclass Classification . 193

E Supplementary Material: Recommender Systems **195**
E.1 Extended Results on the PubMed Dataset . . . 195
E.2 Mean Average Precision Results . . . 195

F Supplementary Material: Lifelong Learning on Graphs **199**
F.1 Proof: k-Neighborhood Time Differences tdiff_k is Equivariant to Temporal Granularity . . . 199
F.2 Details on Changes in the Class Sets . . . 200
F.2.1 PharmaBio . . . 200
F.2.2 DBLP-Easy . . . 201
F.2.3 DBLP-Hard . . . 201
F.3 Extended Results for Unseen Class Detection . . . 203

Bibliography **207**

List of Publications

Here, I list the publications with which I conveyed the research conducted for the present thesis, clarify which publications share material with this thesis, and outline the contributions that others have made to those publications. The following preprints and publications share material with the present thesis.

[GCM+22] Lukas Galke, Isabelle Cuber, Christoph Meyer, Henrik Ferdinand Nölscher, Angelina Sonderecker, and Ansgar Scherp. "General cross-architecture distillation of pretrained language models into matrix embeddings". In: *International Joint Conference on Neural Networks, IJCNN 2022, Padua, Italy, July 18-23, 2022*. IEEE, 2022, pp. 1–10. DOI: 10.1109/IJCNN55064.2022.9892144. URL: https://doi.org/10.1109/IJCNN55064.2022.9892144.

[GFZ+21] Lukas Galke, Benedikt Franke, Tobias Zielke, and Ansgar Scherp. "Lifelong learning of graph neural networks for open-world node classification". In: *International Joint Conference on Neural Networks, IJCNN 2021, Shenzhen, China, July 18-22, 2021*. IEEE, 2021, pp. 1–8. DOI: 10.1109/IJCNN52387.2021.9533412.

[GMS19] Lukas Galke, Florian Mai, and Ansgar Scherp. "What if we encoded words as matrices and used matrix multiplication as composition function?" In: *49. Jahrestagung der Gesellschaft für Informatik, 50 Jahre Gesellschaft für Informatik - Informatik für Gesellschaft, INFORMATIK 2019, Kassel, Germany, September 23-26, 2019*. Vol. P-294. LNI. GI, 2019, pp. 287–288. DOI: 10.18420/inf2019_47.

[GMV+18] Lukas Galke, Florian Mai, Iacopo Vagliano, and Ansgar Scherp. "Multi-modal adversarial autoencoders for recommendations of citations and subject labels". In: *Proceedings of the 26th Conference on User Modeling, Adaptation and Personalization, UMAP 2018, Singapore, July 08-11, 2018*. ACM, 2018, pp. 197–205. DOI: 10.1145/3209219.3209236.

[GS22] Lukas Galke and Ansgar Scherp. "Bag-of-words vs. graph vs. sequence in text classification: Questioning the necessity of text-graphs and the surprising strength of a wide MLP". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 4038–4051. DOI: 10.18653/v1/2022.acl-long.279. URL: https://aclanthology.org/2022.acl-long.279.

[GVF+21] Lukas Galke, Iacopo Vagliano, Benedikt Franke, Tobias Zielke, and Ansgar Scherp. *Lifelong learning in evolving graphs with limited labeled data and unseen class detection*. Under review. 2021. arXiv: 2112.10558 [cs.LG].

[GVS19] Lukas Galke, Iacopo Vagliano, and Ansgar Scherp. "Can graph neural networks go "online"? An analysis of pretraining and inference". In: *Representation Learning on Graphs and Manifolds, ICLR Workshop*. 2019. URL: https://rlgm.github.io/papers/21.pdf.

[MGS19] Florian Mai, Lukas Galke, and Ansgar Scherp. "CBOW is not all you need: Combining CBOW with the compositional matrix space model". In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL: https://openreview.net/forum?id=H1MgjoR9tQ.

[VGS22] I. Vagliano, L. Galke, and A. Scherp. "Recommendations for item set completion: On the semantics of item co-occurrence with data sparsity, input size, and input modalities". In: *Information Retrieval Journal* (Apr. 2022). ISSN: 1573-7659. DOI: 10.1007/s10791-022-09408-9.

Here, I list the roles of my coauthors in these publications on the basis of the Contributor Roles Taxonomy, see: http://credit.niso.org/. I *conceptualized* the research of these publications and in this thesis, while taking advice from Ansgar Scherp. To all these publications, I contributed *Investigation*, *Methodology*, *Software*, *Visualization*, *Writing* and *Data Curation* (when we introduced new datasets).

- Florian Mai has contributed *Investigation*, *Methodology*, and *Software* to [MGS18].
- Iacopo Vagliano has contributed *Investigation*, *Methodology* and *Software* to [VGS22].
- The project group on lifelong learning in graphs, Benedikt Franke and Tobias Zielke, contributed *Investigation* using the methods SGC, GraphSAINT, and JKNet, after they have integrated those methods (*Software*) into the lifelong graph learning framework that I had developed. I have *co-supervised* the project together with Ansgar Scherp. After the project had finished, I *validated* and re-ran the experiments.
- The project group on Transformer distillation: Isabelle Cuber, Christoph Mezer, Ferdinand Nölscher, Angelina Sonderecker have *investigated* and implemented (*Software*) task-specific distillation with matrix-space embedding models. I have *co-supervised* the project along with Ansgar Scherp. After the project had finished, I *validated* the experiments.

During my studies, I have further contributed to the following publications, which do not share material with this thesis.

[GGS18] Lukas Galke, Gunnar Gerstenkorn, and Ansgar Scherp. "A case study of closed-domain response suggestion with limited training data". In: *Database and Expert Systems Applications - DEXA 2018 International Workshops, BDMICS, BIOKDD, and TIR, Regensburg, Germany, September 3-6, 2018, Proceedings*. Vol. 903. Communications in Computer and Information Science. Springer, 2018, pp. 218–229. DOI: 10.1007/978-3-319-99133-7_18.

[GMS+17] Lukas Galke, Florian Mai, Alan Schelten, Dennis Brunsch, and Ansgar Scherp. "Using titles vs. full-text as source for automated semantic document annotation". In: *Proceedings of the Knowledge Capture Conference, K-CAP 2017, Austin, TX, USA, December 4-6, 2017*. ACM, 2017, 20:1–20:4. DOI: 10.1145/3148011.3148039.

[GMS+19] Lukas Galke, Tetyana Melnychuk, Eva Seidlmayer, Steffen Trog, Konrad U. Förstner, Carsten Schultz, and Klaus Tochtermann. "Inductive learning of concept representations from library-scale bibliographic corpora". In: *49. Jahrestagung der Gesellschaft für Informatik, 50 Jahre Gesellschaft für Informatik - Informatik für Gesellschaft, INFORMATIK 2019, Kassel, Germany, September 23-26, 2019*. Vol. P-294. LNI. GI, 2019, pp. 219–232. DOI: 10.18420/inf2019_26. URL: https://doi.org/10.18420/inf2019_26.

[GSL+21] Lukas Galke, Eva Seidlmayer, Gavin Lüdemann, Lisa Langnickel, Tetyana Melnychuk, Konrad U. Förstner, Klaus Tochtermann, and Carsten Schultz. "COVID-19++: A citation-aware COVID-19 dataset for the analysis of research dynamics". In: *IEEE Big Data 2021 Workshop on Big Data Analytics for COVID-19*. IEEE, 2021.

[GSS17] Lukas Galke, Ahmed Saleh, and Ansgar Scherp. "Word embeddings for practical information retrieval". In: *47. Jahrestagung der Gesellschaft für Informatik, Digitale Kulturen, INFORMATIK 2017, Chemnitz, Germany, September 25-29, 2017*. Vol. P-275. LNI. GI, 2017, pp. 2155–2167. DOI: 10.18420/in2017_215. URL: https://doi.org/10.18420/in2017_215.

[LEG+18] Anne Lauscher, Kai Eckert, Lukas Galke, Ansgar Scherp, Syed Tahseen Raza Rizvi, Sheraz Ahmed, Andreas Dengel, Philipp Zumstein, and Annette Klein. "Linked open citation database: Enabling libraries to contribute to an open and interconnected citation graph". In: *Proceedings of the 18th ACM/IEEE on Joint Conference on Digital Libraries*. JCDL '18. Fort Worth, Texas, USA: ACM, 2018, pp. 109–118. ISBN: 978-1-4503-5178-2. DOI: 10.1145/3197026.3197050. URL: http://doi.acm.org/10.1145/3197026.3197050.

[MGS+21] Tetyana Melnychuk, Lukas Galke, Eva Seidlmayer, Konrad U. Förstner, Klaus Tochtermann, and Carsten Schultz. "Früherkennung wissenschaftlicher Konvergenz im Hochschulmanagement". In: *Hochschulmanagement* 16 (1 2021), pp. 24–28.

List of Publications

[MGS18] Florian Mai, Lukas Galke, and Ansgar Scherp. "Using deep learning for title-based semantic subject indexing to reach competitive performance to full-text". In: *Proceedings of the 18th ACM/IEEE on Joint Conference on Digital Libraries, JCDL 2018, Fort Worth, TX, USA, June 03-07, 2018*. ACM, 2018, pp. 169–178. DOI: 10.1145/3197026.3197039.

[SBG+18] Ahmed Saleh, Tilman Beck, Lukas Galke, and Ansgar Scherp. "Performance comparison of ad-hoc retrieval models over full-text vs. titles of documents". In: *Maturity and Innovation in Digital Libraries - 20th International Conference on Asia-Pacific Digital Libraries, ICADL 2018, Hamilton, New Zealand, November 19-22, 2018, Proceedings*. Vol. 11279. Lecture Notes in Computer Science. Springer, 2018, pp. 290–303. DOI: 10.1007/978-3-030-04257-8_30.

[VGM+18] Iacopo Vagliano, Lukas Galke, Florian Mai, and Ansgar Scherp. "Using adversarial autoencoders for multi-modal automatic playlist continuation". In: *Proceedings of the ACM Recommender Systems Challenge, RecSys Challenge 2018, Vancouver, BC, Canada, October 2, 2018*. ACM, 2018, 5:1–5:6. DOI: 10.1145/3267471.3267476.

List of Figures

3.1 The continual multiplication of words (CMOW)/continuous bag-of-words (CBOW)-Hybrid model during pretraining. 35
3.2 Mean of the absolute values of the text embeddings (y-axis) plotted depending on the number of multiplications (x-axis) for the three initialization strategies. The absolute value of the embeddings quickly decreases with more multiplications for standard initialization strategies. When we apply our initialization method, the absolute values of the embeddings have the same magnitude regardless of the sentence length. 38
3.3 Results on linguistic probing tasks. 44
3.4 Results on supervised downstream tasks 44
3.5 Results on the unsupervised downstream tasks 45

4.1 The bidirectional CMOW component of our proposed architecture during pretraining. In this example, the model predicts the masked token at position 4 by concatenating forward and backward matrix embeddings, which are then fed into a masked language modeling head. Bert illustration: Melanie Poech 56
4.2 Separate encoding (DiffCat) for sequence pairs using a Bidirectional CMOW/CBOW-Hybrid model during fine-tuning, optionally, with task-specific distillation with a pre-training of deep bidirectional Transformers (BERT) teacher. Bert illustration: Melanie Poech 59

6.1 Exemplary bipartite graphs of citation relationships between documents (left) and documents annotated with subject labels (right). 98
6.2 A multilayer perceptron (MLP) with two hidden layers 102
6.3 Undercomplete autoencoder 102
6.4 Denoising autoencoder (DAE) 103
6.5 Variational autoencoder (VAE) 103
6.6 Adversarial autoencoder (AAE) 104
6.7 Characteristics of the PubMed dataset 107

6.8 Characteristics of the DBLP dataset 109
6.9 Characteristics of the ACM dataset 110
6.10 Characteristics of the EconBiz Dataset 111
6.11 Characteristics of the IREON dataset 112
6.12 Characteristics of the Reuters dataset 113
6.13 Mean reciprocal rank of predicted citations on the test set with varying minimum item occurrence (pruning) thresholds for the PubMed (top row), DBLP (middle row), and ACM (bottom row) citation datasets. *Left*: Only the partial set of items is given. *Center*: The partial set of items along with the document title is given. *Right*: The partial set of items is given along with the document title, the authors, the journal title and the MeSH labels (if available). MLP can only make use of either titles or titles, authors, journal titles, and MeSH labels. 118
6.14 MRR of predicted subject labels on the test set with varying minimum item occurrence thresholds for the EconBiz (top row), IREON (middle row) and Reuters (bottom row) dataset. *Left*: Only the partial set of items is given. *Center*: The partial set of items along with the document title is given. *Right*: The partial set of items along with the title and authors is given. MLP can only use either titles or titles and authors. 120
6.15 MRR of predicted citations on the test set with varying number dropped elements for the PubMed (top row) and ACM (bottom row) citation datasets. The minimum item occurrence threshold is set to 55. *Left*: Only the partial set of items is given. *Center*: The partial set of items along with the document title is given. *Right*: The partial set of items is given along with the document title, the authors, the journal title, and the MeSH labels. MLP can only make use of either titles or titles, authors, journal titles, and MeSH labels. 121
6.16 MRR of predicted subject labels on the test set with varying number of dropped elements for the EconBiz (top row), IREON (middle row) and Reuters (bottom row) datasets. The minimum item occurrence threshold is set to 20. *Left*: Only the partial set of items is given. *Center*: The partial set of items along with the title is given. *Right*: The partial set of items along with the document title and authors is given. MLP can only use either titles or titles and authors. 123

7.1 Lifelong Open-World Node Classification. At each time t the learner has to classify new vertices of task $\mathcal{T}_t$ (red). The learner may use knowledge from previous tasks to adapt to the current task, eventually cut off by a history size (blue). The current task might come with previously unseen classes. For example, the "c" appears only at task $t-2$ and was subsequently added to the class set. After evaluating each task $\mathcal{T}_t$, we continue with task $\mathcal{T}_{t+1}$. 133
7.2 Example of time differences $\text{tdiff}_2(G)$ for hops at distance of up to 2 from each node. Node annotations show the time difference to all nodes in the two-hop neighborhood, i. e., its contribution to the multiset $\text{tdiff}_2(G)$. The calculation for the node with time $t = 21$ is highlighted in orange. 147
7.3 Distribution of vertices per year on log scale (left column), degree distributions (middle column), label distributions (right column), for our new datasets: DBLP-easy (top row), DBLP-hard (middle row), PharmaBio (bottom row) 153
7.4 Magnitude of the class drift per dataset. The drift within the PharmaBio dataset (no new classes) is lower than the drift of both DBLP variants. Independent and identically distributed data would have drift magnitude zero. 154
7.5 Distributions of time differences tdiff_k (y-axis) for DBLP-easy (left), DBLP-hard (center) and PharmaBio (right) within the k-hop neighborhood for $k = \{1, 2, 3\}$ (x-axis). 154
7.6 Test accuracy after each inference epoch for the many-few settings A *(Top)* and few-many setting B *(Bottom)* on the datasets Cora, Citeseer, and PubMed. Each line resembles the mean of 100 runs and its region shows the standard deviation. The dashed lines show the results with 200 pretraining. The solid lines are the results without pretraining. 158
7.7 Accuracy scores of once-trained, static models (solid lines) are lower than incrementally trained models (dashed lines). 159
7.8 Average accuracy of GraphSAGE with warm restarts across tasks on DBLP-hard under varying label rate 165
7.9 Number of nodes with unseen classes per task on DBLP-hard . . . 166
7.10 Global MCC of GDOC (history size 3, warm) as a function of the SD Factor for risk reduction with different minimum threshold values. 168

E.1 MRR of predicted citations on the test set with varying number dropped elements for the PubMed citation dataset. The minimum item occurrence threshold is set to 55. *Left*: The partial set of items is given along with the document title, the authors, and the journal title. *Right*: The partial set of items is given along with the document title, the authors, the journal title, and the MeSH labels. 196
E.2 MAP of predicted citations on the test set with varying number dropped elements for the PubMed (top row) and ACM (bottom row) citation datasets. The minimum item occurrence threshold is set to 55. *Left*: Only the partial set of items is given. *Center*: The partial set of items along with the document title is given. *Right*: The partial set of items is given along with the document title, the authors, the journal title, and the MeSH labels. MLP can only make use of either titles or titles, authors, journal titles, and MeSH labels. 197
E.3 MAP of predicted subject labels on the test set with varying number of dropped elements for the EconBiz (top row), IREON (middle row), and Reuters (bottom row) datasets. The minimum item occurrence threshold is set to 20. *Left*: Only the partial set of items is given. *Center*: The partial set of items along with the title is given. *Right*: The partial set of items along with the document title and authors is given. MLP can only use either titles or titles and authors. 198

List of Tables

4.1 Comparison of DiffCat encoding and joint BERT-like encoding. Both variants use randomly initialized unidirectional CMOW/CBOW-Hybrid embeddings with MLP under task-specific distillation. The DiffCat encoding improves the average score across two-sentence tasks by 20%. 64

4.2 Comparison of unidirectional and bidirectional (prefix 'B-') variants of CMOW/CBOW-Hybrid under task-specific distillation from random initialization (suffix '-R'). We also report pretrained (suffix '-P') bidirectional Hybrid for reference. All methods use DiffCat encoding and an MLP classifier. 65

4.3 Comparison of general and task-specific distillation using bidirectional CMOW/CBOW-Hybrid embeddings and MLP classifier. Task-specific distillation models have been either randomly initialized (suffix '-R') or initialized from pretraining (suffix '-P'). In 5 out of 9 tasks, general distillation performs better. 65

4.4 Comparison of best embedding-based methods (in bold) with methods from the literature on the validation set of the general language understanding evaluation benchmark (GLUE) benchmark. 66

4.5 Number of parameters and inference time of the models. Inference time is measured as encoding speed without gradient computation on an NVIDIA A100-SXM4-40GB card 66

5.1 Properties of text classification approaches. Graph-based models that rely on having access to unlabeled test documents such as TextGCN and TensorGCN are not capable of inductive learning without specific modifications. 77

5.2 Characteristics of text classification datasets 79

5.3 Accuracy and standard deviation on text classification datasets. Column "Provenance" reports the source. 83

5.4 Parameter counts of the models 84

5.5 Total runtime (training+inference). Average of five runs rounded to minutes. 84

6.1 Notation table 99
6.2 Availability and occurrence of metadata in the datasets considered for the two recommendation tasks. Subject labels and item set occurrences are the same for the subject-label datasets as the subject labels are the items to recommend (but can also be used as additional metadata for the citation tasks). 114
6.3 Characteristics of the citation datasets with respect to different selected pruning thresholds on the minimum item occurrence. 116
6.4 Characteristics of the subject label datasets with respect to different selected pruning thresholds on minimum item occurrence. 117

7.1 Statistics for train-test splits: few-many (A) and many-few (B) settings on the citation networks datasets: Cora, Citeseer, and PubMed. The unseen nodes and edges are available only after the training epochs. The test samples for measuring accuracy are a subset of the unseen nodes. The label rate is the percentage of labeled nodes for training. 151
7.2 Global dataset characteristics: total number of nodes $|V|$, edges $|E|$, features D, classes $|\mathbb{Y}|$ along with # of newly appearing classes (in braces) within the T evaluation tasks 152
7.3 Accuracy (with 95% confidence intervals through 1.96 standard error of the mean) and Forward Transfer (averaged difference of warm and cold restarts) in our datasets with different history sizes (column **c**). The best method per dataset and history size is marked in bold, along with the methods where the 95% CI overlaps. 163
7.4 Results for unseen class detection on DBLP-hard with GraphSAGE as base model (average of 5 repetitions). α indicates that risk reduction is used with the respective factor for the standard deviation, τ is the minimum threshold. Runs named graph deep open classification (GDOC) are trained with weighted cross entropy. Deep Open Classification (DOC) is our baseline. 169

A.1 Published resources in the context of this thesis 183

B.1 Scores for different training objectives on the linguistic probing tasks. 186
B.2 Scores for different training objectives on the supervised downstream tasks. 186

B.3 Scores for different training objectives on the unsupervised downstream tasks. 186
B.4 Scores for initialization strategies on probing tasks. 187
B.5 Scores for initialization strategies on supervised downstream tasks. 187
B.6 Scores for initialization strategies on unsupervised downstream tasks. 187

C.1 Hyperparameter Search Space 189
C.2 Scores on the GLUE development set. Our best performing general distillation and task-specific distillation models are highlighted in bold font per task. References indicate sources of scores. The $\star$-symbol indicates numbers on the official GLUE test set. 'Hybrid' denotes CMOW/CBOW-Hybrid. 190

E.1 MRR of predicted citations for PubMed using either only documents with fewer references than the median number of references or with more references than the median number of references. 196
E.2 MAP of predicted citations for PubMed using either only documents with fewer references than the median number of references (left column) or with more references than the median number of references (right column). 197

F.1 Results for unseen class detection on the development set DBLP-easy with GraphSAGE as base model (average of 5 repetitions). α indicates that risk reduction is used with the respective factor for the standard deviation, τ is the minimum threshold. Runs named gDOC are trained with weighted cross entropy. DOC is our baseline. 204
F.2 Extended results for unseen class detection on DBLP-hard with GraphSAGE as base model (average of 5 repetitions). α indicates that risk reduction is used with the respective factor for the standard deviation, τ is the minimum threshold. The gDOC models are trained with weighted binary-cross entropy. History sizes 1 and 3. . 205
F.3 Extended results for unseen class detection on DBLP-hard with GraphSAGE as base model (average of 5 repetitions). α indicates that risk reduction is used with the respective factor for the standard deviation, τ is the minimum threshold. The gDOC models are trained with weighted binary-cross entropy. History sizes 6 and 3. . 206

List of Acronyms

AAE adversarial autoencoder
AE autoencoder
AI artificial intelligence
BERT pre-training of deep bidirectional Transformers
BoW Bag-of-Words
CBOW continuous bag-of-words
CMOW continual multiplication of words
CMSM compositional matrix-space model
CNN convolutional neural network
CoLA Corpus of Linguistic Acceptability
CR Customer Reviews
DAE denoising autoencoder
DAN deep averaging network
DOC Deep Open Classification
ELMo Embeddings from Language Models
FWT Forward Transfer
GAT graph attention network
GCN graph convolutional network
GDOC graph deep open classification
GloVe global vectors for word representation
GLUE general language understanding evaluation benchmark
GNN graph neural network
GPT-3 3rd version of the generatively pretrained Transformer
GPU graphics processing unit
GraphSAGE graph sampling and aggregating

List of Acronyms

GraphSAINT graph sampling based inductive learning method
IDF inverse document frequency
JKNet jumping-knowledge network
LDA Latent Dirichlet Allocation
LM language model
LSTM long short-term memory
MCC Matthew's correlation coefficient
MeSH medical subject headings
MI mutual information
MLM masked language model
MLP multilayer perceptron
MNLI Multi-Genre Natural Language Inference corpus
MPQA Multi-Perspective Question Answering
MRPC Microsoft Research Paraphrase Corpus
MRR mean reciprocal rank
MR Movie Reviews
NLI natural language inference
NLP natural language processing
PMI pointwise mutual information
PreLM pretrained language model
QNLI Question Natural Language Inference
QQP Quora Question Pairs
ReLU rectified linear unit
RNN recurrent neural network
RoBERTa a robustly optimized BERT pretraining approach
RTE Recognizing Textual Entailment
SD standard deviation
seq2seq sequence-to-sequence

SGC simplified graph convolutional network
SICK Sentences Involving Compositional Knowledge
SOMO Semantic Odd Man Out
SST-2 two-way Stanford Sentiment Treebank
SST-5 five-way Stanford Sentiment Treebank
SST Stanford Sentiment Treebank
STS-B Semantic Textual Similarity Benchmark
STS Semantic Textual Similarity
SVD singular value decomposition
SVM support vector machine
SWEM simple word embedding model
T5 text-to-text transfer Transformer
TF-IDF term frequency—inverse document frequency
TREC Text REtrieval Conference
VAE variational autoencoder
WC Word Content
WNLI Winograd Natural Language Inference
word2vec word-to-vector
IC Item Co-occurrence

Terminology and Notation

Terminology

Here, a few potentially ambiguous terms are clarified. For the context of this thesis, we distinguish between a graph that refers to the data, and networks that refer to the neural network model.

[model] architecture The blueprint for the model

model An instance of a model architecture

parameters The learnable parameters of a model

features Input representation for the model

[neural] network a layered model architecture with learnable parameters

hidden units The "neurons" within in a neural network

hidden layer a collection of hidden units organized in a layer within a multi-layered architecture.

layer The connections between two collections of units. A model with one hidden layer has two layers: input-to-hidden and hidden-to-output.

layer's representation The output of the layer.

weights The weight parameters of a layer

bias Refers to the bias parameters of a layer, sometimes also used on a meta-level for experimental biases. Context should be sufficient to disambiguate.

Transformer A neural network architecture comprising multiple self-attention and fully-connected layers [VSP+17]

graph Graph data comprising nodes and edges.

node A node of the graph

node features Data attached to each node that is used as model input

edge An edge of the graph

graph structure The structure within the graph data, i. e., nodes and edges but no node features.

Notation

We adopt a subset of the notation[1] of Goodfellow, Bengio, and Courville [GBC16] with minor adaptions and some extensions.

a A scalar (integer or real)

$\boldsymbol{a}$ A vector

$\boldsymbol{A}$ A matrix

$\boldsymbol{I}_n$ Identity matrix with n rows and n columns

$\mathbf{A}$ A tensor

a A scalar random variable

$\mathbf{a}$ A vector-valued random variable

$\mathbf{A}$ A matrix-valued random variable

a_i Element i of vector $\boldsymbol{a}$, with indexing starting at 1.

$A_{i,j}$ Element i, j of matrix $\boldsymbol{A}$

$\boldsymbol{A}_{i,:}$ Row i of matrix $\boldsymbol{A}$

$\boldsymbol{A}_{:,i}$ Column i of matrix $\boldsymbol{A}$

$\boldsymbol{A}[i]$ Row i of matrix $\boldsymbol{A}$

$\mathbf{A}[i]$ Matrix i of a rank 3 tensor $\mathbf{A}$, sliced along the first dimension.

$\|$ Concatenation of two vectors (or two tensors along their last dimension)

[1] https://github.com/goodfeli/dlbook_notation

flatten Flatten a matrix into a vector.

softmax The softmax function

$\mathcal{L}$ Loss function

σ Sigmoid activation function, or generic activation function, if mentioned explicitly

BCE Binary Cross-Entropy

$\mathbb{E}$ Expected value

Var Variance

SD Standard deviation

SE Standard error of the mean

D_{KL} Kullback-Leibler Divergence

$\mathbf{x} \sim P$ $\mathbf{x}$ is distributed as P

$\mathcal{G}$ A graph

$\mathcal{V}$ Set of nodes

$\mathcal{E}$ Set of edges

$\boldsymbol{A}$ The adjacency matrix of a graph

Chapter 1

Introduction

Text is a communication medium for language, and language is human's primary tool to convey information. Data in textual format provide a rich source of information for numerous applications, such as text classification and recommender systems. Fueled by representation learning [BCV13; LBH15; GBC16], natural language processing (NLP) has evolved into a powerful tool to take advantage of this rich source of information.

However, a rarely considered factor is context information beyond the text. For humans, context information is crucial to assess a text's importance, quality, and meaning For instance, a text can be a response to some other text, be part of a bigger unit, or at least have context information, such as an author and its production time. Learning systems that use pure text to solve a task disregard information that might be as important as the text itself. *How can we design multimodal models that jointly process text together with more structured data that comes with it? Moreover, does it help?*

In previous work, we have shown that scientific papers can be very well classified into topics purely based on their title [GMS+17; MGS18]. Still, some papers are hard to classify only by their title. For example, consider the paper entitled "Optimal Brain Damage" [LDS89]. Is it medicine, neuroscience, or artificial intelligence (AI)? Hard to say. However, what if we knew that the first author also co-authored a paper entitled "Deep Learning" [LBH15] and that Deep Learning is a well-established subfield of AI. This context might help us determine that we can sort the paper to AI and we might even correctly guess that it is about the pruning of *artificial* neural networks (thankfully).

Still, the "Deep Learning" paper came out 25 years later, and no learning system can learn from the future. The world around us is evolving, and so are its data. In addition, new categories (or classes) may emerge. As models and data grow larger and larger, rebuilding models whenever new data come in becomes more and more expensive. *How can we adapt existing models to changes in evolving data?*

From a high-level perspective, this thesis examines the edge between text and graph representation learning. It aims to create models that jointly process both text and graph data and do so efficiently and effectively while adapting to new data. Having such representation learning models would lead to a more holistic approach of text and graph representation learning and would affect the processing of a wide range of real-world data, as found in social graphs, citation graphs, and collaboration graphs.

However, creating unified models for text and graph data is challenging because recent models for only one of these data types already consume an immense amount of resources. On the one hand, contemporary language models are becoming larger and larger [SGM19; BHA+21]. On the other hand, graph data are often massive in themselves, making them difficult to handle [FLW+21]. Thus, combining the most prominent models of both domains is not feasible, leaving us with various challenges at the intersection of text and graph representation learning, which we outline in more detail below.

1.1 Advances and Challenges in Text and Graph Representation Learning

Representation learning [BCV13] has sparked advances in many research fields, such as computer vision [KSH12; HZR+16], natural language processing [SVL14; VSP+17; DCL+19; BMR+20], and graph representation learning [PAS14; KW17; Ham20]. MT The key idea is to learn a helpful representation of data automatically, optionally, with multiple abstraction layers, called *deep* learning [LBH15; GBC16].

Most deep learning methods rely on deep neural networks. Deep neural networks are a stack of layers in which each layer computes a non-linear transformation of its input. The parameters are then optimized end-to-end by gradient descent. With two layers, neural networks are universal function approximators [Cyb89], which means that they can approximating any (compact) function to an arbitrary degree of accuracy, depending on the width of the intermediate layer.

Neural networks are technically able to adapt to new data, but this comes at the price of catastrophic forgetting [Rob95], which means that old knowledge is quickly lost when adapting to new tasks. Catastrophic forgetting remains a crucial challenge when adapting models to new data [LR17].

In text representation learning for natural language processing, the transfer learning paradigm has emerged with great success [DCL+19; Rud19]. In transfer

learning, we train models on one task then transfer them to another. A typical approach in NLP is to pretrain models on large amounts of unlabeled text and then fine-tune the models for the tasks of interest.

The first example of this paradigm with tremendous success was word embeddings [MSC+13; PSM14], i. e., learned representations of single words. Later, it became popular to encode entire text blocks together [PNI+18; HR18]. Vital transfer learning approaches have emerged with the rise of the Transformer model [VSP+17]. Famous examples are BERT [DCL+19], the multitask text-to-text model T5 [RSR+20], and the purely autoregressive GPT-3 [BMR+20]. We call these models pretrained language models (PreLMs).

However, the recent success of PreLMs comes at the cost of ever-increasing model sizes, which translate into increasing financial and ecological costs [SGM19]. At the time of writing, large PreLMs have billions or even trillions of parameters and require terabytes of textual training data [FZS21]. As a result, the power to design and train these large models lies in the hands of a few global players alone [BHA+21].

Although considerable efforts improve the efficiency of PreLMs [SDC+20; JYS+20; SCG+19; TDB+20], the core concept of self-attention remains an obstacle that prevents us from simply extending the most prominent models to take graph data together with text into account. Thus, in this thesis, we will first explore more efficient text representation models.

However, again, there is no isolated text. The text comes with valuable structured information, such as its author, where it has appeared, or whether it is a response to some other text. We will study recommendation systems as a bridge between text and graph data In recommender systems, side information often improves the performance of collaborative filtering approaches. We will explore using the document's title and other metadata as side information in the context of citation and subject-label recommendation.

Autoencoders (AEs) find widespread applications in recommender systems [LKH+18; LS17]. AEs further stood the test of a reproducibility study by Dacrema, Cremonesi, and Jannach [DCJ19]. Many approaches aim to incorporate side information into autoencoding recommendation systems [CR18; MJ17]. Still, there is a need to study a principled way to use side information with AEs and tackle the critical problem of making predictions for new users. Dealing with new users is crucial for the document-based recommendation tasks we will consider in this work.

When we move to graph representation learning, we have the means to model context information in a more sophisticated way. A graph comprises a set of vertices (or nodes) and a set of edges, each of which optionally comes with additional attributes. For example, each node resembles a document in a citation graph, and edges indicate the citation relationship. The graph neural network (GNN) model [SGT+09] is a generalization of neural networks to graph input data. The introduction of graph convolution [KW17; HYL17] and graph attention [VCC+18] has spurred new interest in the design of model architectures [XLT+18; XHL+19]. In addition, numerous recent approaches aim at scaling graph neural networks to large graphs [CZS18; WJZ+19; ZZS+20; FLW+21].

Several GNN approaches also focus on heterogeneous graphs [WJS+19; YJK+19; ZZP+19; ZSH+19; FZM+20], which process graphs with nodes of different types. However, a recent study by Lv et al. [LDL+21] has shown that a properly configured graph attention network outperforms GNNs explicitly designed for heterogeneous graphs. Standard approaches are powerful when adequately configured and applied to an appropriate input representation, such as modeling the type as an additional attribute.

Interestingly, we make similar observations in the case of topical text classification. Here, the rise of graph neural networks has triggered renewed interest in transforming pure text into graphs [YML19; LYZ+20; DWL+20; RSI+21]. In Chapter 5, we show that the benefit of creating such synthetic graphs is, in fact, marginal. However, there are numerous other instances where NLP benefits from GNNs, for which we refer to a recent survey by Wu et al. [WCS+21].

In graphs, node attributes can hold textual data. However, there has yet to be a principled way of adapting GNNs to new data in evolving graphs. Most existing work studies graphs with a fixed set of nodes, and only the edges or node attributes change over time [TDW+17; SDV+18; TFB+19; KZL19; MRM20; SWG+20; RCF+20]. Only a few works are concerned with the appearance of new nodes over time [PDC+20; XRK+20].

However, even entirely new classes may emerge. In this thesis, we tackle these problems with a lifelong machine learning approach. In lifelong learning [TM95; FWL16; CL18], a single model is maintained over time and continuously adapted to new tasks. The model can make use of implicit or explicit knowledge from previous tasks. Lifelong learning on graphs is a relatively new topic, investigated by a few other works [FXM+22].

Dealing with evolving data also moves us beyond the core assumption of having independent and identically distributed data between training and testing.

Moving beyond the IID assumption enables a more thorough evaluation of the models [GJM+20]. The temporal evolution of the data closely resembles real-world applications, or, as Bottou [Bot19] has said, "Nature does not shuffle the examples. We shouldn't."

In summary, we have identified the key challenges we must overcome to process text and graph-structured data together.

Efficiency Language models grow larger and larger. Similarly, large amounts of graph-structured data take more work to handle. How can we design efficient methods for learning text representation combinable with graph data

Multimodality There is no isolated text. How can representation learning models effectively process text and graph data together? What is the mutual influence?

Adaptability Real-world data evolve. Although language models excel at being transferable to new tasks, this is not the case for graph representation learning models. How can representation learning models for texts and graphs be continually adapted to new data?

1.2 A Unified Perspective on Representation Learning for Texts and Graphs

The contributions of this thesis are mainly concerned with the challenges described above. The overarching aim is to design efficient text and graph representation learning models adaptable to new data.

Although NLP models excel at transfer learning, they lack scalability. On the other hand, commonly used GNN models struggle to process large graphs, which can only be tackled with expensive sampling techniques. However, real-world data, such as social or citation graphs, are large and evolving. Simply putting together the most prominent models, e. g., a Transformer-based PreLM and a GNN, is not a feasible solution because it does not scale.

In this thesis, we will first explore more *efficient* ways of learning text for the learning of text representations. Then we examine the text classification case in more detail. Subsequently, we investigate the mutual influence of text and graph data in *multimodal* AEs for recommendation systems. Finally, we develop methods that *adapt* a single model over time for lifelong open-world node classification in evolving graphs with text as node attributes.

In more detail, we explore the continual multiplication of words (CMOW) as an efficient alternative to Transformer models that, unlike the CBOW representation, considers word order. Specifically, we introduce an unsupervised training algorithm for the compositional matrix space model [RG10], in which each word is represented as a matrix (a word matrix). The multiplication of word matrices forms a representation of a sentence. Through extensive experimentation with ten linguistic probing tasks, 11 supervised, and five unsupervised downstream tasks, we find that word vectors and word matrices have complementary strengths and that a jointly trained hybrid model, coined CMOW/CBOW-Hybrid, outperforms both individual components CBOW and CMOW. Since dimensionality is a crucial factor for sentence representations (higher is better) [WK19], we ensure that each model has the same number of parameters. We will see that the CMOW/CBOW-Hybrid model increases the average scores on the linguistic probing tasks by 8% and by 1.2% on supervised downstream tasks, compared to a CBOW model, which does not take the order of words into account.

As a next step, we will investigate how a popular Transformer-based language model, BERT [DCL+19], can be distilled into such matrix-based embedding models to capture the learned knowledge of BERT in a smaller and faster model. We call this procedure cross-architecture distillation, and it comes with unique challenges, as standard techniques to promote distillation and model compression rely on the smaller model (the student) having the same architecture as the original larger model (the teacher) [SDC+20; JYS+20; SCG+19]. We extend CMOW/CBOW-Hybrid to emit one representation for each token to facilitate knowledge distillation during pretraining with a masked language modeling objective. We further enhance CMOW/CBOW-Hybrid's expressive power with a bidirectional component by introducing a second set of matrix embeddings for each word, which are then multiplied in the reverse direction. Finally, we extend the model with a specific encoding scheme for tasks defined on sentence pairs inspired by approaches before Transformers emerged, such as in Mou et al. [MML+16]. This extension allows us to tackle natural language inference and sentence similarity tasks. We evaluate the developed models on the GLUE benchmark, which comprises nine datasets that cover natural language inference, sentence similarity, and linguistic acceptability. We implement the transfer learning paradigm commonly used by large language models and fine-tune the CMOW/CBOW-Hybrid model for each downstream task. By thoroughly testing the influence of each of the extensions, we find that bidirectionality leads to a 1% increase, and the proposed two-sentence encoding scheme leads to an increase of 20% compared to a sequential encoding. Our results

further indicate that general distillation is preferable over task-specific distillation. Compared to the literature, we find that the proposed method performs better than other cross-architecture distillation approaches while falling behind the predominant yet more expensive Transformer models.

Subsequently, we take a closer look at the text classification task, which can take different forms, such as topical classification or sentiment analysis. In a comparative study, we critically assess recent advances in text classification based on synthetic graphs generated from the raw text corpus. After a thorough literature review, we contrast these models with more efficient, wide MLP models that operate on a Bag-of-Words (BoW) input representation and more expensive Transformer-based sequence models. We compare 16 different methods on five datasets and find that synthetic graph approaches provide no benefit for text classification over a simple MLP model. Our experiments with BERT and DistilBERT set a new state of the art. We conclude that BoW representations are well suited for text classification when considering their advantage in efficiency.

Then, we move on to recommender systems that operate on documents. Providing recommendations can be regarded as a link prediction task in the bipartite user-item graph. We will be concerned with AEs as a recommendation engine and how they can be enhanced with textual side information. We design an experimental protocol that evaluates only unseen documents. This setting is uncommon in other recommender systems, where the users are typically known to the system. It is a unique challenge in document-based recommender systems, where the recommendations should only depend on the document itself. We conducted a wide range of experiments with datasets that included citation graphs and subject labels from a domain-specific thesaurus. The considered domains cover computer science, life science, economics, politics, and news, on which we impose a chronological train/test split to ensure that the setting resembles real-world conditions. We experimented with six datasets of scholarly documents from different domains. We first acknowledge the strength of a purely graph-based item co-occurrence method in citations and the purely text-based BoW-MLP for subject labels. However, we show that the four tested AE variants could consistently benefit from multimodal input. We confirm these results in different configurations for the number of all possible items, i. e., cited documents or subject labels, and the number of currently available items as a basis for the recommendation. This procedure effectively generates numerous versions of the base datasets and thus ensures the reproducibility and generality of the results.

Finally, we explore a more general approach to learning unified representations of text and graphs. More specifically, we are concerned with node classification in graphs whose nodes correspond to scholarly articles with textual features. The edges in the graph come from citations or collaboration relationships. Although this combination of using text as node features is undoubtedly familiar [KW17], we tackle the challenges of evolving graphs, including the maintenance and adaptation of a single model to new tasks that involve new nodes, edges, and even new classes. We frame this problem as an instance of lifelong learning and develop an incremental training algorithm and a method to derive comparable history sizes across datasets We evaluate representative GNNs as base models and a BoW-MLP as a graph-agnostic baseline in this new experimental framework. Our results on three newly contributed evolving graph datasets, two citation graphs, and one collaboration graph show that reusing the previous model parameters allows us to use smaller history sizes without losing (much) accuracy. Precisely, with history sizes comprising only 50% of the GNN's receptive field, we still attain 95% accuracy compared to the same models trained on the entire history of the graph. We then look closely at the hardest dataset in which most new classes appear over time. There, we explore two more variants of the problem: training with limited labeled examples and self-detection of nodes that do not belong to the already known classes. For the latter, we combine existing techniques with GNNs and find that weighted cross-entropy training is essential for dealing with skewed label distributions.

In summary, the experiments cover a wide range of tasks: general text representation learning, text classification, document-based recommendation, and lifelong node classification, along with numerous different representation learning models such as Transformers, MLPs, AEs, GNNs, and a total of 49 datasets. For all newly developed methods, we have carefully investigated the influence of each component. The key insights are the following. A hybrid model composed of vector representations (aggregated by summation) and matrix representations (aggregated by matrix multiplication) performs better than each alone on a wide range of tasks. PreLM can be effectively distilled into such matrix-vector hybrid models and are competitive to RNN-based approaches. The latter ones are also slower. In topical text classification, vector representations combined with an MLP are surprisingly strong compared to recent approaches that rely on building synthetic graphs from pure text. The same MLP model is also strong for recommending subject labels, while structured data from citation graphs are more important when recommending other documents. When classifying text-attributed nodes in a graph,

structured information from the citation graph is helpful for determining the topic of a publication.

Our experiments are designed in a way that is close to real-world applications: we considered inductive learning, chronological train/test splits, and even operating in fully evolving data. Therefore, the results directly affect practitioners concerned with building text classification systems or recommender systems in which text and graph data appear together. For the research community, we have laid the foundation for lifelong learning on graphs while reminding us of effective and efficient models which outperform more sophisticated, recently developed methods.

1.3 Outline and Contributions

Here, we summarize the five research questions. The thesis is structured along these research questions. Each of these is related to the challenges identified in Section 1.1 and is answered by the corresponding chapter's key findings. This overview also serves as an outline of the body of this thesis.

- Between highly efficient word embeddings that discard word order and resource-intensive language models. *Q1: Can we design efficient text representation learning models that capture word order?*

  In Chapter 3 and [MGS19], we tackle the challenge of learning *efficient* text representations and introduce an unsupervised pretraining scheme for the compositional matrix space model as a lightweight order-sensitive sentence encoding model. We show that a jointly trained hybrid model consisting of matrix representations, aggregated by matrix multiplication, and vector representations, aggregated by summation, is superior to each of the components.

- Following on this hybrid model in combination with knowledge distillation methods that aim to reduce model size, we explore cross-architecture distillation. *Q2: Can we distill the knowledge of large-scale pretrained language models into efficient text representation models and adapt those via fine-tuning for downstream tasks?*

  In Chapter 4 and [GCM+22], we continue to work on the challenge *efficiency* in learning text representation while also considering *adaptability*. We extend the combined matrix-vector representation with a bidirectional component and distill BERT as a large PreLM into it. We show that such an efficient general-purpose distillate can be successfully fine-tuned to downstream tasks, also on those defined on sentence pairs. The proposed model outperforms all previous

work on cross-architecture distillation, is comparable to other contextualized language models, and falls only behind Transformer-based PreLMs.

▷ Investigate topical text classification, where graph-based approaches that synthesize a graph of words and documents have recently emerged. *Q3: Are synthetic graph structures derived from raw text necessary for topical text classification?*

In Chapter 5 and [GS22], we tackle the challenge of *efficiency* for text classification and compare models based on BoW, graph, and sequence representations. We show that, surprisingly, a wide BoW-MLP is competitive to many recent approaches that construct a synthetic graph structure from raw text. We further complement the existing literature by applying PreLMs as sequence models to the topical text classification datasets, which expectedly outperform both graph-based and BoW models. This indicates that, contrary to the recent literature, synthetic text-graphs are less helpful for topical classification than assumed. Furthermore, MLPs are much stronger than expected when using a single but wide hidden layer along with modern best practices in tokenization and optimization.

▷ As a bridge from text to (natural) graph data, consider bipartite graphs within recommendation tasks but enriched with textual side information. *Q4: How can we design multimodal representation learning models that jointly process text and graph data for document-based recommendation tasks?*

In Chapter 6 and [GMV+18; VGM+18; VGS22], we tackle the challenge of *multimodality* and systematically investigate the influence of using side information in document-based recommendation tasks: citation recommendation and subject label recommendation. We show that autoencoder-based recommender systems consistently gain improvements from tapping the resource of textual side information, while we systematically control the item set size and the item vocabulary size. Using strong task-specific baselines, we further find that, again, an MLP that does not use graph data is a powerful model to predict subject labels. We hypothesize that the importance of graph data is related to the meaning of item co-occurrence: it only helps when similar items occur together in a single item set.

▷ Investigate continual training and classification of interlinked text documents organized in an evolving graph. *Q5: How can we adapt representation learning models for text and graph data that evolve over time?* In Chapter 7 and [GVS19; GFZ+21; GVF+21], we tackle the challenge of *adaptability*, while also considering

the *efficiency* challenge. We introduce a new experimental framework with a new evaluation protocol for lifelong learning on evolving graphs. In this framework, a single GNN model is continually adapted to new data, including new classes, while periodically being retrained on limited-history snapshots of the graph. We reveal interesting interactions between the considered history size in the temporal evolution and the reuse of parameters. Reusing parameters enables us to use smaller history sizes without losing accuracy. Furthermore, we investigate a method to automatically detect nodes that belong to, at that point, unknown classes, and find that using a weighted cross-entropy loss is crucial for its stability. This sets the foundation for lifelong learning on evolving graphs.

In Chapter 2, background information can be found that is useful for understanding the body of this thesis. The body then consists of the Chapters 3, 4, 5, 6, and 7, which address research questions 1–5, respectively. We close with a general discussion and concluding remarks in Chapter 8.

A summary of the measures taken to ensure reproducibility is provided in Appendix A. Supplementary material corresponding to Chapters 3 to 7 is provided in Appendices B to F, respectively.

Chapter 2

Background

In this chapter, we recapitulate the fundamental concepts necessary to understand the body of this thesis. The basics of representation learning, in general, are briefly described in Section 2.1. Subsequently, the basics of *text* representation learning are described in Section 2.2, before we cover graph representation learning in Section 2.3. The chapter ends with an overview of common learning paradigms on text- and graph-structured data in Section 2.4.

2.1 Representation Learning

Representation learning [BCV13] is a subfield of machine learning that includes deep learning [KSH17; LBH15; GBC16]. The core idea of machine learning is to learn a function f that maps the input x to the output y. This *model* function f is learned from data and should generalize to unseen inputs. In a normal program, the rules to derive y from x are hand-designed. In contrast, a machine learning algorithm only needs training data and *learns* the "rules" on its own. With the abundance of available data, machine learning has become a vital approach. We do not need to know and implement the rules in advance, but we let the machine learning algorithm figure them out. Classical machine learning is based on hand-designed features that are mapped to the output by the model function f. In representation learning, these features are learned as *representation* of the raw input. When stacking multiple layers of representations, we call that *deep* learning [GBC16].

Recently, Bronstein, Bruna, Cohen, and Veličković [BBC+21] have introduced a theory of symmetry and equivariance that unifies the common architectures of deep learning, including convolutional neural network (CNN), recurrent neural network (RNN), and Transformers. A symmetry is defined a transformation of a data object that does not change the data object. The key properties of symmetries are that the identity transformation is a symmetry, that the chaining of symmetries is a symmetry, and that the inverse of a symmetry is a symmetry. Exploiting symmetries, for instance, weight sharing in CNNs, improves sample complexity,

and thus weakens the curse of dimensionality. Under this notion of symmetries, we regard neural networks as function spaces which we can influence by inductive biases.

An **inductive bias** is a constraint that we place on the learnable function space [BHB+18]. These constraints often emerge from domain knowledge, e. g., CNN impose a constraint for locality-sensitive models, while RNNs impose a constraint to process the input in sequential order. Such inductive biases can take the form of decisions within model architecture design, regularization terms, or data augmentation.

For the context of this thesis, we follow the notion of function spaces and expand the notion of a **representation** to its model function $f_\theta : \mathbb{X} \to \mathbb{Y}; \boldsymbol{x} \mapsto \boldsymbol{y}$, where $\boldsymbol{x}$ is the input and $\boldsymbol{y}$ the output. Learning the representation then means learning the parameters θ of the model function f. Denoting the model function f_θ itself as a representation (rather than its output $\boldsymbol{y}$) can be ambiguous, but is most often not problematic. In particular, when we refer to a representation as efficient, multimodal, or adaptable, we refer to the model function f_θ.

We will now recapitulate the fundamental concepts of representation learning that are essential for the body of the thesis.

2.1.1 Multilayer Perceptron

The multilayer perceptron (MLP) is the most basic form of an artificial neural network, in which the inputs of each layer are fully connected to all outputs. An MLP with L layers can be expressed as the composition of parameterized functions $f = f^{(1)} \circ f^{(2)} \circ \cdots \circ f^{(L)}$, where each layer f_l has the form:

$$f^{(l)}(x) = \sigma\left(W^{(l)}\boldsymbol{x} + \boldsymbol{b}^{(l)}\right)$$

with weights $\boldsymbol{W}^{(l)}$ and bias $\boldsymbol{b}^{(l)}$ being learnable parameters and σ being an activation function. For instance, a two-layer MLP, which we also call an MLP with one *hidden* layer, has the form:

$$f(\boldsymbol{x}) = \boldsymbol{W}^{(2)}\sigma\left(\boldsymbol{W}^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)}\right) + \boldsymbol{b}^{(2)}$$

Among many options, the most commonly used activation function is the rectified linear unit [NH10]:

$$\mathrm{ReLU}(x) = \max(0, x)$$

which clamps the output of a hidden unit at 0. The main benefit of the rectified linear unit (ReLU) activation function is that it does not hinder the gradient flow because of its linear growth on the positive side. In contrast, sigmoidal activation functions saturate and, ultimately, slow down the learning process. Still, the logistic sigmoid activation function is still often used for the final output layer to model a binary classification:

$$\text{Sigmoid}(x) = \frac{1}{1 + \exp(-x)}$$

A fundamental property of neural networks is the universal approximation theorem [Cyb89]. The universal approximation theorem states that already a nonlinear neural network with only two layers (or one *hidden* layer) is capable of approximating any compact function to an arbitrary degree of accuracy, depending on the width of the hidden layer. But why do we bother with deeper neural networks? The key argument is that depth provides exponential gains in terms the number of computation paths that an input feature can take to the output [BCV13]. Counterintuitively, depth improves sample complexity in the sense that fewer training examples are required to learn a task.

2.1.2 Autoencoder

Autoencoders (AEs) are a special form of neural networks that aim to reconstruct their input. $\hat{\boldsymbol{x}} = g(f(\boldsymbol{x}))$ optimized with respect to a loss function $\mathcal{L}(\hat{\boldsymbol{x}}, \boldsymbol{x})$. Here, the encoder f encodes the input into a code, which is then decoded by g. The autoencoder is less of a specific architecture, but more of a general paradigm for unsupervised representation learning: learning *good* representations through reconstruction and preferably disentangling explanatory factors of variation within the data [BCV13].

To prevent AEs from merely copying their input to the output, AEs need to be regularized. Among many options, this regularization can be achieved by using a lower dimensionality for the code (undercomplete AEs), or, a Gaussian prior on the code as in VAEs [KW14], or actively corrupting the input and aiming to reconstruct the original, non-corrupted, input, as in DAEs [VLB+08].

2.1.3 Training of Neural Networks

Here, we briefly revisit the basics of training neural networks: optimization, initialization, and regularization. For a more detailed introduction to neural networks, we refer to Goodfellow, Bengio, and Courville [GBC16].

Optimization To optimize neural networks, we make use of a loss function. The loss function compares the outputs of the model $\hat{\boldsymbol{y}}$ with the ground-truth targets $\boldsymbol{y}$.

$$\mathcal{L}(\hat{\boldsymbol{y}}, \boldsymbol{y})$$

A value of the loss function is commonly referred to as *loss* and can be obtained by applying the model to a set of examples. The most common loss function for multiclass classification is the (categorical) cross-entropy:

$$\mathcal{L}_{\text{CE}}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = -\sum \boldsymbol{y} \log \hat{\boldsymbol{y}}$$

where $\boldsymbol{y}$ is the desired output from the ground truth and $\boldsymbol{y}$ is the model output, in multiclass classification typically activated by a softmax. The softmax activation with C components normalizes the model output (or intermediate representation) as follows:

$$\text{softmax}(\boldsymbol{x})_i = \frac{\exp(x_i)}{\sum_{j=1}^{C} \exp(x_j)}$$

The optimization of neural networks is then carried out by error backpropagation [RHW86], i. e., gradient descent, where the gradient of the loss function $\nabla\mathcal{L}$ is computed with respected to the model parameters and subsequently used to update the model parameters:

$$\theta_t^{(l)} = \theta_{t-1}^{(l)} - \alpha \nabla \mathcal{L}$$

with parameters $\theta^{(l)}$ of layer l and learning rate α.

In stochastic gradient descent, the most common way of training neural networks, we calculate the loss on small batches of examples. This is also known as mini-batch gradient descent, while "stochastic gradient descent" originally referred to using only a single example for each update. In contrast, full-batch gradient descent, as often used with GNNs, uses all available examples for calculating the loss in each update step.

Numerous optimizers have been proposed to optimize neural networks such as momentum techniques [SMD+13]. The most commonly used optimizer is Adam [KB15] because it works well for a wide range of models and is comparatively robust to its hyperparameter values. We will use the Adam optimizer for all experiments presented in this thesis.

Initialization An important factor for optimization is weight initialization. If we initialized all weights with zero, all outputs would remain zero, and the weights would co-adapt during training. Therefore, we need to initialize the weights ran-

domly. A common choice for the initialization is Glorot initialization [GB10], in which the input and output dimensions of the weight matrix determine the bounds of random sampling.

$$r = \sqrt{d_{\text{in}}^{\frac{2}{\text{in}}} d_{\text{out}}}$$

Then, in Glorot uniform, each weight is initialized as a sample from the uniform distribution $W_{ij} \sim \mathcal{U}(-r, r)$. In Glorot normal, each weight is sampled from a normal distribution with standard deviation set to r: $W_{ij} \sim \mathcal{N}(0, r^2)$. Both work similarly well. This aims to ensure that the output vector of each layer has zero mean and unit variance at random initialization.

In common frameworks, e. g., pyTorch[1] and Tensorflow[2], a slightly simplified, yet similar technique is used: He initialization [HZR+15]. In this initialization technique, the boundaries are determined only on the basis of each layer's input dimension: $r = \sqrt{\frac{2}{d_{\text{in}}}}$. When not noted otherwise, we use the He initialization for our experiments.

Regularization The term regularization in neural networks refers to any technique that increases the training loss but decreases the validation loss [GBC16] with the goal of improved generalization performance.

A popular regularization technique is Dropout [SHK+14]. Dropout randomly selects a fraction p of hidden units whose output is then set to zero. At inference time, the magnitude of the weights is scaled up by inverse of the dropout probability $\frac{1}{p}$. With dropout, a single neural network can be regarded as an ensemble of neural networks, which improves their generalization capabilities.

A classic regularization technique is L_2-regularization, a.k.a. weight decay, that introduces a penalty for the magnitude of weights: $\sum_{l \leqslant L} |\boldsymbol{W}^{(l)}|_2$. The effect is that the weights are pushed towards zero. In conjunction with the Adam optimizer, L_2 regularization seem to have fallen out of favor, since it makes little difference whether one uses it or not. This issue might have been caused by a flaw in frameworks' implementations of Adam, which has been identified by Loshchilov and Hutter [LH19], who denote their corrected variant as AdamW.

Layer normalization [BKH16] is a different regularization technique that actively normalizes the inputs to each layer to zero mean and unit variance. Especially in deep neural networks, layer normalization is important to stabilize training, as seen for instance in Transformers [VSP+17].

[1] https://pytorch.org/

[2] https://tensorflow.org/

2.2 Text Representation Learning

The history of using neural networks to learn representations of text ranges back to the 1990s with the Elman RNN [Elm91] and Hochreiter and Schmidhuber's long short-term memory (LSTM) [HS97]. In 2008, Collobert and Weston [CW08] have shown how 1-dimensional CNNs can be employed to tackle multiple NLP tasks at once.

In 2013, Mikolov et al. [MSC+13] have then introduced the word-to-vector (word2vec) algorithm, which learns word representations by predicting missing words in large amounts of text. The idea is motivated by the distributional hypothesis [Har54], which states that the meaning of words is primarily determined by the context of their use. Global vectors for word representation (GloVe) [PSM14] is a similar technique that derives word embeddings by factorizing the global word co-occurrence matrix. As a follow-up to word2vec, Bojanowski, Grave, Joulin, and Mikolov [BGJ+17] have introduced fastText, which incorporates subword information.

In parallel, the generic sequence-to-sequence (seq2seq) paradigm has been developed to tackle tasks like machine translation [SVL14]. In 2015, Bahdanau, Cho, and Bengio [BCB15] introduced an attention mechanism for LSTMs in seq2seq to herald the start of the "LSTM hegemony", whereupon LSTMs with attention has been used across almost all NLP tasks [Gol17]. Notably, Howard and Ruder [HR18] and Peters et al. [PNI+18] have found ways to pretrain LSTMs on unlabeled text before *fine-tuning* the learned representations to tackle downstream tasks.

In 2017, Vaswani et al. [VSP+17] have introduced the Transformer model that only uses attention and feed-forward layers, again for machine translation. Not much later, Devlin, Chang, Lee, and Toutanova [DCL+19]'s BERT has led to tremendous success in NLP by pretraining a bidirectional Transformer model. The BERT model has been trained with a self-supervised masked language modeling objective on large amounts of unlabeled text and is then fine-tuned to various downstream tasks on single sentence and sentence pairs, e. g., sentiment analysis or natural language inference.

The text-to-text transfer Transformer (T5) model by Raffel et al. [RSR+20] follows up on the pretraining and fine-tuning strategy but casts all downstream tasks to have text input and, in contrast to BERT, text output. This means that all downstream tasks are transformed into a format, in which the output is mapped to some tokens of the model vocabulary. For example, a binary classification can be

modeled by testing whether the token "yes" or the token "no" is more probable in the model output.

An even larger language model, Brown et al. [BMR+20]'s 3rd version of the generatively pretrained Transformer (GPT-3), is able to tackle downstream tasks by using only a few examples in the prompt of the input, alleviating the need for fine-tuning at all. The GPT-3 model has been trained with a plain language modeling objective of predicting the next token, as opposed to the masked language modeling objective of BERT. This makes GPT-3 more suitable for text generation.

Recently, Bommasani et al. [BHA+21] have unified these new approaches of representation learning under the hood of "foundation models" and outline various applications in other disciplines, while highlighting benefits, but also risks of the concept of foundation models. The considered risks comprise inequity, misuse, economic and environmental impact, as well as legal and ethical considerations.

2.2.1 Tokenization and the Vocabulary

The first step of processing natural language is to convert the text into a machine-readable format such as a sequence of discrete tokens. For this, we separate the input string into tokens, i. e., perform **tokenization**. Tokens can be words but also subword units, e. g., imagine the phrase "quickly starting" being tokenized into "quick@@" "@@ly" "start@@", "@@ing", Here, the special characters "@@" indicate that a single word has been split up, such that the tokenization is invertible.

The next step is to build a **vocabulary** from all words that appear in a certain corpus. The vocabulary $\mathbb{V}$ is set of all possible tokens. Along with the vocabulary, there is a mapping $\text{vocab} : \mathbb{V} \to \mathbb{N}$, such that we can map each token to a unique index. Depending on the model of choice, it can be useful to reserve some special tokens for the start/end of the sequence or out-of-vocabulary words. Note that tokens can be either words or subword units, to which we still refer as words for reasons of simplicity.

In general, one can distinguish between top-down and bottom-up ways of tokenization and building the vocabulary. In the top-down approach, the text is tokenized with a predefined strategy such as splitting on whitespaces. To determine the final vocabulary, one can, for instance, limit the vocabulary size by using only the most frequent words or by requiring a minimum number of occurrences over the corpus. In the bottom-up approach, e. g., WordPiece tokenization [WSC+16], one specifies a desired vocabulary size and subword units are merged together

with the goal of maximum coverage over the training corpus, until the desired vocabulary size is reached.

2.2.2 One-Hot Encoding

The straight-forward way to encode text in a machine readable format is to assign each word w to a unique vector $\boldsymbol{x}_w$ that is all zero but has a one at the position of the respective word in the vocabulary, i. e., $x_i = 1$, if $i = \text{vocab}[w]$ and 0 otherwise. A sequence of tokens is then encoded as a sequence of these one-hot vectors. While this input representation is suitable for models like RNNs that can process arbitrary-length sequences, other architectures such as an MLP require a fixed-size input representation. A fixed size in a sequence can be enforced by a combination of padding with dummy symbols (e. g., all zero vectors) and truncation of long sequences. A different way to build a fixed-size representation is the Bag-of-Words, which we discuss next.

2.2.3 Bag-of-Words Representation

A common fixed-size input representation for processing text is the Bag-of-Words (BoW), which refers to a multiset of words. We count the words within the input sequence and obtain a vector $\boldsymbol{x}$ with the dimension of the vocabulary size. Each entry x_i corresponds to the number of times that word w with $\text{vocab}[w] = i$ has occurred in the input text. It is worth noting that this representation loses any information on word order and word position. In exchange, the sequence is squashed into a single fixed-size vector, optionally with length normalization by dividing through the L_1 or L_2 vector norm.

TF-IDF Weighting

Term frequency—inverse document frequency (TF-IDF) is a heuristic weighting scheme discounts words that appear frequently in a corpus of documents [SB88]. The inverse document frequency (IDF) is defined per word as:

$$\text{IDF}_{\text{corpus}}(w) = \frac{\text{Number of documents}}{\text{Number of documents in corpus that contain word } w}$$

The term frequency is the length-normalized count of the word (or token) w within a document d

$$\text{TF}(w, \boldsymbol{d}) = \frac{\text{Count of } w \text{ in } \boldsymbol{d}}{\text{Count of all words in } \boldsymbol{d}}$$

Then, the TF-IDF score of a word w in a document $\boldsymbol{d}$ is computed as

$$\text{TF-IDF}(w, \boldsymbol{d}) = \text{TF}(w, \boldsymbol{d}) \cdot \text{IDF}(w)$$

As such, TF-IDF can be regarded as a soft form of stop-word removal. The IDF score of a word is minimal when it appears in every document of the corpus. Conversely, rare words are more distinctive and their weight is increased. TF-IDF is a powerful weighting scheme for information retrieval. Similarly, the weighting scheme can be employed to compose a weighted BoW of a document as input representation for machine learning algorithms [GMS+17].

2.2.4 Word Embeddings

A word embedding is a dense low dimensional vector representation for each word of the vocabulary [Gol17]. Such a word embedding can be obtained by multiplying a BoW vector (or a one-hot vector) with the initial weight matrix of a neural network $\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$. The output dimension of the layer then corresponds to the embedding size, while the input is required to have the same size as the vocabulary. For an intuition, consider a one-hot vector, which would effectively select one row of the weight matrix. When we have a BoW vector, the respective rows of the embedding would be summed up.

A more efficient way to obtain word embeddings is to skip the step of the BoW or one-hot encoding and to directly reserve a lookup table for embedding vectors. In this case, we reserve an embedding matrix $\boldsymbol{X}$ of shape vocabulary size $\times$ embedding size. When we need to encode some text, we can look up the word indices i and obtain the word vector $\boldsymbol{X}[i]$ from the embedding right away, whose parameters can be adjusted during optimization. While it is often left as an implementation detail, the lookup table (or embedding) approach is mathematically equivalent to feeding the BoW representation into a fully-connected feedforward layer.

Yet, the lookup table is more efficient because it skips the matrix multiplication with the dimension of the large vocabulary size. The only difference between a lookup table embedding and a fully-connected input layer lies in commonly used initialization techniques and that embeddings are typically not subject to

an activation function. Dropout is, however, still often used on the embedding. Length normalization, i. e., averaging the word vectors, and TF-IDF weighting can be applied at will during aggregation of word embeddings [GSS17].

In general, the concept of an **embedding layer** refers to the idea of a lookup table to encode discrete tokens with continuous vectors. Apart from words, an embedding layer can also be used for a node embedding in graphs or any other form of categorical input. When we are concerned with text, an embedding layer representation that aggregates over the sequence in a BoW fashion is oftentimes called an embedded BoW or a continuous bag-of-words (CBOW) [Gol17].

An important distinction is between pretrained and non-pretrained word embeddings. On the one hand, in non-pretrained word vectors, the embedding layer is initialized randomly and optimized for the current task at hand. On the other hand, we can use a pretrained word embedding that has been trained on large amounts of unlabeled text. Popular examples include word2vec [MSC+13], GloVe [PSM14], and fastText [BGJ+17], which are known to capture syntactic and semantic properties of the words. Goth [Got16] has constituted such pretrained word embeddings as the most important contribution of unsupervised (or more accurately, self-supervised) machine learning to NLP.

Word2vec learns the parameters of the embedding layer by predicting missing words in a fixed-size context window on unlabeled text. It can be regarded as an autoencoder, where the encoder is the embedding layer and the decoder is a logistic regression. Still, the error is back-propagated to the embedding layer and the decoder is discarded after training. GloVe uses a different approach of factorizing the global word–word co-occurrence matrix. It can be shown that word2vec is a local approximation thereof [LG14], where local means that the algorithm operates only on single sentences and does not require setting up a global co-occurrence matrix. FastText is similar to word2vec in its training objective but goes down to the subword level by jointly considering character n-grams. This allows to better generalize to words that are out-of-vocabulary.

In a sense, pretrained word vectors can be regarded as a precursor to today's large pretrained language models, which will be described next. The main difference is that language models jointly encode entire blocks of text rather than only emitting static vectors for single words.

2.2.5 Transformers and Language Models

Language models have a long history in statistics and machine learning with traditional n-gram based language models that seek to predict the next word based on last seen n words (e. g., [Kat87]).

Similarly, RNN models such as LSTMs [HS97] and gated recurrent units [CMB+14] are sequentially applied to one token at a time, while maintaining a hidden state that is updated depending on each newly seen token.

Ever since its introduction, the realm of language models was conquered by the Transformer architecture [VSP+17], spinning out pretrained language model such as BERT [DCL+19], RoBERTa [LOG+19], T5 [RSR+20], or GPT-3 [BMR+20]. In contrast to RNNs, Transformers have neither recurrent connections nor hidden state that is sequentially updated after seeing each token. This removes sequential dependencies and allows a higher degree of parallelization.

Transformers are sequence models that primarily rely on a self-attention mechanism [VSP+17]. This self-attention mechanism, or more specifically: key–value attention, consists of projecting the same input vector $\boldsymbol{x}$ to three new vectors: a query $\boldsymbol{q} := \boldsymbol{W}_q\boldsymbol{x}$, a key $\boldsymbol{k} := \boldsymbol{W}_k\boldsymbol{x}$, and a value $\boldsymbol{v} = \boldsymbol{W}_v\boldsymbol{x}$. These three vectors then interact to resemble a soft table lookup. Each token is projected to a key–value pair (both vectors), and can query other tokens with its query vector, which is again the output of another projection. The values of the other tokens are aggregated on the basis of how similar the respective key and query vectors are. The key–value self-attention mechanism can be formalized in matrix form as follows [VSP+17]:

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}}\right)\boldsymbol{V}$$

where d_k corresponds to the dimension of the key and query vectors. The scaling factor $\frac{1}{\sqrt{d_k}}$ flattens the softmax and thus counteracts small gradients.

After each key–value self-attention layer, there is a feedforward module, an MLP, and layer normalization to prepare the output for the next self-attention layer. Furthermore, multiple so-called attention heads, i. e., different instances of the linear projections $\boldsymbol{W}_q, \boldsymbol{W}_k, \boldsymbol{W}_v$, operate in parallel such that different aspects of each word can be captured.

Originally, Transformers were developed for machine translation in an encoder decoder architecture. Still, not much later, Devlin, Chang, Lee, and Toutanova [DCL+19] have shown how to pretrain a self-supervised language model by using a masked language model objective. A masked language model objective is similar

to the objective of word embeddings, as described above. The idea is to encode text, while some of the words are masked (replaced with special tokens), and then optimize the model to reconstruct the original words in its output [DCL+19]. Intuitively, the masked language modeling objective can be regarded as filling in the blanks.

An interesting property of Transformers is that word position and word order needs to be explicitly encoded. The self-attention mechanism does not distinguish between the different positions in its input. The most common way to inject this information into the model is to use a **position embeddings**. In position embeddings, each discrete position is associated with an embedding vector, which is then added to the word embedding vector before further processing. There are also other techniques for positional encoding, such as using a sinusoid function or a *relative* positional encoding, e. g., see [Che21].

2.3 Graph Representation Learning

In 2009, Scarselli et al. [SGT+09] have introduced a generic way to process graph-structured data with neural networks: the graph neural network model. Eight years later, the success of graph convolution Kipf and Welling [KW17] and Hamilton, Ying, and Leskovec [HYL17] in node classification along has led to widespread interest in graph neural networks [Ham20]. A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a generic data structure that consists of a set of vertices $\mathcal{V}$ and a set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. The entire graph, the nodes or the edges may have attributes. In this work, we consider node attributes in the form of textual node features and node labels, the latter forming the target outputs of our models. This allows us to use a notation for attributed graphs, that is closer to the neural network processing: We denote a graph with n nodes and m node features as $(\boldsymbol{X}, \boldsymbol{A})$, where $\boldsymbol{X} \in \mathbb{R}^{n \times m}$ are the node features and $\boldsymbol{A} \in \{0, 1\}^{n \times n}$ is a binary adjacency matrix. Node labels are then defined as a vector that holds the label for each node $\boldsymbol{y} \in \mathbb{Y}^n$, where $\mathbb{Y}$ is the set of all possible labels, which we encode as values between 0 and $|\mathbb{Y}| - 1$.

2.3.1 Types of Graphs

Undirected and Directed Graphs The edges in a graph can be **undirected** or **directed**. In the undirected case, we ensure that $(u, v) \in \mathcal{E} \Leftrightarrow (v, u) \in \mathcal{E}$. In case of an undirected graph, the adjacency matrix is symmetric, such that $\boldsymbol{A}^T = \boldsymbol{A}$. A directed graph can be converted into an undirected graph by inserting the corresponding

edges. This comes with a loss of information since the graph cannot be converted back to the same directed graph.

Attributed Graphs In principle, attributes can be added to all components of the graph, i. e., to the nodes, the edges, or even the graph itself. Such attributes might take form of time information, a label, or a text. For instance, also the author or the journal of a paper can be modeled as a node attribute rather than having as its own dedicated node. In this work, we will be using mostly textual node attributes in and a class label that is the desired output of the models. We will also see how journal and author information can be used as attributes in bipartite graphs for recommender systems.

Edge Weights A special case of attributes are edge weights. In the standard binary formulation, either there is an edge between two nodes or not. However, some edges might be more important than others, which can be modeled by giving the edges a specific weight. While this could be modeled explicitly per-edge, also principled approaches can be employed to normalize the edge weights [KW17] or compute them dynamically [VCC+18] on the basis of the representations of the source and destination node.

Homogeneous and Heterogeneous Graphs In heterogeneous graphs, nodes can have different types, e. g., there might be author, paper, venue-typed nodes. Then the edges would also have different a interpretation depending on the types of the invovled nodes, such as "written-by" or "published-in". In homogeneous graphs, all nodes have the same type and all edges have the same meaning. Oftentimes, the same underlying data can be regarded under different perspectives, either focusing only on one specific type, e. g., paper nodes and citation edges, or ignoring that the nodes have different types and treating them all equal. In fact, sometimes the homogeneous view is more effective than one would expect [LDL+21]. This is because, for instance, node types can be modeled as node attributes. This way, also a seemingly homogeneous GNN becomes aware of different node types.

Types of Dynamic Graphs The notion of a dynamic graphs can be ambiguous, if not properly defined. We adopt a principled approach and distinguish between several characteristics (not mutually exclusive): **dynamic signal**, **dynamic structure**, and **continuous time**. In graphs with a dynamic *signal*, the node features and/or

node labels may change over time. When the graph has a dynamic *structure*, there can be new nodes and new edges and also removed nodes and edges. A third characteristic is whether the time information is continuous or discrete. With discrete time, the data are effectively organized into snapshots [AS14]. In continuous time, the data comes in a stream and no assumptions are made about snapshots or granularity. When we speak of evolving graphs, we refer to graphs that have a dynamic structure but a static signal.

Tasks on Graph-structured Data Typical tasks on graph-structured data include node classification, graph classification, edge classification, or link prediction. When we model documents as nodes in a graph, node classification would translate to classifying those documents. Moreover, when we aim to suggest new items to users in a recommendation task, we predict links in the bipartite graph between users and items.

Random Walk Approaches As an interesting connection between text and graphs, we briefly outline the DeepWalk [PAS14] or node2vec [GL16] algorithm. The DeepWalk algorithm aims to learn node representation by predicting left-out nodes along random walks. It essentially transfers the Word2vec algorithm, which operates on text sequences, to graphs by sampling random walks. In fact, implementations of the DeepWalk algorithm even use the exact same code base that is used for the Word2vec algorithm.

2.3.2 Graph Neural Networks

A graph neural network (GNN) can be regarded as a message passing network that transforms and aggregates neighborhood information for each node. A single layer of a GNN can be expressed as:

$$\boldsymbol{h}_i^{(l+1)} = \sigma\left(\boldsymbol{b}^{(l)} + \sum_{j \in i \cup \mathrm{Nbrs}(i)} \frac{1}{c_{ij}} \boldsymbol{W}^{(l)} \boldsymbol{h}_j^{(l)}\right),$$

where $\boldsymbol{h}_i^{(0)} = \boldsymbol{x}_i$ are the features of node i, $\mathrm{Nbrs}(i)$ gives the set of adjacent nodes i, and c_ij is the scalar edge weight. The factor c_{ij} depends on the specific model. For instance, in Kipf's graph convolutional network (GCN) [KW17], the authors use the geometric mean of the two nodes' degree $c_{ij} = \sqrt{\deg_i \cdot \deg_j}$.

When storing $\frac{1}{c_{ij}}$ as entry A_ij in the adjacency matrix, we can express above formula equivalently with matrix operations:

$$\boldsymbol{H}^{(l+1)} = \sigma\left(\hat{\boldsymbol{A}}\boldsymbol{H}^{(l)}\boldsymbol{W}^{(l)} + \boldsymbol{b}^{(l)}\right)$$

where each row $\boldsymbol{H}_{i,:}$ corresponds to the representation of a node i.

In the mean aggregation variant of GraphSAGE [HYL17], the mean over the incoming edges is used, i. e., $c_{ij} = \deg_i$, which simplifies the graph-level normalization of the adjacency matrix and facilitates applying the model to new graph data without having to re-normalize. GraphSAGE further uses two different weight matrices: one for self-connections and one for the connections to neighbors:

$$h_i^{(l+1)} = \sigma\left(\frac{1}{c_{ii}}\boldsymbol{W}^{(\text{self},l)}\boldsymbol{h}_i^{(l)} + \sum_{j\in\text{Nbrs}(i)}\frac{1}{c_{ij}}\boldsymbol{W}^{(\text{neigh},l)}\boldsymbol{h}_j^{(l)}\right)$$

Another variant of GNNs is the graph attention network (GAT) [VCC+18]. In GATs, the edge weights between node i and node j are computed dynamically with an attention mechanism on the basis of the node representations of $\boldsymbol{h}_i^{(l)}$ and $\boldsymbol{h}_j^{(l)}$. Graph neural networks that use dynamic edge weights are called anisotropic GNNs, as opposed to isotropic GNNs that use static edge weights [DJL+20].

In a more general formulation, graph neural networks can be expressed with two operations AGGREGATE, which performs the neighborhood aggregation and UPDATE, which updates each node's own representation:

$$\boldsymbol{H}^{(l+1)} := \text{UPDATE}\left(\boldsymbol{H}^{(l)}, \text{AGGREGATE}\left(\boldsymbol{H}^{(l)}, \hat{\boldsymbol{A}}\right)\right)$$

with $\boldsymbol{H}^{(0)} = \boldsymbol{X}$ corresponding to the node features at the input level.

2.3.3 Graph Approaches to NLP

Using graph approaches for NLP has a long history. Early approaches are KeyGraph [OBY98] and TextRank [MT04]. In KeyGraph, a word co-occurrence graph is segmented to determine the main topics of a document. TextRank [MT04] is an adaption of the PageRank algorithm [BP98] to the words within documents for unsupervised keyword and sentence extraction. Similarly, the keyword extraction approach RAKE [REC+10] leverages a graph structure inferred from text, while splitting the text on predefined stop-words.

Recently, also GNNs have been used for NLP tasks such as syntactic parsing, reasoning, and semantic role labeling. For an overview, we refer to [WCS+21].

In particular, a number of approaches have been developed for graph-based text classification such as TextGCN [YML19] and TensorGCN [LYZ+20].

In TextGCN, a graph is set up on the basis of an collection of documents. The graph holds both words and documents as nodes. The word-document edges are determined by a TF-IDF weighted BoW. The word-word edges correspond to (positive) pointwise mutual information (PMI) scores that have been determined with a sliding window over the text. Each node in the graph then receives its own embedding vector by setting $\boldsymbol{X} = \boldsymbol{I}$. A drawback of this approach is that new documents do not have an embedding yet and the model needs to be retrained before it is able to make predictions. TensorGCN then extends the graph construction technique with cosine similarity of LSTM representations and dependency parsing methods in addition to word co-occurrence.

Follow-up approaches already relax this constraint and are able to deal with new documents being inserted after training. Those include HeteGCN [RSI+21], HyperGAT [DWL+20], and DADGNN [LGG+21], which will be described in more detail in Chapter 5.

2.4 Learning Paradigms

Here, we describe the learning paradigms relevant for this thesis.

In **supervised learning**, the goal is to learn a function f that maps data from the input to domain to the desired output domain $\boldsymbol{x} \mapsto \boldsymbol{y}$. For this, the model is trained on a corpus of paired training data $\mathcal{D} := \{(\boldsymbol{x}, \boldsymbol{y})_i\}, i < N$. After training, the model is applied to previously unseen examples from the test set. A related learning paradigm in graph representation learning is **inductive learning**. In inductive learning, the underlying graph can be swapped out completely because the method only operates on the graph structure and attributes such as node features.

Semi-supervised learning refers to having a set of labeled examples and a set of unlabeled examples available at training time. Then, the task is to make predictions for the unlabeled examples. In graph representation learning, this is referred to as **transductive learning**. In node classification, for instance, the full graph is available for training but only some of the nodes are labeled. The task is to predict labels for the unlabeled nodes.

Self-supervised learning and transfer learning refers to taking an existing model and adapting it to a new task. A popular strategy is to exploit large amounts of unlabeled data for self-supervised pretraining, e. g., predicting missing words

from their context, and then use transfer learning to adapt the model to downstream tasks of interest. From a bird's eye perspective, self-supervised pretraining and subsequent transfer learning might seem similar to the semi-supervised learning case. However, there is a subtle but important difference. In self-supervised pretraining, it is not the ultimate goal to predict labels for the unlabeled data. Instead the unlabeled data is consulted to obtain a general-purpose model, e. g., in language, by predicting intentionally left-out words. Only afterwards, the model is then fine-tuned to tasks of interest.

Lifelong learning [Thr98] is the origin of transfer learning. In transfer learning, the idea is also to transfer an existing model to a new task. The lifelong learning goes one step further: a single model is gradually adapted throughout a *sequence* of tasks, while making use of the knowledge acquired in past tasks.

2.4.1 Definition (Lifelong Learning [CL18])**.** At any time t, the learner has performed a sequence of t learning tasks, $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_t$ and has accumulated the knowledge $\mathcal{K}$ learned in these past tasks. At time $t+1$, it is faced with a new learning task $\mathcal{T}_{t+1}$. The learner is able to make use of past knowledge to help perform the new learning task $\mathcal{T}_{t+1}$.

In Chapter 7, we will use a lifelong learning approach to deal with evolving graph data, i. e., each task $\mathcal{T}_i$ will be a task defined on graphs.

Open-world learning is considered a subproblem of lifelong learning. In a closed-world setting, all possible classes are assumed to be known at training time. In contrast, in open-world learning, new classes may appear after training [CL18]. Hence, the model has to deal with data instances from previously unseen classes *at test time*, which is very challenging. A typical way to deal with this problem is to actively reject the classification of those data instances. But how can the model determine when an instance should be rejected? This can, for instance, be modeled in a supervised fashion by adding a virtual class along with noise data instances to the training, or, in an unsupervised fashion by observing the model's outputs for all classes and reject on the basis of the degree of uncertainty within the outputs.

2.5 Summary

We have outlined the key concepts of text representation learning and graph representation learning. Furthermore, we have described the key learning paradigms that are relevant to this thesis: transfer learning, supervised or inductive learning, and lifelong learning, which will be relevant for the upcoming chapters.

Chapter 3

Word Matrices for Text Representation Learning

In this chapter, our aim is to answer *Q1: Can we design efficient text representation learning models that capture word order?*. The results presented in this chapter are based on material from an article published in International Conference for Learning Representations 2019 [MGS19].

Word embeddings [CW08; MSC+13] have been celebrated as one of the most impactful contributions from unsupervised representation learning to natural language processing [Got16]. After unsupervised learning from a large textual corpus, the word embeddings can be transferred to various downstream tasks. Sentence representations are then aggregated by the sum or mean of the words in the sentence, the so-called continuous bag-of-words (CBOW) [MSC+13]. Since these operations are inherently commutative, any information of word order is lost. For example, the following two sentences would produce exactly the same embedding: "The movie was not awful, it was rather great." and "The movie was not great, it was rather awful." A classifier based on the continuous bag-of-words embedding of these sentences would inevitably fail to distinguish the two different meanings [Gol17, p. 151]. Although the use of n-grams is a common choice to make traditional classifiers sensitive to word order, storing embeddings for all n-gram combinations would require a large amount of memory. Other approaches, such as contextualized word representations [PNI+18], also require more parameters and suffer from slow inference times. We identify the need for efficient, order-aware, word embedding models.

We propose to encode each word as a matrix and to use matrix multiplication as a composition function. Due to the associative property, only $O(\log n)$ sequential steps are sufficient to encode a sentence. Frequent n-grams can be precomputed via dynamic programming. The idea was theoretically explored earlier by Rudolph and Giesbrecht [RG10] as the compositional matrix-space model (CMSM) of language but without any learning algorithm. We show that the CBOW training

objective [MSC+13] can be adapted to obtain an efficient and unsupervised training scheme by making two modifications. On the one hand, we modify the initialization scheme so that the expected value of chained matrix multiplications is constant. On the other hand, we chose a random word as target instead of the center word to alleviate bias.

Our experiments show that matrix embeddings yield an increase in 9 out of 10 linguistic probing tasks compared to vector embeddings. We find that matrix- and vector-based models complement each other well. When training a joint model with both matrix- and vector-based components, the model produces a higher performance on 10 of 16 downstream tasks compared to the vector-based approach trained on the same data with the same capacity. The average improvement across the 16 tasks is 1.2%. These results are insofar promising, as they open up new opportunities to efficiently incorporate order awareness into word embedding models.

After outlining the related work below, we will introduce the CMOW architecture along with a special initialization technique and the unsupervised training scheme for CMSMs in Section 3.3. The datasets used for model evaluation are described in Section 3.4. The experiments and results can be found in Section 3.5, which we discuss in Section 3.6, before we summarize the chapter.

3.1 Related Prior Work

We present an algorithm to learn the weights of the CMSM [RG10]. To the best of our knowledge, only [YC11] and Asaadi and Rudolph [AR17] have addressed this. They present complex, multi-level initialization strategies to achieve reasonable results. Both papers train and evaluate their model on sentiment analysis datasets only, but do not evaluate their CMSM as a general-purpose sentence encoder. There is also a work by Borbély [Bor17], which investigates three strategies for learning matrix embeddings, one of which is similar to ours in the sense that the Word2vec objective is also reused. The experiments of this work only evaluate word2vec-style analogy tasks, and the results were not very promising.

Other works have also represented words as matrices, but, unlike our work, they are not within the framework of the CMSM. Grefenstette and Sadrzadeh [GS11] represent only relational words as matrices. Socher, Huval, Manning, and Ng [SHM+12] and Chung, Wang, and Bowman [CWB18] argue that while CMSMs are arguably more expressive than embeddings located in a vector space, the associative

property of matrix multiplication does not reflect the hierarchical structure of language. Instead, they represent the word sequence as a tree structure. Socher, Huval, Manning, and Ng [SHM+12] represent each word directly as a matrix (and a vector) and build recursive neural networks, which combine pairs of words step by step. Chung, Wang, and Bowman [CWB18] present an approach that maps pretrained word embeddings to their matrix representation and a non-linear function composes the constituents.

Sentence embeddings have recently become an active field of research. A desirable property of embeddings is that the encoded knowledge is useful in a variety of high-level downstream tasks. To this end, Conneau and Kiela [CK18] and Conneau et al. [CKL+18] introduced an evaluation framework for sentence encoders that tests both their performance on downstream tasks and their ability to capture linguistic properties. Most of the works focus on either i) the *ability* of encoders to capture appropriate semantics or on ii) training objectives that give the encoders incentive to capture those semantics. Regarding the former, large RNNs are by far the most popular [CKS+17; KZS+15; TJF+17; NBG19; HCK16; MBX+17; PNI+18; LL18], followed by convolutional neural networks [GPH+17].

A third group is efficient methods that aggregate word embeddings [WBG+16; ALM17; PGJ18]. Most of the methods in the latter group are word-order agnostic. Sent2Vec [PGJ18] is an exception in the sense that they also incorporate bigrams. Despite employing an objective similar to CBOW, their work is very different from ours, as they use addition as their composition function.

Regarding training objectives, there is an ongoing debate whether language modeling [PNI+18; HR18], machine translation [MBX+17], natural language inference [CKS+17], paraphrase identification [WBG+16], or a mix of many tasks [STB+18] incentives the models to learn important aspects of language. In our study, we focus on adapting the well-known objective from word2vec [MSC+13] for the CMSM.

3.2 Problem Formulation

We study the problem of learning an encoder model to produce representations for a sequence of words (or a sentence). After learning the encoder model on unlabeled text (pretraining), the model is transferred to downstream tasks of interest. Optionally, for supervised tasks, another task-specific model is learned on top of the pretrained model to tackle the downstream task. The parameters of the pretrained model remain unchanged, i. e., the sentence encoder is frozen

after pretraining. The procedure for an arbitrary task (supervised, unsupervised, or linguistic probing) can be described by the following procedure.

1. Train the encoder model f_{enc} on unlabeled text.
2. Apply encoder model to the task $\mathcal{T}$.
3. If the task is supervised, learn a task-specific model $g_{\mathcal{T}}$, such that $y = g(f(x))$ for paired training data $\{(x, y)_i\}$.
4. Evaluate the encoder model combined with the task-specific model on the test data $\mathcal{T}$.

Note that we only train the encoder in an unsupervised manner on large amounts of unlabeled text. Only thereafter, steps 2–4 are repeated for each task on the basis of the same pretrained model from step 1.

3.3 Methods

We describe the CBOW and CMOW encoders as well as the CMOW/CBOW-Hybrid model. Subsequently, we discuss the training objective and the initialization strategy.

3.3.1 Continuous Bag-of-Words

With standard word vectors, we reserve a lookup table (an embedding) $\boldsymbol{X}^{(\text{CBOW})} \in \mathbb{R}^{d_{\text{vocab}} \times d_{\text{vec}}}$, where d_{vocab} is the vocabulary size and d_{vec} is the dimensionality of each word vector. Given a sequence of tokens $s_1, s_2, \ldots, s_n$, we denote the summation of the word vectors for the continuous bag-of-words (CBOW) encoding as:

$$\text{CBOW}(s) := \sum_{i=1}^{n} \boldsymbol{X}^{(\text{CBOW})}[s_i]$$

3.3.2 Continual Multiplication of Words

Again, we start with a lookup table that contains a matrix for each word (a word matrix), which we encode as a tensor $\mathbf{X}^{(\text{CMOW})} \in \mathbb{R}^{d_{\text{vocab}} \times d \times d}$, where d_{vocab} is the vocabulary size and d is the dimensionality of the square matrices. We denote a specific word matrix of the embedding by $\mathbf{X}[\cdot]$. Now, given a sequence of tokens

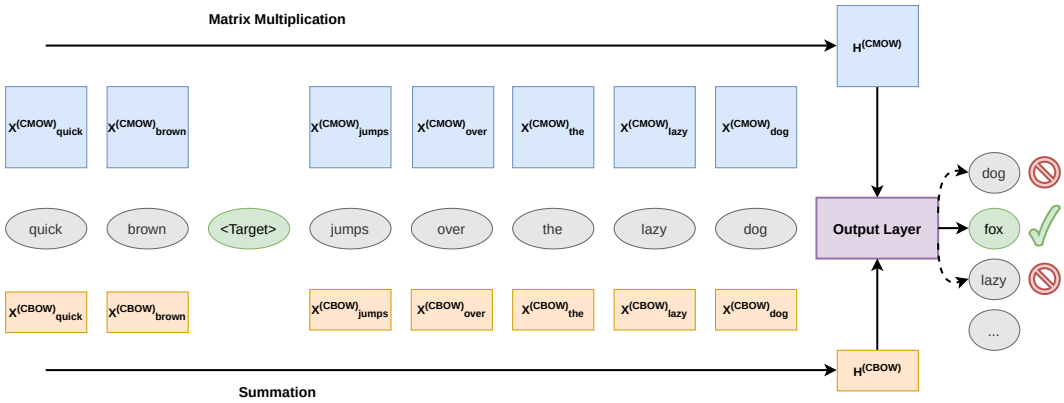


Figure 3.1. The CMOW/CBOW-Hybrid model during pretraining.

$s_1, s_2, \ldots, s_n$. The CMOW text encoder computes:

$$\text{CMOW}(s) := \text{flatten}\left(\mathbf{X}^{(\text{CMOW})}[s_1] \cdot \mathbf{X}^{(\text{CMOW})}[s_2] \cdots \mathbf{X}^{(\text{CMOW})}[s_n]\right)$$

where $\cdot$ denotes matrix multiplication, and flatten flattens the resulting $d \times d$-matrix into a vector of dimension d^2.

The model is well defined for sequences of arbitrary length because the result of the aggregation for any prefix of the sequence is again a square matrix of shape $d \times d$ for both aggregation functions. Thus, it can serve as a general-purpose text encoder.

3.3.3 CMOW/CBOW-Hybrid

The text representation models CBOW and CMOW also extract different linguistic properties of the text. Intuitively, a hybrid model that combines the features of its constituent models also improves the performance on downstream tasks.

The simplest combination is to train CBOW and CMOW separately and concatenate the resulting sentence embeddings at test time. In preliminary experiments, this approach did not work well. We conjecture that there is still considerable overlap in the features learned by each model, which hinders better performance on downstream tasks. To avoid redundancy in the learned features, we combine CBOW and CMOW already during pretraining.

$$\text{CMOW/CBOW-Hybrid}(s) := \text{CMOW}(s) \parallel \text{CBOW}(s)$$

Note that CMOW and CMOW use separate word lookup tables to retrieve the embedding matrices and vectors, respectively.

3.3.4 Training Objective

Motivated by its success, we employ a similar training objective as word2vec [MSC+13]. Intuitively, the model uses logistic regression to predict a left-out word from the concatenation of CBOW and CMOW embeddings. Then, the model parameters are updated according to the backpropagated error signal.

The objective consists of maximizing the conditional probability of a word w_O in a certain context s: $p(w_O \mid s)$. For a word w_t at position t within a sentence, we consider the window of tokens $(w_{t-c}, \ldots, w_{t+c})$ around that word. From that window, we select a target word $w_O := \{w_{t+i}\}, i \in \{-c, \ldots, +c\}$. The remaining $2c$ words of the window form the context s.

We train the model with negative sampling, which is an efficient approximation of the softmax [MSC+13]. For each positive example, we draw k negative examples (noise words) from a noise distribution $P_{\text{noise}}(w)$. The goal is to distinguish the target word w_O from the randomly sampled noise words. Given the encoded input words f(s), a dot product with weights $\boldsymbol{v} \in \mathbb{R}^{d_{\text{vocab}} \times d_{\text{model}}}$ is trained to predicts 1 for context words and 0 for noise words. The negative sampling training objective is:

$$\log \sigma \left(\boldsymbol{v}_{w_O}^T f_{\text{enc}}(s) \right) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_{\text{noise}}(w)} \left[\log \sigma(-\boldsymbol{v}_{w_i}^T f_{\text{enc}}(s)) \right] \tag{3.3.1}$$

where f_{enc} is one of the text representation models described above: $f_{\text{enc}} \in \{\text{CBOW}, \text{CMOW}, \text{CMOW/CBOW-Hybrid}\}$, and d_{model} corresponds to the dimension of the (flattened) sentence representation of the model.

Figure 3.1 depicts the training procedure for the CMOW/CBOW-Hybrid model. For each word, we look up the corresponding word vector and the corresponding word matrix. The word vectors are aggregated by summation. Word matrices are aggregated by multiplying the matrices from left to right. The aggregates of both representations are concatenated, before an output layer (a logistic regression) tries to predict the left-out word.

In the original word2vec [MSC+13], the center word $w_O := w_t$ is used as the target word. However, in our experiments, this objective did not yield satisfactory results. We hypothesized that the model might be biased because it always had an equal number of matrices on the left-hand side and on the right-hand side of the missing word. Instead, we have proposed to select a random output word

$w_O \sim \mathcal{U}(\{w_{t-c}, \ldots, w_{t+c}\})$ from the window [MGS19]. The rationale was that by removing the information about the position of the removed word, the model is forced to build a representation of the entire sequence.

For CMOW, modifying the objective leads to a large improvement on downstream tasks by 20.8% on average, while it does not make a difference for CBOW. We present details in the Appendix (see Appendix B.1).

3.3.5 Initialization

So far, only Yessenalina and Cardie [YC11] and Asaadi and Rudolph [AR17] have proposed algorithms to learn the parameters of the matrices in CMSMs. Both works devote particular attention to the initialization, noting that a standard initialization randomly sampled from $\mathcal{N}(0, 0.1)$ does not work well due to the optimization problem being non-convex. To alleviate this, the authors of both papers propose complex initialization strategies based on a bag-of-words [YC11] or incremental training, starting with two word phrases [AR17]. Instead, we propose an effective yet simple strategy that initializes embedding matrices close to the identity matrix.

We argue that modern optimizers based on stochastic gradient descent have shown to find good solutions to optimization problems even when the problems are non-convex, as in optimizing the weights of deep neural networks. CMOW is essentially a deep linear neural network with flexible layers, where each layer corresponds to a word in the sentence. The output of the final layer is then used as an embedding for the sentence. A subsequent classifier may expect all embeddings to come from the same distribution. We argue that initializing the weights randomly from $\mathcal{N}(0, 0.1)$ or any other distribution that has most of its mass around zero is problematic in such a setting. This includes Glorot initialization [GB10], which was designed to alleviate the problem of vanishing gradients. Figure 3.2 illustrates the problem: With each multiplication, the values in the embedding become smaller (by about one order of magnitude). This leads to the undesirable effect that short sentences have a drastically different representation than larger ones and that the embedding values vanish for long sequences.

To avoid this problem of vanishing values, we propose an initialization strategy, where we initialize each word embedding matrix $\mathbf{X}[w] \in \mathbb{R}^{d \times d}$ as a random deviation from the identity matrix.

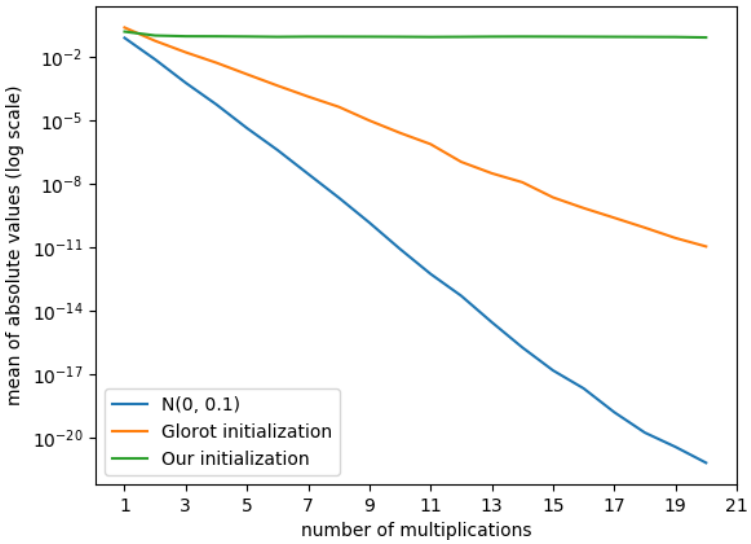


Figure 3.2. Mean of the absolute values of the text embeddings (y-axis) plotted depending on the number of multiplications (x-axis) for the three initialization strategies. The absolute value of the embeddings quickly decreases with more multiplications for standard initialization strategies. When we apply our initialization method, the absolute values of the embeddings have the same magnitude regardless of the sentence length.

$$\mathbf{x}[w] \sim \boldsymbol{I}_d + \begin{pmatrix} \mathcal{N}(0,0.1) & \cdots & \mathcal{N}(0,0.1) \\ \vdots & \ddots & \vdots \\ \mathcal{N}(0,0.1) & \cdots & \mathcal{N}(0,0.1) \end{pmatrix} \tag{3.3.2}$$

The expected value of the multiplication of any number of such word embedding matrices is again the identity matrix [MGS19]. Figure 3.2 shows that this initialization strategy is capable of maintaining a constant magnitude of the values throughout the multiplications. For training CMSMs, we observed an improvement over Glorot initialization of 2.8% on average. We present details in Appendix B.2.

3.4 Datasets

Below, we describe the datasets for pretraining, for the supervised and unsupervised downstream tasks as well as for the linguistic probing tasks. The downstream tasks are close to applications (e. g., sentiment analysis, natural language inference,

sentence similarity), while the linguistic probing tasks test to what extent a model captures certain linguistic properties such as bigram shift or sentence length.

3.4.1 Dataset for Pretraining

For pretraining, we use the unlabeled news corpus from the University of Maryland, Baltimore County (UMBC) [HKF+13], available at `https://ebiquity.umbc.edu/resource/html/id/351`. The UMBC corpus consists of 134 million English sentences with 3 billion tokens in total. Each sentence has 24.8 words on average with a standard deviation of 14.6. The dataset originates from a 2007 web crawl that covered more than 100 million web pages. The uncompressed text data have a size of 48 gigabytes.

3.4.2 Datasets for Linguistic Probing Tasks

All linguistic probing tasks come with $100,000$ training examples and $10,000$ test examples (English) and are evaluated by accuracy. The datasets for these tasks have been assembled by Conneau et al. [CKL+18] and can be downloaded from `https://github.com/facebookresearch/SentEval/tree/main/data/probing`.

Tree Depth prediction (Depth) The task is to predict the depth of the syntactic tree of a sentence. Its ground truth values range between 5 and 12.

Bigram Shift (BShift) In BShift, the task is to predict whether a word bigram has been inverted within the input sentence.

Subject number (SubjNum) The task to predict whether the subject of the main clause of the input sentence is in singular or plural form.

Tense A binary classification task, in which it should be predicted whether the main verb of the sentence is in the present or past tense.

Coordination Inversion (CoordInv) A binary classification task in which it needs to be predicted whether the order of two parts (combined by a conjunction) has been inverted compared to the training data.

Length prediction (SentLen) The task is to predict the length of the input sequence in six possible intervals between 5 and 28.

Object number prediction (ObjNum) Analogous to SubjNum, the ObjNum task is to predict whether the direct object of the input sentence's main clause is in singular or plural form.

Top Constituents prediction (TopConst) The task is to predict the top constituent sequences, e. g., noun-phrase verb-phrase. The number of possible classes is 20 with 19 top-constituent sequences and 1 'other' class.

*Semantic Odd Man Out (*SOMO) A binary classification task in which the task is to determine whether the sentence has occurred as-is in the training data or whether a noun or verb has been replaced.

*Word Content (*WC) is a task to memorize word content, i. e., the model should predict which of 1,000 words are present in a given sentence based on the representation of the aggregated sentence. Note that this task would be trivial with a BoW representation. What is of interest is the extent to which the input words can be reconstructed on the basis of an aggregated low-dimensional representation of a sentence.

3.4.3 Datasets for Supervised and Unsupervised Downstream Tasks

Here, we briefly describe the tasks within the SentEval benchmark[1] by Conneau and Kiela [CK18]. All tasks are evaluated with accuracy, except for Sentences Involving Compositional Knowledge (SICK)-R and the STS variants, which we evaluate with Spearman correlation.

Subjectivity Status (SUBJ) A two-class classification dataset, where the task is to distinguish between subjective reviews and objective plot summaries [WM12] (English). The original dataset had been composed by Pang and Lee [PL04] The data can be found at `https://nlp.stanford.edu/~sidaw/home/projects:nbsvm`.

*Customer Reviews (*CR) CR is a dataset of customer reviews in English [WM12] with the task of distinguishing between positive or negative opinions about products. The dataset has been processed as in Nakagawa, Inui, and Kurohashi [NIK10], while the original dataset is from Hu and Liu [HL04].The data can be found at `https://nlp.stanford.edu/~sidaw/home/projects:nbsvm`.

*Movie Reviews (*MR) The movie reviews dataset consists of 11k training and 11k test examples. The task is to predict the sentiment of sentences within movie reviews. The data can be found under `https://nlp.stanford.edu/~sidaw/home/projects:nbsvm`

[1] `https://github.com/facebookresearch/SentEval`

Opinion Polarity (abbreviated as MPQA) The Opinion-Polarity subtask [WM12] of the Multi-Perspective Question Answering (MPQA) by Wiebe, Wilson, and Cardie [WWC05]. The task is to classify English questions into polarity. The data can be found at `https://nlp.stanford.edu/~sidaw/home/projects:nbsvm`.

*Microsoft Research Paraphrase Corpus (*MRPC) MRPC is a paraphrase detection task consisting of 5,800 pairs of sentences. The task is to determine whether a pair of English sentences is in a relationship of paraphrase (or semantic equivalence) or not. The original data set is from Dolan, Quirk, and Brockett [DQB04]. The data can be found at `https://aclweb.org/aclwiki/Paraphrase_Identification_(State_of_the_art)`

*Text REtrieval Conference (*TREC) Question-type Classification The TREC Question-type classification dataset has been assembled by Li and Roth [LR02]. The task is to classify English questions into question types. The original dataset had 6 coarse and 50 fine-grained classes. The data can be found at `https://cogcomp.seas.upenn.edu/Data/QA/QC/`. The test set contains 500 TREC 10 questions, while the training sets are composed of earlier TREC versions.

*Sentences Involving Compositional Knowledge (*SICK) SICK-E and SICK-R are English natural language inference datasets, in which the task is to determine whether one sentence is implied by another. The task of SICK-E is a three-way classification with classes being 'entailment', 'contradiction', and 'neutral'. The task of SICK-R is to predict a more fine-grained relatedness score in the range of 1 to 5. SICK-E is evaluated with accuracy, whereas the more fine-grained SICK-R is evaluated with Spearman correlation. The data can be found under `https://wiki.cimec.unitn.it/tiki-index.php?page=CLIC`

*Stanford Sentiment Treebank (*SST) Two-way Stanford Sentiment Treebank (SST-2) and five-way Stanford Sentiment Treebank (SST-5) are English sentiment analysis datasets from the Stanford Sentiment Treebank [SPW+13]. SST-2 is a binary sentiment classification task and SST-5 is a fine-grained (five classes) sentiment classification task. The datasets can be found at `https://nlp.stanford.edu/sentiment/index.html`.

*Semantic Textual Similarity Benchmark (*STS-B) STS-B is a collection of the *English* datasets from the annual SemEval challenges between 2012 and 2017. The task is to measure the similarity of two sentences on a scale from 1 to 5. Details can be found in Section 8 of Cer et al. [CDA+17]. We evaluate the

performance on this task as Spearman correlation coefficient (times 100) of the models' output with the similarity scores. The data can be found at `http://ixa2.si.ehu.eus/stswiki/index.php/STSbenchmark`

*Semantic Textual Similarity (*STS) 12–16 STS-12–16 refer to the STS tasks of the annual issues of the SemEval challenge between 2012 and 2016. Again, the task is to measure the similarity of a pair of English sentences. We report the results as Spearman correlation coefficient (times 100) of the models' output with the ground-truth similarity scores. The difference from STS-B is that these tasks are evaluated in an unsupervised manner.

3.5 Experiments

We first describe the procedure for pretraining our models on the UMBC corpus and fine-tuning them on the linguistic probing and downstream tasks.

Following Mikolov et al. [MSC+13], we limit the vocabulary to the 30,000 most frequent words to compare our different methods and their variants. Out-of-vocabulary words are discarded. The optimization is carried out by Adam [KB15] with an initial learning rate of 0.0003 and $k = 20$ negative samples, as suggested by [MSC+13]. For the noise distribution $P_n(w)$, we again follow [MSC+13] and use $\mathcal{U}(w)^{3/4}/Z$, where Z is the partition function to normalize the distribution.

To limit the total batch size and to avoid repeating the expensive tokenization step, we created each batch as follows: 1,024 sentences from the corpus are selected at random. After tokenizing each sentence, we randomly select (without replacement) at maximum 30 words from the sentence to function as center words for a context window of size $c = 5$, i. e., we generate up to 30 training samples per sentence. By padding with copies of the neutral element, we also include words as center words for which there are not enough words in the left or right contexts. For CBOW, the neutral element is the zero vector. For CMOW, the neutral element is the identity matrix.

We use 0.1% of the 134 million sentences for validation. After each 1,000 updates corresponding to 1M training examples, we calculate the validation loss, and training terminates after 10 consecutive validations of no improvement.

We have trained five different models: CMOW and CBOW with $d = 20$ and $d = 28$, which lead to 400-dimensional and 784-dimensional word embeddings, respectively. We also trained the CMOW/CMOW-Hybrid model with $d = 20$ for each component, so that the total model has 800 parameters per word in the

lookup tables. We report the results of two more models: H-CBOW is the 400-dimensional CBOW component trained in Hybrid and H-CMOW is the respective CMOW component. Below, we compare the 800-dimensional Hybrid method to the 784-dimensional CBOW and CMOW models.

After training, we keep only the encoder f of the model. We evaluate the ability to encode linguistic properties by testing on 10 linguistic probing tasks [CKL+18]. In particular, the Word Content task tests the ability to memorize exact words in the sentence. BShift analyzes the encoder's sensitivity to word order. The downstream performance is evaluated on 10 supervised and 6 unsupervised tasks from the SentEval framework [CK18]. We use the standard evaluation configuration that trains a logistic regression classifier on top of the embeddings.

3.5.1 Linguistic Probing Tasks

First, we consider the linguistic probing tasks. The results are shown in Figure 3.3. We find that CBOW and CMOW have complementary strengths and weaknesses. While CBOW gives the highest performance in word content memorization, CMOW outperforms CBOW on all other tasks. Most improvements vary between 1-3 percentage points. The difference is about 8 points for CoordInv and SentLen, and even 21 points for BShift, where the CBOW model remains with random performance as expected.

The hybrid model yields scores close to or even above the better model of the two on all tasks. In terms of relative numbers, the hybrid model improves on CBOW in all probing tasks but WC and SOMO. The relative improvement averaged over all tasks is 8%. Compared to CMOW, the hybrid model shows rather small differences. The largest loss is 4% on the CoordInv task. However, due to the large gain in Word Content (20.9%), the overall average gain is still 1.6%.

We now compare the jointly trained H-CMOW and H-CBOW with their separately trained 400-dimensional counterparts. We observe that CMOW loses most of its ability to memorize word content, while CBOW shows a slight gain. However, H-CMOW shows, among others, improvements on BShift.

3.5.2 Supervised Downstream Tasks

Figure 3.4 shows the scores obtained in the supervised downstream tasks. When comparing the 784-dimensional models CBOW and CMOW seem to complement each other. This time, CBOW has the upperhand, matching or outperforming

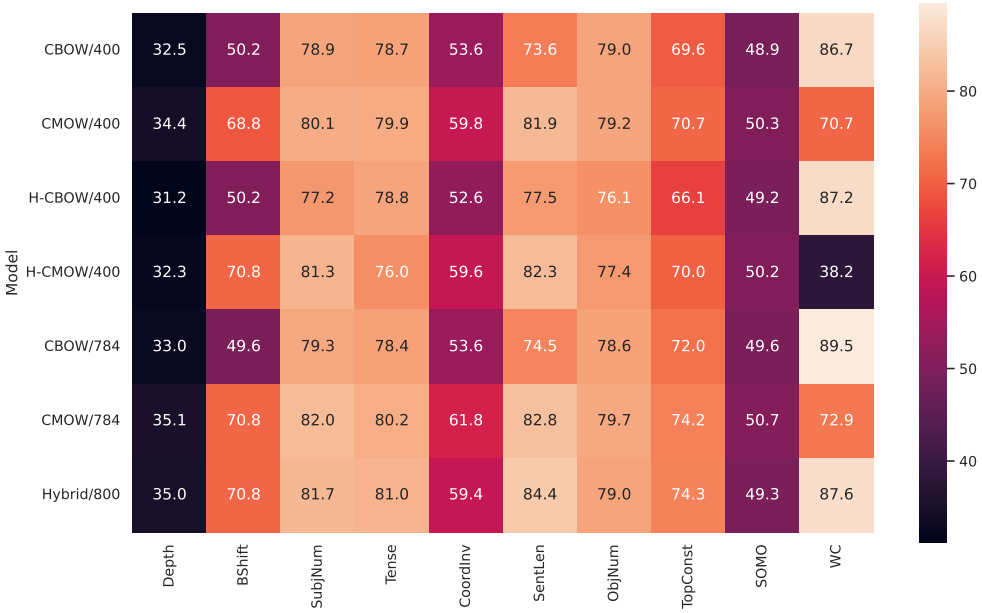


Figure 3.3. Results on linguistic probing tasks.

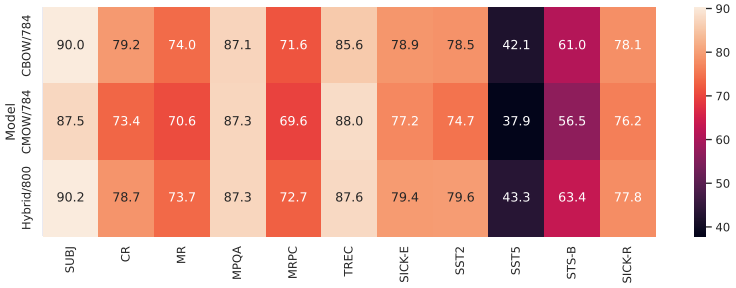


Figure 3.4. Results on supervised downstream tasks

CMOW on all supervised downstream tasks except TREC by up to 4 points. On the TREC task, CMOW outperforms CBOW by 2.5 points. Our jointly trained model is not more than 0.8 points below the better one of CBOW and CMOW on any of the considered supervised downstream tasks. On 7 of 11 supervised tasks, the joint model even improves on the better model, and on two-way Stanford Sentiment Treebank, SST-5, and MRPC the difference is more than 1 point. The average relative improvement over all tasks is 1.2%.

3.5.3 Unsupervised Downstream Tasks

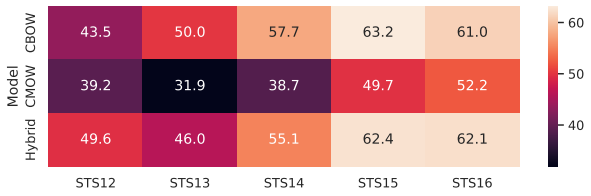


Figure 3.5. Results on the unsupervised downstream tasks

On the unsupervised downstream tasks, CBOW attains higher scores than CMOW on all datasets by wide margins. Figure 3.5 shows the results. For example, on STS-13, CBOW's score is 50% higher. The hybrid model is capable of repairing this deficit, reducing the difference to 8%. It even outperforms CBOW on two of the tasks and yields a slight improvement of 0.5% on average on all unsupervised downstream tasks. However, the variance in relative performance is notably larger than on the supervised downstream tasks.

3.6 Discussion

Key Results Our CMOW model produces sentence embeddings that are at the level of fastSent [HCK16]. Thus, CMOW is a reasonable choice as a sentence encoder. Essential to the success of our training scheme for the CMOW model are two changes to the original word2vec training. First, our initialization strategy improved downstream performance by 2.8% compared to Glorot initialization. Secondly, by choosing the target word of the objective at random, the performance of CMOW on downstream tasks improved by 20.8% on average. Hence, our novel

training scheme is the first that provides an effective way to obtain parameters for the Compositional Matrix Space Model from unlabeled, large-scale datasets.

On linguistic probing tasks, we observe that CMOW embeddings better encode the linguistic properties of sentences than CBOW. CMOW gets reasonably close to CBOW on some downstream tasks. However, in general, CMOW does not supersede CBOW embeddings. This is because CBOW is stronger in word content memorization, which is positively correlated with performance on most downstream tasks [CKL+18]. However, CMOW has increased performance on the TREC question type classification task (88.0 compared to 85.6). The rationale is that this particular TREC task belongs to a class of downstream tasks that require capturing other linguistic properties apart from Word Content [CKL+18].

Due to joint training, our hybrid model learns to pick up the best features of CBOW and CMOW simultaneously. It enables both models to focus on their respective strengths. This is best observed as H-CMOW loses its ability to memorize word content. In return, H-CMOW has more capacity to learn other properties, as seen in the increase in performance at BShift and others. We observe a complementary behavior for H-CBOW, which has increased scores on WC. With an 8% improvement on average, the hybrid model is more linguistically informed than CBOW. This transfers to a total performance improvement by 1.2% on average over 11 supervised downstream tasks, with large improvements on sentiment analysis tasks (SST-2, SST-5), question classification (TREC), and semantic text similarity (STS-B). These tasks arguably depend on word order information. On the other tasks, the differences are small. Again, most tasks in the SentEval benchmark depend mainly on word content memorization [CKL+18], where the hybrid model does not improve on CBOW.

Limitations Please note, these models do not represent the state-of-the-art for sentence embeddings. Perone, Silveira, and Paula [PSP18] show that LSTMs and Transformer models, but also by averaging the word embeddings from fastText [BGJ+17], achieve better scores. These embeddings were trained on the CBOW objective and are thus very similar to our models. However, they are trained on large corpora (600B tokens vs 3B in our study), use large vocabularies (2M vs 30k in our study), and incorporate numerous tricks to further enhance the quality of their models: word subsampling, subword information, phrase representation, n-gram representations, position-dependent weighting, and corpus deduplication. In the present study, we focus on comparing CBOW, CMOW, and the hybrid model in a scenario where we have complete control over the independent variables. Our analysis yields

interesting insights into what our models learn when trained separately or jointly, which we consider to be more valuable in the long term for the research field of text representation learning.

Inference Speed We offer an efficient order-aware extension to embedding algorithms from the bag-of-words family. Our 784-dimensional CMOW embeddings can be computed at the same rate as CBOW embeddings. We measured an encoding speed of 71k for CMOW vs. 61k for CBOW sentences per second. This is due to the fast implementation of matrix multiplication in graphics processing units (GPUs). It allows us to encode sentences about 5 times faster than using a classical Elman RNN [Elm91] of the same size (12k per second). Our matrix embedding approach also offers valuable theoretical advantages over RNNs and other autoregressive models. Matrix multiplication is associative, so that merely $\log_2 n$ sequential steps are necessary to encode a sequence of size n. In addition to parallelization, we could also employ dynamic programming techniques to further reduce the number of matrix multiplication steps, e. g., by precomputing frequent bigrams. Thus, we expect our matrix embedding approach to be specifically well suited for large-scale, time-sensitive text encoding applications. Our hybrid model serves as a template for using CMOW in conjunction with other existing embedding techniques, such as fastText [BGJ+17].

3.7 Summary

We have presented the first efficient unsupervised learning scheme for compositional matrix-space models, an efficient embedding-based model that captures word order. We have shown that the resulting sentence embeddings capture linguistic features that are complementary to the embeddings of CBOW. We then presented a hybrid model with CBOW that is able to combine the complementary strengths of both models for improved performance on downstream tasks, in particular, on tasks that depend on word order information. Thus, our model narrows the gap of representational power between simple word embeddings and highly nonlinear recurrent sentence encoders. To answer the research question *Q1: Can we design efficient text representation learning models that capture word order?*, we have shown that it is possible to learn efficient *and* order-aware representations of text with pretraining on large amounts of unlabeled text.

Chapter 4

Cross-Architecture Distillation with Word Matrices

In this chapter, we will continue to work with word matrices for general text representation learning. In particular, we will address the research question *Q2: Can we distill the knowledge of large-scale pretrained language models into efficient text representation models and adapt those via fine-tuning for downstream tasks?*. This chapter is based on material published as a conference paper [GCM+22] in the International Joint Conference on Neural Networks 2022.

Large pretrained language models (PreLMs) [DCL+19; RSR+20] have emerged as de-facto standard methods for natural language processing [WSM+19; WPN+19]. The common strategy is to pretrain models on enormous amounts of unlabeled text before fine-tuning them for downstream tasks. However, the drawback of PreLMs is that the models are getting larger and larger, with up to several billions of parameters [BMR+20]. This comes with high environmental and economic costs [SGM19] and shifts the development and research into the hands of a few global players [BHA+21, pp. 10-12].

Although a single pretrained model can be reused for multiple downstream tasks, the sheer size of the model is often prohibitive. The immense resource requirements prevent the use of these models in small-scale laboratories and mobile devices, which is also tied to privacy concerns [SWR20].

There is a need for more efficient models or compressed versions of large models to make AI research more inclusive and energy-friendly, while fostering deployment in applications. Reducing the size of PreLMs using knowledge distillation [HVD15] or model compression [BCN06] is an active area of research [SDC+20; JYS+20; SYS+20]. Both knowledge distillation and model compression can be described as teacher-student setups. The student is trained to mimic the predictions of the teacher while using less resources. Typically, a large PreLM takes the role of the teacher, while the student is a smaller version of the same architecture. Sharing the same architecture between the student and the teacher enables the use of

dedicated distillation techniques, e. g., aligning the representations of intermediate layers [SDC+20; SYS+20].

However, using more efficient architectures for the student has already shown promising results, such as the task-specific distillation approaches of Tang et al. [TLL+19] and Wasserblat, Pereg, and Izsak [WPI20]. In their work, student models are LSTMs [HS97] or models based on a CBOW representation.

On the one hand, LSTMs are difficult to parallelize as they need at least $\mathcal{O}(n)$ sequential steps to encode a sequence of length n. On the other hand, CBOW-based models are *not sensitive to word order*, i. e., cannot distinguish sentences with the same words but in different order ("cat eats mouse" vs. "mouse eats cat" are treated as equivalent).

There are, however, more efficient models such as CMOW from the previous Chapter 3 that *do capture word order* by representing each token as a matrix, instead of a vector. We recall that a sequence in CMOW is modeled by non-commutative matrix multiplication, which makes the encoding of a sequence dependent on the word order.

The present work investigates how order-sensitive matrix embeddings can be used as student models in cross-architecture distillation from large PreLM teachers. Thus, we complement the existing body of work that focuses predominantly on same-architecture distillation. All previous cross-architecture approaches are task-specific, whereas we also explore general distillation. We aim to understand to what extent order-sensitive embeddings are suited to capture the teacher signal of a large PreLM such as BERT. To this end, we extend the CMOW/CBOW-Hybrid model from Chapter 3, which unifies the strengths of CBOW and CMOW, with a bidirectional representation of the sequences. Furthermore, we add the ability to emit per-token representations to facilitate the use of a modern masked language model objective [DCL+19]. We investigate both *general distillation*, i. e., the distillation is applied during pretraining on unlabeled text, and *task-specific distillation*, when an already fine-tuned PreLM is distilled per task. We also introduce a two-sentence encoding scheme to CMOW such that it can deal with similarity and natural language inferencing tasks. This improves performance by 20% compared to a naive joint encoding.

Our results show that large PreLMs can be distilled into efficient order-sensitive embedding models and achieve performance comparable to the approach known as Embeddings from Language Models (ELMo) [PNI+18] on the GLUE benchmark. On certain tasks, embedding-based models even challenge other size-reduced BERT

models such as DistilBERT. In summary, the contributions of this chapter are as follows:

- ▻ We extend order-sensitive embedding models with bidirectionality and make them amenable for masked language model pretraining.
- ▻ We explore the use of order-sensitive embedding models as student models in a cross-architecture distillation setup with BERT as a teacher and compare general and task-specific distillation.
- ▻ We introduce the first encoding scheme that enables CMOW/CBOW-Hybrid to deal with two-sentence tasks (20% increase over the naive approach).
- ▻ Our results show that the best distilled embedding models can be on par with more expensive models such as ELMo or DistilBERT on certain tasks of the GLUE benchmark, while having a higher encoding speed (thrice as high as DistilBERT).

In the following, we describe the related work on distillation techniques before we formulate the problem and introduce preliminaries in Section 4.2. Our embedding models, our cross-architecture distillation setup, and our two-sequence encoding scheme are described in Section 4.3. The experiments are presented in Section 4.5, whose results are discussed in Section 4.6, before we summarize the chapter.

4.1 Related Prior Work

We discuss related work on general distillation, task-specific distillation, cross-architecture distillation, and other techniques to reduce the size of large models.

General Distillation In general distillation, a PreLM is distilled into a student model during pretraining. DistilBERT [SDC+20] is such a general-purpose language model that has been distilled from BERT. Apart from masked language modeling and distillation objectives, the authors also introduced a cosine loss term to align the student and teacher's hidden states (layer transfer). Furthermore, the student is initialized with selected layers of the teacher. MobileBERT [SYS+20] introduced a bottleneck to BERT such that layers can be transferred to student models with smaller dimensions. The structure of MobileBERT changes BERT's structure by using an inverted bottleneck before and a bottleneck after the encoder blocks, making

the model deeper and reducing its width. This makes MobileBERT amenable for layer transfer to the smaller student [SYS+20].

Task-specific Distillation In task-specific distillation, the teacher signal is used during fine-tuning. Approaches that use a teacher signal during pretraining *and* fine-tuning also fall into this caterogy.Sun, Cheng, Gan, and Liu [SCG+19] use layer-wise distillation objectives and initialize with teacher weights to train BERT students with fewer layers. They explore which layers of the teacher provide the most information e. g., using upper layers or learning from every kth layer. Jiao et al. [JYS+20] applies knowledge distillation in both stages, pretraining and fine-tuning. The authors also augment the training data [JYS+20], which we have also considered but found no consistent improvement. Turc, Chang, Lee, and Toutanova [TCL+19] analyze the interaction between pretraining and fine-tuning when distilling BERT and report a benefit of applying distillation already during pretraining.

Mao et al. [MWW+20] propose another size-reduced BERT model that combines knowledge distillation with pruning and matrix factorization. Other approaches consider distillation in multilingual [TRJ+19] or multi-task settings [YSG+19].

Cross-architecture Distillation The works described above assume that the teacher and the student share the same architecture. This allows us to use techniques such as layer transfer and loss terms to align hidden states. However, the student model does not need to have the same architecture as the teacher, which we then call cross-architecture distillation. Wasserblat, Pereg, and Izsak [WPI20] use a simple feed forward network with CBOW embeddings and a bidirectional LSTM model as students. Both models perform well in several downstream tasks. Tang et al. [TLL+19] explore distilling BERT into a single-layer BiLSTM without using additional training data or modifications to the teacher architecture. Their distillation-based approach yields improvements compared to a plain BiLSTM without teacher signal: about 4 points on all reported tasks (Quora Question Pairs (QQP), Multi-Genre Natural Language Inference corpus (MNLI), and SST-2). This has motivated us to investigate whether even simpler models can be used as students of a BERT teacher.

Pruning and Quantization Other techniques to reduce the size of a model are pruning and quantization. Pruning approaches such as Sanh, Wolf, and Rush [SWR20] reduce the number of parameters. However, the smaller resulting models

use the same architecture as their larger counterparts and, thus, pruning does not necessarily improve inference speed until dedicated hardware for sparse networks becomes available [SWR20]. Quantization is a common post-processing step to reduce the size of the model by decreasing the floating point precision of the weights [GAG+15; WJZ+20]. Pruning and quantization can be applied in conjunction with knowledge distillation [SWR20; SYS+20]. In addition to techniques to reduce the size of the model, there is also a tremendous effort to improve the efficiency of the Transformer architecture, for which we refer to the recent survey of Tay, Dehghani, Bahri, and Metzler [TDB+20] on efficient Transformers. For now, we focus on distillation, while pruning and quantization might even further increase the efficiency of our proposed method.

Summary The existing literature focuses mainly on reducing the size of PreLMs by distillation, pruning, and quantization. Specialized techniques, such as layer transfer, depend on teacher and student sharing the same architecture and, thus, are not applicable for cross-architecture distillation. So far, only a few recent works consider distilling PreLMs into other architectures such as LSTMs and bag-of-words feed-forward nets. In this work, we show that cross-architecture distillation with order-sensitive embedding models as students outperform previous cross-architecture distillation approaches and achieve scores comparable to ELMo, while using less computational resources. Finally, all previous cross-architecture distillation approaches are task-specific, while we could show that general distillation can lead to scores even higher than those for task-specific distillation.

4.2 Problem Formulation

We study the problems of knowledge distillation in a transfer learning setting. In the following, we first distinguish between the pretraining and fine-tuning stages of knowledge distillation. Then, we describe how knowledge distillation can be applied in these two stages.

Transfer Learning: Pretraining and Fine-tuning In transfer learning, we distinguish between the pretraining and the fine-tuning stage. In the pretraining stage, a representation learning model is trained on large amounts of unlabeled text. The pretrained model is then transferred to downstream tasks. The parameters of the

transferred model are fine-tuned to these task. For each task, we start with a fresh copy of the pretrained model.

Pretraining Train model f_θ on unlabeled text.

Fine-tuning For each downstream task $\mathcal{T}$, continue training of the same model f_θ that has been obtained by pretraining.

The task $\mathcal{T}$ can be an arbitrary supervised downstream task, where paired training data are available. In particular, this includes two-sequence tasks such as natural language inference or textual similarity. The performance measure depends on the respective task. We will use the tasks of the general language understanding evaluation benchmark (GLUE) benchmark to evaluate our models. The difference from the sentence representation learning tasks considered in Chapter 3 is that the encoding part of the model can also be adapted during fine-tuning.

Knowledge Distillation The problem of knowledge distillation [HVD15] or model compression [BCN06] refers to learning a smaller model g that imitates the behavior of a larger model f, such that desirably $g(\boldsymbol{x}) \approx f(\boldsymbol{x})$. We call the smaller model g the student, and the larger model f the teacher. The distillation is carried out by aligning the teacher and student outputs, e. g., through a loss term $\mathcal{L}(f(\boldsymbol{x}), g(\boldsymbol{x}))$, which we call the **teacher signal**. There are more techniques to foster distillation, such as using the teacher's weights as initialization for the student (see Section 4.1), but those are only applicable when teacher and student are of the same model architecture.

To contextualize knowledge distillation with transfer learning, we adopt the distinction between *general* distillation and *task-specific* distillation from Tang et al. [TLL+19].

General Distillation The distillation is carried out *only* in the pretraining stage. After pretraining, the teacher is no longer needed. The student can be fine-tuned to downstream tasks independently.

Task-specific distillation In task-specific distillation, the teacher model can be consulted during fine-tuning for each downstream task.

Note that the task-specific distillation model can still be initialized with a model obtained by general distillation. We consider this procedure also as task-specific distillation, because the teacher has to be consulted for fine-tuning.

The distinction between general and task-specific distillation is important because this affects how the methods can be applied in practice. Imagine that you want to tackle a downstream task on a mobile device. A general distillation model would be able to learn new tasks on its own. With a task-specific distillation approach, the larger teacher model would need to be consulted on the mobile device. Thus, both approaches differ in how they can be applied in practice and should be considered as separate problems.

4.3 Methods

First, we introduce our bidirectional extension to the CMOW/CBOW-Hybrid model. Subsequently, we introduce our approach for cross-architecture distillation that we use during the pretraining and fine-tuning stages. Finally, we introduce a two-sentence encoding scheme for order-sensitive embedding models that is crucial for fine-tuning on downstream tasks with paired sentences.

4.3.1 Extending Matrix Embedding Models

We extend the CMOW/CBOW-Hybrid embeddings of the previous chapter with a bidirectional component and the ability to emit per-token representations as a foundation for cross-architecture distillation. In the following, we briefly recall the CMOW/CBOW-Hybrid embedding model and then describe the extensions that enable distillation, increase expressive power, and improve the processing of sentence pairs.

Recalling CMOW/CBOW-Hybrid CMOW/CBOW-Hybrid embeddings are a combination of matrix embeddings and vector embeddings. Compared to vector-only embeddings, the word order can be captured because matrix multiplication is non-commutative. Given a sequence s of n tokens with each token s_j having its corresponding matrix-space embedding $\boldsymbol{X}_j \in \mathbb{R}^{d \times d}$ and vector-space embedding $\boldsymbol{x}_j \in \mathbb{R}^{d_{\text{vec}}}$, the CMOW/CBOW-Hybrid embedding is the multiplication of embedding matrices $\boldsymbol{X}_i$ concatenated (symbol $\cdot||\cdot$) to the sum of embedding vectors $\boldsymbol{x}_i$:

$$\boldsymbol{H}^{(\text{CMOW})} := \boldsymbol{X}_1^{(\text{CMOW})} \cdot \boldsymbol{X}_2^{(\text{CMOW})} \cdots \boldsymbol{X}_n^{(\text{CMOW})}$$
$$\boldsymbol{h}^{(\text{CBOW})} := \sum_{1 \leqslant j \leqslant n} \boldsymbol{x}_j^{(\text{CBOW})}$$

$$\boldsymbol{h}^{(\text{Hybrid})} := \text{flatten}\left(\boldsymbol{H}^{(\text{CMOW})}\right) \parallel \boldsymbol{h}^{(\text{CBOW})}$$

where flatten collapses the matrix into a vector. As in the previous chapter, we initialize each embedding matrix $\boldsymbol{X}_j$ as the identity plus Gaussian noise $\boldsymbol{I}_d + \mathcal{N}(0, \sigma^2_{\text{init}})$. We use $\sigma_{\text{init}} = 0.01$.

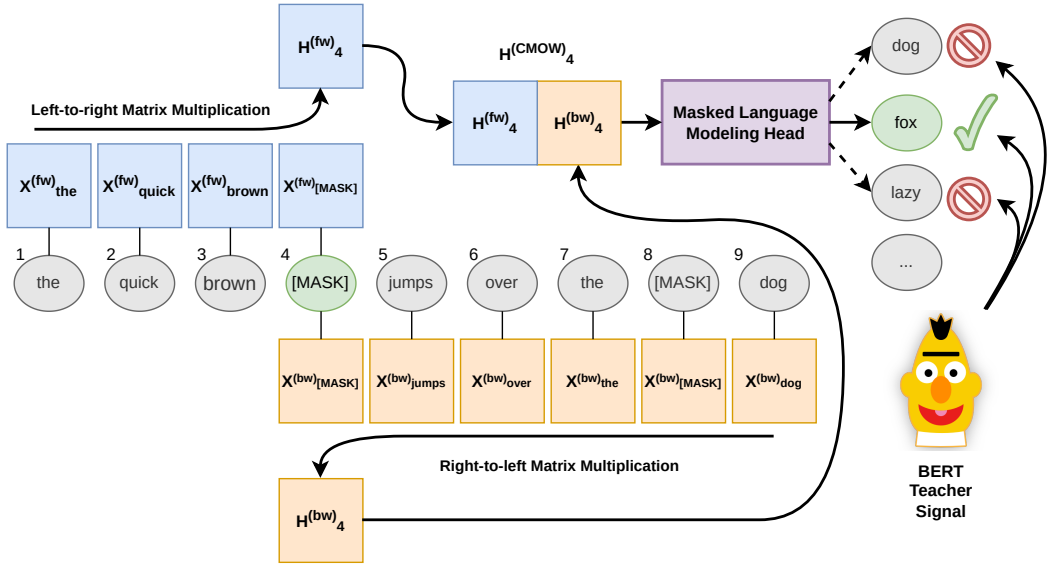


Figure 4.1. The bidirectional CMOW component of our proposed architecture during pretraining. In this example, the model predicts the masked token at position 4 by concatenating forward and backward matrix embeddings, which are then fed into a masked language modeling head. Bert illustration: Melanie Poech

Bidirectional CMOW/CBOW-Hybrid Inspired by the success of bidirection in RNNs [SP97], LSTMs [PNI+18], and Transformers [DCL+19], we extend CMOW by a bidirectional component. Hence, we introduce a second set of matrix-space embeddings that are multiplied in reverse order. We then have two sets of matrix embeddings for each word: one for the forward direction $\mathbf{X}^{(\text{fw})} \in \mathbb{R}^{n_{\text{vocab}} \times d \times d}$ and one for the backward direction $\mathbf{X}^{(\text{bw})} \in \mathbb{R}^{n_{\text{vocab}} \times d \times d}$. Then, we concatenate forward and backward directions. Figure 4.1 illustrates the bidirectional CMOW component.

Furthermore, we emit one representation per token position i, which allows training with a masked language model objective [DCL+19]. Thus, we are able to make use of the BERT teacher signal for pretraining. Since we can reuse computations, $\mathcal{O}(n)$ matrix multiplications are sufficient to encode a sequence of length n.

For these intermediate representations, we also modify the CBOW component in a way that it yields partial sums for the forward and backward directions.

To ease the notation, we use $\boldsymbol{X}_i$ as a shorthand for the embedding matrix of the token s_i at position i, i. e., $\boldsymbol{X}_i = \mathbf{x}[s_i]$. In this notation, CMOW/CBOW-Hybrid from the previous chapter can be expressed as:

$$\begin{aligned}
\boldsymbol{h}_i^{(\text{Bidi. CMOW})} &:= \text{flatten}\left(\boldsymbol{X}_1^{(\text{fw})} \cdot \boldsymbol{X}_2^{(\text{fw})} \cdots \boldsymbol{X}_i^{(\text{fw})}\right) \\
&\quad \parallel \text{flatten}\left(\boldsymbol{X}_n^{(\text{bw})} \cdot \boldsymbol{X}_{n-1}^{(\text{bw})} \cdots \boldsymbol{X}_i^{(\text{bw})}\right) \\
\boldsymbol{h}_i^{(\text{Bidi. CBOW})} &:= \sum_{j=1}^{i} \boldsymbol{x}_j^{(\text{CBOW})} \parallel \sum_{j=i}^{n} \boldsymbol{x}_j^{(\text{CBOW})} \\
\boldsymbol{h}_i^{(\text{Bidi. Hybrid})} &:= \boldsymbol{H}_i^{(\text{Bidi. CMOW})} \parallel \boldsymbol{h}_i^{(\text{Bidi. CBOW})}
\end{aligned}$$

For fine-tuning on tasks with full sentences as input, e. g., natural language inference, we do not need per-token representations. In this case, we compute the representation of a token sequence as:

$$\begin{aligned}
\boldsymbol{h}^{(\text{Bidi. CMOW})} &:= \text{flatten}\left(\boldsymbol{X}_1^{(\text{fw})} \cdot \boldsymbol{X}_2^{(\text{fw})} \cdots \boldsymbol{X}_n^{(\text{fw})}\right) \\
&\quad \parallel \text{flatten}(\boldsymbol{X}_n^{(\text{bw})} \cdot \boldsymbol{X}_{n-1}^{(\text{bw})} \cdots \boldsymbol{X}_1^{(\text{bw})}) \\
\boldsymbol{h}^{(\text{CBOW})} &:= \sum_{j=1}^{n} \boldsymbol{x}_j^{(\text{CBOW})} \\
\boldsymbol{h}^{(\text{Bidi. Hybrid})} &:= \boldsymbol{h}^{(\text{Bidi. CMOW})} \parallel \boldsymbol{h}^{(\text{CBOW})}
\end{aligned}$$

Note that the forward and backward directions of the embedding vectors $\boldsymbol{h}^{(\text{CBOW})}$ conflate to equivalent formulas when we encode entire sequences. Thus, we only need to include a single CBOW representation along with the two CMOW components that yield different results for the forward and backward direction. At inference time, the model can be parallelized along the sequential dimension.

For regularization, we apply a mild dropout (p=0.1) on both the embeddings and their aggregated representations during pretraining. Then we feed them into a linear masked language modeling head during pretraining, see Figure 4.1, or an MLP classification head to tackle downstream tasks. We chose an MLP because it adds nonlinearity to the model without introducing further complexity. In pre-experiments, the MLP downstream classifier has led to the best average performance across all tasks of the GLUE benchmark.

4.3.2 Cross-Architecture Distillation

A central question of our research is whether we can distill a large PreLM, e. g., BERT, into more efficient, non-Transformer architectures such as the proposed bidirectional CMOW/CBOW-Hybrid model. This requires a cross-architecture distillation approach, which we describe below.

In general, the idea of knowledge distillation is to compress the knowledge of a large teacher model into a smaller student model [HVD15; BCN06] . It involves a loss function $\mathcal{L}$ that is a combination of two loss terms, i. e., $\mathcal{L} = \lambda \cdot \mathcal{L}_{\text{hard}} + (1 - \lambda) \cdot \mathcal{L}_{\text{soft}}$ with weighting parameter λ. $\mathcal{L}_{\text{hard}}$ denotes the cross-entropy loss with respect to the ground truth and $\mathcal{L}_{\text{soft}} = \Sigma_i t_i \cdot \log(s_i)$ is the cross-entropy between student logits $\boldsymbol{s}$ and the teacher signal $\boldsymbol{t}$. Optionally, the softmax within $\mathcal{L}_{\text{soft}}$ is flattened by a temperature parameter T. We distinguish between **general distillation**, where BERT's teacher signal is only used during pretraining, and **task-specific distillation**, where the BERT teacher signal is used during fine-tuning for the downstream task.

Considering our goal of designing a cross-architecture distillation, the general distillation approach has the conceptual benefit that the teacher model is not needed for fine-tuning. Thus, the student model is capable of tackling downstream tasks without the supervision of the large teacher. This has the benefit that one does not need to carry around the BERT model to adapt to every new downstream task. Above, we have introduced the ability to emit per-token representations with lightweight (bidirectional) CMOW/CBOW-Hybrid embedding models. This enables us now to use BERT's teacher signal during pretraining together with a masked language modeling objective. In other words, this allows us to perform *general* cross-architecture distillation with matrix embeddings.

We consider three variants of cross-architecture distillation in our experiments: a) When using general distillation, depicted in Figure 4.1, BERT acts as a teacher during pretraining and the model is fine-tuned to downstream tasks on its own. b) For task-specific distillation, BERT acts as a teacher during fine-tuning, as shown in Figure 4.2. For this case, we have the option of either b1) starting with pretrained embeddings (from general distillation, i. e., the a) variant), or b2) starting from scratch with randomly initialized embeddings.

4.3.3 Two-Sequence Encoding with Matrix Embeddings

When fine-tuning our matrix embeddings to downstream tasks, we can deviate from BERT's input processing, even if BERT is used as a teacher. This is because

the distillation loss is computed per sentence (pair) and not per token. The input processing of BERT encodes two sequences by joining them into one sequence. For example, in a natural language inference task, there is a sentence A that potentially entails a sentence B, which is encoded as one sequence using a special separator token. This encoding scheme is less useful to our matrix embeddings without any attention component, since order-sensitive matrix multiplications would blend the representation of the two sequences.

To develop an appropriate two-sequence encoding scheme for matrix embeddings, we take inspiration from the pre-Transformer era, e. g., Mou et al. [MML+16], and from SentenceBERT [RG19]. The key idea is to encode two sentences A and B separately before combining them. As a combination operation, we use the absolute elementwise difference and concatenate it to the representations of A and B, which we denote as DiffCat:

$$\boldsymbol{h}^{(\text{DiffCat})} = \boldsymbol{h}^{(\text{A})} \parallel |\boldsymbol{h}^{(\text{A})} - \boldsymbol{h}^{(\text{B})}| \parallel \boldsymbol{h}^{(\text{B})}$$

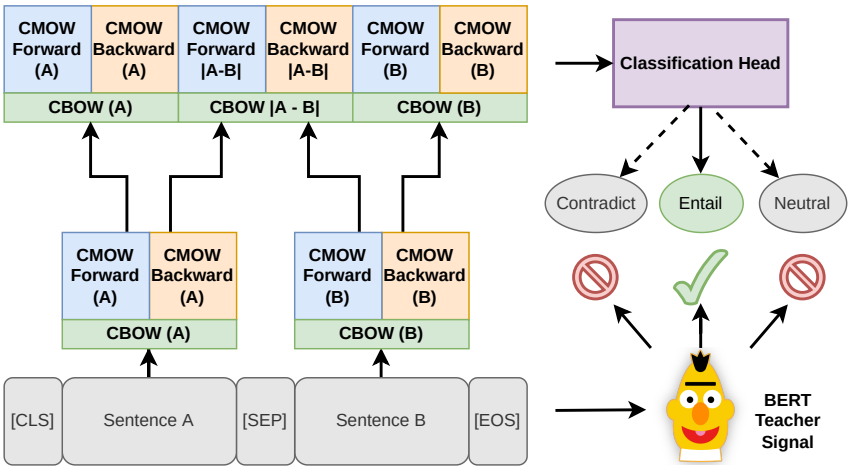


Figure 4.2. Separate encoding (DiffCat) for sequence pairs using a Bidirectional CMOW/CBOW-Hybrid model during fine-tuning, optionally, with task-specific distillation with a BERT teacher. Bert illustration: Melanie Poech

We illustrate this separate encoding scheme during task-specific distillation in Figure 4.2. The rationale for using a concatenation of both sequence representations along with their difference is that we add a component for the similarity of the two sequence representations, without loss of expressive power.

4.4 Datasets

Here, we briefly describe the datasets used for pretraining as well as the downstream tasks from the general language understanding evaluation benchmark (GLUE) benchmark, which we use for evaluation.

4.4.1 Dataset for Pretraining

As dataset for pretraining, we use a combination of English Wikipedia (about 2,500M words) and the Toronto BookCorpus (800M words). We select this dataset because it has been also used in the original work on BERT [DCL+19]. We also use the same vocabulary and tokenizer as in BERT to preprocess the dataset. The BERT tokenizer uses the WordPiece algorithm [WSC+16], which yields a high coverage while maintaining a small vocabulary. This ensures that the teacher and the student have the same vocabulary.

4.4.2 Datasets for Downstream Tasks

For model evaluation, we use the general language understanding evaluation benchmark (GLUE) benchmark [WSM+19] The GLUE benchmark consists of nine tasks for English language comprehension. These tasks comprise natural language inference (MNLI, QNLI, WNLI, RTE), sentence similarity (QQP, STS-B, MRPC), linguistic acceptability (CoLA), and sentiment analysis (SST-2). All tasks are defined on pairs of sentences except for CoLA and SST-2, which are single-sentence tasks.

In the GLUE benchmark, it is allowed to use different fine-tuning strategies for different tasks. This is different to the SentEval benchmark used in the previous Chapter 3, where the goal was to learn a fixed sentence representation. Instead, on GLUE, fine-tuning of the entire model for each task is encouraged.

The GLUE benchmark is widely used to evaluate natural language processing models. Therefore, we only briefly describe the tasks of the benchmark for the sake of completeness. All datasets for the tasks as well as a leaderboard is available under `https://gluebenchmark.com/`.

*Corpus of Linguistic Acceptability (*CoLA) CoLA is a dataset, in which each example is a single sentence. The dataset contains 9k training and 1k test examples and the task is to determine whether a sentence is linguistically acceptable.

*Multi-Genre Natural Language Inference corpus (*MNLI) In MNLI, each example comprises two sentences. Given a premise and a hypothesis, the natural language

inference task is to predict whether the premise entails the hypothesis. The dataset offers two test sets: MNLI-matched (in-domain) and MNLI-mismatched (cross-domain).

*Microsoft Research Paraphrase Corpus (*MRPC) The MRPC dataset consists of pairs of sentences from online news sources. The task is to predict whether two sentences are semantically equivalent, i. e., whether one is a paraphrase of the other.

*Question Natural Language Inference (*QNLI) The QNLI task is derived from the Stanford Question Answering Dataset [RZL+16] and consists of question-paragraph pairs where one of the paragraph's sentences contains the answer to the question. The original question answering task is converted to a sentence pair classification task: decide whether one specific sentence answers the question.

*Quora Question Pairs (*QQP) In QQP, the task is to decide whether two questions are semantically equivalent.

*Recognizing Textual Entailment (*RTE) The RTE task is another two-sentence task, where the task is to decide whether one text can be inferred (entailed) from the other.

*Two-way Stanford Sentiment Treebank (*SST-2) The SST dataset is a single-sentence sentiment analysis task [SPW+13]. The sentences are from movie reviews. In SST-2, we have a two-way split (positive / negative) and only use sentence-level labels.

*Semantic Textual Similarity Benchmark (*STS-B) The STS-B dataset is a two-sentence similarity task. It is the same dataset as in Chapter 3. The task is to predict similarity scores from 1 to 5.

*Winograd Natural Language Inference (*WNLI) The WNLI task is natural language inference on sentence pairs. In particular, the task tests reading comprehension that requires world knowledge. Given a sentence with an ambiguous pronoun, the task is to determine the referred noun. The task is converted into sentence pair classification task by substituting the pronoun with each possible referent and predicting whether the new sentence is entailed by the original sentence.

The performance on all four natural language inference (NLI) tasks (MNLI, QNLI, RTE, WNLI) as well as the sentiment analysis task SST-2 is measured with

accuracy. Linguistic acceptability (CoLA) is evaluated by Matthew's correlation coefficient. The performance on sentence similarity tasks is measured as the average of Pearson and Spearman correlation for the STS-B task, and as the average of accuracy and F_1-score for MRPC and QQP.

4.5 Experiments

Below, we provide the details of the experimental setup, before we provide the results of our experiments in Section 4.5.2 to Section 4.5.5

The experimental procedure is divided in pretraining on unlabeled text and fine-tuning on the downstream tasks, as described in Section 4.2. In the pretraining stage, we train our proposed bidirectional CMOW/CBOW-Hybrid model with a masked language model (MLM) objective [DCL+19] on large amounts of unlabeled text. The MLM objective is to predict the omitted words from their context. During pretraining, we employ an already-pretrained BERT model to act as a teacher for knowledge distillation. The training procedure is visualized in Figure 4.1.

We put equal weights on the MLM objective and the teacher signal from BERT ($\alpha = 0.5$). As suggested by Liu et al. [LOG+19] and Sanh, Debut, Chaumond, and Wolf [SDC+20], we do not use the next-sentence-prediction objective of BERT, but only the MLM objective.

To reduce the environmental footprint of our experiments, we have only pretrained a single bidirectional CMOW/CBOW-Hybrid model with BERT-base as a teacher on the full unlabeled training data, after pre-experiments on 10% of the training data showed that the selected CMOW/CBOW-Hybrid with distillation exceeded the performance of the baseline.

In the fine-tuning stage, as shown in Figure 4.2, the pretrained model is adapted for each downstream task separately. The training objective for fine-tuning is either cross-entropy with the ground truth (in general distillation) or a mixture of the ground truth loss and cross-entropy with respect to the teacher's logits (in task-specific distillation). Again, we put equal weight on ground truth and teacher signal ($\alpha = 0.5$) and use unit temperature for the softmax.

In the case of task-specific distillation, we cast the regression task STS-B into a classification task by binning the scores into intervals of 0.2, as also done by the T5 model [RSR+20]. This is necessary to facilitate distillation because the distillation procedure relies on a softmax teacher signal as obtained in classification, but not regression objectives.

For task-specific distillation, we employ an uncased BERT-base model[1] from the Huggingface repository that has already been fine-tuned for each task of the GLUE benchmark. We have fine-tuned the BERT model ourselves on the tasks STS-B, where we applied binning, and MNLI, on which the already fine-tuned model has led to sub-par results. We use the same fine-tuned BERT model as a teacher for all reported results with task-specific distillation.

We seek a fair comparison between the unidirectional CMOW/CBOW-Hybrid baseline model and our bidirectional model. As such, we allow both models to equally benefit from the BERT's teacher signal during fine-tuning. For this comparison, we use random initialization for both models because the pretrained embeddings from the previous chapter would come with a different vocabulary that covers only 53% of the one of BERT. We also use this strategy for the comparison of two-sentence encoding schemes, where we compare the proposed DiffCat encoding (see Section 4.3.3) with a joint encoding similar to BERT. In the other experiments, we initialize our bidirectional CMOW/CBOW-Hybrid with the pretrained embeddings from general distillation, while we isolate the effect of task-specific distillation in a dedicated experiment.

In terms of hyperparameter choices, we use matrix embeddings of size $20 \times 20 = 400$ ($d = 20$) for both the CMOW directions (forward and backward) and vector embeddings of size $d_{\text{vec}} = 400$. We optimize the learning rates in the range $[10^{-3}, 10^{-6}]$. To determine the best model, we use each task's evaluation measure on the development set. We run each model for 20 epochs with early stopping (5 epochs patience). We select appropriate batch sizes on the basis of preliminary experiments and training data sizes. The exact bounds and optimization method for the hyperparameters are reported in Section C.1 To fine-tune the CMOW/CBOW-Hybrid model from general distillation, we use an MLP classification head with three layers (or two hidden layers), a ReLU activation after the first layer, and a mild dropout with probability 0.1 plus layer normalization before the final classification layer. The hidden size of the MLP classifier is 1,200, which is the same as the embedding dimension: 400 for each direction of the CMOW component plus 400 for the CBOW component.

We present the results along the design choices introduced in Section 4.3: two-sequence encoding scheme, bidirection, cross-architecture distillation variants, before we compare our best embedding-based methods with ELMo and BERT distillates from the literature.

[1] https://huggingface.co/textattack

4.5.1 DiffCat Encoding versus Joint Encoding

First, we compare the encoding schemes for two-sentence tasks. On the one hand, we have the BERT-like encoding that encodes the two sentences together, separated by a special token. On the other hand, we have the proposed DiffCat encoding, which encodes each sentence separately before combining the representations. For a fair comparison, we use a randomly initialized unidirectional CMOW/CBOW-Hybrid model under task-specific distillation.

The result (see Table 4.1) shows that DiffCat encoding improves the results consistently with the largest margin on STS-B. The most notable improvement is the improvement from 18.5 to 58.6 in the sentence similarity task STS-B when encoding the sentence pair input via DiffCat. The average improvement for the two-sentence tasks of the GLUE benchmark is 20%, when the DiffCat encoding is used over a BERT-like joint encoding of sentence pairs. In subsequent experiments, we only report scores with DiffCat encoding.

Table 4.1. Comparison of DiffCat encoding and joint BERT-like encoding. Both variants use randomly initialized unidirectional CMOW/CBOW-Hybrid embeddings with MLP under task-specific distillation. The DiffCat encoding improves the average score across two-sentence tasks by 20%.

Two-Sentence Encoding	**Avg.**	MNLI-m	MRPC	QNLI	QQP	RTE	STS-B	WNLI
Joint Encoding	55.8	50.0	73.0	60.4	78.6	53.8	18.5	**56.3**
DiffCat Sep. Encoding	**66.8**	**62.5**	**74.3**	**71.5**	**86.6**	**58.1**	**58.6**	**56.3**

4.5.2 Bidirectional versus unidirectional CMOW/CBOW-Hybrid

To isolate the effect of the bidirectional component, we compare unidirectional with bidirectional CMOW/CBOW-Hybrid under the same conditions. Therefore, we train both variants starting with a random initialization for the downstream tasks, while using a BERT's teacher signal. Table 4.2 shows the results of comparing unidirectional Hybrid embeddings with the proposed bidirectional Hybrid embeddings. Having a bidirectional model helps for the tasks MNLI, MRPC, QNLI, SST-2, STS-B, and WNLI. On the other tasks, the difference between the two variants is small (0.1 on CoLA, 0.5 on QQP, 0.7 on RTE). We have an average improvement of 1% of the bidirectional model over the unidirectional model across all tasks of the GLUE benchmark.

Table 4.2. Comparison of unidirectional and bidirectional (prefix 'B-') variants of CMOW/CBOW-Hybrid under task-specific distillation from random initialization (suffix '-R'). We also report pretrained (suffix '-P') bidirectional Hybrid for reference. All methods use DiffCat encoding and an MLP classifier.

Model Type	Score	CoLA	MNLI-m	MRPC	QNLI	QQP	RTE	SST-2	STS-B	WNLI
Hybrid-R	62.5	**13.1**	62.5	74.3	71.5	**86.6**	**58.1**	83.1	58.6	56.3
B-Hybrid-R	**63.2**	13.0	**63.3**	**75.7**	**72.6**	86.1	57.4	**83.3**	**59.7**	**57.7**
B-Hybrid-P	**64.6**	**23.3**	61.8	75.0	72.0	86.3	**59.9**	82.9	**62.9**	**57.7**

4.5.3 General Distillation versus Task-specific Distillation

Next, we compare general distillation with task-specific distillation. As shown in Table 4.3, the use of general distillation leads to better results for five tasks (MNLI, MRPC, QQP, STS-B, and RTE) compared to task-specific distillation. For the other four tasks (CoLA, QNLI, SST-2, and WNLI), task-specific distillation achieves higher scores. The average score for general distillation is higher than for task-specific distillation in both pretrained and random initialization cases.

Table 4.3. Comparison of general and task-specific distillation using bidirectional CMOW/CBOW-Hybrid embeddings and MLP classifier. Task-specific distillation models have been either randomly initialized (suffix '-R') or initialized from pretraining (suffix '-P'). In 5 out of 9 tasks, general distillation performs better.

Distillation Type	Score	CoLA	MNLI-m	MRPC	QNLI	QQP	RTE	SST-2	STS-B	WNLI
General	**66.6**	16.7	**66.6**	**79.7**	71.7	**87.2**	**61.0**	82.9	**76.9**	56.3
Task-specific-R	63.2	13.0	63.3	75.7	**72.6**	86.1	57.4	**83.3**	59.7	**57.7**
Task-specific-P	64.6	**23.3**	61.8	75.0	72.0	86.3	59.9	82.9	62.9	**57.7**

4.5.4 Comparing Bidirectional CMOW/CBOW-Hybrid to the Literature

Finally, Table 4.4 shows the results of the bidirectional CMOW/CBOW-Hybrid variants that perform best using any of the three distillation methods considered. As described by Wasserblat, Pereg, and Izsak [WPI20], a model needs to capture context and linguistic structure to perform well on CoLA. We achieved higher scores on CoLA and SST-2 compared to the results of the cross-architecture distillation reported by Wasserblat, Pereg, and Izsak [WPI20]. Our best models scored higher than ELMo [PNI+18] on the tasks MRPC, QNLI, QQP, RTE, and WNLI. We achieve higher scores than DistilBERT on RTE and WNLI. In a recent extension of

CMOW, Word2rate [PLP21], the matrices are considered statistical transitions (rate matrices) in a Taylor series. The authors report the Word2rate scores on SST-2 and STS-B, which are lower than the CMOW/CBOW-Hybrid scores from the previous chapter. The CMOW/CBOW-Hybrid model of the previous chapter is, in turn, outperformed by the one explored in this chapter.

Table 4.4. Comparison of best embedding-based methods (in bold) with methods from the literature on the validation set of the GLUE benchmark.

Method	**Score**	CoLA	MNLI-m	MRPC	QNLI	QQP	RTE	SST-2	STS-B	WNLI
ELMo [PNI+18]	68.7	44.1	68.6	76.6	71.1	86.2	53.4	91.5	70.4	56.3
DistilBERT [SDC+20]	77.0	51.3	82.2	87.5	89.2	88.5	59.9	91.3	86.9	56.3
MobileBERT [SYS+20]	—	51.1	84.3	88.8	91.6	70.5	70.4	92.6	84.8	—
CBOW [WPI20]	—	10.0	—	—	—	—	—	79.1	—	—
BiLSTM [WPI20]	—	10.0	—	—	—	—	—	80.7	—	—
Hybrid [MGS19]	—	—	—	—	—	—	—	79.6	—	—
Word2rate [PLP21]	—	—	—	—	—	—	—	65.7	—	—
Bidi. Hybrid + MLP	**68.0**	**23.3**	**66.6**	**80.9**	**72.6**	**87.2**	**61.0**	**84.0**	**76.9**	**59.2**

4.5.5 Comparison of Parameter Count and Runtime Performance

Table 4.5. Number of parameters and inference time of the models. Inference time is measured as encoding speed without gradient computation on an NVIDIA A100-SXM4-40GB card

Model	# Parameters	Encoding speed (sentences / sec)
ELMo	94M	1.1k
BERT-base	109M	4.6k
DistilBERT-base	66M	9.2k
MobileBERT	**25M**	5.5k
Bidi. CMOW/CBOW-Hybrid	37M	**30.0k**

We compare the parameter count and runtime performance of the bidirectional CMOW/CBOW-Hybrid model with ELMo, BERT, and two BERT distillates: DistilBERT and MobileBERT. To compare runtime performance, we generate 1,024 batches with 256 random sequences of length 64 and measure the time to encode the sequences with gradient computation disabled. As shown in Table 4.5, CMOW/CBOW-Hybrid is more than 3 times faster than the fastest competitor,

DistilBERT, and only uses about half of its parameters. The inference speed of CMOW/CBOW-Hybrid could be increased even further because the $\mathcal{O}(n)$ steps to encode a sequence of length n can be parallelized to $\mathcal{O}(\log n)$ *sequential* steps, as matrix multiplication is associative.

4.6 Discussion

Key Results We have shown that BERT can be successfully distilled into efficient matrix embedding models. To facilitate this success, we have introduced intermediate representations, a bidirectional representations, and a two-sequence encoding scheme for matrix embedding models. We have observed that the general distillation approach, i. e., using the BERT teacher only during pretraining, leads to results that are often even better than those achieved with task-specific distillation. This is an interesting result because all previous work on cross-architecture distillation relied on task-specific distillation. The proposed model offers an inference speed that is three times faster than that of DistilBERT, while reaching similar performance on several downstream tasks such as QQP and RTE.

Limitations Currently, matrix embeddings still fall behind other BERT distillates such as MobileBERT or DistilBERT in many of the downstream tasks. In particular, detecting linguistic acceptability remains a challenge for non-Transformer models. However, the proposed approach provides an increased downstream task performance compared to previous cross-architecture distillation approaches.

Potential Benefits for Explainability There is criticism on large language models and the amplification of bias through the reduction of model size [BGM+21]. This is particularly a problem when the model is not explainable. In the proposed method of this chapter, we distill large-scale pretrained language models into models that do not have a nonlinearity (except for the final classification head). Since nonlinearities make explainability more difficult, following this approach may lead to more explainable language models, although explainability is not the focus of the present work.

General vs. Task-specific Distillation A central issue is whether the distillation procedure should be carried out in a general pretraining stage or in a task-specific training stage. On the one hand, the conceptual benefit of general distillation is that

the distilled model can be reused for multiple downstream tasks without requiring the teacher. On the other hand, general distillation is more resource-intensive than task-specific distillation because general distillation requires training on large amounts of unlabeled text. In our experiments, general distillation and task-specific distillation led to similar results. However, the average score for general distillation was higher than for task-specific distillation. This result is in line with the findings of Turc, Chang, Lee, and Toutanova [TCL+19] for same-architecture distillation. This suggests that general distillation is a direction that is worth further exploration.

Threats to Validity For the comparison of CMOW/CBOW-Hybrid with the best models from the literature, we reported the best-performing models from either general or task-specific distillation. Although this needs to be taken into account for interpreting the results, using a different training strategy per task is valid under the specifications of the GLUE benchmark [WSM+19]. We have ensured that all of the introduced extensions are compared to their respective baselines under fair and equal conditions: the encoding scheme, the component for bidirectional processing, and general vs. task-specific distillation. Specifically, we resorted to using non-pretrained models for a fair comparison between the unidirectional and bidirectional variants.

Generalizabilty We have evaluated the proposed methods on the GLUE benchmark. The GLUE benchmark spans a wide range of language-related tasks including natural language inference, sentence similarity, linguistic acceptability, and sentiment analysis. The performance on the GLUE benchmark hints at generalizable linguistic capabilities of the model, especially regarding two-sequence tasks. Particularly, for such two-sequence tasks, the DiffCat encoding scheme for matrix embeddings has improved the performance by 20% compared to a BERT-like joint encoding. We expect this encoding scheme to be particularly valuable in other settings, where the relationship between two sentences is crucial, such as information retrieval.

Practical Impact The methods presented in this chapter are especially useful in applications, where the encoding speed is of central importance. Matrix embeddings offer a speed-up with factor 3 compared to DistilBERT and with factor 6 compared to BERT. Given this advantage, matrix embedding approaches could be valuable in high-throughput areas, such as in social media.

4.7 Summary

We have introduced three extensions to the CMOW/CBOW-Hybrid model: a bidirectional component, a separate two-sequence encoding scheme, and the ability to emit per-token representations. These per-token representations allow us to distill BERT into CMOW/CBOW-Hybrid already during pretraining with a masked language modeling objective. Our results show that a separate encoding scheme improves the performance of CMOW/CBOW-Hybrid on two-sentence GLUE tasks by 20%, while adding a bidirectional component improves performance by 1% compared to the unidirectional model. Furthermore, the results suggest that general distillation is sufficient and task-specific distillation is not necessary for most GLUE tasks. Compared with more expensive models from the literature, our embedding-based approach achieves scores that match or exceed the scores of ELMo and are competitive to DistilBERT on QQP and RTE with only half of DistilBERT's parameters and three times its encoding speed. While linguistic acceptability remains a challenge for non-Transformer models, the approach presented in this chapter yields notably higher scores than previous cross-architecture distillation approaches. Lastly, we note how order-sensitive matrix embeddings can be implemented and scaled efficiently, as scaling issues that apply to Transformer-based architectures do not apply to matrix embeddings models.

Chapter 5

Wide Multilayer Perceptrons for Text Classification

In this chapter, we answer *Q3: Are synthetic graph structures derived from raw text necessary for topical text classification?* The chapter is based on material from a publication [GS22] in the Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (2022).

Text classification, or text categorization, is the task of automatically assigning categories to text units such as documents, social media postings, or news articles. Text classification research is an active research field, as the sheer amount of new methods in recent surveys shows [BKR21; LPL+21; ZGL+20; KMH+19; Kad19]. Topical text classification finds application in information retrieval scenarios where text units need to be made accessible for search. Naively extracting category names from text units is not enough, because texts can also fall into a certain category without explicitly naming it [GMS+17].

The traditional way to compose an input representation of text is to count word occurrences. We then get a feature vector with the dimension of the vocabulary of all possible words, in which each entry corresponds to the number of times that the respective word (or token) appears in the text. This is what we call a Bag-of-Words (BoW) in terms of a multiset of tokens (see Section 2.2.3).

Most traditional text classification methods rely on this BoW input representation, optionally normalized in length and/or weighted by TF-IDF. Such traditional approaches include support vector machines (SVMs), logistic regression, or k-nearest neighbors. There are also deep text classification approaches that rely on a BoW such as deep averaging networks (DANs) [IMB+15], a deep MLP model that operates by averaging the input BoW, simple word embedding model (SWEM) [SWW+18] that explores different pooling strategies for pretrained word embeddings, and fastText [BGJ+17], which uses a linear layer on top of pretrained word embeddings. Because these models simply count occurrences of the tokens in the input sequence, they disregard word position and order, and then

rely, for instance, on pretrained word embeddings and fully connected feedforward layer(s). We call these **BoW-based models**.

Recently, models that make up a graph representation from words and documents for text classification became popular. For example, TextGCN [YML19] first induces a synthetic word–document co-occurrence graph across the entire corpus and then applies a graph neural network (GNN) to perform the classification task. In addition to TextGCN, there are follow-up works that employ a similar strategy, such as HeteGCN [RSI+21], TensorGCN [LYZ+20], and HyperGAT [DWL+20], which we collectively call **graph-based models**.

Finally, we have the well-known Transformer [VSP+17] models such as BERT [DCL+19], RoBERTa [LOG+19], T5 [RSR+20], and GPT-3 [BMR+20] as well as size-reduced variants such as DistilBERT [SDC+20] and others [TDB+20; FCA21]. Here, the input is a (fixed-length) sequence of tokens, which is then fed into multiple layers of self-attention. Lightweight versions such as DistilBERT use fewer parameters but operate on the same type of input. Together with recurrent models like the LSTM [HS97], we call these **sequence-based models**.

In this chapter, we hypothesize that text categorization can be performed very well by simple but effective BoW-based models. We investigate this research question in three steps: First, we conduct an in-depth analysis of the literature. We review key research in the field of modern and classical machine learning methods for text categorization. From this analysis, we derive the different families of methods, the established benchmark datasets, and identify the top-performing methods. We decide for which models we report numbers from the literature and which models we run on our own. In total, we compared 16 different methods from the families of BoW-based models (8 methods), sequence-based models (3 methods) and graph-based models (5 methods). We ran our own experiments for 7 of these methods on five text categorization datasets, while we report the results from the literature for the remaining methods.

The result is surprising. The BoW-based MLP, denoted WideMLP, with only one but wide hidden layer, outperforms many of the recent graph-based models for text categorization [YML19; LYZ+20; RSI+21], when the test documents are unseen at training time, i. e., the inductive learning setup. Furthermore, we did not find any reported scores for BERT-based methods from the sequence-based family. Thus, we fine-tuned BERT [DCL+19] and DistilBERT [SDC+20] ourselves, which sets a new state of the art on the considered datasets.

At the meta-level, our study shows that MLPs have been largely ignored as competitor methods in experiments. It seems as if MLPs have been forgotten as

a baseline in the literature, which instead is focusing mostly on other advanced Deep Learning architectures. However, considering strong baselines is an important means to argue about *true* scientific advancement [SWW+18; DCJ19].

In the following, we introduce our methodology and the results from the literature study. Subsequently, we introduce the families of models in Section 5.3. The datasets are described in Section 5.4. We present the experiments in Section 5.5 and discuss our findings in Section 5.6, before we summarize the chapter.

5.1 Related Prior Work

We analyze the literature on text classification. In a first step, we have consulted recent surveys and comparison studies [BKR21; LPL+21; ZGL+20; KMH+19; Kad19; GMS+17; ZWY+16]. These cover the range from shallow to deep classification models. Second, we have screened the literature in key NLP and AI venues. Finally, we have complemented our search by checking the results and papers on `paperswithcode.com`. Based on this input, we have determined three families of methods and benchmark datasets (see Table 5.2). We focus our analysis on identifying models per family showing strong performance and select the methods to include in our study. We check whether modified versions of the datasets have been used (e. g., less classes). For all results, we have verified that the same train-test split is used on all datasets to avoid bias and incorrectly giving advantages.

5.1.1 Bag-of-Word-based Models

Classical machine learning models that operate on a BoW-based input are extensively discussed in two surveys [KMH+19; Kad19] and other comparison studies [GMS+17]. Iyyer, Manjunatha, Boyd-Graber, and III [IMB+15] proposed DAN, which combine word embeddings and deep feedforward networks. It is an MLP with 1-6 hidden layers, non-linear activation, dropout, and AdaGrad as the optimization method. The results suggest using pretrained word embeddings such as GloVe [PSM14] over a randomly initialized continuous BoW [KGB14] as input. In fastText [BGJ+17; JGB+17] a linear layer is used on top of pretrained embeddings for classification. Furthermore, Shen et al. [SWW+18] explore embedding pooling variants and find that SWEM can rival approaches based on RNNs and 1D-CNNs. We consider fastText, SWEM, and a DAN-like deeper MLP in our comparison.

Note that those approaches that rely on logistic regression on top of pretrained word embeddings, e. g., fastText, share an architecture similar to that of an MLP

with one hidden layer. However, the standard training protocol involves pretraining the word embedding on large amounts of unlabeled text and then freezing the word embeddings while training only a logistic regression [MSC+13].

5.1.2 Graph-based Models

Using graphs induced from text for the task of text categorization has a long history in the community. An early work is the term co-occurrence graph of the KeyGraph algorithm [OBY98]. The graph is divided into segments representing the key concepts of the document. Co-occurrence graphs have also been used for automatic keyword extraction, such as in the influential early work of the automatic keyword extraction method RAKE [REC+10] and can also be used for classification [ZDX+21].

Modern approaches exploit this idea in combination with graph neural networks (GNN) [Ham20]. Examples of GNN-based methods operating on a word-document co-occurrence graph are TextGCN [YML19] and its successor TensorGCN [LYZ+20] aswell as HeteGCN [RSI+21], HyperGAT [DWL+20], and DADGNN [LGG+21]. We briefly discuss these models.

In TextGCN, the authors created a graph based on word-word connections given by pointwise mutual information (PMI) and word-document TF-IDF scores based on sliding windows. They use a one-hot encoding as node features and apply a two-layer GCN [KW17] on the graph to perform the node classification task.

TensorGCN uses multiple ways of converting text data into graph data, including a semantic graph created with an LSTM, a syntactic graph created by dependency parsing, and a sequential graph based on word co-occurrence. HeteGCN combines ideas from Predictive Text Embedding [TQM15] and TextGCN and divides the adjacency matrix into its word–document and word–word submatrices and fuses the different layer representations when required. HyperGAT extended the idea of text-induced graphs for text classification to hypergraphs. The model uses graph attention and two kinds of hyperedges. Sequential hyperedges represent the relationship between sentences and their words. Semantic hyperedges for word-word connections are derived from topic models [BNJ03]. Finally, DADGNN is a graph-based approach that uses attention diffusion and decoupling techniques to tackle oversmoothing of the GNN, and thus stack more layers.

In TextGCN's original transductive formulation, the entire graph including the test set, needs to be known for training. This may be prohibitive in practical applications, as each batch of new documents would require retraining of the

model. When these methods are adapted for inductive learning, where the test set is unseen, they achieve considerably lower scores [RSI+21].

In general, many of the GNNs for text classification exploit corpus-level statistics, e. g., PMI, to connect related words in a graph [YML19]. When these word-word connections were omitted, the GNNs would collapse to bag-of-words MLPs. GloVe [PSM14] also captures PMI corpus statistics by factorizing the PMI matrix into word embeddings. Therefore, we include an MLP on GloVe input representations in our comparison.

5.1.3 Sequence-based Models

In the following, we discuss sequence models, which we further divide into the RNN and CNN families and the Transformer family.

Sequence models based on RNNs and CNNs RNNs are a natural choice for any NLP task However, it turned out to be challenging to find numbers reported on text categorization in the literature that can be used as references. The bidirectional LSTM with two-dimensional max-pooling BLSTM-2DCNN [ZQZ+16] has been applied to a stripped-down version with only 4 classes of the 20ng dataset. Therefore, the high score of 96.5 reported for 4ng cannot be compared to the papers applied on the entire 20ng dataset. Also TextRCNN [LXL+15], a model that combines recurrence and convolution. uses only the four main categories in the 20ng dataset. The results of Text-RCNN are identical to those of BLSTM-2DCNN. For the MR dataset, BLSTM-2DCNN does not provide information on the specific split of the dataset. RNN-Capsule [WSH+18] is a sentiment analysis method that achieves an accuracy of 83.8 on the MR dataset, but with a different train-test split. Lyu and Liu [LL20] combine a 2D-CNN with bidirectional RNN. Another work that applies a combination of a convolutional layer and a LSTM layer is Wang et al. [WLC+19]. The authors experimented with five English and two Chinese datasets, which are not in the set of representative datasets we identified. The authors report that their approach outperforms existing models such as fastText on two of the five English datasets and both Chinese datasets.

Sequence models based on Transformers Essentially, all Transformer-based language models are pre-trained in a self-supervised fashion on large text corpora and subsequently fine-tuned in supervised learning on a specific task. The leader-

board of the GLUE benchmark datasets[1] (and SuperGLUE) has become the key battleground for comparing the performance of BERT [DCL+19] and other large pretrained language models [LOG+19; RSR+20; BHA+21]. Despite the different characteristics of the GLUE tasks (e. g., natural language inference, textual similarity, linguistic acceptability), they basically all resemble a classification objective. Even similarity tasks that are evaluated by correlation with ground truth can be transformed into classification tasks [RSR+20]. Given that classification is a key objective of BERT-style methods, it would be assumed that there is work that considers the benchmark datasets for document-based text classification.

Recent work shows that BERT outperforms the classic TF-IDF BoW approaches on English, Chinese, and Portuguese text classification datasets [GG21]. We have not found any results of Transformer-based models reported on the text classification datasets that are commonly used in graph-based approaches.

Therefore, we fine-tune BERT [DCL+19] and DistilBERT [SDC+20] on these datasets ourselves. BERT is a large pretrained language model on the basis of Transformers. DistilBERT [SDC+20] is a distilled version of BERT with 40% reduced parameters while retaining 97% of the score of BERT on the GLUE benchmark. TinyBERT [JYS+20] and MobileBERT [SYS+20] would be similarly suitable alternatives, among others. We chose DistilBERT because it can be fine-tuned independently of the BERT teacher. and its inference times are 60% faster than BERT, making it more likely to be reusable with limited resources.

5.1.4 Summary

From our literature survey, we see that all recent methods are based on graphs. BoW-based methods are scarce in experiments, whereas, likewise surprisingly, Transformer-based sequence models are extremely scarce in the literature on topical text classification. Recent surveys on text classification include both classical and Deep Learning models, but none considered a simple MLP except for the inclusion of DAN [IMB+15] in Li et al. [LPL+21].

Table 5.1 summarizes the basic properties of the methods found in the literature. Most importantly, graph approaches are not inductive in their basic formulation (as in TextGCN and TensorGCN). Follow-up works such as HeteGCN and HyperGAT relax this constraint. Both the BoW approaches and (most) graph-based approaches are not sensitive to word order, but rather build an input representation by counting word occurrences. Although sequence models naturally take word order and

[1] https://gluebenchmark.com/leaderboard

Table 5.1. Properties of text classification approaches. Graph-based models that rely on having access to unlabeled test documents such as TextGCN and TensorGCN are not capable of inductive learning without specific modifications.

Model	Builds Graph	Position-Aware	Arbitrary Length	Inductive
Bag-of-Words	No	No	Yes	Yes
Graph: TextGCN	Yes	No	Yes	No
Graph: TensorGCN	Yes	Yes	Yes	No
Graph: HeteGCN/HyperGAT	Yes	No	Yes	Yes
Sequence: RNN/CNN	No	Yes	Yes	Yes
Sequence: BERT/DistilBERT	No	Yes	No	Yes

position into account, there is a difference between the RNNs- and CNN-based approaches and Transformers. RNNs and CNNs can deal with sequences of arbitrary length because they combine the sequence into a single representation. Transformers, on the other hand, typically have a maximum sequence length. However, there are already efforts to design more efficient Transformers that can also be applied to long sequences [TDA+21].

5.2 Problem Formulation

We study the problem of text classification. A model is trained on paired training data $\mathcal{D} = \{(\boldsymbol{x}, \boldsymbol{y})_i\}$, before being applied to test data $\mathcal{D}_{\text{test}}$, where $\mathcal{D} \cap \mathcal{D}_{\text{test}} = \varnothing$.

We distinguish between transductive and inductive settings (cf. Section 2.4). In the transductive setting, the model has access to the unlabeled examples of $\mathcal{D}_{\text{test}}$ during training. This is also called semi-supervised learning. In contrast, the test data $\mathcal{D}_{\text{test}}$ remain completely unseen in an inductive setting.

This distinction is important because it determines how the model can be applied in practice. In practical applications, an inductive model is much more useful, as it can be applied to new data without retraining [HHY+19].

5.3 Methods

We formally introduce the three families of text classification models, namely the BoW-based, graph-based, and sequence-based models. Table 5.1 summarizes the key properties of the approaches: whether they require a synthetic graph, whether

the position of words is reflected in the model, whether the model can deal with text of arbitrary length, and whether the model is capable of inductive learning.

5.3.1 Bag-of-Words-based Text Classification

Under pure BoW-based text categorization, we denote approaches that are not sensitive to word order and operate only on the multiset of words from the input document. Given paired training examples $(\boldsymbol{x}, y) \in \mathcal{D}$, each consisting of a BoW $\boldsymbol{x} \in \mathbb{R}^{n_{\text{vocab}}}$ and a class label $y \in \mathbb{Y}$, the goal is to learn a generalizable function $\hat{\boldsymbol{y}} = f_\theta^{(\text{BoW})}(\boldsymbol{x})$ with parameters θ such that $\arg\max(\hat{\boldsymbol{y}})$ preferably equals the true label y for input $\boldsymbol{x}$.

As models based on BoW input representation, we consider one hidden layer WideMLP (i. e., two layers in total). We further experiment with pure BoW, TF-IDF weighted or averaged GloVe input representations and two hidden layers WideMLP-2. We also list the numbers for fastText, SWEM, and logistic regression from Ding et al. [DWL+20] in our comparison.

5.3.2 Graph-based Text Classification

Graph-based text categorization approaches first set up a graph *synthetic* based on the text corpus $\mathcal{D}$ in the form of an adjacency matrix $\hat{\boldsymbol{A}} := \text{make-graph}(\mathcal{D})$. For instance, TextGCN the graph is set up in two parts: word-word connections modeled by pointwise mutual information, and word-document edges resemble that the word occurs in the document. Then, a parameterized function $f_\theta^{(\text{graph})}(\boldsymbol{X}, \hat{\boldsymbol{A}})$ is learned that uses the graph as input, where $\boldsymbol{X}$ are the node features. The graph is made up of word nodes and document nodes, each receiving its own embedding (by setting $\boldsymbol{X} = \boldsymbol{I}$). However, in inductive learning, there is no embedding of test documents. Note that graph-based approaches from the current literature, such as TextGCN, also disregard word order, similar to the BoW-based models described above. A detailed discussion of the connection between TextGCN and MLP is provided in Appendix D.2.

We consider the best graph-based models in the literature, namely TextGCN along with its successors HeteGCN, TensorGCN, HyperGAT, DADGNN, and simplified GCN (SGC) [WJZ+19]. We do not run our own experiments for graph-based models, but we rely on the original work and extensive studies by Ding et al. [DWL+20] and Ragesh et al. [RSI+21].

5.3.3 Sequence-based Text Classification

As sequence-based text classification models, we consider models such as RNNs, LSTMs, and Transformers. These models are sensitive to the ordering of words in the input text. Thus, the key difference between the BoW-based and graph-based families is that the order of the words is reflected in the sequence-based model. The model signature is $\hat{y} = f_\theta^{(\text{sequence})}((x_1, x_1, \ldots, x_k))$, where k is the (maximum) sequence length. In Transformers, the position and order of words are modeled by a dedicated positional encoding. For example, in BERT each position is associated with an embedding vector that is added to the word embedding at the input level.

For the sequence-based models, we run our own experiments with BERT and DistilBERT, while reporting the scores of a pretrained LSTM from Ding et al. [DWL+20] for comparison.

5.4 Datasets

Table 5.2. Characteristics of text classification datasets

Dataset	N	#Train	#Test	#Classes	Avg. length and SD
20ng	18,846	11,314	7,532	20	551 $\pm$ 2,047
R8	7,674	5,485	2,189	8	119 $\pm$ 128
R52	9,100	6,532	2,568	52	126 $\pm$ 133
ohsumed	7,400	3,357	4,043	23	285 $\pm$ 123
MR	10,662	7,108	3,554	2	25 $\pm$ 11

We use the same datasets and train-test split as in TextGCN [YML19], which we describe here only briefly since they are common benchmark datasets for test classification. These datasets are 20ng, R8, R52, ohsumed, and MR.

Twenty Newsgroups (20ng) The 20ng dataset contains long posts categorized into 20 newsgroups. The mean sequence length is 551 words with a standard deviation of 2,047. The dataset can be found at `http://qwone.com/~jason/20Newsgroups/`.

Reuters datasets: R8 and R52 The datasets R8 and R52 are well-established subsets of the Reuters 21578 news dataset with 8 and 52 classes, respectively. The mean length of the sequence and SD is 119 $\pm$ 128 words for R8, and 126 $\pm$ 133 words for R52.

Ohsumed Ohsumed is a corpus of medical abstracts from the MEDLINE database that are classified into diseases (one per abstract). The mean sequence length is 285 ± 123 words. The dataset can be found under `http://disi.unitn.it/moschitti/corpora.htm`.

Movie Reviews (MR) The MR dataset contains a binary sentiment analysis task at the sentence level (mean sequence length and SD: 25 ± 11). The original dataset has been assembled by Pang and Lee [PL05]. We reuse the train/test-split by Tang, Qu, and Mei [TQM15]. The dataset can be found at `https://www.cs.cornell.edu/people/pabo/movie-review-data/`.

A summary of the basic characteristics of the data set, i. e., the number of examples in the train and test set, the number of classes, and the sequence lengths are shown in Table 5.2. Note that 20ng has the longest sequences, while MR has the shortest.

5.5 Experiments

Below, we describe the experimental procedure before providing the results along the dimensions effectiveness in Section 5.5.1 and efficiency in Section 5.5.2.

Procedure We distinguish between a transductive and an inductive setup for text categorization. In the transductive setup, as used in TextGCN, the (unlabeled) test documents are visible and are actually used to prepare the graph and train the model. In the inductive setting, the test documents remain unseen until the test time (i. e., they are not available at preprocessing/training time).

In practice, a transductive setup would correspond to completing the annotations in a partially annotated corpus. On the other hand, an inductive model would be able to seamlessly handle new documents continuously appearing over time, where a transductive model would require retraining the model whenever a new document appears.

We report the scores of the graph-based models for both setups from the literature, where available. BoW-based and sequence-based models are inherently inductive. Ragesh et al. [RSI+21] have evaluated a variant of TextGCN that is capable of inductive learning, which we also include in our results.

We have extracted accuracy scores from the literature according to our systematic selection from Section 5.1. In the following, we provide a detailed description

of the procedure for the models that we have run ourselves. We borrowed the tokenization strategy from BERT [DCL+19] along with its uncased vocabulary of 30k tokens. The tokenizer is based mainly on WordPiece [WSC+16] for high coverage while maintaining a small vocabulary.

Hyperparameter settings Our WideMLP has one hidden layer with 1,024 rectified linear units (one input-to-hidden and one hidden-to-output layer). We apply dropout after each hidden layer, notably also after the initial embedding layer. Only for GloVe+WideMLP, neither dropout nor ReLU is applied to the *frozen* pretrained embeddings, but only on subsequent layers. The rationale is that we want to avoid cutting the entire negative range from the pretrained GloVe vectors by applying a ReLU. The variant WideMLP-2 has two ReLU-activated hidden layers (three layers in total) with $1,024$ hidden units each. Although this might be overparameterized for single-label text classification tasks with few classes, we rely on recent findings that overparameterization leads to better generalization [NLB+18; NKB+20]. In the pre-experiments, we realized that MLPs are not very sensitive to hyperparameter choices. Therefore, we optimize for cross-entropy with Adam [KB15] and its default learning rate of 10^{-3}, a linearly decaying learning rate schedule, and train for a large number of steps [NKB+20] (we use 100 epochs) with small batch sizes (we use 16) for sufficient stochasticity, along with a dropout ratio of 0.5.

For BERT and DistilBERT, we fine-tuned for 10 epochs with a linearly decaying learning rate of $5 \cdot 10^{-5}$ and an effective batch size of 128 through a gradient accumulation of 8 x 16 batches. We truncate all inputs to 512 tokens. To isolate the influence of word order on BERT's performance, we experiment with two further variants. First, we set all position embeddings to zero and disable their gradient updates (BERT w/o pos emb). By doing this, we force BERT to operate on a BoW without any notion of word order or position. Second, we shuffle each sequence to augment the training data. We use this augmentation strategy to double the number of training examples (BERT w/ shuf. aug.).

Measures We report accuracy as the main evaluation metric, which is equivalent to Micro-F1 in single-label classification (see Appendix D.3). We repeat all experiments five times with different random initializations of the parameters and report the mean and standard deviation of these five runs.

In the following, we report the results along the dimensions of effectiveness and efficiency.

5.5.1 Classification Accuracy

Table 5.3 shows the accuracy scores for the text classification models on the five datasets. All graph-based models in the transductive setting show similar accuracy scores (maximum difference is 2 points). As expected, scores decrease in the inductive setting to a point where our WideMLP matches or even outperforms them.

In the inductive setting, the WideMLP models perform best among the BoW models, in particular, TF-IDF+WideMLP and WideMLP on an unweighted BoW. The best performing graph-based model is HyperGAT, yet DADGNN has a slight advantage on R8, R52, and MR. For sequence-based models, BERT achieves the highest scores, closely followed by DistilBERT.

We complement the results with the scores for the pretrained bidirectional CMOW/CBOW-Hybrid + MLP model from the previous chapter. The model achieves reasonable scores, which are comparable to WideMLP on R8, R52, and MR. On datasets with longer texts (20ng and MR), the BoW-based MLPs yield higher scores.

The strong performance of WideMLP rivals all graph-based techniques reported in the literature, particularly the recently published graph-inducing methods. MLP only falls behind HyperGAT, which relies on topic models to set up the graph. Another observation is that one hidden layer (but wide) is sufficient for the tasks considered, as the scores for MLP variants with 2 hidden layers are lower. We further observe that both pure BoW and TF-IDF weighted BoW lead to better results than approaches that exploit pretrained word embeddings such as GloVe-MLP, fastText, and SWEM.

With its immense pretraining, BERT attains the highest overall scores, closely followed by DistilBERT. DistilBERT outperforms HyperGAT by 7 points on the MR dataset while being on par with the others. BERT outperforms the strongest graph-based competitor, HyperGAT, by 8 points on MR, 1.5 points on ohsumed, 1 point on R52 and R8, and 0.5 points on 20ng.

Augmenting BERT's data with shuffled sequences has led to neither a consistent decrease nor an increase in performance.

5.5.2 Efficiency

Parameter Count of the Models Table 5.4 lists the parameter counts of the models. Although MLP is fully connected on top of a bag-of-words with the dimensionality

Table 5.3. Accuracy and standard deviation on text classification datasets. Column "Provenance" reports the source.

Inductive Setting	**20ng**	**R8**	**R52**	**ohsumed**	**MR**	**Provenance**
BoW-Models						
Log. Regression	83.70	93.33	90.65	61.14	76.28	[RSI+21]
SWEM	85.16 (0.29)	95.32 (0.26)	92.94 (0.24)	63.12 (0.55)	76.65 (0.63)	[DWL+20]
fastText	79.38 (0.30)	96.13 (0.21)	92.81 (0.09)	57.70 (0.49)	75.14 (0.20)	[DWL+20]
TF-IDF + WideMLP	84.20 (0.16)	97.08 (0.16)	93.67 (0.23)	66.06 (0.29)	76.32 (0.17)	own exp.
WideMLP	83.31 (0.22)	97.27 (0.12)	93.89 (0.16)	63.95 (0.13)	76.72 (0.26)	own exp.
WideMLP-2	81.02 (0.23)	96.61 (1.22)	93.98 (0.23)	61.71 (0.33)	75.91 (0.51)	own exp.
GloVe+WideMLP	76.80 (0.11)	96.44 (0.08)	93.58 (0.06)	61.36 (0.22)	75.96 (0.17)	own exp.
GloVe+WideMLP-2	76.33 (0.18)	96.50 (0.14)	93.19 (0.11)	61.65 (0.27)	75.72 (0.45)	own exp.
Graph-based Models						
TextGCN	80.88 (0.54)	94.00 (0.40)	89.39 (0.38)	56.32 (1.36)	74.60 (0.43)	[RSI+21]
HeteGCN	84.59 (0.14)	97.17 (0.33)	93.89 (0.45)	63.79 (0.80)	75.62 (0.26)	[RSI+21]
HyperGAT	86.62 (0.16)	97.07 (0.23)	94.98 (0.27)	69.90 (0.34)	78.32 (0.27)	[RSI+21]
DADGNN	—	98.15 (0.16)	95.16 (0.22)	—	78.64 (0.29)	[LGG+21]
Sequence-based Models						
LSTM (pretrain)	75.43 (1.72)	96.09 (0.19)	90.48 (0.86)	51.10 (1.50)	77.33 (0.89)	[DWL+20]
DistilBERT	86.24 (0.26)	97.89 (0.15)	95.34 (0.08)	69.08 (0.60)	85.10 (0.33)	own exp.
BERT	87.21 (0.18)	98.03 (0.24)	96.17 (0.33)	71.46 (0.54)	86.61 (0.38)	own exp.
BERT w/o pos emb	81.47 (0.49)	97.39 (0.20)	94.70 (0.27)	65.18 (1.53)	80.35 (0.20)	own exp.
BERT w/ shuf. aug.	86.46 (0.42)	98.07 (0.21)	96.48 (0.18)	70.94 (0.60)	86.23 (0.33)	own exp.
Bidi. Hybrid	79.63	97.17	93.38	64.09	76.45	own exp.
Transductive Setting	**20ng**	**R8**	**R52**	**ohsumed**	**MR**	**Provenance**
Graph-based Models						
TextGCN	86.34	97.07	93.56	68.36	76.74	[YML19]
SGC	88.5 (0.1)	97.2 (0.1)	94.0 (0.2)	68.5 (0.3)	75.9 (0.3)	[WJZ+19]
TensorGCN	87.74	98.04	95.05	70.11	77.91	[LYZ+20]
HeteGCN	87.15 (0.15)	97.24 (0.51)	94.35 (0.25)	68.11 (0.70)	76.71 (0.33)	[RSI+21]

of the vocabulary size, it has only half of the parameters as DistilBERT and a quarter of the parameters of BERT. Using TF-IDF does not change the number of model parameters. Due to the high vocabulary size, GloVe-based models have a high number of parameters, but the majority of these are frozen and do not receive gradient updates during training. Note that the majority of MLP parameters (about 30M for the first layer) can be implemented as an embedding layer. This way, a table lookup is sufficient rather than a large, dense matrix multiplication.

Runtime Performance of the Models We provide the *total* running times in Table 5.5 as observed while conducting the experiments on a single NVIDIA A100-

Table 5.4. Parameter counts of the models

Model	#parameters
WideMLP	31.3M
WideMLP-2	32.3M
GloVe+WideMLP	575,2M (frozen) + 0.3M
GloVe+WideMLP-2	575,2M (frozen) + 1.3M
DistilBERT	66M
BERT	110M

SXM4-40GB card. All WideMLP variants are an order of magnitude faster than DistilBERT when considering the average runtime *per epoch*. DistilBERT is twice as fast as the original BERT. The Transformers are only faster than the BoW models on the MR dataset. This is because the sequences in the MR dataset are much shorter and fewer $\mathcal{O}(L^2)$ attention weights must be computed.

Table 5.5. Total runtime (training+inference). Average of five runs rounded to minutes.

Model	#epochs	20ng	R8	R52	ohsumed	MR
WideMLP	100	7min	3min	4min	3min	4min
TF-IDF+WideMLP	100	9min	4min	4min	3min	4min
WideMLP-2	100	9min	5min	5min	3min	6min
GloVe+WideMLP	100	6min	3min	4min	3min	4min
GloVe+WideMLP-2	100	6min	4min	4min	3min	4min
DistilBERT	10	8min	4min	5min	3min	1min
BERT	10	15min	7min	8min	5min	2min

5.6 Discussion

Key Results Our experiments show that our wide MLP models that use the BoW input representations outperform the recent graph-based models TextGCN and HeteGCN in inductive text classification. Furthermore, the MLP models are comparable to HyperGAT. Only Transformer-based BERT and DistilBERT models outperform our wide MLP and set a new state-of-the-art. This result is important for two reasons: First, the strong performance of a pure BoW-MLP questions the

added value of synthetic graphs in models like TextGCN to the text categorization task. Only HyperGAT, which uses the expensive Latent Dirichlet Allocation for computing the graph, slightly outperforms our BoW-WideMLP in two of five datasets. Therefore, we argue that the use of strong baseline models for text classification is important to assess true scientific progress [DCJ19].

Second, in contrast to conventional wisdom [IMB+15], we find that pretrained embeddings, e. g., GloVe, can have a detrimental effect compared to using an MLP with a single wide hidden layer. Such an MLP bypasses the bottleneck of the small dimensionality of word embeddings and has a higher capacity. Furthermore, we experiment with more hidden layers (see WideMLP-2), but do not observe any improvement when the single hidden layer is sufficiently wide. A possible explanation is that a single hidden layer is sufficient to approximate any compact function to an arbitrary degree of accuracy depending on the width of the hidden layer [Cyb89].

Finally, the Transformer model BERT sets a new state of the art. However, as our efficiency analysis shows, MLPs require only a fraction of the parameters and are faster in their combined training and inference time, except for the MR dataset. The attention mechanism of (standard) Transformers is quadratic in sequence length, which leads to slower processing of long sequences. With larger batches, the speed of MLP could be increased even further.

Discussion on the Use of Graphs for Text Classification Graph-based models come with high training costs, as not only the graph has to be first computed, but also a GNN has to be trained. For standard GNN methods, the entire graph has to fit into the GPU memory, and mini-batching is non-trivial but possible with dedicated sampling techniques for GNNs [FLW+21]. Furthermore, the original TextGCN is inherently transductive, so it needs to be retrained whenever new documents appear. Strictly transductive models are effectively useless in practice [HHY+19] except for applications where a partially labeled corpus needs to be fully annotated. However, recent extensions such as HeteGCN, HyperGAT, and DADGNN already relax this constraint and enable inductive learning. However, word-document graphs require $\mathcal{O}(N^2)$ space, where N is the number of documents plus the size of the vocabulary, which is a hurdle for large-scale applications.

There are also tasks where the *natural* structure of the graph data provides more information than the mere text, e. g., citation graphs or connections in social graphs. In such cases, the performance of graph neural networks is state of the art [KW17; VCC+18] and is superior to MLPs that use only the node features and

not the graph structure [SMB+19]. GNNs also find application in various tasks NLP, in addition to pure classification [WCS+21].

Influence of Word Order and Positional Embeddings An interesting factor is the ability of the models to capture word order. BoW models disregard word order and still give good results, but fall behind order-sensitive Transformer models. In a detailed study, Conneau et al. [CKL+18] have shown that memorizing the word content (which words appear at all) is highly indicative of the performance on downstream tasks. Sinha et al. [SJH+21] have experimented with pretraining BERT by disabling word order during pretraining and show that it makes surprisingly little difference for fine-tuning. In their study, word order is preserved during fine-tuning. In our experiments, we have conducted complementary experiments: we have used a BERT model that is pretrained with word order, but we have deactivated the position encoding during fine-tuning. Our results show that the removal of position embeddings in BERT leads to a notable decrease in performance. This is presumably because the position embeddings were present during pretraining. Still, it should be noted that the model does not completely fail without position embeddings, e. g., it is still better than all other models on the MR task.

Other NLP tasks such as question answering [RZL+16] or natural language inference [WSM+19] can also be regarded as a text classification on a technical level. Here, the positional information of the sequence is more important than for pure topical text categorization, so that we expect that capturing word order is more important for these types of task.

Generalizability We expect similar observations to be made in other text classification datasets because we have already covered a variety of different characteristics: long and short texts, topical categorization (20ng, Reuters, and Ohsumed), and sentiment prediction (MR) in the domains of forum posts, news, movie reviews, and medical abstracts. Our results are in line with those of other fields, who have reported a resurgence of MLPs. For example, in business prediction, a MLP baseline outperforms various other Deep Learning models [VTF+21]. In computer vision, the models by Tolstikhin et al. [THK+21] and Melas-Kyriazi [Mel21] with attention-free MLPs are on par with the Vision Transformer [DBK+21]. In natural language processing, Liu, Dai, So, and Le [LDS+21] show similar results, while acknowledging that a small attention module is necessary for some tasks. These findings point in a similar direction, even though the pure MLP-based Transformer

replacements are conceptually different (position aware) from the *bag-of-words* MLP that we considered for text classification.

Threats to Validity We acknowledge that the experimental datasets are limited to English. Although word order is important in the English language, it is noteworthy that methods that disregard word order still work well for text categorization. Another possible bias is the comparability of the results. However, we carefully checked all relevant parameters, such as the train/test-split, the number of classes in the datasets, if the datasets have been preprocessed in such a way that, e. g., makes the task easier, such as reducing the number of classes, the training procedure, and the reported evaluation metrics. Regarding our efficiency analysis, we make sure to report the numbers for the parameter count *and*, a measure of speed other than FLOPs, as recommended by Dehghani et al. [DAB+21]. Since runtime is heavily dependent on training parameters, such as batch size, we complement this with asymptotic complexity.

Practical Impact Our findings have an immediate impact on practitioners who seek to employ robust text classification models in research projects and in industrial operational environments. Furthermore, we advocate the use of a MLP baseline in future text classification research, for which we provide concrete guidelines in Appendix D.1.

5.7 Summary

We argue that a wide multilayer perceptron enhanced with current best practices should be considered as a strong baseline for text classification tasks. In fact, our experiments show that our WideMLP is often on par or even better than recently proposed models that synthesize a graph structure from the text. Based on the strong performance of models with a BoW input representation, we will use this representation in subsequent chapters.

Chapter 6

Multimodal Autoencoders for Document-based Recommendations

In this chapter, we seek to answer *Q4: How can we design multimodal representation learning models that jointly process text and graph data for document-based recommendation tasks?* This chapter is based on material published in the Proceedings of the 26th Conference on User Modeling, Adaptation and Personalization, UMAP 2018 [GMV+18] and an extension published in the Springer Information Retrieval journal [VGS22].

We investigate to what extent side information in the form of textual bibliographic metadata improves the performance of recommender systems. In particular, we investigate recommendation tasks, in which a document is the sole basis for recommended items, which we call **document-based recommender systems**.

Applications of document-based recommender systems include multi-label classification [TK07; GMS+17; MGS18; NMK+17], a.k.a. subject indexing in the digital library community [ISO96], document-level citation recommendation [CSM+13; EF17; ZYW21], and research paper recommendation [BGL+16; RFP17]. A successful recommender system for these tasks helps researchers in finding relevant literature. In particular, we choose two scenarios, which fall in the category of document-based recommender systems: citation recommendation and subject label recommendation.

We choose citation and subject-label recommendation as our scenarios because of the difference in what it means when two items occur together. In the citation scenario, when two papers are cited together, these two papers are related to each other [Sma73]. On the contrary, in the subject indexing scenario, the professional librarians attach labels to the documents that represent the different covered topics. In other words, the labels of a single document are dissimilar. These two considered tasks are two extrema with regard to the degree of similarity within item sets. In other tasks, such as the recommendation of movies or music, the similarity of the items is more mixed. Users can like very different genres, such as jazz and electronic music, but the songs within each genre are rather similar. In [VGM+18],

we explored automatic playlist continuation as one of such tasks with the same methods that we use throughout this chapter.

Challenges of Document-based Recommendation Tasks A particular challenge in these recommendation tasks is that it is desirable that only the document itself is the basis for recommendations. In other recommendation systems, the user and its metadata are the central factor for recommendations (e. g., "What articles does the user know already?"). However, the researcher might very well be involved in the writing of numerous different papers with different topics, which supports the analogy that we regard the documents themselves as pseudo-users [MAC+02]. Furthermore, professional subject indexers presumably know all candidate subjects but need to know which fits best for the current paper in question. Therefore, we ensure that all subject indexers receive the same recommendation for a given document, and all coauthors of a paper draft receive the same recommendations for articles to cite. We call these tasks **document-based recommendation tasks** because the document, along with its metadata, is the basis for recommendations.

Therefore, it is crucial in such document-based recommendation tasks that the system is able to provide recommendations for documents that were previously unknown to the system, known as the **cold start** problem [SCP+19]. In recommender systems, cold start refers to the problem of having insufficient information about users or items to make accurate recommendations. A subproblem of the cold start problem is the new user problem, which corresponds, in our document-based recommendation tasks, to the problem of dealing with new documents after training. This is a critical factor because the drafts that are being written are not yet available at training time. Similarly, subject indexers usually annotate new documents.

Autoencoders as Recommendation Models A commonly used model for recommender systems is the autoencoder (AE) [LS17; LKH+18], sometimes also enriched with side information [SGM16; MJ17], where side information refers to additional user or item data aside from the ratings. AEs capture patterns in the training data by learning to reconstruct their input. The input here is the item set, and the reconstructed item set then determines the recommendations while removing any items that were already present in the input.

However, previous autoencoder approaches, and also more general approaches such as IRGAN [WYZ+17], struggle to deal with new unseen documents. This is

because the methods rely on learning an embedding for each user during training and there would be no such embedding for new users. In our experiments on document-based recommendation, all test documents are unseen by the system during training, whereas in other recommendation scenario, the assumption is that only a fraction of the ratings matrix remains unseen. This presents a special challenge, which we address by building autoencoder systems that rely only on the document and its metadata.

To alleviate the cold start problem, we extend the autoencoders to take multi-modal input so that they become hybrid recommenders that can take advantage of both the ratings in terms of the set of partial item set and the content in terms of additional metadata. These extended models can receive different metadata fields as side information, such as the title of the document, the authors, and the venue.

We compare the autoencoders with three strong baselines for a fair evaluation and to address the question of whether neural architectures are making progress on the considered recommendation tasks [DCJ19]. As baselines, we have selected an Item Co-occurrence (IC) method inspired by the Co-Citation Score [Sma73], which is reminiscent Common Neighbors from link prediction [New01; BJN+02]. We provide an efficient implementation for IC via matrix multiplication. Second, we have a singular value decomposition (SVD) matrix factorization approach [CSM+13], which we extend to also take into account the textual side information. Finally, an MLP motivated by its success in the previous Chapter 5 on text classification and in prior work [GMS+17; MGS18].

Evaluation We evaluate the models on two recommendation tasks (citations and subject labels) with three different options for side information (none, only titles, all metadata) together with the item set (either citations or subject labels). We support each task with 3 different datasets, which makes up a total of 6 datasets from five domains: medicine, computer science, economics, politics, and news. In addition to evaluating only previously unseen test documents, i. e., a pure cold start evaluation, we also conduct a *chronological* split between training and test data. The split along the time axis is especially important, since only already existing papers can be cited, and thus recommended for a citation. Similarly, subject labels suffer from concept drift [WHC+16], so a chronological split is more representative of real-world applications. Moreover, we provide an extensive analysis of two experimental conditions: dataset pruning, which controls the size of the dataset by setting a minimum threshold on the number of times an item occurs, and a drop parameter, which controls the size of the input item sets, i. e., the stage of

the iterative recommendation process. This effectively creates a new version of each dataset for each value of the pruning and the drop parameter. To ensure reproducibility with respect to the models' random initialization and the dropped items within dataset preparation, we repeat each experiment three times for each configuration of models and the datasets including their variations in terms of pruning and drop parameter.

Results The results show that side information improves the recommendations of autoencoder-based models for both citations and subject labels. We observe that autoencoder models even *need* the side information to compete with strong collaborative filtering baselines. On the subject label side, we find that a simple MLP, a pure content-based approach, yields the best results without using the ratings matrix at all. We explain this phenomenon by the meaning of item co-occurrence, which is relatedness in co-citations and diversity in co-annotations. When item co-occurrence resembles relatedness or similarity, the ratings are more important than when item co-occurrence resembles diversity or dissimilarity. Intuitively, it is easier to predict similar items on the basis of others, while finding items that *complement* an existing set is more difficult solely on the basis of other items. Instead, it is easier to consult the side information, e. g., the document's title, to find such complementing items. We conclude that the meaning of item co-occurrence is a critical factor for the choice of an appropriate model.

By systematically analyzing the influence of dataset pruning, we confirm that the ranking between methods remains stable. However, absolute recommendation performance increases with a smaller set of possible items, i. e., with a lower degree of sparsity. For the item set size, we find that the recommendation performance is best when a medium amount of items is available in the input, while a medium amount of items are still to be recommended. Still, we found that VAEs deviate from this behavior and provides comparatively good recommendations on small item sets. This may serve as an explanation for why VAEs [KW14] are among the most popular models in recommender systems [LS17; LKH+18; CR18; BB19].

The remainder of this chapter is structured as follows. In the following, we review previous work on recommender systems with a focus on autoencoder-based approaches and methods for citation and subject recommendation. Subsequently, a formal problem statement is provided in Section 6.2. In Section 6.3, we introduce the multimodal autoencoder models and the three baselines, before describing our datasets in Section 6.4. The experiments are described in Section 6.5, the results of which are discussed in Section 6.6, before we summarize the chapter.

6.1 Related Prior Work

Recommender systems are typically separated into three types: **Collaborative filtering** approaches operate only on the rating matrix between users and items [FJN+13]. Approaches that provide recommendations based on item or user content are called **content-based** [LGS11] recommender systems. **Hybrid recommender systems** [ÇM17; LWK+18] combine these two strategies and consider content and user-item ratings. In hybrid recommender systems, one can further distinguish between loose coupling and tight coupling [WWY15]. In loose coupling, collaborative filtering and content-based models are separate and their final output is combined, whereas in tight coupling, a joint model operates on both input modalities.

When assuming that the documents serve as pseudo-users, our approaches behave as collaborative filtering when only the item set is used, while they are content-based when only the side information is used, and they behave as hybrid approaches when both the item set and side information are used. We now review previous work on recommender systems with a focus on autoencoder-based approaches. Then, we discuss methods and systems specifically designed for citation and subject-label recommendation.

6.1.1 Autoencoders as Recommendation Engines

Autoencoder (AE) have recently gained great popularity for recommendation tasks [PHY20; LKH+18; Ste19; CYL17; ZZQ+17; SMS+15], often using side information [BB19; HMZ19; CR18; LWK+18; WWY15; MJ17; LS17; ZYX+17; SGM16; LKF15]. AE are especially suitable for collaborative filtering and content-based recommendations with tight coupling [BB19; WWY15; LS17]. A common strategy is to combine the encoded features of the items with a latent factor model of the ratings [BB19; HMZ19; LS17]. Some works also fuse user/item side information with the respective rows/columns of the rating matrix as input to the AE [CR18; MJ17]. Majumdar and Jain [MJ17] investigated the pure cold start problem and the partial cold start problems, in which they assume that 10% or 20% of the ratings are present.

Regarding model architectures, the variational autoencoder (VAE) [KW14] has recently been used more frequently for recommendation tasks [LKH+18; BB19; HMZ19; CR18; LS17], although other work has used (stacked) denoising autoencoder [LWK+18; WWY15; LKF15]. AEs find widespread applications in recommender systems. The most prominent variants [LKH+18; LS17] stood the test

of a reproducibility study of deep learning recommender systems [DCJ19]. The study has shown that all the autoencoder variants tested, namely Collaborative VAE [LS17] and Mult-VAE [LKH+18] could be reproduced.

Compared to previous work, we focus on using AE for the new user problem [SCP+19] as modeled by a chronological split of the datasets along the time axis. In contrast, most of the existing literature assumes that all users are already present during training. Furthermore, we explicitly investigate the influence of different meanings of item co-occurrence on the recommendation performance. We also consider different completeness levels of the partial set in our experiment, beginning from early to mid-late stages of the two scenarios of citation and subject-label recommendation.

6.1.2 Research Paper and Citation Recommendation

Research paper recommendation is a well-known and popular topic [BGL+16; AUK+21]. Specifically, in citation recommendation [FJ20], recommendations are distinguished between recommendations based on a partial set of references and recommendations based on the content of a manuscript [HKC+12]. While the former aims at identifying missing citations on the broader document level, the latter is suited to find matching citations for a given statement during writing. Citation recommendation recently focuses on these context-sensitive applications, in which concrete sentences need to be assigned to preferably relevant citations [MZL20; EF17; ZYW21; BGL+16; HKC+12]. Instead, we revisit the problem of completion of the reference set and do not take into account the context of the citation, as the full text of a paper is rarely available in large-scale metadata sources [MGS18]. Co-citation analysis assumes that two articles are more related to each other, the more they are co-cited [Sma73]. Following this idea, Caragea, Silvescu, Mitra, and Giles [CSM+13] used singular value decomposition as a more efficient and extendable approach to citation recommendation. Other approaches make use of deep learning techniques for citation recommendation but focus on context-sensitive scenarios [EF17; HWL+15; SAH20; ZM20; TSZ+20; CYL+20; AKM+20]. We recognize the need for new methods on the document level that are not only based on item co-occurrence but also consider additional metadata for document-based recommendations.

Other approaches in the areas of network analysis include node embeddings [PAS14; GL16], link prediction with graph neural networks [KW16; ZC18], and dynamic graph representation learning [KZL19]. However, these methods and also

retrieval methods such as IRGAN [WYZ+17] do not apply to our problem because they require that all documents are known in advance, while our methods need to be applicable to new unseen data without further training. In graph representation learning, the former is known as transductive learning, while the latter is called inductive learning (cf., Section 2.4). Here, we assume that the test documents are completely unseen during training, i. e., an inductive learning problem. We need to predict citations for a paper at the time of writing that are not available during model training. Such a scenario is particularly challenging, as it corresponds to having only new users at test time.

6.1.3 Subject Label Recommendation

Subject label recommendation is similar to tag recommendation. In both cases, the goal is to suggest a descriptive label for some content. Boughareb et al. [BKB+20] show an approach to recommend tags for scientific papers, which defines the relatedness between the tags assigned by users and the concepts extracted from the available sections of scientific papers based on statistical, structural, and semantic aspects. Sun et al. [SZJ+21] presented a hierarchical attention model for personalized tag recommendation. Lei, Fu, Yang, and Liang [LFY+20] introduce a tag recommendation by text classification that uses the capsule network with dynamic routing for tag recommendation. The capsule network encodes the intrinsic spatial relationship between a part and a whole, constituting viewpoint-invariant knowledge that automatically generalizes to novel viewpoints. Zhou, Xia, Wan, and Zhang [ZXW+20] propose a hybrid method based on multimodal content analysis, in which keywords are recommended to compose titles and tags for video upload. They combine textual semantic analysis of original tags and recognition of video content with deep learning. Sen, Vig, and Riedl [SVR09] proposed algorithms that predict users' preferences for items based on their inferred preferences for tags. Montañés, Quevedo, Díaz, and Ranilla [MQD+09] exploited probabilistic regression for collaborative tag recommendation, while Krestel, Fankhauser, and Nejdl [KFN09] relied on latent Dirichlet allocation. Similarly, Sigurbjörnsson and Zwol [SZ08] proposed a tag recommender for Flickr to support the user in the photo annotation task, while Posch, Wagner, Singer, and Strohmaier [PWS+13] predicted hashtag categories on Twitter. Dellschaft and Staab [DS12] measure the influence of tag recommenders on indexing quality in collaborative tagging systems. While these works focus on tags for social media, we consider subjects from a standardized thesaurus for scientific documents. The recommendation of subject

labels is also related to social tagging systems, such as [DZS+16; WSS+14], in which users share content-dependent tags with each other.

Furthermore, tag or label recommendation is related to the problem of multi-label classification. Multi-label classification typically requires a hard decision per label [GMS19]. When considering a recommendation task, we instead evaluate the ranking of recommended items [TZY+08]. Tsoumakas, Katakis, and Vlahavas [TKV11] propose to use random subsets of label permutations. Zhang and Zhou [ZZ14] give an overview of approaches to multi-label learning. Zhang and Wu [ZW15] exploit label-specific features to discriminate between classes. Nam, Mencía, Kim, and Fürnkranz [NMK+17] propose an encoder-decoder architecture based on sequence models as a special case of the generic set2set method [VBK16]. Here, the sequential decoder can take into account the dependencies between the labels, similar to classifier chains [RPH+11]. They extended this approach with an expectation maximization model to provide context-dependent label permutations in sequential mode [NKM+19].

In our previous work on multi-label classification, we analyzed the influence of using title-only versus full-text input data on scientific and news corpora [GMS+17]. We found that using only the title retains 90% of the accuracy obtained by using the full text. Among many baselines, a wide MLP with a single hidden layer is preferable. Furthermore, we showed that using only titles can be even better than using the full text for multi-label classification tasks, when more training data is available [MGS18]. This motivates us to use additional bibliographic metadata in the two recommendation scenarios, rather than the full text, which is often not available even in open access datasets [MGS18].

6.1.4 Summary

Autoencoder have been shown to be effective in recommendation tasks and have been widely used. To our knowledge, no previous work has studied in detail the recommendation of items at different stages of completeness of the input item set, i. e., with a varying number of items in the input and in the new user settings. Furthermore, we consider the differences in the meanings of item co-occurrence in the two recommendation scenarios of citations and subject labels, as well as different degrees of sparsity and the influence of using metadata.

6.2 Problem Formulation

We first describe two scenarios and how to incorporate them into a common framework for document-based recommendations with side information. Subsequently, we provide a formal problem statement for the task and describe its evaluation.

6.2.1 Scenarios and Common Framework

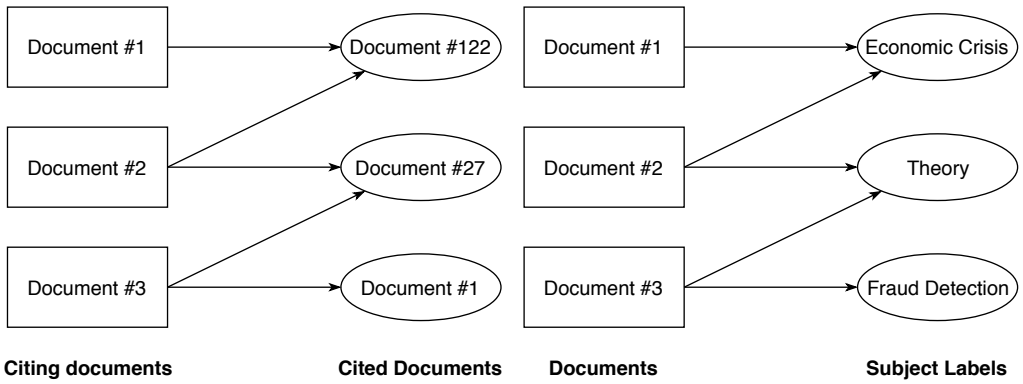
Scenario 1: Citation Recommendation When writing a new article, it is essential that the authors refer to other publications that are key in the respective field of study or relevant to the article being written. Reviewers can negatively rate failure to do so in a peer-reviewing process. However, due to the increasing volume of scientific literature, even some key papers are sometimes overlooked. We address this challenge by studying the problem of recommending publications for consideration as citation candidates. Given that the authors have already selected some references for their scientific paper, we use these references as input for the recommendations of other candidate documents.

Using the set of documents already cited comes with a further challenge. At the initial stage of writing, the set has only few elements, making it difficult to infer the topic of the draft, but there are more opportunities to make a "good" prediction. At a later stage, the set has grown, and, while inferring the topic becomes easier, finding the missing item becomes more difficult. In our experiments, we explicitly consider the influence of the degree of completeness of the input set on the recommendation performance.

Scenario 2: Subject Label Recommendation Subject indexing is a common task in scientific libraries to make scientific documents accessible for search. New documents are manually annotated by qualified subject indexers with a set of subject labels, i. e., categories from a, typically domain-specific, thesaurus. Fully automated multi-label classification approaches to subject indexing are promising [NMK+17], even when only the title metadata of the publication is used [GMS+17]. Still, professional subject indexers use the result of these approaches only as recommendations such that human-level quality can be guaranteed. This motivates us to investigate subject labeling from a recommender system perspective.

Similarly to the citation task, professional subject indexers add labels one by one to the publications. Following internationally standardized guidelines [ISO96], the indexers ensure that the labels are covering the diverse scientific aspects,

contributions, and methods of the papers. To this end, the indexers scan the paper's title, section headings, and partially read their content. Thus, properly indexing a scientific paper with subject labels is a difficult task, which can be well supported by a recommender system. The recommender takes the set of already assigned subject labels as input and produces recommendations for new labels that refer to aspects of the paper that are not yet covered. Following the same experimental approach as for the citation task, we explicitly evaluate the influence of the different levels of completeness of the already assigned subject labels on the recommendation performance. Thus, we use different sizes of the input set of labels and measure the quality of the recommendations.



(a) The citation recommendation scenario. **(b)** The subject indexing scenario.

Figure 6.1. Exemplary bipartite graphs of citation relationships between documents (left) and documents annotated with subject labels (right).

Common Framework for Document-based Recommendation Tasks Figure 6.1 shows how the citation recommendation and the subject label recommendation share the structure of a bipartite graph. In both cases, the document is linked to multiple *items*, which are either cited documents in the one case and assigned subject labels in the other. Thus, the task of recommending such items can be regarded as predicting links in the bipartite document-item graph.

6.2.2 Formal Problem Statement

Traditionally, the recommendation problem is modeled as a matrix completion problem. The goal is to predict missing ratings in a $\mathbb{U} \times \mathbb{I}$ matrix, where $\mathbb{U}$ is

Table 6.1. Notation table

Symbol	Description
$\mathbb{D}$	Set of m documents
$\mathbb{I}$	Set of n items
$\boldsymbol{X} \in \{0,1\}^{m \times n}$	Sparse ratings matrix
$\boldsymbol{S} \in \mathbb{R}^{m \times d}$	Side information from document metadata
$\boldsymbol{x}, \boldsymbol{s}$	Row vectors of $\boldsymbol{X}$ or $\boldsymbol{S}$, respectively
$\boldsymbol{x} \parallel \boldsymbol{s}$	Concatenation of vectors $\boldsymbol{x}$ and $\boldsymbol{s}$
$\bowtie$	Natural join (on document identifiers)
$\boldsymbol{I}$	Identity matrix

the set of users and $\mathbb{I}$ is the set of items. As illustrated in Figure 6.1, for citation recommendation (Scenario 1), the users are the newly authored papers and the recommended items are the papers that are recommended to be cited. While in subject label recommendation (Scenario 2), the users are the newly annotated papers and the items are recommended subject labels. As in previous work [MAC+02], we consider a matrix $\mathbb{U} \times \mathbb{I}$, where the set of users is in our context the set of documents, i. e., the training corpus of documents. The rationale is that in Scenario 1, all authors for a given paper should receive the same recommendations of which papers to cite. Analogously for Scenario 2, a given paper should receive the same recommendations for adding subjects labels, independently of the current subject indexer responsible for annotating it.

Document-based Recommendations Given a set of m documents, $\mathbb{D}$, and a set of n items, $\mathbb{I}$, the typical recommendation task is to model the spanned space, $\mathbb{D} \times \mathbb{I}$. We model the ratings as a sparse matrix $\boldsymbol{X} \in \{0,1\}^{m \times n}$, in which X_{jk} indicates implicit feedback from document j to item k. To simulate a real-world scenario, we split the documents $\mathbb{D}$ into m_{train} documents for training, $\mathbb{D}_{\text{train}}$, and m_{test} documents for evaluation, $\mathbb{D}_{\text{test}}$, such that $\mathbb{D}_{\text{train}} \cap \mathbb{D}_{\text{test}} = \varnothing$. More precisely, we conduct this split into training and test documents based on the publication year. All documents that were published before a certain year are used as training, and the remaining documents are used as test data. This leads to an experimental setup that is close to real-world applications of citation recommendation and subject-label recommendation. All models are supplied with the users' ratings $\boldsymbol{X}_{\text{train}} = \mathbb{D}_{\text{train}} \bowtie \boldsymbol{X}$ along with the side information $\boldsymbol{S}_{\text{train}} = \mathbb{D}_{\text{train}} \bowtie \boldsymbol{S}$ for training. As side information $\boldsymbol{S}$, we use the documents' various metadata fields

such as title, authors, and venue. The test set $\boldsymbol{X}_{\text{test}}, \boldsymbol{S}_{\text{test}}$ is obtained analogously. A summary of the notation used in this chapter can be found in Table 6.1.

Evaluation For evaluation, we remove randomly selected items in $\boldsymbol{X}_{\text{test}}$ by setting a fraction of non-zero entries in each row to zero. We denote the hereby created test set by $\tilde{\boldsymbol{X}}_{\text{test}}$. The model ought to predict values $\boldsymbol{X}_{\text{pred}} \in \mathbb{R}^{m_{\text{test}} \times n}$, given the test set, $\tilde{\boldsymbol{X}}_{\text{test}}$, along with the title information, $\boldsymbol{S}_{\text{test}}$. Finally, we compare the predicted scores, $\boldsymbol{X}_{\text{pred}}$, with the true ratings, $\boldsymbol{X}_{\text{test}}$, via ranking metrics. The goal is that those items, that were omitted in $\tilde{\boldsymbol{X}}_{\text{test}}$ are highly ranked in $\boldsymbol{X}_{\text{pred}}$.

Note that a research paper is usually both a cited and a citing paper, which is modeled here through separate roles. We follow this approach because the number of citations that point out of the dataset is so high that there are an order of magnitude more cited papers than citing papers. For example, the PubMed citation dataset has 224,092 documents that cite 2,896,764 other distinct documents. Therefore, it is reasonable to distinguish documents based on their role in the citation relationship, i. e., citing versus cited paper. A different perspective will be considered in Chapter 7, in which the cited and cited research papers are treated in an interconnected graph.

6.3 Methods

Autoencoder (AE) are well suited for our task as they learn to reconstruct their input. We make use of this capability to produce recommendations. In the following, we describe the employed models. We first start by recapturing two well-known baselines, singular value decomposition and item co-occurrence. Then, we describe the employed AE base models and introduce a generic mechanism to make use of the side information in AEs.

6.3.1 Singular Value Decomposition

We use Singular value decomposition (SVD) to factorize the co-occurrence matrix of items $\boldsymbol{X}^T \cdot \boldsymbol{X}$. Caragea, Silvescu, Mitra, and Giles [CSM+13] have successfully used SVD for citation recommendation. We include an extended version of SVD in our comparison, which can incorporate textual side information [GMV+18]. We concatenate the textual features as TF-IDF weighted bag-of-words (see Section 2.2) with the items and perform a singular value decomposition on the resulting matrix.

As the final predictions, we use only those indices of the reconstructed matrix that are associated with items.

6.3.2 Item Co-Occurrence

With the goal of developing a strong non-parametric baseline, we take inspiration from Small [Sma73]'s co-citation score and early work in document-level citation recommendation [MAC+02]. The co-citation score counts how often two documents have been cited together, indicating the similarity of the two documents. McNee et al. [MAC+02] have used this co-citation score together with other collaborative filtering approaches in a recommender scenario. The rationale is that two papers, which have been often cited together in the past, are more likely to be cited together in the future.

We generalize this method to arbitrary recommender scenarios by operating on the ratings matrix $\boldsymbol{X}$. Given training data, $\boldsymbol{X}_{\text{train}}$, we compute the full item co-occurrence matrix $\boldsymbol{C} = \boldsymbol{X}_{\text{train}}^T \cdot \boldsymbol{X}_{\text{train}} \in \mathbb{R}^{n \times n}$. Each element C_ij holds the number of times that item i has co-occurred with item j. On the diagonal of $\boldsymbol{C}$, we have the total number of times that each item has occurred. At prediction time, we obtain the scores by aggregating the co-occurrence values via matrix multiplication $\boldsymbol{X}_{\text{test}} \cdot \boldsymbol{C}$. Note that the diagonal of $\boldsymbol{C}$ only affects the predicted scores of those items already present in the input, and thus does not alter the list of recommended items. All matrix multiplications can be performed in a sparse format for more efficient computation.

This item co-occurrence method is reminiscent of the Common Neighbors method for link prediction in network data, in which the score is computed as the number of common neighbors of any two nodes [New01]. In our terms, that would correspond to two items having a document in common, i. e., co-occur in an item set. Since we are considering recommender systems, we denote this method as item co-occurrence (IC).

6.3.3 Multilayer Perceptrons

A multilayer perceptron (MLP) is a fully-connected feedforward neural network with one or multiple hidden layers (Figure 6.2). The output is calculated by consecutive applications of $\boldsymbol{h}^{(i)} = \sigma(\boldsymbol{h}^{(i-1)} \cdot \boldsymbol{W}^{(i)} + \boldsymbol{b}^{(i)})$, with σ being a nonlinear activation function. In the description of the following models, we abbreviate a two-hidden-layer perceptron module by MLP-2.

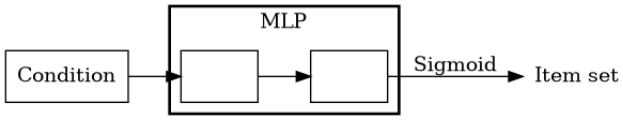


Figure 6.2. A multilayer perceptron (MLP) with two hidden layers

Furthermore, we use an MLP-2 as a standalone recommendation engine, which only operates on document metadata. In this case, we optimize binary cross-entropy $\mathrm{BCE}(\boldsymbol{x}, \mathrm{MLP\text{-}2}(\boldsymbol{s}))$, where the title and other metadata $\boldsymbol{s}$ are used as input and citations or subject labels $\boldsymbol{x}$ as target output. We use dropout [SHK+14] and Adam optimizer [KB15]. For the purpose of a fair comparison, the MLP operates on the same input representation, $\boldsymbol{s}$, which we will also use as the representation of side information for the autoencoders, which are described below.

6.3.4 Undercomplete Autoencoders

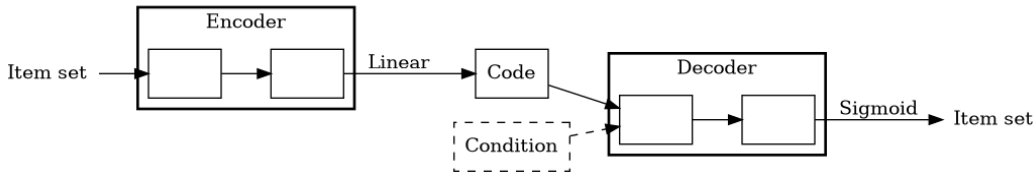


Figure 6.3. Undercomplete autoencoder

An autoencoder (AE) has two main components: the encoder enc and the decoder dec (Figure 6.3). The encoder transforms the input $\boldsymbol{x}$ into a hidden representation, the code $\boldsymbol{z} = \mathrm{enc}(\boldsymbol{x})$. The decoder aims to reconstruct the input from the code $\boldsymbol{r} = \mathrm{dec}(\boldsymbol{z})$. The two components are jointly trained to minimize the binary cross entropy, $\mathrm{BCE}(\boldsymbol{x}, \boldsymbol{r})$. To avoid learning to simply copy the input $\boldsymbol{x}$ to the output $\boldsymbol{r}$, autoencoders need to be regularized. The most common way to regularize autoencoders is by imposing a lower dimensionality on the code (undercomplete autoencoder). In short, autoencoders are trained to capture the most important explanatory factors of variation for reconstruction [BCV13]. For both the encoder and the decoder we chose an MLP-2 module, such that the model function becomes

$$\boldsymbol{r} = \mathrm{MLP\text{-}2}_{\mathrm{dec}}(\mathrm{MLP\text{-}2}_{\mathrm{enc}}(\boldsymbol{x}))$$

6.3.5 Denoising Autoencoders

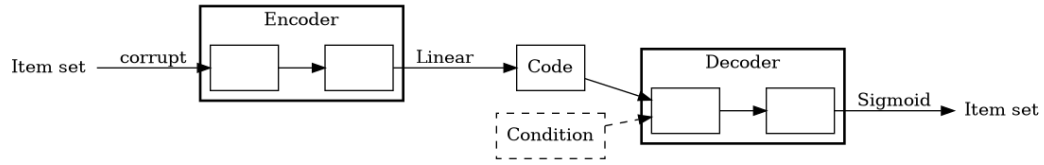


Figure 6.4. Denoising autoencoder (DAE)

A denoising autoencoder (DAE) is an autoencoder that receives corrupted data as input and aims to predict the original data [VLB+08] (Figure 6.4). During training, the initial input $\boldsymbol{x}$ is corrupted into $\tilde{\boldsymbol{x}}$ through stochastic mapping $\tilde{\boldsymbol{x}} \sim q_D(\tilde{\boldsymbol{x}}|\boldsymbol{x})$, e. g., randomly forcing a fraction of entries to zero (white noise). The corrupted input $\tilde{\boldsymbol{x}}$ is then mapped to a hidden representation (the code) $\boldsymbol{z} = \text{enc}(\tilde{\boldsymbol{x}})$ in the same way as the standard undercomplete autoencoder, and from the hidden representation the model reconstructs $\boldsymbol{r} = \text{dec}(\boldsymbol{z})$. Both the encoder and the decoder rely on a MLP-2 module, such that the model function becomes $\boldsymbol{r} = \text{MLP-2}_{\text{dec}}(\text{MLP-2}_{\text{enc}}(\tilde{\boldsymbol{x}}))$. The loss function is again binary cross-entropy.

6.3.6 Variational Autoencoders

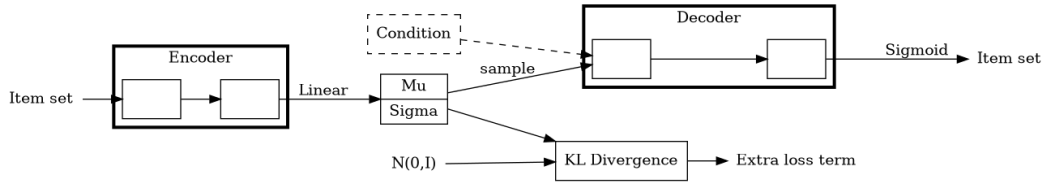


Figure 6.5. Variational autoencoder (VAE)

A variational autoencoder (VAE) is a generative model whose posterior is approximated by a neural network, forming an autoencoder architecture [KW14], as shown in Figure 6.5. VAEs use a variational approach to learn latent representations, i. e., minimize the empirical lower bound of the data distribution, in which it is assumed that the underlying data-generating distribution $p_{\text{data}}(\mathbf{x}|\mathbf{z})$ is a mixture of latent variables $\boldsymbol{z}$. The encoder $p_{\text{encoder}}(\mathbf{z}|\mathbf{x})$ learns to infer these latent variables, while the decoder $p_{\text{decoder}}(\mathbf{x}|\mathbf{z})$ learns to generate samples from these latent variables. The goal is that $p_{\text{decoder}}(\boldsymbol{x}|\boldsymbol{z})$ approximates p_{data} after successful

training. A crucial component of VAEs is the reparametrization trick that facilitates backpropagation through random operations [KW14]. Given a prior distribution $p_{\text{prior}}(\mathbf{z})$ on the code $\mathbf{z}$, a deterministic encoder learns to predict the parameters of this distribution. The prior distribution $p_{\text{prior}}(\mathbf{z})$ of a standard VAE is Gaussian [KW14]. The deterministic output of the encoder $\text{enc}(\boldsymbol{x})$ is first divided into two halves: one for the mean, μ, and one for the standard deviations, σ. Then we independently sample $\epsilon \sim \mathcal{N}(0,1)$ and compose the code as $z = \mu + \sigma\epsilon$ to be fed into the deterministic decoder. To encourage a normal distribution on the code, the Kullback-Leibler (KL) divergence with respect to $\mathcal{N}(0,1)$ is added as an additional loss term. Intuitively, the encoder learns how much noise to inject at the code level. We use an MLP-2 in both the encoder and the decoder and optimize the reconstruction loss (by binary cross entropy) along with the KL divergence term.

6.3.7 Adversarial Autoencoder

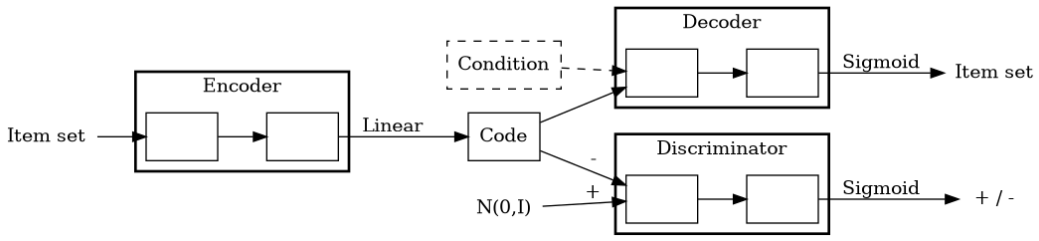


Figure 6.6. Adversarial autoencoder (AAE)

Adversarial autoencoders (AAEs) [GMV+18; MSJ+16] combine generative adversarial networks [GPM+14] with AEs (Figure 6.6). The AE component reconstructs the sparse item vectors, while the discriminator distinguishes between the generated codes and samples from a selected prior distribution. Therefore, the selected prior distribution shapes the distribution of the latent code. Although this is similar to the VAE, which uses an explicit Gaussian prior, the AAE provides more flexibility in the choice of the prior distribution.

The latent representations learned by distinguishing the code from a smooth prior lead to a model that is more robust to sparse input vectors than undercomplete AEs because smoothness is the main criterion for good representations that disentangle the explanatory factors of variation [BCV13]. Formally, we first

compute $\boldsymbol{z} = \text{MLP-2}_{\text{enc}}(\boldsymbol{x})$ and $\boldsymbol{r} = \text{MLP-2}_{\text{dec}}(\boldsymbol{z})$ and then update the parameters of the encoder and the decoder with binary cross-entropy, $\text{BCE}(\boldsymbol{x}, \boldsymbol{r})$.

Therefore, in the regularization phase, we draw samples $\tilde{\boldsymbol{z}} \sim \mathcal{N}(0, \boldsymbol{I})$ from independent Gaussian distributions matching the size of $\boldsymbol{z}$. The parameters of the discriminator $\text{MLP-2}_{\text{disc}}$ are then updated, to minimize $\log \text{MLP-2}_{\text{disc}}(\tilde{\boldsymbol{z}}) + \log(1 - \text{MLP-2}_{\text{disc}}(\boldsymbol{z}))$ [GPM+14].

The encoder parameters are updated to maximize $\log \text{MLP-2}_{\text{disc}}(\boldsymbol{z})$, such that the encoder is trained to fool the discriminator. Thus, the encoder is jointly optimized to match the prior distribution and reconstruct the input [MSJ+16]. At prediction time, we perform one reconstruction step by applying one encoding and one decoding step.

6.3.8 Conditioning Autoencoder on Side Information

An advantage of AE-based recommender systems is that AEs can be conditioned on side information. This conditioning may be performed with different strategies, e. g., by providing the side information as an additional input [BAB+17]. In this work, we chose to impose the condition $\boldsymbol{s}$ on the code level of the AE. The rationale for this strategy is that the side information may help the decoder reconstruct the item set. This strategy is akin to the supervised case of the original work on AAEs that used classes as a condition for reconstructing images [MSJ+16], and has the advantage that we can apply it in the same way to all AE variants without further adaptions specific to an individual variant (such as disabling DAE's noise or modifying the VAE's KL divergence objective). Thus, the encoder still operates solely on the partial item set while the decoder is conditioned on side information to reconstruct the original item set: $\boldsymbol{r} = \text{dec}\left(\text{enc}(\boldsymbol{x}) \parallel \boldsymbol{s}\right)$, where $\cdot \parallel \cdot$ denotes concatenation. Furthermore, the training objective remains to optimize the reconstruction of the item set rather than the reconstruction of the side information. In practice, we embed the document titles into a lower-dimensional space by using pre-trained word embeddings such as word2vec [MSC+13]. More precisely, we use a TF-IDF-weighted bag of embedded words representation that was shown to be useful for information retrieval [GSS17]. We use the same strategy for journal names, as they often contain indicative words. For authors, we learn a categorical embedding from scratch: we optimize a randomly initialized embedding vector for each author during training. In our scenarios, the side information is composed of the title of the documents, the name of the journal, and the authors. We consider three cases for our experiments: (1) no conditioning, (2) conditioning on the title, and

(3) conditioning on all metadata. When multiple conditions are used, we combine them again by concatenation: $s = s_{\text{title}} \parallel s_{\text{author}} \parallel s_{\text{journal}}$. In our experiments, we evaluated these three cases with all the AE variants described above.

For the title data s_{title}, we use a TF-IDF weighted embedded bag-of-words representation [GSS17], which we have also used as input for the standalone MLP recommendation engine. For the journal data s_{journal}, we use the same technique, because journal names often contain indicative terms, while less indicative terms receive a low weight by TF-IDF. For author data s_{author}, which is a categorical attribute, we seek a different strategy: we employ an embedding table such that each author is associated with its own learnable embedding vector.

6.4 Datasets

We consider six datasets for our experiments. Three datasets comprise scientific publications in the domains of medicine and computer science for the citation recommendation task. We also used three datasets from the domains of economics, politics, and news for the subject-label recommendation task.

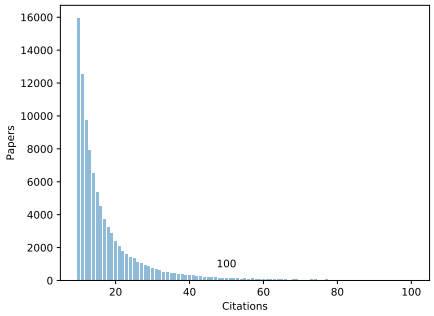
Power Law Coefficient To characterize the six different datasets, we use the power law coefficient. The power-law coefficient allows us to assess how skewed the distribution of documents' citations or subjects is. As such, we estimate the power law coefficient a through maximum likelihood [New05]: $a = 1 + n\left(\sum_{u \in V} \ln \frac{\deg_u}{\deg_{\min}}\right)^{-1}$, where $\deg_{\min}$ is equal to 1. Still, the pure value of the power-law coefficient does not tell us whether there is a power-law in the underlying distribution. Therefore, we also plot the distribution of items (i. e., citations or labels) for visual inspection [New05].

Mutual Information (MI) We further compute the mutual information among the items in each dataset. This quantifies the information that each item conveys about other items. Mutual information (MI) can be computed as the KL-Divergence of the joint distribution with respect to the product of marginals: $\mathrm{I}(X;Y) = D_{KL}(p(x,y)||p(x)p(y))$ [CT06]. The distribution $p(\mathbf{x}, \mathbf{y})$ models the number of documents in which two items occur together, while the marginals $p(\mathbf{x})$ and $p(\mathbf{y})$ are estimated using the empirical distribution based on the number of observed documents that have the item x_i. As $p(\mathbf{x})$ equals $p(\mathbf{y})$ in our case, we can

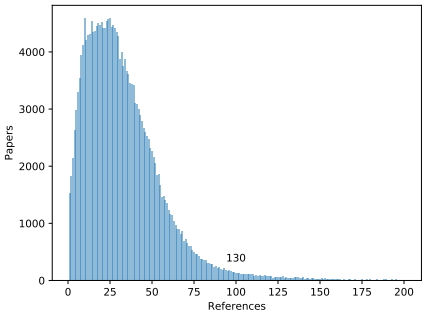
normalize mutual information with the entropy of $\mathbf{X}$: $H(\mathbf{X})$, so that we obtain $\frac{\mathrm{I}(\mathbf{X};\mathbf{X})}{H(\mathbf{X})}$ as normalized mutual information.

In the following, we outline the datasets used for our experiments, organized by recommendation task or item type. First, we introduce three datasets for citation recommendation in Section 6.4.1. Subsequently, we introduce three datasets for subject label recommendation Section 6.4.2.

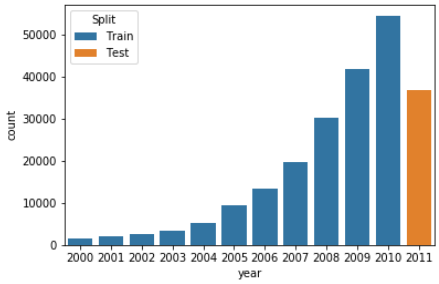
6.4.1 Datasets for Citation Recommendation



(a) Citation distribution



(b) Number of references per document



(c) Number of documents by publication year with chronological train/test split in blue and orange, respectively.

Figure 6.7. Characteristics of the PubMed dataset

PubMed Citation Dataset The CITREC[1] PubMed citation dataset [GML15] consists of 7,546,982 citations. The dataset comprises 224,092 distinct citing documents published between 1928 and 2011 and 2,896,764 distinct cited documents. Each document has an identifier, an article title, the title of the journal in which it was published, a list of authors, medical subject headings (MeSH)[2] labels, and the publication year. The documents are cited between 1 and 3,247 times with a median of 1 and a mean of 2.61 (SD: 6.71). The citing documents hold on average 33.68 (standard deviation, SD: 27.49) citations to other documents (minimum: 1, maximum: 2,242) with a median of 29. The citation distribution of this dataset follows a power law, which is typical for citation networks (cf. Figure 6.7a). The power-law coefficient for the PubMed dataset is 1.47 for citations and 1.30 for the number of cited documents. The normalized MI for PubMed is 0.5996.

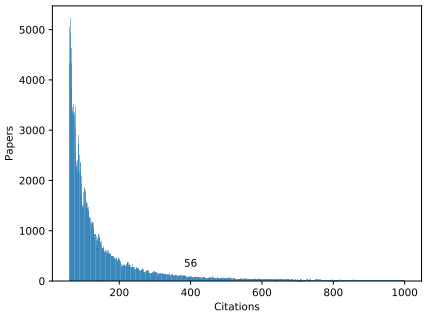
DBLP Citation Dataset The DBLP Citation Graph[3] [TZY+08] includes 25,166,994 citations. The dataset comprises 3,079,007 distinct citing documents published between 1936 and 2018 and 1,985,921 distinct cited documents. Each document has an identifier, a title, a publication venue, a list of authors, a number of citations, a publication date, and optionally an abstract. The documents are cited between 1 and 16,229 times with a median of 4 and a mean of 12.67 (SD: 56.17). The citing documents hold on average 8.17 (SD: 9.71) citations to other documents (minimum: 0, maximum: 1,532) with a median of 6. The power-law coefficient is 1.58 for citations and 1.28 for the number of cited documents. The normalized mutual information is 0.5407.

ACM Citation Dataset The ACM Citation Network[3] [TZY+08] contains 11,344,141 citations. The dataset comprises 2,385,066 distinct citing documents published between 1936 and 2016 and 2,631,128 distinct cited documents. Each document is characterized by its identifier and comes with a title, a publication venue, a list of authors, a publication date, and optionally an abstract. The documents are cited between 0 and 810 times with a median of 1 and a mean of 4.76 (SD: 7.74). The citing documents hold on average 4.31 (SD: 580.96) citations to other documents (minimum: 1, maximum: 938,039) with a median of 1. The power-law coefficient is 1.53 for citations and 1.32 for the number of cited documents, while the normalized mutual information is 0.5282.

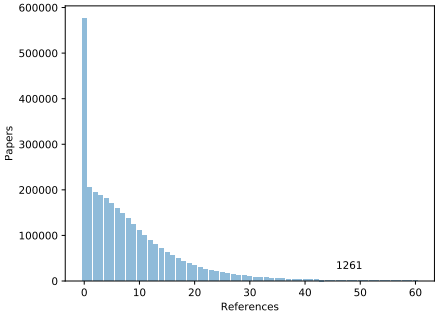
[1] https://www.isg.uni-konstanz.de/projects/citrec/

[2] https://www.nlm.nih.gov/mesh/

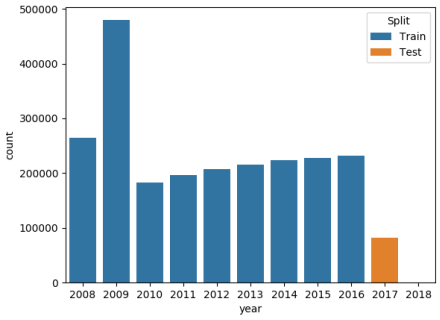
[3] https://aminer.org/citation



(a) Citation distribution



(b) Number of references per document.



(c) Number of documents by publication year with chronological train/test split in blue and orange, respectively.

Figure 6.8. Characteristics of the DBLP dataset

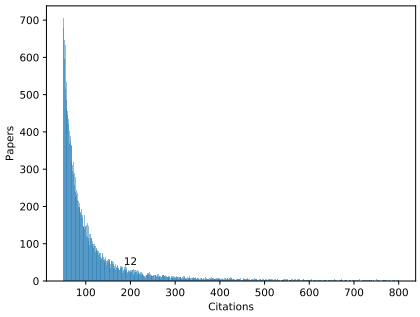
6.4.2 Datasets for Subject Labels Recommendation

EconBiz Subject Labels The EconBiz[4] dataset [GMS+17] provided by ZBW — Leibniz Information Centre for Economics[5] consists of 61,619 documents with label annotations from professional subject indexers [GMS+17; GNS15]. The controlled vocabulary for the subject labels is Standardthesaurus Wirtschaft[6], a professional thesaurus for economics. The subset of the thesaurus that has been used within the dataset has the size of 4,669 subject labels. Every document has an identifier,

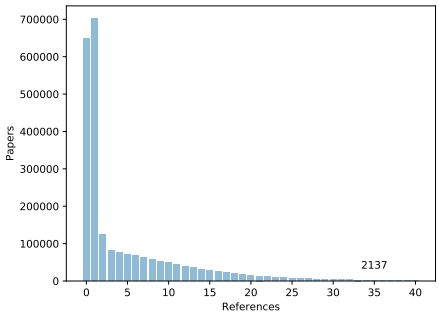
[4] https://econbiz.de/

[5] https://zbw.eu/

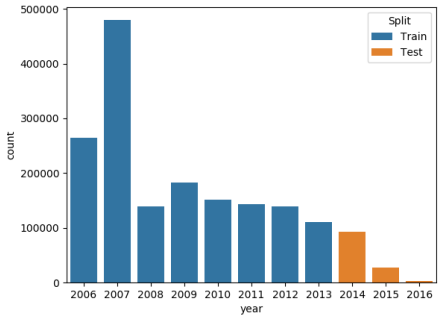
[6] http://zbw.eu/stw/version/latest/about



(a) Citation distribution



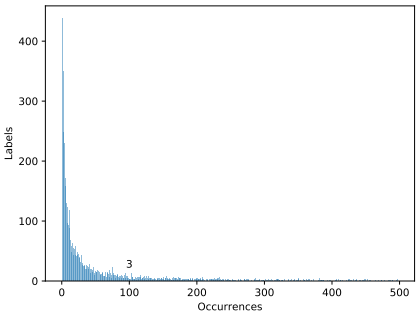
(b) Number of references per document



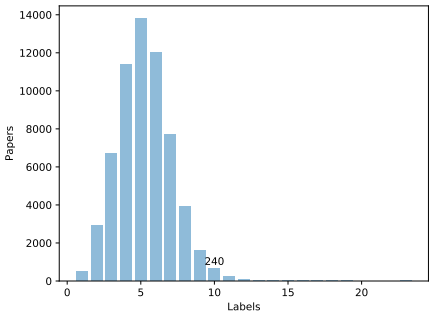
(c) Number of documents by publication year with chronological train/test split in blue and orange, respectively.

Figure 6.9. Characteristics of the ACM dataset

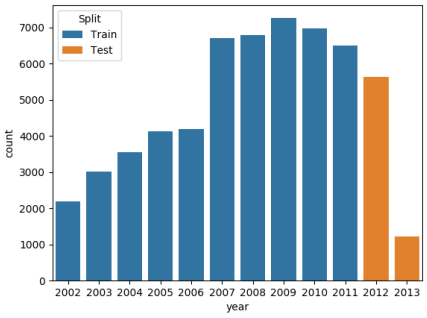
title, authors, language label(s), subject labels, and publication year, as well as optionally the publisher, publication country, and series. The number of documents to which a subject label is assigned ranges between 1 and 13,925 with a mean of 69 (SD: 316) and a median of 14. We plot the label distribution in Figure 6.10a. The label annotations of a document range between 1 and 23 with a mean of 5.24 (SD: 1.83) and a median of 5 labels. Similar to the citation datasets, EconBiz follows a power law distribution. The power law coefficient for the EconBiz dataset is 1.96 for the label occurrences and 1.19 for the number of assigned labels. The normalized mutual information is 0.2970.



(a) Label distribution



(b) Number of labels per document



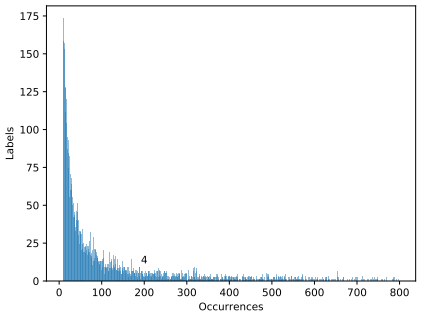
(c) Number of documents by publication year with chronological train/test split in blue and orange, respectively.

Figure 6.10. Characteristics of the EconBiz Dataset

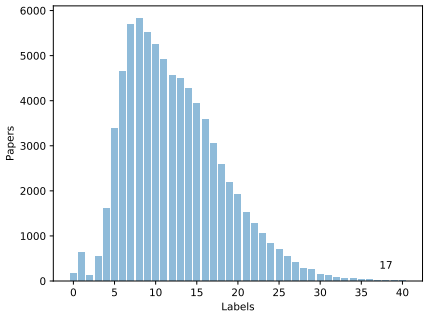
IREON Subject Labels The IREON political sciences dataset has 76,359 documents provided by the German Information Network for International Relations and Area Studies[7]. Each document has an identifier, a title, a list of authors, a language label, a list of subject labels, and a publication year. The 10,440 subject labels assigned to the articles are taken from the thesaurus for International Relations and Area Studies[8]. The number of documents to which a label is assigned ranges between 1 and 13,895 with a mean of 90.68 (SD: 338.63) and a median of 13. The label annotations of a document range between 0 and 70 with a mean of 12.40 (SD: 5.91) and a median of 11. The power-law coefficient is 1.94 for the

[7] http://www.fiv-iblk.de/eindex.htm

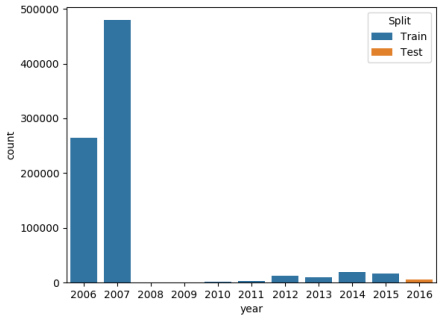
[8] http://www.fiv-iblk.de/information/information_thesaurus.htm



(a) Label distribution



(b) Number of labels per document

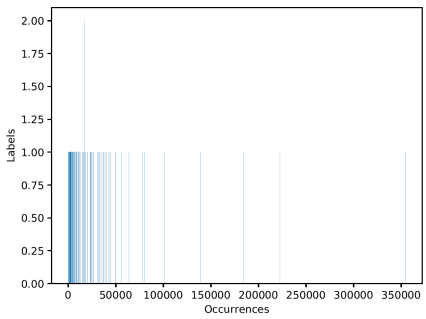


(c) Number of documents by publication year with chronological train/test split in blue and orange, respectively.

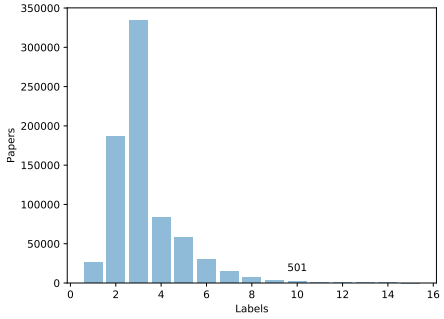
Figure 6.11. Characteristics of the IREON dataset

label occurrences and 1.21 for the number of assigned labels, while the normalized mutual information is 0.1977.

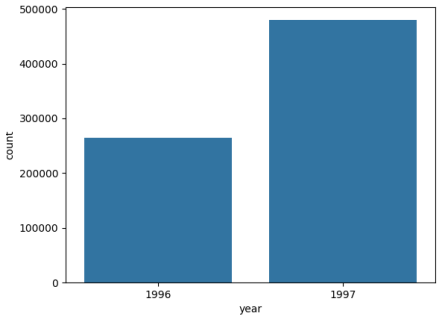
Reuters Subject Labels The Reuters RCV1-v2 dataset contains 744,693 news articles and a thesaurus that provides a set of 104 labels [LYR+04]. Every document includes an identifier, title, subject labels, and publication date. The number of documents to which a label is assigned ranges between 5 and 354,437 with mean 23,217.35 (SD: 47,313.01) and median 7,315. The number of annotated labels of a document range between 1 and 17 with a mean of 3.24 (SD: 1.42) and a median of 3 labels. Its label distribution does not follow a power-law distribution (Figure 6.12a).



(a) Label distribution



(b) Number of labels per document



(c) Number of documents by publication year

Figure 6.12. Characteristics of the Reuters dataset

The power-law coefficient is 148.15 for label occurrence and 1.14 for the number of assigned labels. The normalized mutual information is 0.3207.

6.4.3 Availability of Side Information

Table 6.2 illustrates the metadata fields available in the considered datasets and what percentage of the document has a value for the field. Title data are present in all datasets. The author data are available in all datasets except Reuters. Abstracts are only available in DBLP (89%) and a few (4%) in ACM. The journal information is available in PubMed, DBLP (83%) and ACM. Subject labels are available in PubMed, EconBiz, IREON, and Reuters. Citations are available in PubMed, DBLP, and ACM.

Table 6.2. Availability and occurrence of metadata in the datasets considered for the two recommendation tasks. Subject labels and item set occurrences are the same for the subject-label datasets as the subject labels are the items to recommend (but can also be used as additional metadata for the citation tasks).

Metadata field	PubMed	DBLP	ACM	EconBiz	IREON	Reuters
Title	100 %	100 %	100 %	100 %	100 %	100 %
Author	100 %	100 %	93 %	98 %	83 %	-
Abstract	-	89 %	4 %	-	-	-
Venue/Journal title	100 %	83 %	100 %	-	-	-
Subject labels	77 %	-	-	72 %	100 %	100 %
Citations	100 %	89 %	81 %	-	-	-

Based on the availability of side information, we derive three conditions for the input modalities, alongside which we will report our results.

Item set only We only use the item set as input to the model. The item set consists of cited documents for PubMed, DBLP, and ACM, while it consists of subject labels for EconBiz, IREON, and Reuters. We discard all documents that are not associated with any items.

Item set and title Here, we use the item set together with the document title as side information. The MLP baseline does not use the item set at all.

All metadata Here, we use all available metadata as side information, except for the abstract, because it has only reasonable coverage in DBLP. The MLP baseline uses all metadata except for the item set. As an interesting cross-over case, we include the subject labels in the all-metadata case on PubMed, where we predict citations. Scores without these additional metadata are reported in Section E.1.

6.4.4 Chronological Train-Test Splits

To simulate a real-world citation prediction setting, we split the data on the time axis of the citing documents. This resembles the natural constraint that the authors cannot cite other publications that do not yet exist. Given a specific publication year T, we ensure that the training set $\mathbb{D}_{\text{train}}$ consists of all documents published earlier than in year T, and use the remaining documents as test data $\mathbb{D}_{\text{test}}$. Regarding our evaluation, we select the year 2011 for PubMed, 2017 and 2018 for DBLP, and 2014 to 2016 for ACM to obtain a 90:10 ratio between training and test documents.

For the subject labeling datasets, EconBiz and IREON, we also split the data into training and test set along the time axis (cf. Figures 6.10c and 6.11c). This resembles a challenge because label annotations suffer from concept drift over time [TS20]. We use the years 2012 and 2013 as test documents for EconBiz and the year 2016 for IREON to obtain a 90:10 train-test ratio, as in the citation recommendation datasets described above. Since Reuters contained only two years (1996 and 1997), we randomly select 10% of documents for the test set, regardless of the year.

6.4.5 Evaluation Measures

For the evaluation, certain items were omitted on purpose in the test set. For each document, the models should predict the omitted items as well as possible. Thus, we choose mean reciprocal rank (MRR) as our evaluation metric [Cra09].

We are given a set of predictions, $\boldsymbol{X}_{\text{pred}}$, for the test set, $\tilde{\boldsymbol{X}}_{\text{test}}$. For each row, we compute the reciprocal rank of the missing items $\boldsymbol{x}_{\text{test}} - \tilde{\boldsymbol{x}}_{\text{test}}$. The reciprocal rank corresponds to one divided by the position of the first omitted item in the sorted set of predictions, $\boldsymbol{x}_{\text{pred}}$. We then averaged all documents of the test set to obtain the MRR.

To alleviate the random effects of model initialization, training data shuffle, and selection of elements to omit, we performed three runs for each of the experiments. For a fair comparison, the items removed from the test set remain the same for all models during a run with a fixed pruning parameter.

6.5 Experiments

We present the results alongside the experimental settings of dataset pruning and the degree of completeness of the input item sets. We first consider how the pruning of the dataset influences the performance of the models in Section 6.5.1. Then, we investigate the influence of the size of the input item sets in Section 6.5.2,

Within each of these two experimental settings, we provide results for three citation recommendation tasks and three subject-label recommendation tasks to contrast the different meanings of item co-occurrence. Furthermore, we analyze the main research question of how the use of additional metadata influences the performance of autoencoders throughout all settings.

6.5.1 Experiments under Varying Total Number of Items

Experimental Setup For dataset preprocessing, we perform the following three steps: First, we build a vocabulary on the training set with items that have been cited or assigned more than k times, which we denote as the pruning threshold. Second, we filter both the training set and the test set and retain only items from the vocabulary. Third, we remove documents with fewer than two of the vocabulary items in their item set.

The pruning threshold k is crucial since it affects both the number of considered items and the number of documents. Therefore, we systematically control the pruning threshold k of the datasets and analyze how this influences the different models and input modalities. Table 6.3 shows the characteristics of PubMed, DBLP, and ACM with respect to k, while Table 6.4 illustrates the same for EconBiz, IREON, and Reuters.

Table 6.3. Characteristics of the citation datasets with respect to different selected pruning thresholds on the minimum item occurrence.

Dataset	Pruning	Cited docs	Citations	Documents	Density	a	MI
PubMed	20	20,270	878,359	121,374	0.000357	1.5870	0.4265
	30	8,906	568,563	96,980	0.000658	1.6465	0.3958
	40	4,939	413,746	79,830	0.001049	1.7090	0.3737
	50	3,185	324,693	67,703	0.001506	1.7755	0.3587
DBLP	20	251,405	16,340,121	1,955,132	0.000033	1.5960	0.4879
	30	157,203	13,977,243	1,839,181	0.000048	1.6046	0.4750
	40	109,469	12,271,346	1,742,180	0.000064	1.6127	0.4658
	50	81,817	10,991,096	1,660,462	0.000081	1.6206	0.4591
ACM	20	78,805	5,590,751	786,216	0.000090	1.5840	0.4650
	30	46,782	4,752,086	751,981	0.000135	1.6099	0.4521
	40	31,780	4,189,331	725,635	0.000182	1.6354	0.4435
	50	23,177	3,767,585	702,158	0.000232	1.6610	0.4374

Hyperparameter Optimization The hyperparameters are selected by conducting preliminary experiments on PubMed considering only items that appear 50 or more times in the entire corpus. We chose this scenario because this aggressive pruning

Table 6.4. Characteristics of the subject label datasets with respect to different selected pruning thresholds on minimum item occurrence.

Dataset	Pruning	Classes	Labels	Documents	Density	a	MI
EconBiz	1	4,568	323,670	61,104	0.001160	1.9612	0.2970
	5	3,259	320,048	60,983	0.001610	2.0018	0.2917
	10	2,597	314,738	60,778	0.001994	2.0483	0.2857
	20	1,924	303,693	60,272	0.002619	2.1379	0.2768
IREON	1	10,324	945,888	75,558	0.001213	1.9382	0.1977
	5	6,971	938,677	75,555	0.001782	1.9644	0.1928
	10	5,612	928,701	75,551	0.002190	1.9930	0.1881
	20	4,304	909,156	75,535	0.002797	2.0463	0.1809
Reuters	1	104	2,340,132	744,693	0.030215	148.15	0.3207
	5	104	2,340,132	744,693	0.030215	148.15	0.3207
	10	103	2,340,127	744,693	0.030509	146.71	0.3207
	20	103	2,340,127	744,693	0.030509	146.71	0.3207

results in numbers of distinct items and documents that are similar to those of the subject-label recommendation datasets.

For the MLP modules, we perform a grid search with hidden layer sizes between 50 and $1,000$, an initial learning rate between 0.01 and 0.00005, activation functions Tanh, ReLU [NH10], SELU [KUM+17] along with dropout [SHK+14] (or alpha-dropout in case of SELUs) probabilities between 0.1 and 0.5, and as optimization algorithms a standard stochastic gradient descent and Adam [KB15].

For the autoencoder-based models, we consider code sizes between 10 and 500, but only if the size was smaller than the hidden layer sizes of the MLP modules. For AAEs, we experimented with Gaussian, Bernoulli, and multinomial prior distributions, and with linear, sigmoid, and softmax activation on the code layer, respectively.

Although a certain set of hyperparameters may perform better in a specific scenario, we select the following, most robust set of hyperparameters: hidden layer sizes of 100 with ReLU [NH10] nonlinearities and drop probabilities of 0.2 after each hidden layer. The optimization is carried out by Adam [KB15] with an initial learning rate 0.001. The AE variants use a code size of 50. We further select a Gaussian prior distribution for the adversarial autoencoder. For SVD, we consecutively increased the number of singular values up to $1,000$. Higher amounts

of singular values decreased the performance. We keep this set of hyperparameters across all models and subsequent experiments to ensure a reliable comparison of the models.

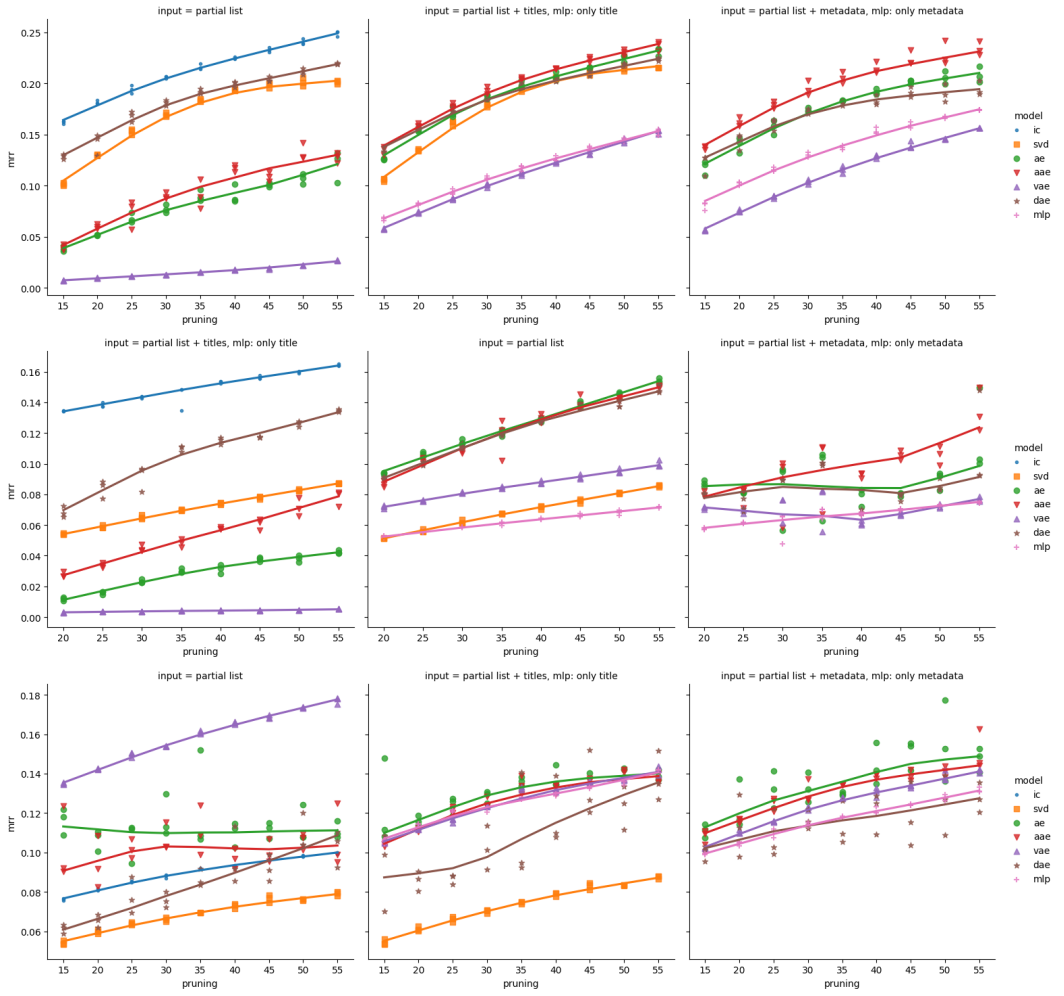


Figure 6.13. Mean reciprocal rank of predicted citations on the test set with varying minimum item occurrence (pruning) thresholds for the PubMed (top row), DBLP (middle row), and ACM (bottom row) citation datasets. *Left*: Only the partial set of items is given. *Center*: The partial set of items along with the document title is given. *Right*: The partial set of items is given along with the document title, the authors, the journal title and the MeSH labels (if available). MLP can only make use of either titles or titles, authors, journal titles, and MeSH labels.

Results for Different Pruning Thresholds in Citation Recommendation Figure 6.13 shows the results for the models with respect to the pruning parameter that controls the number of items considered and the sparsity (see Table 6.3) on the PubMed, DBLP, and ACM datasets, respectively. We observe a trend that a more aggressive pruning threshold leads to higher scores among all models on all three datasets. However, this phenomenon seems to be more attenuated on the ACM dataset. In particular, on this data set, AE and AAE seem to be unaffected by the threshold, or even there seems to be a slight decrease for higher thresholds. When no title information is given, the item co-occurrence approach performs best on PubMed and DBLP, while VAE obtained the best scores on ACM. When titles are used, autoencoders (AAE, AE, and DAE) become competitive with the item co-occurrence approach and outperformed other models on PubMed and DBLP. The same holds when additional metadata are available. The results on the ACM dataset show a similar pattern, except that DAE performs worse than VAE. Surprisingly, more metadata yield worse results than using only the title data.

Results for Different Pruning Thresholds in Subject Labeling Figure 6.14 shows the results of the models with respect to the pruning parameter that controls the number of items considered and the corresponding sparsity (see Table 6.4) on the EconBiz, IREON, and Reuters datasets, respectively. When no title information is available, autoencoders (but VAE) are competitive to the item co-occurrence approach. With titles, the models achieve considerably higher scores than all models operating without this information. Specifically, VAE achieves the best results, closely followed by MLP, on EconBiz, while AAE is the best-performing model on IREON. On Reuters, MLP and VAE achieve the same performance. Similarly to the citation task, using additional metadata decreases the performance compared to using only titles, although in IREON the decline is notably lower. VAE is particularly negatively affected by more metadata on EconBiz. Reuters only provides titles, so it was impossible to use additional metadata.

6.5.2 Experiments under Varying Number of Items per Document

Experimental Setup In these experiments, we investigate the influence of completeness of the partial input set through the number of dropped elements and analyze how different models and modalities behave. We run multiple experiments by dropping different percentages of elements with respect to the size of the original item set. We perform experiments for some given pruning thresholds on PubMed,

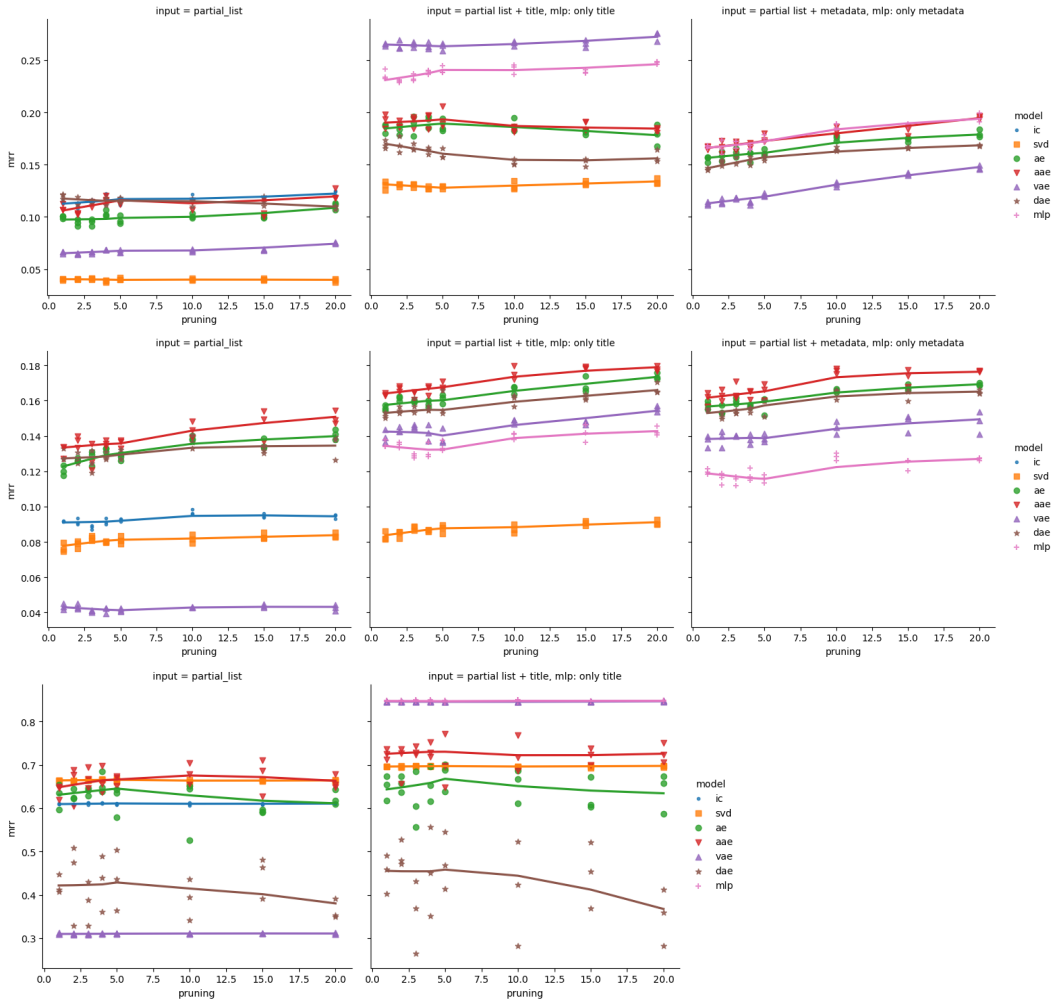


Figure 6.14. MRR of predicted subject labels on the test set with varying minimum item occurrence thresholds for the EconBiz (top row), IREON (middle row) and Reuters (bottom row) dataset. *Left*: Only the partial set of items is given. *Center*: The partial set of items along with the document title is given. *Right*: The partial set of items along with the title and authors is given. MLP can only use either titles or titles and authors.

ACM, and all subject indexing datasets, we expect a similar behavior for other thresholds.

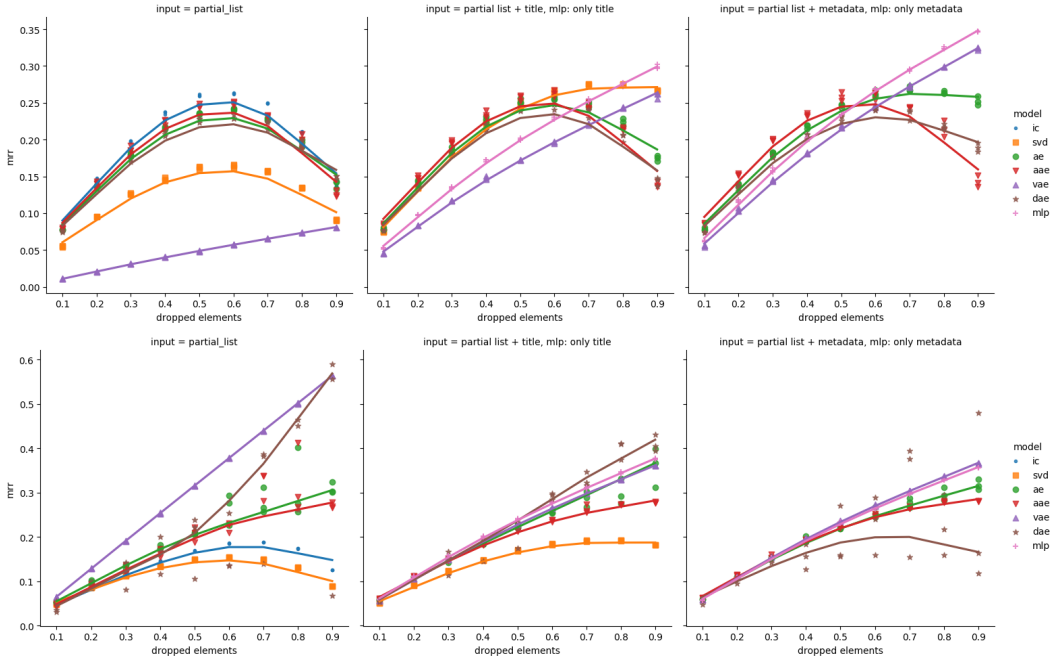


Figure 6.15. MRR of predicted citations on the test set with varying number dropped elements for the PubMed (top row) and ACM (bottom row) citation datasets. The minimum item occurrence threshold is set to 55. *Left*: Only the partial set of items is given. *Center*: The partial set of items along with the document title is given. *Right*: The partial set of items is given along with the document title, the authors, the journal title, and the MeSH labels. MLP can only make use of either titles or titles, authors, journal titles, and MeSH labels.

Results for Different Item Set Sizes in Citation Recommendation Figure 6.15 shows the results for the models on the citation recommendation task with respect to the drop parameter that controls the percentage of elements dropped in the original item sets, i. e., how the completeness of the partial set of items influences the provided recommendation, on PubMed and ACM, respectively. On PubMed, most models peak around a drop threshold of about 50 %, independently if metadata are used or not. The exceptions are MLP and VAE, which also increase with higher percentages of dropped elements. SVD with titles and AE with more metadata plateau when 60 % of elements are dropped. MLP and SVD have low results with

few dropped items, but achieve good performance with high drop thresholds. The same holds for VAE when titles or even more metadata are available. On ACM, only item co-occurrence and SVD decrease with more than 60 % of elements dropped. However, the different autoencoders have a lower improvement with many dropped elements, depending on their type and additional information used (only partial set, partial set and titles, or partial set, titles, and more metadata). On the contrary, DAE with only the partial set tends to increase more with a drop threshold greater than 50,%. The more metadata used, the lower the improvement with many dropped items.

Results for Different Item Set Sizes in Subject Labeling Figure 6.16 shows the results for the models for the subject label task with respect to the number of dropped items on EconBiz, IREON, and Reuters, respectively. As for citations, we performed experiments only for some given pruning thresholds, but we expect a similar behavior for other thresholds. When no title information is available, most of the models peak or plateau around a drop threshold of 50 % and 60 % on Reuters and EconBiz, respectively. On IREON, item co-occurrence plateaus at 70 %, SVD peaks at 60 %, the other models show a steady increase. With titles only, SVD performs poorly when few elements are dropped, while outperforms various models with many elements dropped (except for IREON). When titles and additional metadata are given, only AE, AAE, and DAE increase considerably less with many elements dropped on EconBiz, while they peak at about 50 % on Reuters. On IREON, all the models suffer less when many elements are dropped. VAE and MLP are generally the best performing models in all datasets, although the most effective varies with the dataset and the metadata used.

6.6 Discussion

We first recall the main findings and then compare the different meanings of item co-occurrence in the two tasks. Subsequently, we discuss the results of the citation and subject-label recommendation tasks separately with respect to dataset pruning and the degree of completeness of the input item set.

6.6.1 Key Results

The two tasks of citation recommendation and subject label recommendation, which are similar from a structural perspective, behave very differently in our experiments.

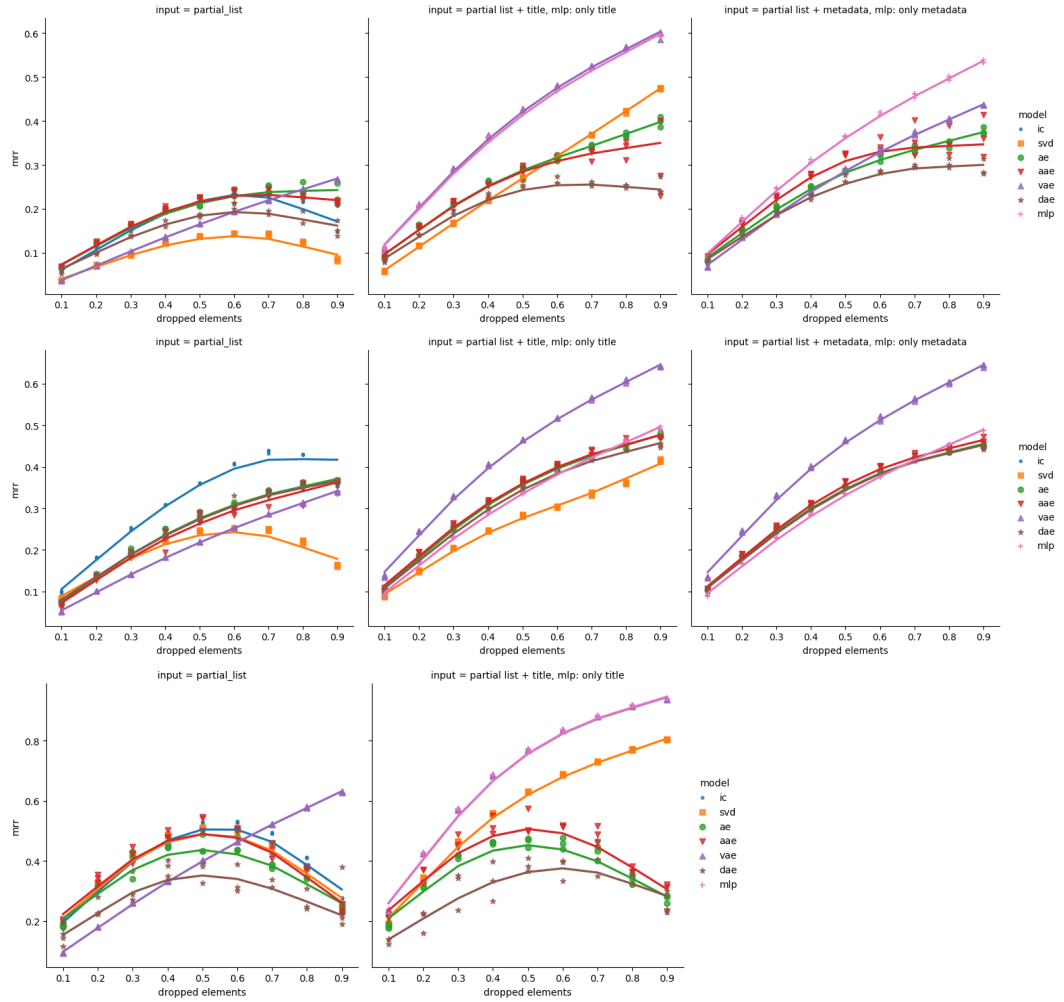


Figure 6.16. MRR of predicted subject labels on the test set with varying number of dropped elements for the EconBiz (top row), IREON (middle row) and Reuters (bottom row) datasets. The minimum item occurrence threshold is set to 20. *Left*: Only the partial set of items is given. *Center*: The partial set of items along with the title is given. *Right*: The partial set of items along with the document title and authors is given. MLP can only use either titles or titles and authors.

Our experiments show that what is already cited is much more informative than which subject labels are already assigned. This is supported by mutual information, which is greater than 0.5 for the citation datasets, but below or close to 0.3 for the subject label datasets. In the citation recommendation task, where item co-occurrence implies relatedness, the approach based on the co-occurrence is a strong baseline. VAE (using only item sets) outperforms this baseline on ACM, but VAE performs worse when titles are combined with other metadata. In the subject label recommendation task, where co-occurrence of items implies diversity, item co-occurrence is not effective, and the best method depends on the dataset: AAE in IREON, VAE in EconBiz (followed by MLP), and MLP and VAE (with titles) in Reuters. DAE seems to be less stable in optimization, since its results usually vary greatly between multiple runs on the same dataset.

In practical applications, it is desirable to have low pruning thresholds to avoid the "rich get richer" phenomenon, where highly cited documents or frequently assigned labels are privileged. In fact, highly cited documents, or often used subject labels, are also the ones most likely to be known, while recommendations should also lead to discovering previously unknown items. However, our results confirm that the ranking between methods is, for the most part, stable under different pruning thresholds.

Our experiment on the varying number of dropped items has shown that the models that do not show the boomerang-shaped curve when using metadata are the ones that rely more on titles and other metadata than on the partial set of items. This also enables them to perform well when very few items are given as input in both the considered tasks. Surprisingly, aside from MLPs, where this was expected, the VAE also shows this comparatively strong performance when only a few input items are present. This might explain the popularity of VAEs in recommender systems.

Finally, our results show that, counterintuitively, it is not always better to use more metadata fields as input. Titles are usually helpful, but models that rely solely on the partial set of items are sometimes the top performers. The title field has been the most useful metadata field in our experiments. Adding additional metadata did not improve performance and sometimes even decreased it.

6.6.2 Meanings of Item Co-occurrence

We observe different relationships between the factors of (i) type of recommendation task, each with different meanings of item co-occurrence, and the performance

with (ii) varying completeness of the partial set, (iii) degree of sparsity, as well as (iv) input modalities. We discuss the results by type of recommendation task in Section 6.6.3 and Section 6.6.4.

The performance of the recommendation depends greatly on the completeness of the partial set. Most models perform best having around 50 % of items from the original set, but VAE and MLP achieve good performance also with 90 % of elements dropped. Note that pruning almost never changes the order of the top performers, whereas the top performers differ by varying the number of dropped elements.

Second, by applying several thresholds on minimum item occurrence, we controlled the number of considered items and thus the degree of sparsity. The result is that all considered models are similarly affected by the increased sparsity and difficulty caused by the larger number of considered items. It is not surprising that the results decrease with lower pruning thresholds, as pruning can strongly influence the results [BGL+16]. Interestingly, there are differences between the two tasks: Although the decrease is pronounced in the citation datasets, it is very low in the other ones. This decrease is low in the subject label datasets presumably because they are considerably less sparse (cf. Table 6.4 and Table 6.3). Differences persist even when datasets are similar. EconBiz has 4,568 classes without pruning and PubMed has 4,939 cited documents with pruning at 40 and 3,904 with pruning at 45. On PubMed (citations), the best MRR is about 21 % with item co-occurrence (not improving with metadata), while on the EconBiz (subject labels) dataset, all models yield similar results only when using metadata, and without metadata are below 13 %.

Lastly, on the citation task, the partial set of citations is the most important information to recommend potentially missing citations. For the subject label recommendation task, the MLP model, which uses only titles, achieves the best performance in one dataset and is generally competitive. On the citation recommendation task, AE and SVD become competitive with the strong cocitation baseline when titles.

6.6.3 Discussion of the Citation Recommendation Task

The number of elements dropped in the partial input sets affects the models differently. Most models improve until about 50 % of elements are dropped, then their performance decreases. In fact, the more elements are dropped, the less

information is used as input, but the task becomes also easier as there is more than one correct answer (one document or label to predict).

When metadata are provided together with the partial set, the results generally improve. This is particularly the case when the partial input set is small, that is, many elements are removed, since additional information compensates for fewer items in the partial set.

The results of the MLP also improve with more dropped elements because it does not use the partial set of items. VAE shows similar behavior. It is capable of providing good predictions even when only a few items are given as input. This could be due to the generalization provided through the Gaussian prior to the code. The reason is that the latent representations learned by distinguishing the code from a smooth prior make the model more robust to sparse input vectors as smoothness is key to good representations that disentangle the explanatory factors of variation [BCV13]. Although both VAE and AAE impose a Gaussian prior on the code, VAE is better in some cases. A possible explanation is that VAE is more stable during optimization [TBG+18].

On the ACM dataset, the performance continues to improve even with very few items in the partial set (i. e., many elements are dropped), except for item co-occurrence and SVD. This may be due to special characteristics of the ACM dataset. Mutual information for ACM (0.5282) is the lowest among the citation datasets (0.5407 for DBLP and 0.5996 for PubMed). This means that already cited documents are less informative in ACM compared to the other two datasets. Furthermore, the difference between most cited and less cited documents is lower in ACM (700) than in the other two datasets (5,000 in DBLP and 16,000 in PubMed), as shown by the y-axes in Figures 6.7a, 6.8a, and 6.9a. So, more documents have many references. In this case, although many references may have been dropped, more documents may still have enough references as a basis for recommendation.

Regarding the results for the experiments with varying pruning, co-citation is known to be an important baseline [Sma73]. However, we have shown that this is not always the case. For the ACM dataset, VAE is the best method. This is interesting because in other cases, such as DBLP and PubMed, the VAE is particularly weak, especially without metadata. This may be due to ACM special characteristics, as discussed above. Specifically, documents already cited are less informative in ACM compared to the other two datasets due to the lower mutual information.

In terms of metadata, MLPs and AEs have the advantage of making use of the side information (AEs together with the partial set of citations), which improves the

results. The use of language attributes, i. e., titles, is most effective. The use of other metadata attributes, such as authors and venues, on top of titles does not improve the recommendation performance. From the partial input set, autoencoders can learn the co-occurrence patterns within item sets and can also learn to reflect the prior probabilities of the items in the bias parameters, if it is helpful for the overall objective. In this task of citation recommendation, the co-occurrence of documents within item sets is of great importance, as related papers are typically cited together, which explains the strength of the item co-occurrence baseline. Although MLP can also learn bias in the output layer to model the item's prior distribution, autoencoders can model the relationship between cited documents from the partial input set. MLP cannot model the relationship between cited documents because it only uses the titles as input and not the partial set. When considering different types of metadata in the case of MLP, using titles is generally as effective as, or even more effective than, using available additional metadata (together with titles). This suggests that it is because the title is more indicative than other metadata when deciding whether to cite a paper. Although researchers may tend to cite more often some papers from well-known venues and some authors because of similarity in topics or because they know them, the title has a stronger relation to the paper than the authors and venues. Furthermore, the advantage of the title is that these metadata are always available, even in the news domain, whereas other metadata are not. For example, in PubMed only about 77% of the documents have MeSH labels, in ACM about 93% of the documents include authors, and in DBLP around 83% of the papers hold the venue (see Table 6.2), while all documents have a title.

6.6.4 Discussion of Subject Label Recommendation Task

The number of elements dropped has a less powerful effect on the models compared to the citation recommendation task. Most models improve until about 50% of the elements in the original set are dropped, then their performance plateaus or decreases slightly. Nevertheless, some models can also provide good predictions with few elements in the partial set. As in the citation recommendation task, the results of MLPs and VAEs improve even with many dropped elements. Only with the Reuters dataset, many models show a notable decline. This could be due to its distribution of label occurrences, which is the only one that does not follow the power-law distribution. In contrast, there is a low number (roughly 100 compared to 4,600 in EconBiz and 10,000 in IREON) of fairly well-balanced labels to choose from (see Figure 6.12a).

Another reason could be that in the Reuters dataset the labels have no hierarchy, contrary to the other two subject-label datasets, which are based on a professional taxonomy. Subject indexers usually assign the ancestor instead of the child subjects when two or more subject labels with a common ancestor in the hierarchical thesaurus match [GNS15]. Thus, two subjects that are semantically related because they share a common ancestor are unlikely to co-occur in the annotations of a single document. Without a hierarchy, different subject labels can be similar, and no common ancestor is available to use instead. For example, the news "*Clinton signs law raising minimum wage to $5.15*" is assigned to subjects *EMPLOYMENT/LABOUR*, *LABOUR*, *LABOUR ISSUES* which are rather similar.

Finally, since the main purpose of subject indexing is retrieval, the fact that many labels are frequently used suggests that recall is preferred over precision. While in a library it may be desirable to retrieve fewer results, but all highly relevant to a query, in the news domain it may be best to retrieve as many results as possible, although some could be less relevant.

In experiments with varying pruning, MLPs and VAEs achieve the highest MRR on two of the three datasets. Thus, already assigned subjects are less informative for a subject-label recommendation task than titles. Two subjects that are semantically related because they share a common ancestor are unlikely to co-occur in the annotations of a single document since subject indexers tend to rely on the ancestor instead of the child subjects. Instead, on the IREON dataset, autoencoders and, notably, AAEs, are more effective. A potential reason is that IREON has many more different subject labels than the other datasets (see Table 6.4). Therefore, the co-occurrence statistics (which are modeled by AE variants) might be more informative.

Regarding the influence of metadata, adding other metadata in addition to titles generally decreases performance. For citations, other metadata seem to be less indicative of the content of the article. Furthermore, only the authors are available in addition to titles. The authors are present in roughly 98 % and 83 % of documents, in EconBiz and IREON, respectively. VAE is the best model (together with MLP) in Reuters, but it performs poorly without titles. This suggests that when given titles, VAEs learn to ignore the input item set, which is not possible with only the partial set of items as input.

6.6.5 Threats to Validity

The availability and occurrence of metadata fields vary between datasets, as shown in Table 6.2. Our citations datasets have had more metadata attributes available than the subject-label datasets. Titles and authors can be used in all three citation datasets; in PubMed also journal and MeSH labels can be exploited, and venues are present in ACM and DBLP. For the subject label datasets, only titles and authors can be used, apart from Reuters which offers only titles. Therefore, it was only partially possible to use the same or similar set of metadata fields among the datasets. In particular, we could not use abstract information because it was always or often missing. Only ACM and DBLP contain this information, but it is rarely available in ACM (less than 4%), so we decided not to use it in our experiments.

We tested the models' performance and the impact of different meanings of item co-occurrence, the completeness of partial input set, the pruning, and metadata on various datasets, for each of the two recommendation tasks. As the results are consistent among the datasets, we have good reasons to assume that the results can also be generalized to other similar datasets in the considered domains. Our method of adding metadata is general and can handle different types of metadata. Thus, the models can also be applied to similar tasks in other domains, which could also benefit from the use of metadata. For example, we have previously shown that metadata are beneficial for automatic playlist continuation [VGM+18].

6.6.6 Practical Impact

Our experiments have high practical relevance, as they are close to real-world settings. This comes in three aspects. First, we use six real-world datasets from five different domains. Thus, the experimental results show how the recommender models would behave in real-world contexts. Second, the splitting of the documents in training and test set along the time axis resembles the natural constraint that newly written publications can only cite already published works and only papers published before are already annotated. Furthermore, applying this chronological split to subject labels also accounts for the concept drift [WLG+18]. Third, by taking into account the typical long-tail distribution in user feedback on items (in our case, citations and label annotations), we have also investigated performance with low pruning thresholds, i. e., including items with few citations and labels rarely used to annotate documents. This further strengthens the reliability of our experimental

results in real-world settings, in contrast to existing studies with limited datasets induced by only a fixed frequency-based pruning.

6.7 Summary

We have shown that using text as side information increases the performance of autoencoders as recommendation engines. The only exception is VAE, which is interesting because VAE is used most frequently in the literature to tackle recommendation tasks. Aside from that, we have analyzed the interactions between text and graph data on a wide range of datasets to argue about the relationship between the meaning of item co-occurrence and the importance of language data.

Different meanings of item co-occurrence in recommendation tasks highly affect the preferable input modalities. When item co-occurrence resembles relatedness, such as in citations, the set of already cited documents is beneficial. In subject recommendations, co-occurring subject labels do not imply that these subjects are similar. Instead, the actual research subject of a document must be described using multiple, diverse subject labels as annotations. Incorporating multiple input modalities offers a conceptual benefit, but not always adding more metadata is useful, and relying on just titles or a partial set of items can be more effective. All evaluated methods are similarly sensitive to data sparsity, but variational autoencoders and multilayer perceptrons are more robust with few items in the partial set. This is likely because they rely more on titles and other metadata than on the initial item set.

In future systems and studies, the meaning of item co-occurrence should be taken into account when deciding whether the partial list of items should be supplied to a recommendation model as input. Regarding the two scenarios considered, we can state the following. In citation recommendation, where co-occurring items are similar, the model can perform well without using additional metadata. In subject indexing, where co-occurring items are diverse, using the content is more effective than using the partial item set.

Coming back to *Q4: How can we design multimodal representation learning models that jointly process text and graph data for document-based recommendation tasks?*, we have shown that textual side information is crucial to tackle recommendation tasks in which item co-occurrence does not entail the similarity of the two items. When item co-occurrences do entail relatedness of items, then using text data as side information leads to substantial improvements for autoencoder-based methods.

Chapter 7

Lifelong Learning on Evolving Graphs

In this chapter, we apply the lifelong learning paradigm to tackle the challenges of evolving graphs and answer the question *Q5: How can we adapt representation learning models for text and graph data that evolve over time?*. This chapter is based on material from a contribution to the Representation Learning on Graphs and Manifolds workshop at ICLR'19 [GVS19], material published in the International Joint Conference on Neural Networks, IJCNN 2022 [GFZ+21], and an extension that is currently under consideration for publication in a journal [GVF+21].

Large-scale graph data in the real world are often dynamic rather than static. The data change with new nodes, edges, and even classes appearing over time, such as in citation graphs and collaboration graphs, which typically come with textual node attributes. Here, we tackle the problem of node classification in evolving graphs and investigate retraining strategies, history sizes, and the automatic detection of new classes.

As a motivation, we consider the compelling argument by Geirhos et al. [GJM+20] that neural networks (as humans) favor the easiest way to solve a task. The authors conclude that evaluating learning systems on standard benchmarks is not sufficient to assess their quality. Rather, it is necessary to evaluate their performance under more challenging testing conditions, such as real-world scenarios. In such scenarios, the assumption of independent and identically distributed data no longer holds. This aspect is even more crucial when dealing with graph data. Apart from the raw conditional distribution, $p(y|\boldsymbol{x})$, the graph structure also evolves over time, i. e., new nodes and edges are being inserted into the graph. Worse still, the set of classes itself, Y, changes over time with the emergence of new classes.

Moreover, building representation learning models for large and growing graphs is particularly challenging because of the limited scalability of the methods [ZZS+20]. In the standard formulation of graph convolution [KW17], the entire graph has to fit into the GPU memory. Still, simply removing old data points to reduce the required memory is also problematic because they could be connected to very recent data points on the time axis. At the same time, having

well-performing learning systems for evolving graphs is highly valuable because graph representations are versatile and needed in many applications [Ham20].

GNNs [SGT+09] have emerged as state-of-the-art methods in numerous tasks on graph-structured data such as node classification [KW17; HYL17; VCC+18; KBG19], graph classification [YYM+18], link prediction [ZC18], and unsupervised node representation learning [VFH+19]. An intriguing property of GNNs is that they are capable of inductive learning [HYL17]. An inductive model for graph data depends only on the node features and the graph structure given by its edges. In many cases, this is an advantage over models that rely on a static node embedding [HYL17; XRK+20; ZZS+20]. Such a static node embedding would need to be retrained as soon as new data arrive [PAS14]. This is known as the transductive learning scenario (as we have introduced in Section 2.4). On the contrary, *inductively* trained GNNs can be applied to new data – or even a completely different graph – without any retraining because they depend only on node features and edges, but not on a static lookup table for node embeddings.

However, the ability to apply the same model to unseen data also presents challenges that have not been adequately addressed in the literature so far. We assume that we have a node classification model and new data streams over time, i. e., new edges and nodes arrive; then even *new classes* may emerge. This raises several questions.

- Do we need to retrain the model and when do we need to retrain it?
- How much of the past data should be preserved for retraining?
- Is it helpful to preserve implicit knowledge within the model parameters or should we retrain from scratch?
- How much new labeled data are needed for stable training?
- How can we automatically detect in an unsupervised manner whether a new class has emerged in the dataset?

Approach To answer these questions, we frame the problem as an instance of lifelong machine learning [TM95; FWL16; CL18]. In lifelong learning, as illustrated in Figure 7.1, the learner must perform a sequence of tasks $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_t$, and may use knowledge $\mathcal{K}$ gained in previous tasks to perform the task $\mathcal{T}_t$. In our case, each task consists of classifying nodes given an attributed graph. Knowledge $\mathcal{K}$ can be stored explicitly (the training data of previous tasks) or implicitly within

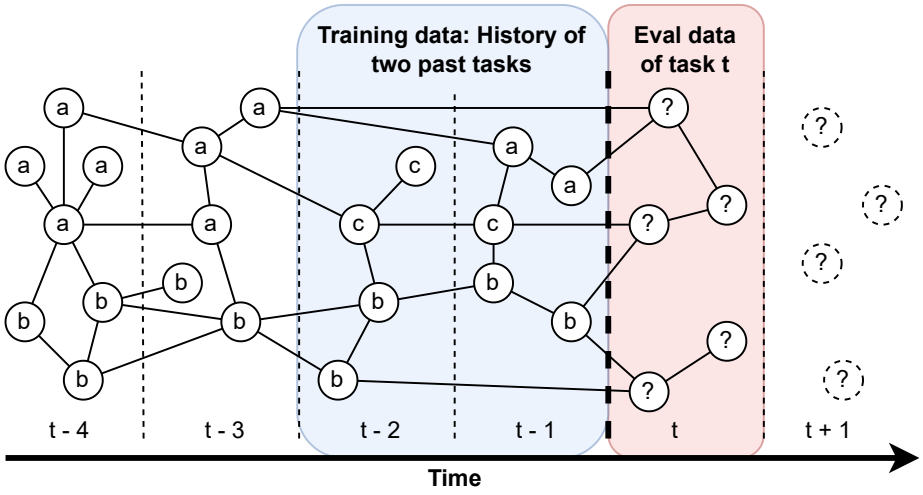


Figure 7.1. Lifelong Open-World Node Classification. At each time t the learner has to classify new vertices of task $\mathcal{T}_t$ (red). The learner may use knowledge from previous tasks to adapt to the current task, eventually cut off by a history size (blue). The current task might come with previously unseen classes. For example, the "c" appears only at task $t-2$ and was subsequently added to the class set. After evaluating each task $\mathcal{T}_t$, we continue with task $\mathcal{T}_{t+1}$.

the model parameters. A particular challenge of lifelong learning in the context of graph data is that nodes cannot be processed independently because models typically take connected nodes into account. We also consider the challenge that the set of classes in task $\mathcal{T}_t$ differs from the classes in previous tasks, which is known as the **open-world classification** [CL18] problem.

To address these challenges, we introduce an incremental training method that retrains the model for each task. In our experiments, we thoroughly evaluate representative and scalable GNN architectures and a graph-agnostic multilayer perceptron (MLP). We use a *history size* that limits the amount of past data (called here: *explicit knowledge*) available for training and compare *limited history-size* retraining against unlimited *full-history* retraining. Furthermore, we compare the reuse of model parameters from previous tasks (*warm restart*) against retraining from scratch (*cold restarts*) to analyze the influence of *implicit knowledge*.

However, absolute history sizes hardly generalize to other graphs that evolve at a different pace. Therefore, we introduce a measure tdiff_k to derive comparable history sizes between datasets. The measure operates on time differences in the

local neighborhood of each node and, thus, captures temporal connectivity patterns within the graph. We show that this measure is equivariant to different temporal granularities of the data.

Comparison of Inductive and Transductive Learning However, we first go one step back and investigate the edge between inductive and transductive learning on three static graph datasets. We compare models that were (pre-)trained only on the labeled subgraph of the training data against models that are trained on the entire graph. This is an important distinction, as it corresponds to going from task t to task $t + 1$. We find that the performance of models that were pretrained on the labeled subgraph remains on the same level when the new unlabeled data is added.

Lifelong Learning on a Sequence of Tasks In a next step, we confirm that incremental training is necessary when applying GNNs to evolving graphs. We compare incrementally training models with once-trained models in the lifelong learning scenario. Our results show that, as expected, the performance level of static models decreases over time, whereas incrementally trained models maintain a constant level of accuracy. Subsequently, we continue investigating the task sequence setting and compare using different history sizes and the difference between cold and warm restarts. We find interesting interactions between explicit and implicit knowledge. When fewer explicit knowledge is available (smaller history size), the influence of implicit knowledge (warm restarts) is stronger. Conversely, reusing model parameters helps to be more data efficient. To facilitate our analyses, we provide three new datasets: one co-authorship and two citation graph datasets with different degrees of changes in the class set. On these datasets, we have experimented with 48 different configurations, namely six GNN base models $\times$ four history sizes $\times$ cold restarts and warm restarts. Apart from this basic lifelong learning scenario, we consider two more scenarios that are motivated by the challenges of real-world graph data: robustness to limited labeled data and unseen class detection.

Limited Labeled Data In the first scenario, we limit the number of training nodes that come with labels. We use the best performing base model and the most challenging dataset to investigate limited labeled data with 8 different label rates. The motivation for this experiment is that collecting ground-truth labels in

real-world applications is expensive. Oftentimes, it is only feasible to manually annotate a small fraction of the data. We simulate this in a dedicated experiment by varying the label rate in our datasets. Our results are also confirmed in this setting. We further observe a slight trend that, with decreasing label rates, using warm restarts becomes even more important.

Unseen Class Detection We extend our experimental setting in a way that the models need to actively detect instances of unseen classes, while classifying instances from known classes as usual. We do *not* introduce any additional data (a "garbage class") that could be used as supervised targets for unseen classes. Instead, we focus on unsupervised detection of instances from unseen classes, i. e., by analyzing the distribution of the model outputs per class (logits). In order to stay close to real-world applications, we further focus on crisp decisions rather than a soft outlier score. The model has to decide fully autonomously whether a node comes from a previously unknown class such that its classification can be rejected without human intervention.

In order to automatically detect unseen classes, we propose an adaptation of the deep open classification (DOC) [SXL17] method for text classification to the graph domain. We further integrate this class-detection mechanism as a generic module in our incremental training algorithm. Our main finding is that when extending DOC to graphs (which we call gDOC), it is crucial to account for class imbalance by reflecting the class distribution in the binary-cross-entropy loss function. Surprisingly, we also find some indication that using fewer past data together with warm restarts is even beneficial for unseen class detection when there is a high number of classes.

The related work is discussed below. A problem formulation is provided in Section 7.2. We explain our proposed training procedure, the new measure to determine an optimal history size for graphs, and the GNN methods used in Section 7.3. Our datasets are described in Section 7.4, while the results of our experiments are reported in Section 7.5. We discuss the results in Section 7.6, before we summarize the chapter.

7.1 Related Prior Work

We discuss work on lifelong machine learning and lifelong learning on graphs. Subsequently, we discuss methods for evolving graphs, out-of-distribution detection, as well as methods regarding history sizes.

7.1.1 Lifelong Learning

Lifelong Learning on Image Data Lifelong learning, or continual learning [LR17]), has been present in machine learning research since the mid 1990s [TM95; Thr98; SYL13; Liu17]. The goal of lifelong learning is to develop approaches that can adapt existing models to new tasks. Although similar on a superficial level, it differs from online learning [HPW05], in which the focus is on processing a data stream efficiently. Ruvolo and Eaton [RE13] introduced a lifelong learning algorithm with convergence guarantees that employs multitask learning so that later tasks can improve earlier tasks. Fei, Wang, and Liu [FWL16] analyzed SVMs in a lifelong learning setting and introduced cumulative learning. Cumulative learning is related to our approach since we consider that some data is shared among the tasks. Lopez-Paz and Ranzato [LR17] have introduced a gradient episodic memory framework for the image domain, where examples can be processed independently, and tackle the catastrophic forgetting problem, i. e., the loss of previously learned information when new information is learned [Rob95]. For an overview of lifelong learning, we refer to a recent textbook [CL18].

Similarly to our work, Wang, Chen, Li, and Chen [WCL+21] decompose lifelong machine learning into the subproblems of rejecting unknown instances, classifying accepted instances, and reducing the cost of learning. However, the work of Wanget al. is on image data, in which the examples are independent of each other, and therefore the challenges of dealing with graph data are not reflected. Another promising approach to lifelong learning and, in particular, class-incremental learning, is iCaRL [RKS+17], in which the prototype vectors of known classes are stored and the classification is carried out by taking the closest distance to these prototypes. However, applying this method to graph data is nontrivial because the nodes are not independent from each other, but connected via edges. For an overview of lifelong learning in general (not specific to graphs), we refer to a recent textbook [CL18].

Lifelong Learning on Evolving Graphs We now focus on the related work on lifelong learning *on graphs*. The challenge of dealing with graph data is a special challenge for lifelong learning approaches. This is because in graphs the nodes are not independent of each other because they are connected through edges. Related work on lifelong learning *on graphs* is still rather limited. We refer to Febrinanto et al. [FXM+22] for a recent survey that covers five recent works on graph lifelong learning.

The most similar approach to ours is Experience Replay GNN [ZC21], which proposed to overcome catastrophic forgetting [Rob95], i. e., the problem of previous knowledge being quickly forgotten when models are adjusted to new tasks. The Replay GNN adapts to new tasks with the help of an experience replay buffer. The buffer holds a subset of the graph that is determined on the basis of different selection strategies: mean of features, coverage maximization, or influence maximization. This work is conceptually similar to ours. However, we use the time information from the nodes in conjunction with a history size to determine which part of the graph is kept in memory. Wang, Qiu, Gao, and Scherer [WQG+22] proposed a very different strategy to tackle lifelong learning on graphs. The main goal was again to alleviate catastrophic forgetting. The authors explored a preprocessing step that transforms the node classification task into a graph classification task, i. e., each node is converted into a feature graph. Therefore, nodes become independent so that they can follow the lifelong learning approach from Lopez-Paz and Ranzato [LR17] (see above).

Continual-GNN [WSW+20] addressed the issue of catastrophic forgetting with a regularization approach. The authors detected new patterns in the data (but not involving any new classes) with an information propagation method. Then, they used a combination of experience replay and model regularization to avoid catastrophic forgetting. The result was that their approach leads to performance comparable to model retraining. In relation to this work, we also compare our lifelong-learned models against models retrained from scratch for each task (cold restart), but additionally consider other conditions such as the history size. Another recent approach [CWG+22] uses neural architecture search to find a suitable model architecture for lifelong learning on graphs. In particular, the proposed approach focuses on multimodal inputs, such as features extracted via BERT [DCL+19] and vision Transformers [DBK+21], rather than dealing with new classes and how to detect them.

So far, none of the related works on lifelong learning in graphs have considered the problem of detecting unseen classes and rejecting the classification of the

respective nodes. Knowing when a model is likely to make mispredictions is a crucial property for deploying reliable systems in practice, which motivates us to explore the combination of lifelong learning on graphs and unseen class detection. Moreover, also labeled data is often not fully available in real-world conditions, which we investigate here because it has not yet been considered in previous work on lifelong graph learning, too.

7.1.2 Graph Neural Networks

Dwivedi et al. [DJL+20] distinguish between isotropic and anisotropic GNN architectures. In isotropic GNNs, all edges are treated equally. Apart from graph convolutional networks [KW17], examples of isotropic GNNs are the mean aggregation variant of graph sampling and aggregating (GraphSAGE) [HYL17], differentiable pooling [YYM+18], and graph isomorphism networks [XHL+19]. In anisotropic GNNs, the weights for the edges are calculated dynamically. Instances of anisotropic GNNs include graph attention networks [VCC+18] and MoNet [MBM+17]. There are further approaches, which have been specifically proposed to scale GNNs to large graphs. These approaches fall into two categories: subgraph sampling [HYL17; HZR+18; CLS+19; ZZS+20] and separating neighborhood aggregation from the neural network [WJZ+19; FRE+20; BKP+20]. From each of these four categories (anisotropic versus isotropic GNN, and preprocessing versus sampling), we select one representative for our experiments, which will be discussed in more detail in Section 7.3.4).

Pretraining Graph Neural Networks Hu et al. [HFC+19] have explored different strategies for pretraining GNNs, including self-supervised objectives derived from the graph structure. Hu et al. [HLG+20] similarly explore pretraining strategies. They find that previous strategies, comprising local (node-level) and global (graph-level) objectives, yield only small improvements and even decrease the performance on some downstream tasks. Instead, their approach of graph-level property prediction and structural similarity prediction improves downstream performance. While these unsupervised pretraining approaches would be technically applicable, our setting is different in the sense that we do have supervised data from previous tasks with which we can pretrain for the next task.

Dynamic Graphs Different methods for *dynamic graphs* have been proposed. This body of work includes dynamic embedding methods [NLR+18; LNR+20],

autoencoder-based methods [GKH+18; GCC20], GNNs for graphs with a fixed node set [TDW+17; SDV+18; KZL19; TFB+19; MRM20; SWG+20; RCF+20], and inductive GNN methods that can deal with previously unseen nodes [PDC+20; XRK+20]. These methods focus on the case with dynamic signal (cf. Section 2.3.1). That means that a node can be in class a at time t and in class b at time $t+1$. However, in our case, the output variable is static, but the graph itself is evolving with new nodes, edges, and classes appearing over time. In contrast, the related approaches assume a static set of nodes, which renders them applicable to the problem of lifelong learning in evolving graphs.

7.1.3 Unseen Class Detection and Out-of-Distribution Detection

Unseen class detection, or open-world learning, is considered a subcategory of lifelong learning [CL18]. Still, more general methods for out-of-distribution (OOD) detection are also related to the problem of detecting unseen classes.

Unsupervised Out-of-Distribution Detection A key challenge is that softmax activation, which is typically used as the final layer for classification, leads to highly confident mispredictions even when the input data is far away from the training distribution. To address this, Liang, Li, and R. [LLR18] resort to temperature scaling, while Lee, Lee, Lee, and Shin [LLL+18] propose to use Mahalanobis distance. Both approaches rely on dedicated preprocessing of the input. Macêdo et al. [MRZ+21] and Macêdo and Ludermir [ML21] replace the softmax activation by an entropy-aware IsoMax activation. However, we are particularly interested in methods that emit a crisp decision whether the classification of an instance should be rejected. In this regard, there are several approaches to detect new classes with classic machine learning methods [MGK+11; BB16; FWL16].

Wu, Pan, and Zhu [WPZ20] have used variational graph autoencoders for uncertain node representation learning. They generate multiple versions of features and test the certainty of a node belonging to a known class.

Supervised Out-of-Distribution Detection Other approaches rely on explicit outlier data that can be used for supervised training of the outlier module [DGB18; HMD19]. This is difficult to apply here because we do not distinguish between out-of-distribution and in-distribution but between previously seen classes and previously unseen classes. When we had the appropriate training data for the unseen classes, we could train directly on them rather than considering them as OOD.

For a detailed discussion of OOD methods, we refer to recent surveys [YZL+21; PSC+21].

Crisp and Unsupervised Unseen Class Detection We are particularly interested in methods that emit a crisp decision whether the classification of an instance (a new vertex) should be rejected. In this regard, there are several approaches to detect new classes using classic machine learning methods [MGK+11; BB16; FWL16]. For example, Wu et al. [WPZ20] have used variational graph autoencoders for uncertain node representation learning. They generate multiple versions of features and test the certainty of a node belonging to a known class.

In Deep Open Classification (DOC) [SXL17], the authors proposed a method for the detection of new classes in text categorization. To perform the detection, the final softmax activation of a neural network is replaced by elementwise sigmoid activation. Then, they derived a threshold for unseen class detection by measuring the logits' standard deviation across the training set. Their experiments on datasets with a uniform class distribution indicated that DOC is preferable to OpenMax [BB16] and cbsSVM [FWL16]. Thus, we use a DOC-inspired module on top of graph neural networks to develop our new gDOC method for unseen class detection.

7.1.4 Summary

In summary, lifelong learning on graphs is so far a fairly unexplored topic (only six previous works) and so far an unexplored topic. In particular, none of the discussed works analyzes the problem of *open-world classification* in graph data and how much past training data is necessary—or how few are enough—to maintain good predictive power.

7.2 Problem Formulation

We define the problem of open-world classification of nodes in an evolving graph as a form of lifelong learning [TM95; Thr98; CL18]. We recall that in lifelong learning, a model is gradually adapted to a sequence of t learning tasks, $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_t$, and has accumulated knowledge $\mathcal{K}$ learned in these past tasks. At time $t+1$, it is faced with a new learning task $\mathcal{T}_{t+1}$. The learner is able to use previous knowledge to help perform the new learning task $\mathcal{T}_{t+1}$. We cast this definition into a lifelong

graph learning problem by considering each task $\mathcal{T}_t := (\mathcal{G}_t, \boldsymbol{X}^{(t)}, \boldsymbol{y}^{(t)})$ to be a node classification task with graph $\mathcal{G}_t = (V_t, E_t)$, corresponding node features $\boldsymbol{X}^{(t)} \in \mathbb{R}^{|V_t| \times D}$, and node labels $\boldsymbol{y}^{(t)} \in \mathbb{N}^{|V_t|}$. We denote the set of all classes at time t as $\mathbb{Y}_t$.

To ensure that past knowledge is helpful in performing $\mathcal{T}_t$, we impose $\mathcal{G}_{t-1} \cap \mathcal{G}_t \neq \varnothing$. We assume that the features and labels of nodes do not change: $\boldsymbol{X}_u^{(t-1)} = \boldsymbol{X}_u^{(t)}, \boldsymbol{y}_u^{(t-1)} = \boldsymbol{y}_u^{(t)}$ if $u \in V_{t-1} \cap V_t$. The task is to predict the class labels for the new nodes $V_t \backslash V_{t-1}$. These nodes may come with new unseen classes, as $\mathbb{Y}_t$ may differ from $\mathbb{Y}_{t-1}$. In other words, we are considering the dynamic structure, the static signal case of dynamic graphs (cf. Section 2.3.1). Note that such changes can still be modeled by inserting a new node and removing the old one.

Furthermore, we analyze the effect of a history size c, which limits the available past data. We call these past data *explicit knowledge*. In this case, we set $\tilde{\mathcal{T}}_t := (\tilde{\mathcal{G}}_t, \tilde{\boldsymbol{X}}^{(t)}, \tilde{\boldsymbol{y}}^{(t)})$ with $\tilde{\mathcal{G}}_t := \mathcal{G}_t \backslash (\mathcal{G}_1 \cup \mathcal{G}_2 \cdots \cup \mathcal{G}_{t-c-1})$, and remove the corresponding features and labels to construct $\tilde{\boldsymbol{X}}_t$ and $\tilde{\boldsymbol{y}}_t$. Still, *implicit knowledge* acquired in previous tasks, e. g., within the model parameters, may be used for the task $\tilde{\mathcal{T}}_t$.

We further distinguish between several variants of this problem statement for lifelong learning on graphs.

Two-task Setting This is a simplified setting with only two tasks. This setting is suitable for applying our approach to any (non-temporal) standard data set by defining $\mathcal{T}_1$ as the training graph, and $\mathcal{T}_2$ as the test graph. We use this setup for experiments that compare transductive and inductive learning in Section 7.5.1. The idea is to compare models that have been inductively pretrained on the labeled subgraph of the training data against models that have been trained transductively on the training plus unlabeled test data.

Task-sequence Setting We have a chronological sequence of tasks $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_T$, with new vertices appearing over time. All past vertices have ground truth labels, which can be used for training. Unseen classes are present and also considered in the evaluation, but no special methods are employed to actively detect unseen classes. This setting is considered in the experiments of Section 7.5.2 and Section 7.5.3.

Task-sequence Setting with Limited Labeled Data In this variant of the standard setting, only a fraction of ground truth labels are available for training, rather than all labels of all past nodes, as in the previous setting. This setting is reflected in the experiments of Section 7.5.4.

Task-sequence Setting with Unseen Class Detection In the final variant, we seek to analyze the models' capabilities to actively detect unseen classes. In addition to classification, models now need to make a binary decision per node, whether it belongs to a previously known class (in-distribution) or to a new, unseen class (out-of-distribution). This setting is reflected in the experiments of Section 7.5.5.

7.3 Methods

In the following, we introduce the incremental training procedure, by which we approach the lifelong learning problem. Subsequently, we describe our gDOC method for unseen class detection, before we describe our method tdiff_k to harmonize absolute window sizes. Finally, we describe the base GNN models that we will use in the experiments.

7.3.1 Incremental Training for Lifelong Graph Learning

Our incremental training algorithm for GNNs is shown in Algorithm 1. We assume that we have a sequence of T tasks $\mathcal{T}_1, \ldots, \mathcal{T}_T$ and a model f with parameters θ. Throughout the sequence of tasks, the graph changes in the sense that nodes and edges are inserted and deleted. Crucially, the new nodes can come with new classes that have not been part of the training data before. To address these changes, we explore a simple but effective incremental training technique to adapt neural networks to new graph-structured tasks.

As a preparation for task $\mathcal{T}_t$, we retrain f on the labels of $\mathcal{T}_{t-1}$ to obtain $\theta^{(t)}$. Whenever l new classes appear in the training data, we add a corresponding number of parameters to the output layer of $f^{(t)}$. Therefore, we have $|\theta^{(t)}_{\text{output weights}}| = |\theta^{(t-1)}_{\text{output weights}}| + l \cdot n_{\text{hidden}}$ and $|\theta^{(t)}_{\text{output bias}}| = |\theta^{(t-1)}_{\text{output bias}}| + l$, where n_{hidden} is the number of hidden units in the penultimate layer. These new parameters that correspond to the new classes are initialized randomly. For the other parameters, we consider two options in our incremental training procedure: *warm restarts* and *cold restarts*. With *cold restarts*, we reinitialize $\theta^{(t)}$ and retrain from scratch. On the contrary, when using *warm restarts*, we initialize the parameters for training on task $\mathcal{T}_t$ with the final parameters of the previous task $\theta^{(t-1)}$. Furthermore, we consider a generic module (lines 14–15) for unseen class detection in the incremental training algorithm. This operates on the logits of the final output layer and determines

whether the classification of a particular node should be rejected because it belongs to a class that was not part of the training data.

Algorithm 1: Incremental training for lifelong graph learning under cold-start vs. warm-start condition

Input : Sequence of tasks $\tilde{\mathcal{T}}_0, \cdots, \tilde{\mathcal{T}}_T$, model f with parameters θ, flag for cold or warm restarts

Output: Predicted labels for new nodes of each task

known_classes $\leftarrow \varnothing$

$\theta \leftarrow$ initialize_parameters()

`/* Iterate through task indices */`

for $t \leftarrow 1$ **to** T **do**

  new_classes $\leftarrow$ set$(\tilde{\boldsymbol{y}}^{(t-1)}) \backslash$known_classes

  if *new_classes* $\neq \varnothing$ **then**

    $\theta' \leftarrow$ expand_output_layer(θ, |new_classes|)

  end

  $\theta' \leftarrow$ initialize_parameters()

  if $t > 1$ **and** *do_warm_restart* $=$ `TRUE` **then**

    `/* Reuse previous model */`

    $\theta' \leftarrow$ copy_existing_parameters(θ)

  end

  `/* Train model with labels from previous task */`

  $\theta' \leftarrow \text{train}(\theta', \tilde{\mathcal{G}}_{t-1}, \tilde{\boldsymbol{X}}^{(t-1)}, \tilde{\boldsymbol{y}}^{(t-1)})$

  `/* Predict classes for new nodes */`

  $\tilde{\boldsymbol{y}}_{\text{pred}} \leftarrow \text{predict}(\theta', \tilde{\mathcal{G}}_t, \tilde{\boldsymbol{X}}^{(t)})$ for $nodes V_t \backslash V_{t-1}$

  `/* Out-of-distribution detection */`

  $\boldsymbol{m}_{\text{ood}}^{(t)} = \text{unseen\_class\_detection}(\hat{\boldsymbol{y}}_{\text{logits}}^{(t)})$

  $\hat{\boldsymbol{y}}_{\text{pred},i}^{(t)} = \begin{cases} \texttt{OOD} & \text{if } \boldsymbol{m}_{\text{ood},i}^{(t)} = \texttt{TRUE} \\ \arg\max(\hat{\boldsymbol{y}}_{\text{logits}^{(t)},i}), & \text{otherwise} \end{cases}$

  `/* Prepare for next task */`

  known_classes $\leftarrow$ known_classes $\cup$ new_classes

  $\theta \leftarrow \theta'$

end

7.3.2 Unseen Class Detection

A successful model for lifelong learning would not only classify new data into known classes, but would also detect when an instance belongs to a previously unseen class. We seek to develop a generic method that is not specific to any particular GNN architecture. Thus, we take inspiration from the Deep Open Classification (DOC) [SXL17] approach that has been proposed for text classification. The key idea is to replace the final softmax activation with element-wise sigmoid activation. Hence, the training objective becomes binary cross-entropy rather than categorical cross-entropy. Then, thresholds on the logit distribution over all known classes are used to determine whether the new example belongs to an already known class or not. Below, we briefly summarize the key risk reduction technique proposed in the original DOC, before we describe our extensions.

Thresholds and Risk Reduction in DOC To make a clear decision, a threshold is necessary to determine whether a vertex is considered out of distribution (OOD) at the test time. When the output for all the known classes falls below the threshold, the classification of that vertex is rejected, i. e., the vertex is considered OOD. Such thresholds can be global or class-specific. A natural choice for a global threshold τ is the inflection point of the sigmoid function, i. e., setting $\tau = 0.5$. However, estimating class-specific thresholds can further reduce the risk of incorrectly rejecting the classification of a known class. A strategy to estimate class-specific thresholds is to consult the standard deviation of logits on the training data [SXL17]. To determine a threshold τ_i for class i, the risk reduction technique proposed in DOC [SXL17] collects all model outputs for instances of class i. For all these outputs $\hat{y} \in [0, 1]$, a mirror point $1 + (1 - \hat{y})$ is created, assuming a Gaussian distribution with mean 1. On this distribution, the standard deviation SD_i is calculated to assign the class-specific threshold $\tau_i := \max\{\tau_{\min}, 1 - \alpha \cdot \mathrm{SD}_i\}$, where α is a scaling factor for the standard deviation and $\tau_{\min}$ is the minimum threshold. For α, the original work suggests a value of 3. The authors use a fixed $\tau_{\min} = 0.5$.

$$\tau_i := \max\{\tau_{\min}, 1 - \alpha \cdot \mathrm{SD}_i\}$$

where α is a scaling factor for the standard deviation and $\tau_{\min}$ is the minimum threshold. The original DOC [SXL17] has used a fixed minimum threshold of $\tau_{\min} = 0.5$, while we leave it open to investigate different values for $\tau_{\min}$. For α, the original work suggests a value of 3.

Extension for Dealing with Class Imbalance (gDOC) Here, we transfer the DOC method to the graph domain. This comprises changing the base model from a 1D-CNN on text to a GNN operating on graphs, as well as changing the loss function for node classification from categorical cross-entropy to binary cross-entropy. In this way, we can employ the same strategy for detecting new classes that was used in the original work on DOC. Throughout this work, we denote this adaptation from text to graph data as DOC.

We propose an extension, which we denote as gDOC, to make DOC more suitable for lifelong learning on graphs, where we have to deal with a highly imbalanced class distribution. Aside from using a GNN model to emit the logits to begin with, we also adjust the loss scaling of binary cross-entropy to account for class imbalance, which is inevitable in real-world graph data. This is particularly important for unseen class detection, because here the magnitude of all outputs is relevant for the final decision, rather than only their maximum value. In detail, if class i appears n^+ times in the training data, we multiply the loss of output i by the factor $\frac{n-n^+}{n^+}$. This is a standard weighting procedure for binary cross-entropy that increases the loss according to the fraction of positive versus negative examples within the training data (cf. [AAC+19]). We denote this variant as gDOC. Furthermore, our experiments will carefully investigate different values for τ_{min}, while the original DOC [SXL17] used a fixed minimum threshold of $\tau_{\text{min}} = 0.5$. Similarly, we also closely investigate the effect of the risk reduction factor α.

7.3.3 Measure of k-Neighborhood Time Differences

Real-world graphs grow and change at different speeds [AS14]. Some graphs change quickly, such as social networks, while others evolve rather slowly, such as citation networks. Furthermore, the graphs show different change behavior, i. e., different patterns in how nodes and edges are added and removed over time. Therefore, depending on the specific graph data, a different history of the data must be used for training to take these factors into account. To obtain absolute history sizes that are comparable across different temporal graphs, a measure is needed that provides a history size that is agnostic to the specific change dynamics of the graph (slow vs. fast). Below, we introduce such a measure.

Formal Definition of the tdiff_k **Measure** The k-neighborhood Time Difference Distribution measure tdiff_k [GVF+21] enumerates the distribution of time differences within the k-hop neighborhood of each node. This corresponds to the *receptive*

field [CZS18] of a GNN with k-many graph convolutional layers. Intuitively, we collect the time differences between all pairs of nodes v and w, which are reachable within at most k edges. We aggregate these time differences based on frequency, i. e., we obtain the number of times a certain time difference has been observed between v and w in the dataset. On this distribution of time differences (represented as a multiset), we compute the percentiles and use them as candidate history sizes.

7.3.1 Definition (k-Neighborhood Time Difference Distribution)**.** Given graph $\mathcal{G} = (V, E)$ and let $\mathcal{N}^k(u)$ be the k-hop neighborhood of vertex $u \in V$ with respect to E, i. e., the set of nodes that are reachable from u by traversing at most k edges. Let time : $V \to \mathbb{N}$ be a function that returns the time information for each node v (timestamp metadata), e. g., the year of publication when considering a citation graph. We then define $\text{tdiff}_k(\mathcal{G})$ as *multiset of time differences*, computed over all vertices $u \in V$ to their at most k-distant neighboring vertices $v \in \text{BFS}_{\mathcal{G}}^{(k)}(u)$ that occurred before node u.

$$\text{tdiff}_k(\mathcal{G}) := \{\text{time}(u) - \text{time}(v) \mid \forall u \in V,\, \forall v \in \text{BFS}_{\mathcal{G}}^{(k)}(u) \text{ with } \text{time}(v) \leqslant \text{time}(u)\}$$

Here, breadth-first search $\text{BFS}_{\mathcal{G}}^{(k)}(u)$ recursively enumerates adjacent nodes of the graph $\mathcal{G}$ starting with node u, up to a maximum path length of k. The multiset tdiff_k, which maps each time difference to the respective number of occurrences, is interpreted as a distribution over time differences. It is used to analyze a dataset's temporal distribution (percentiles) and to make datasets comparable. Figure 7.2 presents an exemplary computation of the k-neighborhood time differences tdiff_2 on a graph with five nodes and five edges. In this example, the 25th percentile of tdiff_2 is 0, the 50th is 1 (also known as the median), and the 75th is 2. The 100th percentile, or the maximum time difference in a two-hop distance is 4.

The tdiff_k measure is used in our experiments to compare models trained with a *limited history size* against models trained with *full history*. Therefore, we calculate the 25th, 50th, and 75th percentiles of the tdiff_k distribution, which we then compare against the entire graph (100th percentile) to analyze the influence of explicit knowledge.

Equivariance to Temporal Granularity Any good measure to determine the discrete history sizes $c : (V, E, t) \mapsto \mathbb{N}$ in evolving graphs should be *equivariant to granularity* to ensure comparability between different datasets and different granularities. This means that if we change the perspective, for instance, from years

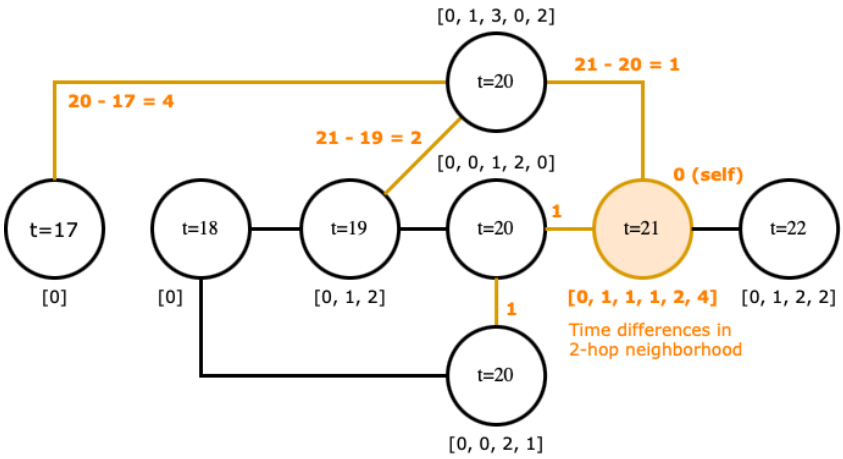


Figure 7.2. Example of time differences $\text{tdiff}_2(G)$ for hops at distance of up to 2 from each node. Node annotations show the time difference to all nodes in the two-hop neighborhood, i. e., its contribution to the multiset $\text{tdiff}_2(G)$. The calculation for the node with time $t = 21$ is highlighted in orange.

to months, we should get history sizes that are about 12 times larger (on the same data).

Formally, consider two different time measurement functions $t_y, t_m : \mathcal{V} \to \mathbb{N}_{>0}$ whose values differ by a constant factor $a \in \mathbb{R}^+$ such that $t_y(u) = \lfloor \frac{t_m(u)}{a} \rfloor$ for all $u \in \mathcal{V}$. For example, $a = 12$ when comparing the granularities of months t_m and years t_y. In fact, two arbitrary discrete-time measurement functions differ by a constant factor, one coarser-grained (larger denominator) than the other, or both equal ($a = 1$). Then, the derived history sizes should not differ by more than the ratio between the granularity values, i. e., for the measure c to determine the history sizes it should hold that $a \cdot c(V, E, t) - a \leqslant c(V, E, t') \leqslant a \cdot c(V, E, t) + a]$ where $t' \in [a \cdot t - a; a \cdot t + a]$ for all $u \in V$. When the difference in granularity is 12, a good measure c should return a history size times 12 plus/minus 12, when we switch the perspective from the year level t to the month level t' (ratio: $a = 12$) on the same data. This property is also crucial for comparable history sizes across datasets with different temporal granularities. A proof that tdiff_k is equivariant to temporal granularity in Section F.1.

7.3.4 Base Graph Neural Network Models

The techniques described above can be applied to arbitrary base models. In the following, we describe the base models that we considered for our experiments. The success of graph convolutional networks (GCNs) [KW17] has triggered a resurgence of interest in graph neural networks [SGT+09]. In a generic formulation, the hidden representation of node i in layer l is defined as:

$$\boldsymbol{h}_i^{(l+1)} = \sigma\left(\sum_{j \in i \cup \text{Nbrs}(i)} \frac{1}{c_{ij}} \boldsymbol{W}^{(l)} \boldsymbol{h}_j^{(l)}\right)$$

where $\text{Nbrs}(\cdot)$ refers to the set of adjacent nodes and σ is a non-linear activation function. The normalization factor c_{ij} depends on the respective model: Kipf's original graph convolutional network [KW17] use $c_{ij} = \sqrt{\text{deg}_i \cdot |\text{deg}_j|}$. GraphSAGE with mean aggregation uses $c_{ij} = |\mathcal{N}(i)|$, and GAT use learned attention weights instead of $\frac{1}{c_{ij}}$ which are computed by a non-linear transformation from the concatenation of $h_i^{(l)}$ and $h_j^{(l)}$.

For our experiments, we systematically select representative GNN architectures, as well as scalable GNN techniques for our experiments on lifelong learning. For this, we consider the different types of GNNs anisotropic vs. isotropic and standard vs. scalable approaches, as outlined in Section 7.1. Our goal is to understand how different approaches of GNNs react to situations of changing graphs and new classes.

We select graph attention network (GAT) [VCC+18] as representative of the class of anisotropic GNNs. In GATs, the representations in layer $l+1$ for node i are computed as follows:

$$\hat{\boldsymbol{h}}_i^{l+1} = \alpha_{ii}^l \boldsymbol{h}_i^l + \sum_{j \in \text{Nbrs}(i)} \alpha_{ij}^l \boldsymbol{h}_j^l$$

$$\boldsymbol{h}_i^{l+1} = \sigma(\boldsymbol{U}^{(l)} \hat{\boldsymbol{h}}_i^{(l+1)})$$

where $\text{Nbrs}(i)$ is the set of adjacent nodes to node i, U^l are learnable parameters, and σ is a non-linearity. The edge weights α_{ij} are computed by a self-attention mechanism based on h_i and h_j, i. e., the softmax of $a(\boldsymbol{U}^l \boldsymbol{h}_i \parallel \boldsymbol{U}^l \boldsymbol{h}_j)$ over the edges, where a is an MLP and $\cdot \parallel \cdot$ is the concatenation operation.

We select GraphSAGE-Mean [HYL17] as a representative for isotropic GNNs because its special treatment of each node's own previous representation has been shown to be beneficial [DJL+20]. The representations of self-connections are

concatenated with averaged neighbors' representations before multiplying with the parameters. In GraphSAGE-Mean, the procedure for obtaining representations on layer $l+1$ for node i is given by the equations:

$$\hat{\boldsymbol{h}}_i^{l+1} = \boldsymbol{h}_i^l \parallel \frac{1}{\deg_i} \sum_{j \in \mathrm{Nbrs}(i)} \boldsymbol{h}_j^l$$
$$\boldsymbol{h}_i^{l+1} = \sigma(\boldsymbol{U}^l \hat{\boldsymbol{h}}_i^{l+1})$$

We select simplified graph convolutional network (SGC) [WJZ+19] as a representative to shift the neighborhood aggregation to preprocessing. SGC is a scalable variant of GCN [KW17] that admits regular minibatch sampling. SGC removes nonlinearities and collapses consecutive weight matrices into a single one. Thus, SGC can be described by the equation

$$\hat{\boldsymbol{y}}_{\mathrm{SGC}} = \mathrm{softmax}(\boldsymbol{S}^k \boldsymbol{X} \boldsymbol{W})$$

where $\boldsymbol{S}$ is the normalized adjacency matrix and $\boldsymbol{W}$ is the weight matrix. The hyperparameter k has a similar effect as the number of layers in regular GCNs. Instead of using multiple layers, the k-hop neighborhood is computed by $\boldsymbol{S}^k$, such that $\mathbf{S}^k \boldsymbol{X}$ can be precomputed. This makes SGC efficient, while not necessarily harming the performance [WJZ+19].

Furthermore, we include GraphSAINT [ZZS+20] as state-of-the-art subgraph sampling technique. In GraphSAINT, entire subgraphs are sampled for training GNNs. Subgraph sampling introduces a bias that is counteracted by normalization coefficients for the loss function. The authors propose different sampling methods: node sampling, edge sampling, and random-walk sampling. We use the best-performing random-walk sampling for our experiments. The underlying GNN is exchangeable, but the authors suggest to use jumping-knowledge networks (JKNets) [XLT+18]. JKNets introduce skip-connection to GNNs. Each hidden layer has a direct connection to the output layer, in which the representations are aggregated, e. g., by concatenation. To isolate the effect of GraphSAINT sampling, we also include JKNets in our experiments.

7.4 Datasets

Adapting models to new data is an important problem whenever machine learning models are deployed in production. However, many graph benchmark datasets are

stripped off any temporal data, which is needed to divide the data into realistic partitions, i. e., tasks. We scanned the literature (e. g., [DJL+20; PDC+20; XRK+20]) and common collections (OpenGraphBenchmark [HFZ+20], KONECT[1], and PyTorch Geometric Temporal[2]) for datasets that meet the following criteria:

- Attributed nodes
- Node labels
- Time information on the nodes
- *Evolving* set of nodes (and thus also edges) over time
- *Evolving* set of classes over time

Surprisingly, graph datasets that meet these criteria are rare. Still, the problem is extremely relevant for practical applications, such as social networks or publication metadata. In those datasets with time information, either the graph is static or the set of classes is static. Concurrent work on lifelong learning synthesizes an ordering of nodes in standard datasets [WQG+22]. In this work, we seek to understand how our methods deal with *naturally* evolving datasets. For our first experiment, we use two different splits on standard benchmark datasets, which are described next. For the other three lifelong learning experiments, we constructed three entirely new datasets that we describe thereafter.

7.4.1 Static Graph Datasets

We use standard citation datasets: Cora, Citeseer, and PubMed [SNB+09] for our first experiments on transductive versus inductive learning. Nodes are research papers represented by textual features and annotated with a class label. Edges resemble citation relationships, which are represented as bidirectional edges. These datasets are often used in transductive (or semi-supervised) learning environments [YCS16; KW17; VCC+18]. However, in our experimental setup with unseen nodes, we use an inductive learning strategy.

We use two different train-test splits for each dataset. Setting A is derived from the train-test split for transductive tasks [KW17]. It consists of few labeled nodes that induce our training set and many unlabeled nodes. Setting B instead comprises many training nodes and few test nodes. We set it up by inverting the train-test mask of Setting A and assign the edges accordingly. Setting B is motivated by

[1] http://konect.cc/

[2] https://pytorch-geometric-temporal.readthedocs.io/

Table 7.1. Statistics for train-test splits: few-many (A) and many-few (B) settings on the citation networks datasets: Cora, Citeseer, and PubMed. The unseen nodes and edges are available only after the training epochs. The test samples for measuring accuracy are a subset of the unseen nodes. The label rate is the percentage of labeled nodes for training.

Dataset	**Cora**		**Citeseer**		**PubMed**	
Classes	7		6		3	
Features	1,433		3,703		500	
Nodes	2,708		3,327		19,717	
Edges	5,278		4,552		44,324	
Avg. Degree	3.90		2.77		4.50	
Setting	**A**	**B**	**A**	**B**	**A**	**B**
Train Nodes	440	2,268	620	2,707	560	19,157
Train Edges	342	3,582	139	2,939	34	41,858
Unseen Nodes	2,268	440	2,707	620	19,157	560
Unseen Edges	4,936	1,696	4,413	1,613	44,290	2,466
Test Samples	1,000	440	1,000	620	1,000	560
Label Rate	16.2%	83.8%	18.6%	81.4%	2.8%	97.2%

applications, in which a large graph is already known and incremental changes occur over time, such as for citation recommendations, link prediction in social networks, and others [AS14; GFZ+21]. We refer to Table 7.1 for the details of the datasets and the two settings. We use these three data sets with two different train-test splits in our first experiment described in Section 7.5.1.

7.4.2 Evolving Graph Datasets

We provide three new graph datasets for lifelong learning based on scientific publications: a new coauthorship graph dataset (PharmaBio) as well as two newly compiled citation graph datasets based on DBLP (DBLP-easy and DBLP-hard). For PharmaBio, the classes are categories of journals. For DBLP, we use the conferences and journals in which the papers have been published, as classes. Since we select those venues with the most publications, this serves as a proxy for a broad categorization. When new conferences and journals emerge, as they do in computer science, new classes will be introduced to the data. The datasets were generated by imposing a minimum threshold of publications per class per year: 100 for

DBLP-easy, 45 for DBLP-hard, and 20 for PharmaBio. For the coauthorship graph PharmaBio we additionally require a minimum of two publications per author per year. In all datasets, the node features are normalized TF-IDF representations of the title of the publication.

Table 7.2. Global dataset characteristics: total number of nodes $|V|$, edges $|E|$, features D, classes $|\mathbb{Y}|$ along with # of newly appearing classes (in braces) within the T evaluation tasks

Dataset	$\lvert V\rvert$	$\lvert E\rvert$	D	$\lvert\mathbb{Y}\rvert$	T
DBLP-easy	45,407	112,131	2,278	12 (4 new)	12
DBLP-hard	198,675	643,734	4,043	73 (23 new)	12
PharmaBio	68,068	2,1M	4,829	7	18

Basic Characteristics Table 7.2 summarizes the basic characteristics of the datasets. DBLP-easy and DBLP-hard are organized into 12 annual snapshots, while PharmaBio has 18 annual snapshots. DBLP-easy has 45k nodes, 112k edges and a feature dimension of 2,278. The nodes are assigned to one of 12 classes, of which four only appear at some time during the sequence of snapshots, i. e., they are not present in the first snapshots. DBLP-hard has 199k classes, 644k edges, and a feature dimension of 4,043. Twenty-three of the 73 classes appear only during subsequent snapshots. PharmaBio comes with 68k nodes, 2.1M edges, feature dimension 4,829, 7 classes, and 18 snapshots. The number of edges is much higher than in the DBLP variants because PharmaBio is a coauthorship graph, which is more dense than the citation graphs. Note that DBLP-easy is a subset of DBLP-hard as both were generated by applying a minimum threshold on the number of publications per class.

We report the label distribution of the datasets, the degree distribution, and the distribution over time in Figure 7.3. The annual number of publications grows with time. Only in PharmaBio, there is a higher amount between 1991-1997 than between 1998 and 2003. The global degree distributions of DBLP-easy and DBLP-hard seem to follow a power-law distribution [New05] as the degree distribution is almost a straight-line except for the blurry tail. For PharmaBio, the degree distribution is more blurry, while a trend line can still be identified. Furthermore, we observe that the number of examples per class is imbalanced in all three datasets. Although the

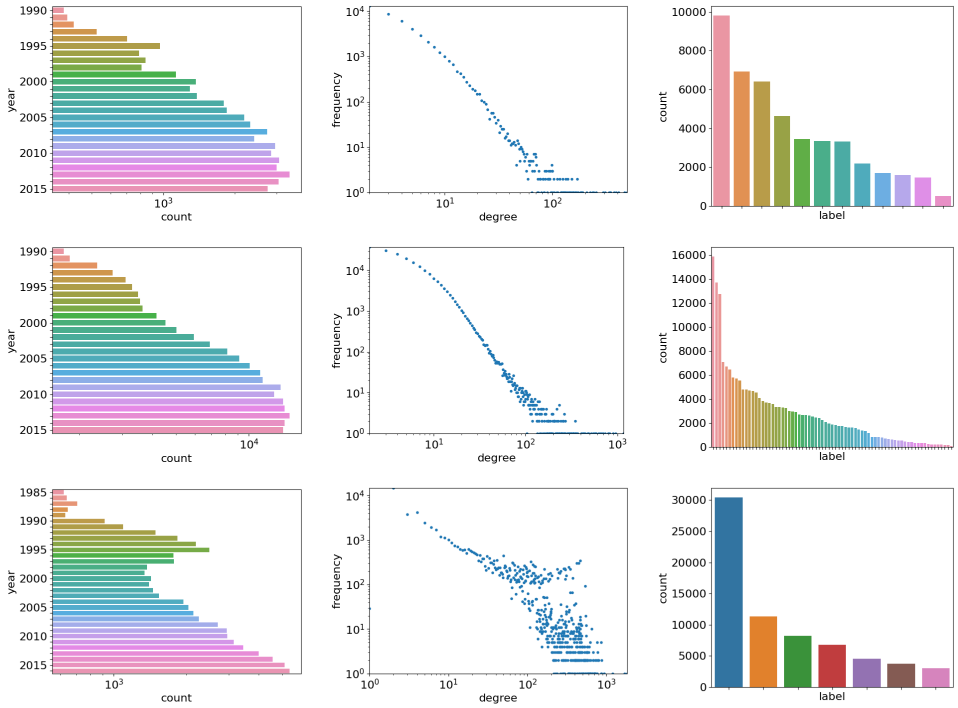


Figure 7.3. Distribution of vertices per year on log scale (left column), degree distributions (middle column), label distributions (right column), for our new datasets: DBLP-easy (top row), DBLP-hard (middle row), PharmaBio (bottom row)

three classes have different numbers of classes, the shape of the label distributions is similar.

Unseen Classes and Distribution Shift With respect to changes in the class set, DBLP-easy has 12 venues in total, including one biannual conference and four new venues appearing in 2005, 2006, 2007, and 2012. DBLP-hard has 73 venues, including one discontinued, nine biannual, six irregular venues, and 23 new venues. To quantify changes in the class set, we calculate the magnitude of the class drift as the total variation distance [WHC+16; WLG+18]:

$$\sigma_{t-1,t} = \frac{1}{2} \sum_{y \in \mathbb{Y}_{t-1} \cup \mathbb{Y}_t} |P_{t-1}(y) - P_t(y)|$$

where $P_t(y)$ is the observed class probability at time t. We visualize the drift

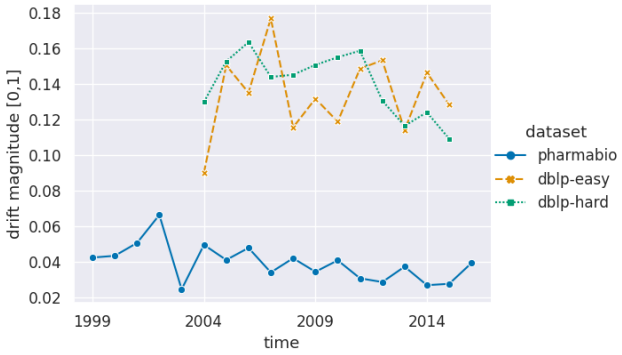


Figure 7.4. Magnitude of the class drift per dataset. The drift within the PharmaBio dataset (no new classes) is lower than the drift of both DBLP variants. Independent and identically distributed data would have drift magnitude zero.

magnitudes per dataset in Figure 7.4. An i.i.d. dataset would have a drift magnitude of zero by definition. As expected, the drift magnitude is high (between 0.12 and 0.16) for the two datasets with new classes: DBLP-easy and DBLP-hard. On PharmaBio, which does not have new classes, the drift magnitude is consistently lower than 0.07. A detailed description of which classes are introduced at which time is provided in Section F.2.

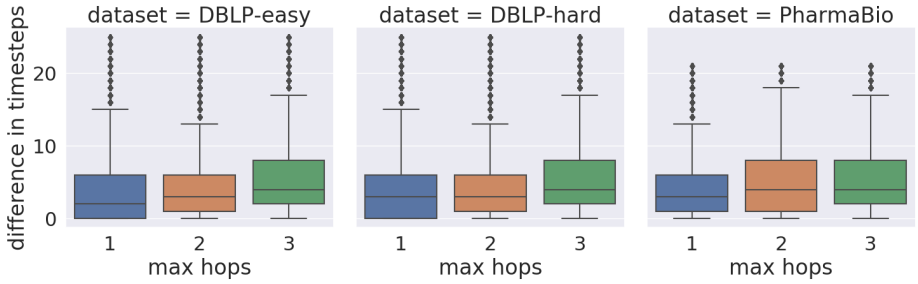


Figure 7.5. Distributions of time differences tdiff_k (y-axis) for DBLP-easy (left), DBLP-hard (center) and PharmaBio (right) within the k-hop neighborhood for $k = \{1, 2, 3\}$ (x-axis).

Analyzing Time Differences Next, we analyze the k-neighborhood time differences tdiff_k, which we have introduced in Section 7.3.3. In Figure 7.5, we show the resulting distribution for three different values of $k = 1, 2, 3$. As expected, the time

differences increase if we allow a longer maximum path length k. For our experiments, we will use GNN models with 2 layers, i. e., which take into account the two-hop neighborhood of each node. Thus, we use the time difference distribution tdiff_2 to derive the candidate history sizes. These are 1, 3, 6, 25 as history sizes for DBLP-{easy,hard} and 1, 4, 8, 21 as history sizes for PharmaBio according to the 25th, 50th, 75th, and 100th percentiles of tdiff_2.

Task Sequences for Lifelong Learning For each dataset, we construct the sequence of tasks $\tilde{\mathcal{T}}_1, \ldots, \tilde{\mathcal{T}}_T$ based on the publication year along with a given history size c. For each task $\tilde{\mathcal{T}}_t$, we construct a graph with publications from time $[t-c, t]$, where publications from time t are the test nodes, and $t < c$ training nodes (transductive). In the inductive case that GraphSAINT uses in our experiments, we train exclusively on $\tilde{\mathcal{T}}_{t-1}$, but still evaluate the test nodes of $\tilde{\mathcal{T}}_t$. We set the first evaluation task $\tilde{\mathcal{T}}_1$ to the time at which 25% of the total number of publications are available. Therefore, when mapping the data sets to our problem statement (see Figure 7.1), our first evaluation task $t = 1$ corresponds to year 1999 in PharmaBio (total range: 1985–2016) and 2004 in DBLP-{easy,hard} (1990-2015). We continue with the next years for subsequent tasks. We will use these datasets in the experiments described in Sections 7.5.3, 7.5.4, and 7.5.5.

7.5 Experiments

In the following, we describe our experiments to analyze transductive vs. inductive learning, lifelong learning, open-world learning, and learning with limited labeled data. We use standard benchmark datasets (described in Section 7.4.1) for the first experiment described in Section 7.5.1, and then use our new datasets (described in Section 7.4.2) for subsequent experiments, which are covered in Sections 7.5.2 to 7.5.5.

7.5.1 Transductive versus Inductive Learning

In the first experiment, our objective is to learn whether accuracy increases when we add unlabeled data to the graph after having trained a model only on the portion of the graph that has labeled vertices. This is important for later experiments because it affects how we move from task t to task $t+1$. We answer whether we need to retrain a model with unlabeled data from the graph at $t+1$, or is it sufficient to wait

until the new labeled data become part of the training set. This research question can be very well investigated with the static graph datasets that we introduced in Section 7.4.1. We use the training set of the static graphs as step one and the unlabeled part of the test set as step two. In order to obtain generalizable results, we consider two different train-test splits for each dataset, which we call setting A (few training, many test examples) and setting B (many training, few test examples), as described in more detail in Section 7.4.1.

In the context of lifelong learning, settings A and B correspond to different stages of the incremental training procedure. At the very beginning, we start with a few labeled data. After a few tasks, the amount of labeled data increases, and, then, any new data added to the training set will make only a smaller fraction of the already known labeled data.

In the following, we describe the procedure, hyperparameter, and metrics of our experiments to analyze transductive vs. inductive learning on standard benchmark datasets with two complementary train-test splits. The aim is to analyze the effect of adding unlabeled data after (pre-)training and comparing inductively pretrained models to models that have been trained transductively *including* unlabeled test data. We will show that the addition of unlabeled data does not further improve the performance of the inductively pretrained models.

Experimental Setup We construct a dedicated experimental setup to assess the influence of pretraining GNNs. We include edges in the training set if and only if its source and destination node are both in the training set. The training process is then divided into two steps. First, we pre-train the model on the labeled training set. Then we insert previously unseen nodes and edges into the graph and continue training for a limited number of inference epochs. The unseen nodes do not introduce any new labels. Instead, the unseen nodes provide features and may be connected to known labeled nodes. We evaluate the accuracy on the test nodes, which are a subset of the unseen nodes, before the first and after each inference epoch. For each model, we compare using 200 pretraining epochs versus no pretraining. In the latter case, training begins during inference, which is equivalent to retraining from scratch whenever new nodes and edges are inserted. This allows us to assess whether pretraining is helpful for applying graph neural networks on dynamic graphs.

Hyperparameters All employed graph neural networks use two graph convolution layers that aggregate neighbor representations. The output dimension of the second layer corresponds to the number of classes. Thus, the features within the two-hop neighborhood of each labeled node are taken into account for its prediction. We adopted the same hyperparameter values as proposed in the original work. For GCN, we use 16 or 64 hidden units per layer (denoted by GCN and GCN-64, respectively), ReLU activation, 0.5 dropout rate, along with a learning rate of 0.005 and weight decay $5 \cdot 10^{-4}$ [KW17]. For GAT, we use 8 hidden units per layer and 8 attention heads in the first layer. The second layer has 1 attention head (8 on PubMed). We set the learning rate to 0.005 (0.01 on PubMed) with weight decay 0.0005 (0.001 on PubMed) [VCC+18]. For GraphSAGE, we use 64 hidden units per layer with mean aggregation, ReLU activation, and a dropout rate of 0.5. We set the learning rate to 0.01 with weight decay $5 \cdot 10^{-4}$ [HYL17]. Our baseline MLP has one hidden layer with 64 hidden units, ReLU activation, a dropout rate of 0.5, a learning rate of 0.005 and weight decay $5 \cdot 10^{-4}$. In all cases, we use Glorot initialization[GB10] and Adam [KB15] to optimize cross-entropy. We initialize the optimizer at the beginning of the inference epochs.

Measures Our main evaluation measure is accuracy. We train each model for 35 epochs and repeat the training 100 times with different seeds. We then report the mean accuracy plus the standard deviation of the models at each of these training epochs. Furthermore, we compute the Jenson-Shannon divergence [Lin91] on the accuracy distributions to quantify the similarity of the distributions in the two different pretraining configurations (with or without) and in the two different settings (A and B). Since the two distributions are of the same kind, we use a symmetric measure to compare them. The Jenson-Shannon divergence (D_{JS}) is a symmetric measure. It compares two distributions P and Q by calculating the (asymmetric) Kullback-Leibler divergence (D_{KL}) in both directions:

$$D_{\mathrm{JS}}(P||Q) = \frac{1}{2} D_{\mathrm{KL}}(P||Q) + \frac{1}{2} D_{\mathrm{KL}}(Q||P)$$

As D_{JS} is a divergence measure, lower values indicate more similar distributions.

Results Figure 7.6 shows the results of the three models on the three datasets: Cora, Citeseer, and PubMed. Pretrained models score consistently higher and have less variance than non-pretrained models. The accuracy of the pretrained models plateaus after a few inference epochs (up to 10 on Cora-A and PubMed-B). Without

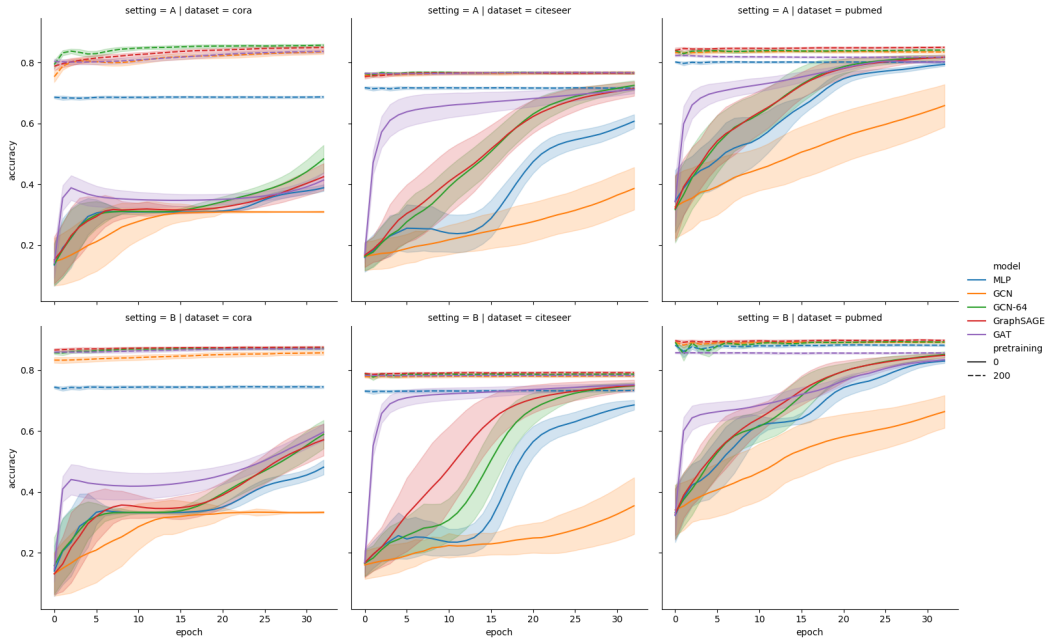


Figure 7.6. Test accuracy after each inference epoch for the many-few settings A *(Top)* and few-many setting B *(Bottom)* on the datasets Cora, Citeseer, and PubMed. Each line resembles the mean of 100 runs and its region shows the standard deviation. The dashed lines show the results with 200 pretraining. The solid lines are the results without pretraining.

any pretraining, GAT shows the fastest learning process. The absolute scores of the pretrained graph neural networks are higher than those of MLP. From a broad perspective, the scores of pretrained graph neural networks are all on the same level: GCN falls behind the others on Cora-B and GAT falls behind the others on PubMed. The scores of many-few setting B are higher than those of few-many setting A by a constant margin.

We compare the results between both settings A and B by measuring the Jensen-Shannon divergence between the accuracy distributions. The Jenson-Shannon divergence between the two settings is lower with pretraining (between 0.0057 for GAT and 0.0115 for MLP) than without pre-training (between 0.0666 for GraphSAGE and 0.1013 for GCN). This indicates that pretraining is helpful in both train/test splits.

In summary, our results show that inductive graph neural networks perform well even though we insert new *unlabeled* nodes and edges after training. For all

three considered in this study, the accuracy plateaus after very few inference epochs. This observation holds for both train-test split settings: many-few and few-many. This motivates the warm-restart strategy, i. e., reusing previous parameters, which we use in the following lifelong learning experiments. In other words, we have not observed any gain from continued training of an inductive model on additional *unlabeled* data.

7.5.2 Incrementally-trained vs Once-trained Models

So far, we have only added new unlabeled data after the initial training; now we will tackle the task-sequential lifelong learning setup, in which *labeled* data are added sequentially throughout the tasks and the models have to adapt over time. We compare once-trained trained models (static) against incrementally trained models (incremental).

Experimental Setup We train static models for 400 epochs on the data prior to the first evaluation time step, which comprises 25% of the total nodes. We train incremental models for 200 epochs with history sizes of 3 time steps (4 on the PharmaBio dataset) before evaluating each task. Each training and evaluation run is repeated ten times with different random seeds.

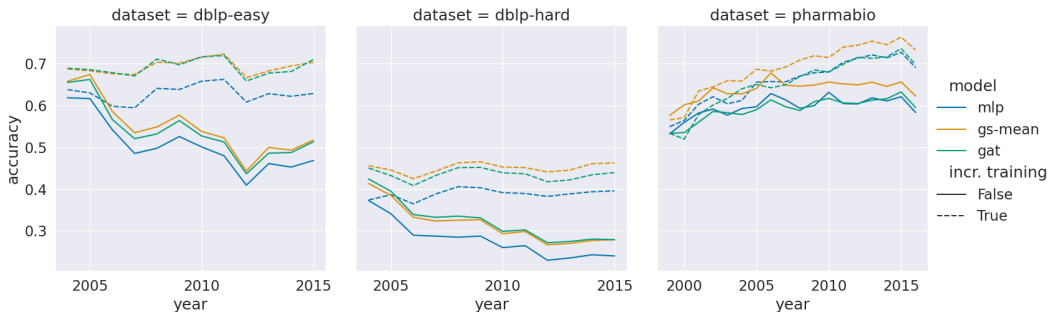


Figure 7.7. Accuracy scores of once-trained, static models (solid lines) are lower than incrementally trained models (dashed lines).

Results In Figure 7.7, we see that the accuracy of the static models decreases over time on DBLP-easy and DBLP-hard, where new classes appear over time. On PharmaBio (fixed class set), the accuracy of static models plateaus, whereas the

accuracy of incrementally trained models increases. Since incremental training is beneficial in all cases, we will use only incrementally trained models in subsequent experiments.

7.5.3 Lifelong Learning on Graphs

From previous experiments, we know that inductively trained models are stable when adding unlabeled data after training and that incremental training is necessary to adapt to new data. Now, we focus on the case in which we continually add more labeled data to the graph, even including new classes in addition to new nodes and edges. The aim is to determine whether parameter reuse is helpful. We consider this question in the context of whether and how many old nodes (and their edges) can be discarded when dealing with evolving graphs.

Now, we report the results of our main experiments to explore the standard lifelong learning setting, as described in Section 7.2. In this setting, the models have to sequentially adapt to new tasks with new labeled data, including unseen classes.

Experimental Setup The evolving graph is divided into tasks according to the time slices in years (see Section 7.4.2). We apply the incremental training algorithm of Section 7.3.1 to each of the considered models, GraphSAGE, GAT, SGC, GraphSAINT, JKNet, along with a graph-agnostic MLP baseline. The rationale for the selection of these particular base GNN models is provided in Section 7.3.4.

For each model, we distinguish between warm restart and cold restart configurations, which determines whether the previous parameters are reused as initialization for the next task (warm restart) or not (cold restart).

Furthermore, we consider the history size as a controlled parameter and vary it according to the percentiles of tdiff_k, as determined in our analyses of the datasets in Section 7.4.2. Corresponding to two layers of graph convolution, which our models use, the quartiles consider 25%, 50%, and 75% of the tdiff_2 distribution and are in terms of history sizes $c = 1$, 3, and 6 for both DBLP datasets, and 1, 4, and 8 for the PharmaBio dataset. We compare these limited-history settings with full-graph training, which corresponds to keeping an unlimited history of the entire timeline of the graph.

All methods are trained in a transductive fashion, except for GraphSAINT, which needed to use the inductive setting. However, we have ensured that the evaluation is fair (see Section 7.4) and we have confirmed in Experiment 1 (see

Section 7.5.1) that the difference between inductive and transductive training is negligible.

We run our incremental training method for graph learning from Section 7.3.1 for each of the models in warm-restart and cold-restart configurations and for the different history sizes. Experiments are repeated 10 times with different random seeds.

Hyperparameters We constrain all models to two graph convolutional layers, comparable hidden dimensions (2x32 GraphSAGE, 4x8 GAT, 2x2x16 JKNet, 64 MLP), and a 0.5 dropout rate. We fix an update step budget of 200 per task and use Adam [KB15] to optimize cross-entropy. We implemented GAT, GraphSAGE, SGC, and JKNet with *dgl* [ZWG+20] and use *torch-geometric* [FL19] for GraphSAINT. We had to disable GraphSAINT's norm recomputation for each task so that our experiments could finish in a reasonable time. We also optimized the weight decay, whose effect was negligible.

For the sake of a fair comparison, we have optimized the hyperparameters separately for each possible history size and restart configuration. In more detail, for each combination of base model, history size, and restart configuration, we tune the learning rate on DBLP-easy. The search space for the learning rate is $\{0.1, 0.05, 0.01, 0.005, 0.001, 0.0005\}$. We consider DBLP-easy as our development dataset to tune the learning rate, which we then use for DBLP-hard and PharmaBio.

Evaluation Measures Our primary evaluation measure for lifelong node classification f is accuracy. With $\text{acc}_t(f^{(t)})$, we denote the accuracy of model $f^{(t)}$ on task $\mathcal{T}_t$. We aggregate accuracy scores over the sequence of tasks $\mathcal{T}_1, \ldots, \mathcal{T}_T$ by using their unweighted average [LR17]:

$$\text{acc}(f) = \frac{1}{T} \sum_{t \in 1,\ldots,T} \text{acc}_t(f^{(t)})$$

Following Lopez-Paz and Ranzato [LR17], we use Forward Transfer (FWT) to quantify the effect of reusing the previous parameters. This is reflected in the accumulated differences in accuracy between the f_{warm} and f_{cold} models, defined below:

$$\text{FWT}(f_{\text{warm}}, f_{\text{cold}}) = \frac{1}{T-1} \sum_{t \in 2,\ldots,T} \text{acc}_t(f^{(t)}_{\text{warm}}) - \text{acc}_t(f^{(t)}_{\text{cold}})$$

To aggregate the results over the 10 repititions with different random seeds, we report the mean accuracy plus/minus 1.96 times the standard error of the mean.

Results Table 7.3 shows the aggregated results of 20,160 evaluation steps (48 configurations with 10 repetitions on two datasets with 12 tasks each and one dataset with 18 tasks). We consider the method A better than the method B when the mean accuracy of A is higher than that of B and the 95% confidence intervals, as estimated by 1.96 times the standard error of the mean, do not overlap [GBC16]. In terms of the absolute best methods per setting (= dataset $\times$ history size), we find that GraphSAGE consistently gives the highest scores except for DBLP-hard, where it is challenged by SGC .

Regarding the comparison of history sizes (i. e., *explicit knowledge* the highest scores are achieved in almost all cases by using an unlimited history size, i. e., using the full graph's history. However, on all datasets, the scores for training with limited window sizes larger than 1 are close to those for full-graph training. With history sizes that cover 50% of the GNN's receptive field, all methods achieve at least 95% relative accuracy compared to the same model under full-history training. When 75% of the receptive field are covered, the models yield at least 99% relative accuracy. To compute these percentages, we have selected the best of both cold and warm restarts for each method.

Regarding the influence of *implicit knowledge*, we find that reusing parameters (warm restarts) leads to notably higher scores than retraining from scratch, when the history size is one (see column FWT with history size $c = 1$): The average Forward Transfer across all models and datasets with history size $c = 1$ is five accuracy points.

Regarding isotropic vs. anisotropic GNNs, we find that GAT and GraphSAGE perform similarly well on DBLP-easy (on which the learning rate was tuned). However, GraphSAGE yields higher scores on DBLP-hard and PharmaBio, which could indicate that the GraphSAGE-mean is more robust to hyperparameters than GAT.

Regarding memory-efficient methods, we observe that the scores of SGC are among the highest of all methods on DBLP-hard. To understand this result, we recall that SGC uses only one single weight matrix of shape $n_{\text{features}} \times n_{\text{outputs}}$, which leads to 300,000 learnable parameters on DBLP-hard, but only 27,000 and 34,000 on DBLP-easy and PharmaBio, respectively. For comparison, GraphSAGE has 146,000 learnable parameters on DBLP-easy, 264,000 on DBLP-hard, and 310,000 on PharmaBio. On the other hand, GraphSAINT yields high scores on PharmaBio, comparable to GraphSAGE, but lower scores on both DBLP datasets.

To summarize this experiment, the results show that in the three analyzed datasets, with only history sizes of 3 or 4 (corresponding to 50% coverage of the

Table 7.3. Accuracy (with 95% confidence intervals through 1.96 standard error of the mean) and Forward Transfer (averaged difference of warm and cold restarts) in our datasets with different history sizes (column **c**). The best method per dataset and history size is marked in bold, along with the methods where the 95% CI overlaps.

		GAT			GraphSAGE-Mean			MLP (Baseline)		
		avg. accuracy		FWT	avg. accuracy		FWT	avg. accuracy		FWT
		cold	warm		cold	warm		cold	warm	
Dataset	**c**									
DBLP-easy	**1**	60.8 ± 0.5	$\mathbf{64.9 \pm 0.4}$	$+4.5$	60.4 ± 0.5	$\mathbf{65.1 \pm 0.4}$	$+5.2$	56.1 ± 0.4	62.2 ± 0.5	$+6.6$
	3	$\mathbf{68.9 \pm 0.3}$	$\mathbf{69.3 \pm 0.3}$	$+0.2$	$\mathbf{68.7 \pm 0.3}$	$\mathbf{69.3 \pm 0.3}$	$+0.7$	61.0 ± 0.5	62.9 ± 0.4	$+2.0$
	6	$\mathbf{70.3 \pm 0.4}$	70.2 ± 0.4	-0.1	$\mathbf{71.1 \pm 0.4}$	$\mathbf{70.9 \pm 0.4}$	-0.3	62.7 ± 0.3	62.7 ± 0.4	-0.2
	full	70.2 ± 0.4	70.2 ± 0.4	$+0.1$	$\mathbf{71.6 \pm 0.4}$	$\mathbf{71.4 \pm 0.3}$	-0.2	63.4 ± 0.3	61.9 ± 0.4	-1.2
DBLP-hard	**1**	39.4 ± 0.2	39.1 ± 0.2	-0.1	34.5 ± 0.4	40.0 ± 0.2	$+5.9$	31.6 ± 0.3	38.3 ± 0.3	$+7.4$
	3	44.0 ± 0.2	43.7 ± 0.2	-0.4	44.3 ± 0.2	$\mathbf{45.1 \pm 0.2}$	$+0.8$	33.7 ± 0.3	38.9 ± 0.2	$+5.6$
	6	45.1 ± 0.3	45.3 ± 0.3	$+0.2$	$\mathbf{46.5 \pm 0.3}$	$\mathbf{46.7 \pm 0.3}$	$+0.2$	39.2 ± 0.2	38.3 ± 0.2	-0.7
	full	45.6 ± 0.3	45.6 ± 0.3	-0.1	46.8 ± 0.2	47.1 ± 0.3	$+0.4$	38.2 ± 0.2	36.7 ± 0.2	-1.1
PharmaBio	**1**	61.6 ± 0.9	65.4 ± 0.9	$+3.8$	65.4 ± 0.9	$\mathbf{68.6 \pm 1.0}$	$+3.3$	62.7 ± 0.9	66.3 ± 0.9	$+3.9$
	4	64.5 ± 0.8	65.3 ± 0.9	$+0.9$	$\mathbf{68.0 \pm 0.8}$	$\mathbf{69.0 \pm 0.8}$	$+1.1$	66.3 ± 0.7	65.7 ± 0.8	-0.7
	8	65.1 ± 0.8	65.4 ± 0.8	$+0.3$	$\mathbf{68.8 \pm 0.7}$	$\mathbf{69.0 \pm 0.8}$	$+0.2$	64.2 ± 0.8	65.3 ± 0.7	$+0.9$
	full	64.3 ± 0.8	65.4 ± 0.8	$+0.2$	$\mathbf{69.0 \pm 0.7}$	$\mathbf{68.4 \pm 0.7}$	-0.7	65.4 ± 0.8	64.4 ± 0.6	-1.1

		SGC			GraphSAINT			Jumping Knowledge		
		avg. accuracy		FWT	avg. accuracy		FWT	avg. accuracy		FWT
		cold	warm		cold	warm		cold	warm	
DBLP-easy	**1**	57.1 ± 0.4	63.7 ± 0.4	$+7.2$	62.1 ± 0.3	63.2 ± 0.4	$+1.2$	56.2 ± 0.5	61.4 ± 0.5	$+5.6$
	3	66.4 ± 0.3	67.4 ± 0.3	$+1.2$	66.4 ± 0.4	65.3 ± 0.5	-0.9	65.2 ± 0.3	65.9 ± 0.5	$+1.0$
	6	69.3 ± 0.4	69.3 ± 0.4	$+0.1$	68.1 ± 0.4	65.5 ± 0.7	-2.1	68.0 ± 0.4	66.9 ± 0.6	-0.7
	full	$\mathbf{71.0 \pm 0.4}$	70.0 ± 0.4	-1.0	68.4 ± 0.5	65.7 ± 0.5	-2.8	68.7 ± 0.4	66.3 ± 0.4	-2.5
DBLP-hard	**1**	34.5 ± 0.3	$\mathbf{41.0 \pm 0.3}$	$+7.0$	35.9 ± 0.3	35.6 ± 0.4	$+0.5$	33.0 ± 0.2	35.3 ± 0.3	$+2.9$
	3	44.1 ± 0.2	$\mathbf{44.8 \pm 0.3}$	$+0.8$	39.3 ± 0.3	38.1 ± 0.5	-0.6	39.1 ± 0.3	38.8 ± 0.4	$+0.3$
	6	$\mathbf{46.9 \pm 0.3}$	46.2 ± 0.3	-0.4	40.6 ± 0.3	38.8 ± 0.6	-1.2	41.0 ± 0.3	40.1 ± 0.5	-0.3
	full	$\mathbf{48.8 \pm 0.4}$	47.5 ± 0.3	-1.2	41.0 ± 0.4	40.7 ± 0.4	-0.3	41.6 ± 0.3	40.8 ± 0.2	-0.9
PharmaBio	**1**	62.3 ± 0.9	64.5 ± 0.8	$+2.3$	65.7 ± 0.8	$\mathbf{68.6 \pm 0.8}$	$+3.0$	64.1 ± 0.9	$\mathbf{68.3 \pm 0.9}$	$+4.3$
	4	64.4 ± 0.8	64.4 ± 0.8	-0.0	67.3 ± 0.8	$\mathbf{68.4 \pm 0.7}$	$+1.0$	67.1 ± 0.8	$\mathbf{68.2 \pm 0.8}$	$+1.1$
	8	65.3 ± 0.8	64.0 ± 0.7	-1.4	$\mathbf{68.1 \pm 0.8}$	$\mathbf{68.0 \pm 0.7}$	-0.1	$\mathbf{67.8 \pm 0.8}$	$\mathbf{67.7 \pm 0.7}$	-0.3
	full	62.4 ± 0.8	61.7 ± 0.6	-0.8	$\mathbf{68.2 \pm 0.8}$	66.1 ± 0.8	-2.2	66.8 ± 0.8	64.5 ± 0.7	-2.6

receptive field of a 2-layer GCN), almost all methods obtain 95% accuracy compared to the same model under full-history training. Moreover, with very small history sizes, such as using only one past task, using warm restarts is important to maintain a high level of accuracy.

7.5.4 Lifelong Learning with Limited Labeled Data

Until now, we have assumed that the true labels of nodes become part of the training data for subsequent tasks. Now we relax this assumption and release only a portion of the labeled data in task t for training in the subsequent task $t+1$. This resembles real-world applications, such as the indexing of scientific articles in libraries [MGS18]. The motivation is that labeled data is expensive to "produce". Again, we work with the most challenging dataset, DBLP-hard, for this experiment, because it has the highest number of new classes.

Experimental Setup To implement the idea of learning with only a fraction of the labeled data, we randomly sample a subset of nodes, for which the true class label is available for training. We denote this fraction by *label rate*. For experiments, it is important to sample globally rather than per task because otherwise we would get inconsistencies: node i could have a label in task t but not in task $t+1$. Therefore, we sample the entire data set before splitting it into tasks. In this way, the subset of nodes that comes with classes is fixed for the entire duration of the experiments. Furthermore, we use the same subset with all different configurations and with all repetitions of the experiment. We sample uniformly at random at the node level without stratification between classes.

For this experiment, we use GraphSAGE-Mean as GNN model because it has achieved the best results in the previous experiment, where the label rate was not restricted. As before, we evaluate different history sizes (1, 3, 6) and both restart configurations (warm and cold). As dataset, we chose to use DBLP-hard because it has the highest number of classes and is the most challenging.

Hyperparameters Again, as in the previous experiment, the optimal hyperparameters were determined on DBLP-easy, the subset of DBLP-hard that we use consistently to tune the hyperparameters. We have not tuned the hyperparameters separately for each label rate, but we reuse the optimal hyperparameters from training with a 100% label rate. The hyperparameter search space is the same as in the previous experiment.

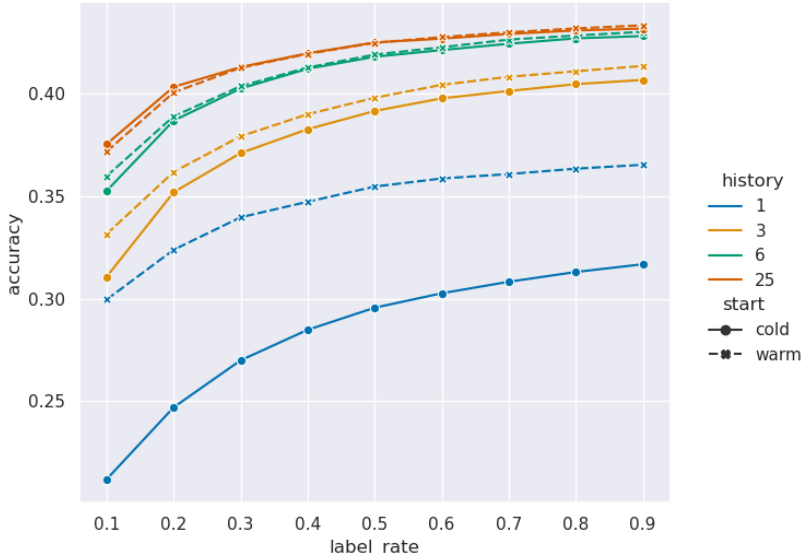


Figure 7.8. Average accuracy of GraphSAGE with warm restarts across tasks on DBLP-hard under varying label rate

Measures As in the previous experiment, we use the average accuracy over tasks as evaluation metric.

Results In Figure 7.8, we plot the average accuracy across tasks as a function of label rate. We confirm that the results of our previous experiments also hold when labeled data are limited. Warm restarts yield higher scores than cold restarts. This is more pronounced when the history size is small. As expected, the absolute accuracy values decrease as the label rate decreases.

When comparing history sizes, we again observe that a larger history size leads to better results. In particular, using the entire history gives the best results, closely followed by a history size of 6. Still, when the label rate is decreased, the difference between the history sizes remains constant.

With very low label rates (in the range between 10% and 30%), the accuracy of the cold restart strategy drops faster than the accuracy with warm restarts. In other words, the use of warm restarts leads to more stable models when dealing with lower label rates.

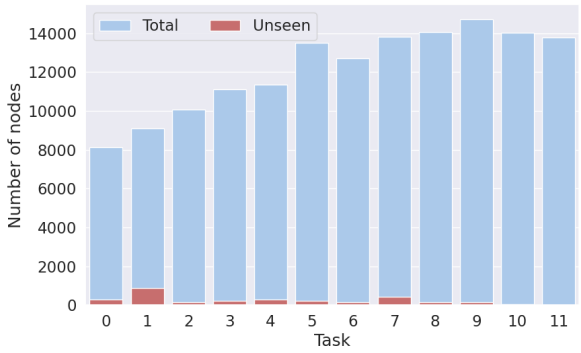


Figure 7.9. Number of nodes with unseen classes per task on DBLP-hard

7.5.5 Self-Detection of Unseen Classes

In our evolving graphs, we have to deal with previously unseen classes. In previous experiments on lifelong learning, these unseen classes (and the vertices that have these classes) were already part of the test data. However, the models did not have the opportunity to actually predict those classes, as no dedicated technique has been used to detect nodes from unseen classes. Here, we evaluate our adaptation of the unseen class detection method DOC to graph data, called gDOC, as introduced in Section 7.3.2. The experiments comprise a crisp unsupervised detection of instances of unseen classes. At the same time, the models need to make predictions as usual for the known classes.

Experimental Setup In previous experiments, unseen classes were part of the test data, while there was no active treatment of having them detected automatically. In this experiment, we seek to evaluate the performance of the gDOC method in detecting unseen classes. As before, we train on task $t-1$ and evaluate on t over a sequence of T tasks. However, for each node, we use our unseen class detection module gDOC to predict whether this node belongs to a previously known class or not. If the prediction is that the node does not belong to any previously known class, we reject its classification and assign a special virtual class ("unseen"). As unseen class detection modules, we compare the original DOC as a baseline with our proposed gDOC method.

We use the DBLP-hard dataset, which has 23 new classes. In addition to the analysis of the datasets in Section 7.4.2, we show in Figure 7.9 how many nodes

belong to unseen classes in the DBLP-hard dataset. We observe that in every task except for the last, there are nodes with unseen classes.

As the base GNN model, we use the GraphSAGE-mean model that has performed best in previous experiments along with gDOC for unseen class detection that we have introduced in Section 7.3.2. Our baseline is the original DOC method, also applied to the outputs of the GraphSAGE-mean.

Hyperparameters As in the previous experiments, we optimize the model hyperparameters in our development data set DBLP-easy. We repeat the hyperparameter optimization because the loss function has changed from categorical to binary cross-entropy. As before, the best learning rate is selected based on the best accuracy on DBLP-easy and transferred to DBLP-hard.

Measures We evaluate how well the models detect unseen classes. For this purpose, we use two measures: Macro-F1 with a special class for instances of unseen classes [SXL17], which we call Open Macro-F1 and the Matthews correlation coefficient (MCC). Note that Macro-F1 averages the F1 scores over classes such that the effect of the 'unseen' class is taken into account as any of the known classes. In detail, *Open Macro-F1* is calculated as

$$\text{Open Macro-F1} := \frac{1}{T}\sum_{t=1}^{T}\text{Macro-F1}(\boldsymbol{y}'(t), \boldsymbol{y}'_{\text{pred}}(t))$$

with

$$\boldsymbol{y}'_{\text{pred},i} := \begin{cases} \text{'unseen', if example } i \text{ is detected as OOD} \\ \boldsymbol{y}_{\text{pred},i}, \text{ otherwise} \end{cases}$$

$$\boldsymbol{y}'_i := \begin{cases} \boldsymbol{y}_i \text{ if class } \boldsymbol{y}_i \text{ is known} \\ \text{'unseen', otherwise} \end{cases}$$

where $\boldsymbol{y}_i$ are the true labels and $\boldsymbol{y}_{\text{pred}}$ are the predicted class labels. The arg max of the output is replaced by a special symbol when the method has emitted an 'unseen' decision for that instance. The true labels $\boldsymbol{y}$ are preprocessed similarly, so that instances of previously unseen classes receive a special class symbol.

In the pre-experiments, we found that the best Open Macro-F1 scores are achieved when the thresholds are high, i. e., no nodes are detected as coming from

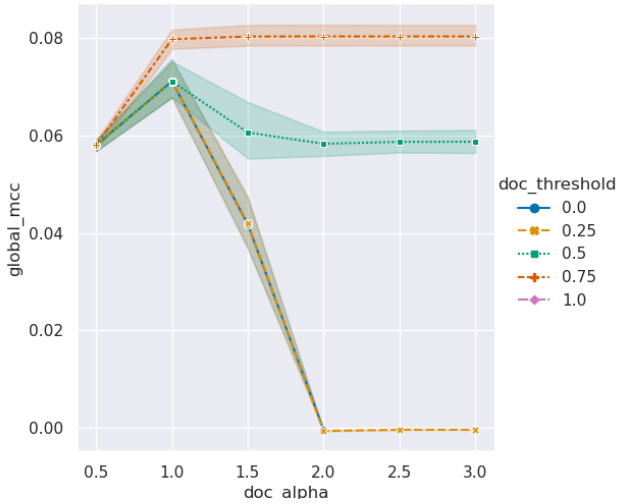


Figure 7.10. Global MCC of GDOC (history size 3, warm) as a function of the SD Factor for risk reduction with different minimum threshold values.

an unseen class. This is because we have a large number of classes, and the special class contributes only very little to the overall Macro-F1 score. Thus, a large number of false rejects diminishes the overall performance in terms of Macro-F1. False rejects are vertices that should *not* be rejected (as their true class is known), but are rejected.

The Macro-F1 score is limited in its expressiveness with respect to the detection of unseen classes because, when averaging over classes, the 'unseen' pseudo-class plays only a minor role compared to all other classes. Thus, we decided to report a further score, the Matthew's correlation coefficient (MCC) of the 'unseen' class vs. all other classes (i. e., the set of known classes). MCC is a popular measure for evaluating binary classification that accounts for class imbalance [CJ20]. Dealing with this class imbalance is important because the number of vertices from the known classes is much larger than the number of vertices from the unseen class. It ranges from -1 to 1, where zero corresponds to random prediction. In more detail, we compute the MCC score as:

$$\mathrm{MCC} = \frac{\mathrm{TP} \cdot \mathrm{TN} - \mathrm{FP} \cdot \mathrm{FN}}{\sqrt{(\mathrm{TP}+\mathrm{FP})(\mathrm{TP}+\mathrm{FN})(\mathrm{TN}+\mathrm{FP})(\mathrm{TN}+\mathrm{FN})}}$$

where TP are true positives or correctly rejected instances, TN true negatives, FP false positives, and FN false negatives. We accumulate these numbers over the entire sequence of tasks.

Table 7.4. Results for unseen class detection on DBLP-hard with GraphSAGE as base model (average of 5 repetitions). α indicates that risk reduction is used with the respective factor for the standard deviation, τ is the minimum threshold. Runs named GDOC are trained with weighted cross entropy. DOC is our baseline.

		MCC		**Open Macro-F1**	
		cold	warm	cold	warm
c	**Open Learning Method**				
1	DOC ($\tau = 0.50$)	.01	.04	.01	.01
	DOC ($\tau = 0.50, \alpha = 3.0$)	.01	.02	.01	.01
	GDOC ($\tau = 0.50$)	.04	.05	**.13**	**.13**
	GDOC ($\tau = 0.50, \alpha = 3.0$)	.04	.05	**.13**	**.13**
	GDOC ($\tau = 0.75$)	.04	**.09**	**.13**	**.13**
3	DOC ($\tau = 0.50$)	.02	.03	.02	.05
	DOC ($\tau = 0.50, \alpha = 3.0$)	.02	.03	.02	.05
	GDOC ($\tau = 0.50$)	.05	.06	**.15**	**.15**
	GDOC ($\tau = 0.50, \alpha = 3.0$)	.05	.06	**.15**	**.15**
	GDOC ($\tau = 0.75$)	.05	**.08**	**.15**	**.15**
6	DOC ($\tau = 0.50$)	.02	.03	.05	.08
	DOC ($\tau = 0.50, \alpha = 3.0$)	.02	.03	.05	.08
	GDOC ($\tau = 0.50$)	.05	.06	**.16**	**.16**
	GDOC ($\tau = 0.50, \alpha = 3.0$)	.05	.06	**.16**	**.16**
	GDOC ($\tau = 0.75$)	.05	**.07**	**.16**	**.16**
∞	DOC ($\tau = 0.50$)	.02	.04	.08	.12
	DOC ($\tau = 0.50, \alpha = 3.0$)	.02	.04	.08	.12
	GDOC ($\tau = 0.50$)	.04	.05	**.16**	**.16**
	GDOC ($\tau = 0.50, \alpha = 3.0$)	.05	.05	**.16**	**.16**
	GDOC ($\tau = 0.75$)	.05	**.07**	**.16**	**.16**

Results Table 7.4 shows the results for the detection of unseen classes in DBLP-hard. We observe that the MCC scores, which measure the correct detection of

new classes, are consistently higher for gDOC than for plain DOC. The same holds for the Open Macro-F1 scores, which measure the overall performance of OOD detection + classification: gDOC is consistently better than plain DOC.

The F1 score of plain DOC with limited history size is very low: between 0.01 for history size 1 and 0.12 for unlimited history size. In the same setting, gDOC achieves much higher scores: already 0.13 with history size 1 and 0.16 with at least history size 6. This shows that the class-weighted binary cross-entropy in gDOC is necessary to achieve reasonable F1 scores.

For thresholds, the results indicate that a high threshold (0.75) is preferable to lower thresholds. We further note that the combination of warm restarts and a small history size leads to the highest MCC score (0.09) on DBLP-hard, while on DBLP-easy, on which the hyperparameters have been tuned, the MCC score is higher for larger history sizes.

In Figure 7.10, we show that risk reduction, i. e., lowering the detection threshold based on the class-specific standard deviation, does not help to increase performance. With a low minimum threshold (e. g., 0), we see the pure performance of the risk reduction technique, which peaks at $\alpha = 1$ before it decreases. When using a high minimum threshold (0.5, 0.75, 1.0), applying risk reduction only decreases the OOD performance. In other words, the absolute best OOD detection performance is achieved when the minimum threshold τ is set to 0.75, regardless of the risk reduction factor α. Therefore, the usefulness of risk reduction for our heavily imbalanced datasets is questionable.

To understand this result, we recall that risk reduction is a technique to calculate class-specific thresholds τ_i (see Section 7.3.2). However, this is only possible up to the global minimum threshold τ. Thus, even with risk reduction, class-specific thresholds cannot go below τ.

The results for the DBLP-easy dataset, on which we have tuned the hyperparameters, as well as the full results for the DBLP-hard dataset can be found in Appendix F.3.

In summary, this experiment has shown that weighting the binary cross-entropy loss function in gDOC is essential for unseen class detection in imbalanced graph data. We also learned that the risk reduction technique (as proposed in DOC [SXL17]) is not helpful on our imbalanced graph datasets. That is because the variance among predictions in the unbalanced case is so high that the (minimum) threshold effectively never changed. The only exceptions are very small factors of standard deviation (< 1). But this only decreases the unseen class detection performance as measured by MCC. In the end, we recommend using gDOC with

weighted binary cross-entropy training to account for class imbalance, but we could not find any benefits of the risk reduction technique proposed in the original DOC.

7.6 Discussion

Our experiments show several key results. First, we have shown in Section 7.5.1 that it is *not* necessary to up-train GNNs when new unlabeled data arrive. Rather, the performance of inductively pretrained GNNs remains stable, even when new unlabeled data are added to the graph.

From the incremental training experiments with limited history sizes in Section 7.5.3, we obtain results that are almost as good as when using the full history of the graph: With window sizes of 3 or 4 (50% receptive field coverage), GNNs achieve at least 95% accuracy compared to using all previous data for incremental training. With window sizes of 6 or 8 (75% receptive field coverage), at least 99% accuracy can be retained. This result holds for standard GNN architectures and also for scalable and sampling-based approaches. This has direct consequences for lifelong learning of GNNs in evolving graphs, as it impacts how GNNs can be employed in real-world applications. More specifically, we have investigated whether to reuse parameters from previous tasks (warm restarts). We find that reusing an "old" model is a viable strategy, even though new classes appear during the sequence of tasks and the history size is limited. We have shown that reusing parameters from previous tasks becomes more important when the history sizes are small. This is because there is less explicit knowledge available.

We have shown in Section 7.5.4 that the methods work well, even when the labeled data are limited. This is important for real-world applications because data annotation is expensive.

With the introduction of gDOC, we have made a first step to introduce new class detection in lifelong graph learning in Section 7.5.5, by combining graph neural networks with the DOC [SXL17] module and extending it to take into account class imbalance. Our experiments on new class detection show that it is necessary to adjust the weights of binary cross entropy training in gDOC to account for the imbalanced label distribution. Contrary to the original DOC, we have not observed any improvements when risk reduction (through the standard deviation of logits) was used. Instead, the best results were achieved with an appropriate threshold ($\tau = 0.75$) regardless of the risk reduction factor α. We acknowledge that emitting a

crisp decision in unsupervised unseen class detection is an extremely challenging problem.

Another interesting result is that the combination of warm restarts with small history sizes has led to increased MCC scores on the most challenging DBLP-hard dataset. It seems that omitting old data helps to better detect out-of-distribution examples.

Regarding generalizability of our results,we have shown that our incremental training approach can be applied to various GNN models and is orthogonal to sampling and preprocessing approaches. In general, our incremental training procedure can be applied to any GNN architecture with few caveats. If the GNN architecture depends on transductive learning, this constraint should be considered during incremental training. Similarly, any pre-computation steps, such as computing normalizing constants such as in GCN [KW17] or GraphSAINT [ZZS+20], must be performed again when adapting the model to a new task.

To reflect our work in the broader context of lifelong or continual learning, we reconsider the gradient episodic memory framework [LR17] for image data, in which the examples are independent. To cast graph data into independent examples for node classification, certain preprocessing steps are required, such as transforming each node into a graph [WQG+22]. This increases the number of inference steps by $\mathcal{O}(|V|)$ compared to our approach.

7.7 Summary

Coming back to *Q5: How can we adapt representation learning models for text and graph data that evolve over time?*, we have introduced a generic experimental framework that enables lifelong learning for evolving graph data. We have introduced an incremental training algorithm and a new measure for k-neighborhood time differences tdiff_k that captures temporal granularity and connectivity patterns in graphs. Using percentiles of these time differences, we can select history sizes that are comparable across different datasets.

In this framework, we have experimented with six representative base models, using different history sizes, and using either warm or cold restarts. Our results on three newly compiled datasets have shown that high levels of accuracy can be preserved even when training only on a fraction of past data. In particular, using warm restarts becomes more important when few past data are available. We then

confirmed our results in a setup with limited label data and observed a trend that warm restarts become even more important when less labeled data is available.

Subsequently, we explored an open-world learning environment in which the models have to detect instances from previously unseen classes while classifying others as before. We adapted a method for crisp detection of unseen classes to the graph case, while stressing the importance of reflecting the class distribution within the loss function of the binary-cross-entropy training objective. Our results further indicate that removing old data is beneficial for unseen class detection.

The settings explored with evolving graph data, limited labels, and new classes reflect the difficulties that practitioners face when using graph neural networks in real-world applications. This chapter sets the ground for the development of additional techniques for lifelong learning in evolving graphs to narrow the gap between research and application.

Chapter 8

Conclusion

The overarching aim of this thesis was to investigate representation learning methods that process texts and graphs together. In the introduction, we agitated three challenges that we needed to overcome to achieve that goal: efficiency, multimodality, and adaptability. We address these challenges by investigating efficient models for text representation learning (Chapters 3, 4, and 5), multimodal models to combine text and graph representation learning (Chapters 6 and 7), and ways to adapt models to new data to deal with evolving graphs with text-attributed nodes (Chapter 7). Below, we summarize the contributions along the three challenges, before we discuss word order, synthetic graphs, and the practical impact of this thesis. We close with some ideas for future work.

8.1 Summary of Contributions

Efficiency In Chapter 3, we explored efficient sentence representation models. We showed that a hybrid model between vector representations, aggregated by summation, and matrix representations, aggregated by matrix multiplication, performs better than each alone on a wide range of tasks. Subsequently, in Chapter 4, we have shown that the knowledge of large pretrained language models, such as BERT, can be successfully distilled into matrix representation models. In this way, matrix embedding models become competitive with RNN-based language models such as ELMo [PNI+18] while being more efficient. However, we have found in Chapter 5 that a bag-of-words representation in conjunction with an **MLP!** already forms a powerful model for classifying texts into topics. In particular, we have shown that a wide MLP is competitive with many recent approaches that rely on building synthetic graphs from pure text. In Chapter 7, we considered efficiency in the sense of learning with limited history sizes in evolving text and graph data. In growing graphs, e. g., social graphs or citation graphs, limiting the history helps to maintain scalability and accuracy We have shown that maintaining a single model over time allows us to use smaller history sizes.

Multimodality In Chapter 6, we analyzed multimodal autoencoders for document-based recommendation tasks. We confirm the findings from Chapter 5 that a BoW-MLP is a strong predictor of topical recommendations when only the text is available. However, we found that the graph structure is more important when dealing with citation graphs. We explain this phenomenon by the semantics of item co-occurrence: in citation graphs, the cited works are similar to each other [Sma73], while in the subject labeling tasks, co-occurring items are dissimilar to cover the different aspects of a single research paper. We hypothesize that predicting similar items is more straightforward than predicting dissimilar items only based on other items.

Nonetheless, the effectiveness of all autoencoder models improves when conditioning on textual side information. Thus, AEs serve as a solid general-purpose model when faced with a new recommendation task and when multiple modalities play a role. In Chapter 7, we then investigated graph representation learning as a more general input representation. Specifically, we investigated a node classification task in evolving graphs of interlinked documents with textual features. The structured information from the citation graph is helpful for the classification of publications.

Adaptability The adaptability of learning models that learn combined text and graph representations was the central topic of Chapter 7. We have introduced three datasets and an experimental framework for lifelong learning in evolving graphs with nodes with textual attributes. We also considered the task of crisp detection of previously unseen classes on the most challenging dataset. We introduced an incremental training algorithm, a new method to derive comparable history sizes in evolving graph data. We showed that reusing parameters is more critical when fewer past data points are available. We found that reflecting class imbalance within the binary cross-entropy criterion is crucial for detecting new classes. We found that removing old data with parameter reuse is beneficial when detecting instances from unseen classes.

8.2 General Discussion

Word Order A central factor for efficiency is whether the models consider the order of words. Models that discard word order are more efficient than sequence models, such as Transformers and the models presented in Chapters 3 and 4.

We have shown in Chapter 3 that modeling word order is beneficial for some tasks, such as question-type classification, while the results of Chapter 5 show that word order is less important for classifying texts into topics. We reflect on Harris [Har54]'s distributional hypothesis, which states that the context of its use determines the meaning of a word. In other words, the exact order of the words is less important than their co-occurrence. With static sentence representations, Conneau et al. [CKL+18] have found that the ability to memorize which words are in a sequence is more indicative of downstream performance than other linguistic properties.

The powerful Transformer-based language models consider word order by actively encoding each word's position. However, Sinha et al. [SJH+21] also have pretrained language models on shuffled text with surprisingly good performance on downstream tasks. We complement those results in Chapter 5 by removing the positional encoding after pretraining. While there was a drop in accuracy, presumably because the model relied on those encodings during pretraining, the model performance decreased slightly. In the end, whether models should consider word order remains to be decided on a per-task basis. However, eliminating word order improves sample complexity, i. e., the number of necessary training examples, because all permutations of a sequence conflate to an indentical input representation. Based on our results, we hypothesize that when learning to infer the topic of a text, the gains in sample complexity outweigh the loss of word-order information.

Synthetic Graphs At first glance, the results of Chapter 5 and Chapter 7 might seem to contradict each other. In Chapter 5, we see that a wide MLP is competitive with recent graph-based approaches, while in Chapter 7, GNNs outperformed the MLP. However, it is essential to note that the graphs are fundamentally different. In Chapter 5, the datasets consisted of pure text. There, graphs were synthesized from pure text, e. g., word-word and word-document relationships. In contrast, in Chapter 7, we worked with graph data, where the edges corresponded to citations. Here, the graph provides more information than what would have been available from the text. Kipf and Welling [KW17] had already expected that graph convolution is "especially powerful in scenarios where the adjacency matrix contains information [otherwise] not present in the data". Now, synthetic graphs derived only from the pure text cannot produce new information (cf. the data processing inequality[1][CT+91]). Hence, we question the usefulness of synthetic graphs for

[1]If $p_{\text{data}} \to \boldsymbol{X} \to \mathcal{G}$, then $I(p_{\text{data}}; \mathcal{G}) \leqslant I(p_{\text{data}}; \boldsymbol{X})$

classification. Empirically, we observe that graph-based approaches only exceeded the accuracy of a wide MLP with very expensive preprocessing Still, even the best graph-based approaches fell behind pretrained Transformer models that only use the text.

Significance for Practice We designed the new experimental frameworks presented in this thesis in a way close to real-world applications: we focus on transfer learning, inductive learning, chronological train/test splits, and dealing with evolving graph data. The findings directly affect practical applications with text and graph data that appear together, such as text classification or recommender systems. In particular, lifelong learning on graphs, as presented in Chapter 7, is highly relevant for practical applications related to social graphs or publication graphs. This thesis provides the tools and knowledge necessary for practitioners to determine which models and modalities best suit the task and the trade-off between efficiency and effectiveness.

In a different regard, this work also relates to privacy considerations: First, smaller and more efficient models are better suited to perform on-device computation without communicating with an extensive computing infrastructure. Second, with document-based recommender systems, we have explored techniques that work without storing user profiles. Third, we found that deleting historical data does not harm accuracy in lifelong node classification.

Because we designed the experimental frameworks in a way close to practice, we observed that many seemingly related approaches were, in fact, not applicable because they could not deal with newly appearing data points after training, which is crucial for practical applications. For instance, recommender system approaches that rely on user embeddings cannot be applied to new users at test time without retraining. Similarly, in evolving graphs, most approaches from the literature are concerned with the dynamic signal case. They assume that the node set is static through time, unlike the dynamic structure case explored in this thesis. All of these approaches certainly have merits and possible applications, yet their applicability to the problems investigated in this thesis is limited. Therefore, we used strong baselines in our new experimental frameworks and explored many representative base models, such as different types of AEs and GNNs, for which we tested the training procedure and extensions.

Significance within Field of Study We shed light on efficient alternatives to heavy-weight Transformer-based language models, including matrix-based sentence representations, cross-architecture distillation, and text classification. While Transformers are still more powerful, the methods presented in this thesis fill a gap between highly efficient word embeddings and more expensive sequence models. In particular, we have shown that a wide BoW-MLP is a powerful model for text classification. Although the method (under different configurations) has existed for decades, it was rarely used as a baseline in research on text classification.

As for recommender systems, a unique contribution is the analysis of the influence of the total number of considered items through dataset pruning and the influence of the fraction of already present items in each example. On the one hand, the ranking of methods remained stable concerning pruning, which justifies the development of new methods on pruned datasets. On the other hand, some methods are better than others when only a little knowledge is available. In particular, the variational autoencoder and the multilayer perceptron are better than other methods when only a few items are available. This result shows that the stage of the recommendation process is an essential factor for evaluating recommender systems.

Moreover, this is one of the first works (along with a few others [FXM+22]) that studies lifelong learning on graph data. We designed an experimental framework and introduced three graph datasets to investigate the maintenance of a single model throughout the evolution of the graph. In contrast to the other works, our datasets resemble the natural evolution of the data, and we also consider the problem detecting unseen classes. This first step opens up various opportunities to further develop and evaluate new methods for lifelong learning in evolving graphs.

8.3 Future Work

In this thesis, we have addressed the challenges of efficiency, multimodality, and adaptability for learning text and graph representations. A logical continuation of this work would employ powerful large pretrained language models and incorporate graph data alongside the text. For now, the size of large language models hindered applying large language models to graphs with textual attributes. This approach will only become feasible with further developments in efficient and long-range Transformers. Note that there are already Transformer variants generalized to graph data [DB21; KBH+21; YCL+21]. Still, none of them can keep the text process-

ing capabilities of large pretrained language models. An idea for this combination is to spread the self-attention mechanisms across adjacent nodes in the graph, i. e., to propagate the query representations of each token to neighboring nodes and harvest the result of the soft key-value lookup from the neighbors. However, there are other possible ways for combined text and graph representation learning with Transformers. A different idea would be to flatten the adjacent nodes' local subgraph and supply it as extra input along with the text to a large language model. Considering lifelong learning on graphs, future work could investigate whether it would be beneficial to maintain a representative graph in memory throughout the evolution of the graph. For the automatic detection of unseen classes, future methods might make use of the graph structure more explicitly, which we used only implicitly within the graph convolution.

8.4 Summary

The key challenges discussed in this thesis will remain relevant for future research in joint representation learning for texts and graphs. For the time being, we have explored efficient matrix embedding models for text representation learning. When focusing on topical text classification, we found that a wide multilayer perceptron operating on a bag-of-words is surprisingly powerful. We continued with the bag-of-words representation for the analysis of multimodality, first with autoencoders in bipartite graphs and then with graph neural networks in evolving graphs. Adaptability to new graph data was a critical challenge we tackled with a lifelong learning approach. However, this is not the end of the story, and the hope is that this thesis inspires future work to continue the exploration of a more holistic perspective on representation learning for texts and graphs.

Appendix A

Reproducibility and Published Resources

A.1 Reproducibility of Relevant Literature

Here, we briefly summarize reproducibility issues in the related work and show that the results presented in this thesis are unaffected by those.

In recommender systems, Dacrema, Cremonesi, and Jannach [DCJ19] have shown that many recently proposed approaches are, in fact, not reproducible. Only autoencoders, which we have also used as a basis for Chapter 6 were among the reproducible ones.

In graph neural networks, Shchur, Mumme, Bojchevski, and Günnemann [SMB+19] have shown that commonly used benchmark datasets (Cora, Citeseer, PubMed) are *not* sufficient to draw conclusions about advances in graph neural networks. Other dataset splits led to different rankings of the models. Only in following benchmarking and reproducibility studies [DJL+20] and by using more profound benchmarking dataset collections [HFZ+20], the key components of successful GNN architectures could be derived. We have considered these results in our experiments with graph neural networks in Chapter 7.

Then again, Lv et al. [LDL+21] raise the same question "are we really making much progress?" as in [DCJ19], but now for heterogeneous graph neural networks. The authors could show that properly configured standard approaches such as graph attention network, could outperform other approaches that have been specifically designed for heterogeneous graphs. The results presented in Chapter 7 are not affected by these issues, since we have worked with homogeneous GNNs.

In the research on large pretrained language model (LM), the matter of reproducibility is different. What is now known as Foundation Models [BHA+21], is effectively a different learning setting. Due to their ever-increasing scale, the research and training of these large models is shifted into the hands of only few global players. Countless technical decisions are necessary and important to build such models, yet it is, for the most part, not clear what model design decisions

are important. For instance, Liu, Dai, So, and Le [LDS+21] have recently raised the question to what extent self-attention is necessary in Transformers.

Dehghani et al. [DAB+21] provide an overview of measures for efficiency and argue that reporting multiple of them is important to fully determine the efficiency of a method. The authors further point out that real time measurements are more accurate as the number of floating point operations (FLOPs), as those do not account for parallelism. Throughout our experiments on efficiency, we report the number of parameters along with a speed measure: throughput or wall-clock time.

A.2 Reproducibility of Our Experiments

To ensure reproducibility, we have followed the key principle of investing the same effort into hyperparameter tuning of the baselines and hyperparameter tuning of the proposed methods. Furthermore, we adhered to at least one of the two strategies a) to evaluate the same model on multiple tasks and datasets and b) to repeat the experiments on the same dataset multiple times.

In more detail, in Chapters 3, 4, and 5, we followed evaluation protocols and conventions of the community. That is the SentEval framework [CK18] to evaluate sentence representations, the GLUE benchmark [WSM+19] to evaluate general text representation learning models, and five commonly used datasets for (topical) text classification [YML19]. In Chapter 6 and Chapter 7, we have created our own experimental frameworks and respective evaluation protocols. In Chapter 6, the difference to almost all existing recommender systems was that *all* test data comes from new, previously unseen documents. In Chapter 7, we have introduced the first generic framework for lifelong learning on graphs. In both experimental frameworks, we opted to tune the hyperparameters on a single dataset, and then applying the determined hyperparameters to the other datasets. In order to draw a rich picture in these two new settings, we have evaluated a wide range of different models (7 in Chapter 6, and 6 in Chapter 7) including strong baselines.

Each experiment has been repeated five times in Chapter 5, three times in Chapter 6, and ten times in Chapter 7. In the other Chapters 3 and 4, we evaluated the same models on multiple different tasks, while using the same original model for all tasks. In Chapter 6 and Chapter 7, we have conducted chronological splits of the data to take a realistic distribution shift into account. References to the code for reproducing all the experiments is provided in the following section.

A.3 Published Resources

In Table A.1, we provide an overview of the resources published with this thesis (code, data, and models) to ensure reproducibility and enable reuse.

Table A.1. Published resources in the context of this thesis

Chapter	Resource Name	Type	Location
3	Word matrices	code	https://github.com/lgalke/word2mat
3	CMOW/CBOW models	models	https://doi.org/10.5281/zenodo.3933322
4	Cross-architecture distillation	code	https://github.com/lgalke/cross-architecture-distillation
4	Distilled models	models	https://doi.org/10.5281/zenodo.6533889
5	WideMLP and language models	code	https://github.com/lgalke/text-clf-baselines
6	Multimodal autoencoders	code	https://github.com/lgalke/aaerec
7	Lifelong learning on graphs	code	https://github.com/lgalke/lifelong-learning
7	Datasets for lifelong learning	dataset	https://doi.org/10.5281/zenodo.3764770

Appendix B

Supplementary Material: Word Matrices

B.1 Comparison of Training Objectives

In Section 3.3.4, we describe a more general training objective than the classical CBOW objective by Mikolov et al. [MSC+13]. The original objective always sets the center word from the window of tokens $(w_{t-c}, \ldots, w_{t+c})$ as target word, $w_O = w_t$. In preliminary experiments, this did not yield satisfactory results. We believe that this was because the model relied too much on always having the same number of matrices before and after the missing word. Hence, we experimented a variant where the target word is sampled randomly from a uniform distribution, $w_O := \mathcal{U}(\{w_{t-c}, \ldots, w_{t+c}\})$.

To test the effectiveness of this modified objective, we evaluate it with the same experimental setup as described in Section 3.5. Table B.1 lists the results on the linguistic probing tasks. CMOW-C and CBOW-C refer to the models where the center word is used as the target. CMOW-R and CBOW-R refer to the models where the target word is sampled randomly. While CMOW-R and CMOW-C perform comparably on most probing tasks, CMOW-C yields 5 points lower scores on WordContent and BigramShift. Hence, CMOW-R also outperforms CMOW-C on 10 out of 11 supervised downstream tasks and on all unsupervised downstream tasks, as shown in Table B.2 and Table B.3, respectively. On average over all downstream tasks, the relative improvement is 20.8%. For CBOW, the scores on downstream tasks increase on some tasks and decrease on others. The differences are small, presumably because CBOW is not sensitive to word order. On average over all 16 downstream tasks, CBOW-R scores 0.1% lower than CBOW-C.

B.2 Comparison of Initialization Strategies

In Section 3.3.5, we have suggested to use an initialization strategy that is specifically tailored towards word matrices. We argue why it is more adequate for training

Table B.1. Scores for different training objectives on the linguistic probing tasks.

Method	Depth	BShift	SubjNum	Tense	CoordInv	Length	ObjNum	TopConst	SOMO	WC
CMOW-C	36.2	66.0	81.1	78.7	61.7	83.9	79.1	73.6	50.4	66.8
CMOW-R	35.1	70.8	82.0	80.2	61.8	82.8	79.7	74.2	50.7	72.9
CBOW-C	34.3	50.5	79.8	79.9	53.0	75.9	79.8	72.9	48.6	89.0
CBOW-R	33.0	49.6	79.3	78.4	53.6	74.5	78.6	72.0	49.6	89.5

Table B.2. Scores for different training objectives on the supervised downstream tasks.

Method	SUBJ	CR	MR	MPQA	MRPC	TREC	SICK-E	SST2	SST5	STS-B	SICK-R
CMOW-C	85.9	72.1	69.4	87.0	71.9	85.4	74.2	73.8	37.6	54.6	71.3
CMOW-R	87.5	73.4	70.6	87.3	69.6	88.0	77.2	74.7	37.9	56.5	76.2
CBOW-C	90.0	79.3	74.6	87.5	72.9	85.0	80.0	78.4	41.0	60.5	79.2
CBOW-R	90.0	79.2	74.0	87.1	71.6	85.6	78.9	78.5	42.1	61.0	78.1

CMSMs than classic strategies that initialize all parameters with random values close to zero, and use it in our experiments to train CMOW.

To verify the effectiveness of our initialization strategy empirically, we evaluate it with the same experimental setup as described in Section 3.5. The only difference is the initialization strategy, where we include Glorot initialization [GB10] and the standard initialization from $\mathcal{N}(0, 0.1)$.

Table B.4 shows the results on the probing tasks. While Glorot achieves slightly better results on BShift and TopConst, CMOW's ability to memorize word content is improved by a wide margin by our initialization strategy. This again affects the downstream performance as shown in Table B.5 and B.6, respectively: 7 out of 11 supervised downstream tasks and 4 out of 5 unsupervised downstream tasks improve. On average, the relative improvement of our strategy compared to Glorot initialization is 2.8%.

Table B.3. Scores for different training objectives on the unsupervised downstream tasks.

Method	STS12	STS13	STS14	STS15	STS16
CMOW-C	27.6	14.6	22.1	33.2	41.6
CMOW-R	39.2	31.9	38.7	49.7	52.2
CBOW-C	43.5	49.2	57.9	63.7	61.6
CBOW-R	43.5	50.0	57.7	63.2	61.0

Table B.4. Scores for initialization strategies on probing tasks.

Initialization	Depth	BShift	SubjNum	Tense	CoordInv	Length	ObjNum	TopConst	SOMO	WC
$\mathcal{N}(0, 0.1)$	29.7	71.5	82.0	78.5	60.1	80.5	76.3	74.7	51.3	52.5
Glorot	31.3	72.3	81.8	78.7	59.4	81.3	76.6	74.6	50.4	57.0
Ours	35.1	70.8	82.0	80.2	61.8	82.8	79.7	74.2	50.7	72.9

Table B.5. Scores for initialization strategies on supervised downstream tasks.

Initialization	SUBJ	CR	MR	MPQA	MRPC	TREC	SICK-E	SST2	SST5	STS-B	SICK-R
$\mathcal{N}(0, 0.1)$	85.6	71.5	68.4	86.2	71.6	86.4	73.7	72.3	38.2	53.7	72.7
Glorot	86.2	74.4	69.5	86.5	71.4	88.4	75.4	73.2	38.2	54.1	73.6
Ours	87.5	73.4	70.6	87.3	69.6	88.0	77.2	74.7	37.9	56.5	76.2

Table B.6. Scores for initialization strategies on unsupervised downstream tasks.

Initialization	STS12	STS13	STS14	STS15	STS16
$\mathcal{N}(0, 0.1)$	37.7	26.5	33.3	44.7	50.3
Glorot	39.6	27.2	35.2	46.5	51.6
Ours	39.2	31.9	38.7	49.7	52.2

Appendix C

Supplementary Material: Cross-Architecture Distillation

C.1 Hyperparameters for Distillation

Table C.1. Hyperparameter Search Space

Hyperparameter	Range	Optimization method
	— *General Distillation* —	
Learning rate	$\{10^{-3}, 5 \cdot 10^{-4}, 10^{-4}, 5 \cdot 10^{-5}, 10^{-5}\}$	grid search
Warmup steps	$\{0, 500\}$	grid search
Embedding dropout	$\{0, 0.1\}$	grid search
Hidden unit dropout	$\{0.2\}$	fixed
Batch size	{1,8,32,64,128,256}	manual
	— *Task-specific Distillation* —	
Learning rate	$\{10^{-3}, 5 \cdot 10^{-4}, 10^{-4}, 5 \cdot 10^{-5}, 10^{-5}, 5 \cdot 10^{-6}\}$	grid search

We list hyperparameter search spaces along with their optimization methods in Table C.1. We optimize over all six initial learning rates, namely $\{10^{-3}$, $5 \cdot 10^{-4}$, 10^{-4}, $5 \cdot 10^{-5}$, and $10^{-5}\}$. All initial learning rates decay linearly over the course of training. For the fine-tuning we further test an even smaller learning rate $5 \cdot 10^{-6}$ Note, we also experimented with using warmup steps versus no warmup for the learning rate schedule with negligible differences.

C.2 Extended Results

In Table C.2, we report an extended version of the comparison with the literature. The best performing model per task are marked in bold. Here, we also include our BERT-base teacher model as well as TinyBERT [JYS+20] and Tang et al. [TLL+19]'s distilled BiLSTM. However, note that TinyBERT and BiLSTM are not fully compara-

ble, because the reported numbers are on the GLUE test set, while we have used the development set for our experiments.

Table C.2. Scores on the GLUE development set. Our best performing general distillation and task-specific distillation models are highlighted in bold font per task. References indicate sources of scores. The ⋆-symbol indicates numbers on the official GLUE test set. 'Hybrid' denotes CMOW/CBOW-Hybrid.

	Score	CoLA	MNLI-m	MRPC	QNLI	QQP	RTE	SST-2	STS-B	WNLI
— large-scale pre-trained language models —										
ELMo [SDC+20]	68.7	44.1	68.6	76.6	71.1	86.2	53.4	91.5	70.4	56.3
BERT-base [SDC+20]	79.5	56.3	86.7	88.6	91.7	89.6	69.3	92.7	89.0	53.5
BERT-base (teacher)	78.9	57.9	84.2	84.6	91.4	89.7	67.9	91.7	88.0	54.9
Word2rate Hybrid [PLP21]	—	—	—	—	—	—	—	65.7	53.1	—
— general distillation baselines —										
DistilBERT [SDC+20]	77.0	51.3	82.2	87.5	89.2	88.5	59.9	91.3	86.9	56.3
MobileBERT [SYS+20]	—	51.1	84.3	88.8	91.6	70.5	70.4	92.6	84.8	—
— task-specific distillation baselines —										
⋆TinyBERT [JYS+20]	—	54.0	84.5	90.6	91.1	88.0	70.4	93.0	90.1	—
⋆BiLSTM [TLL+19]	—	—	73.0	—	78.2	—	—	90.7		
CBOW-FFN [WPI20]	—	10.0	—	—	—	—	—	79.1	—	—
BiLSTM [WPI20]	—	10.0	—	—	—	—	—	80.7	—	—
— general distillation (ours) —										
Bidi. Hybrid + Linear	65.1	15.0	63.6	**80.9**	70.7	84.3	56.7	**84.0**	71.1	**59.2**
Bidi. Hybrid + MLP	66.6	**16.7**	**66.6**	79.7	**71.7**	**87.2**	**61.0**	82.9	**76.9**	56.3
— task-specific distillation (ours) —										
Hybrid + MLP (rand. init.)	62.5	13.1	62.5	74.3	71.5	**86.6**	58.1	83.1	58.6	56.3
Bidi. Hybrid + MLP (rand. init.)	63.2	13.0	**63.3**	75.7	**72.6**	86.1	57.4	83.3	59.7	57.7
Bidi. Hybrid + MLP (pretrained)	64.6	**23.3**	61.8	75.0	72.0	86.3	**59.9**	82.9	62.9	57.7

Appendix D

Supplementary Material: Text Classification

D.1 Practical Guidelines for Designing a WideMLP

On the basis of our results, we provide recommendations for designing a WideMLP baseline.

Tokenization We recommend using modern subword tokenizers such as BERT-like WordPiece [WSC+16] that yield a high coverage while needing a relatively small vocabulary.

Input representation In contrast to conventional wisdom [IMB+15], we find that pretrained embeddings, e. g., GloVe, can have a detrimental effect when compared to using an MLP with one wide hidden layer. Such an MLP circumvents the bottleneck of the small dimensionality of word embeddings and has a higher capacity.

Depth vs. width In text classification, width seems more important than depth. We recommend to use a single, wide hidden layer, i. e., one input-to-hidden and one hidden-to-output layer, e. g., with 1,024 hidden units and ReLU activation. While this might be overparameterized for single-label text classification tasks with few classes, we rely on recent findings that overparameterization leads to better generalization [NLB+18; NKB+20].

We further motivate the choice of using wide layers our previous results in multilabel text classification [GMS+17], which has shown that MLP outperforms all tested classical baselines such as support vector machines, k-Nearest Neighbors, and logistic regression. Our follow-up work [MGS18] then found that also CNN and LSTM models do not substantially improve over the wide MLP.

Having a fully-connected layer on-top of a bag-of-words leads to a high number of learnable parameters. Still, the wide first input-to-hidden layer can be implemented efficiently by using an embedding layer followed by aggregation, which avoids large matrix multiplications.

In our experiments, we did not observe any improvement with more hidden layers (WideMLP-2), as suggested by Iyyer, Manjunatha, Boyd-Graber, and III [IMB+15], but it might help for other, more challenging, datasets.

Optimization and regularization We seek to find an optimization strategy that does not require dataset-specific hyperparameter tuning. This comprises optimizing cross-entropy with Adam [KB15] and default learning rate 10^{-3}, a linearly decaying learning rate schedule and training for a high amount of steps [NKB+20] (we use 100 epochs) with small batch sizes (we use 16) for sufficient stochasticity. For regularization during this prolonged training, we suggest to use a high dropout ratio of 0.5. Regarding initialization, we rely on framework defaults, i. e., $\mathcal{N}(0, 1)$ for the initial embedding layer and random uniform $\mathcal{U}(-\sqrt{d_{\text{input}}}, \sqrt{d_{\text{output}}})$ for subsequent layers' weight and bias parameters.

D.2 Connection between BoW-MLP and TextGCN

TextGCN uses the PMI matrix to set up edge weights for word-word connections. A single layer Text-GCN is a BoW-MLP, except for the document embedding. The one-hop neighbors are words which are aggregated after a nonlinear transform. The basic GCN equation $\boldsymbol{H} = \sigma(\hat{\boldsymbol{A}}\boldsymbol{X}\boldsymbol{W})$ reveals that the order of transformation and neighborhood aggregation is irrelevant. The document embedding implies that TextGCN is a semisupervised technique. Truly new documents, as in inductive learning scenarios, would need a special treatment such as using an all zero embedding vector.

A two-layer MLP can be characterized by the equation $\hat{y} = \boldsymbol{W}^{(2)}\sigma(\boldsymbol{W}^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)}) + \boldsymbol{b}^{(2)}$. On bag-of-words inputs, the first layer $\boldsymbol{W}^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)}$ can be replaced by an equivalent embedding layer with weighting (e. g., TF-IDF or length normalization) being applied during aggregation of the embedding vectors.

The first layer of TextGCN is equivalent to aggregating embedding vectors. A standard GCN layer with shared weights has the form (assuming self-loops have been inserted)

$$\boldsymbol{h}_i = \sum_{j \in N(i)} a_{ij} \boldsymbol{W}^{(1)} \boldsymbol{x}_j + \boldsymbol{b}^{(1)}$$

Now in TextGCN node features are given by the identity, such that $\boldsymbol{x}_j$ is one-hot. Then we can rewrite the first layer of Text-GCN as an aggregation of embeddings $\boldsymbol{E}$. We gain

$$\boldsymbol{h}_i = \sum_{j \in N(i)} a_{ij} \boldsymbol{E}_j$$

as $\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$ may again be replaced by an embedding matrix if applied to one hot vectors $\boldsymbol{x}$. Now $\boldsymbol{E}$ contains two types of embedding vectors: word embeddings and document embeddings corresponding to word nodes and document nodes. We see that the first layer of TextGCN is essentially an aggregation of word embeddings plus the document embedding. Only with a second layer, TextGCN considers the embedding of other documents whose words are connected to the present documents' words.

D.3 Equivalence of Micro-F1 and Accuracy in Multiclass Classification

In multiclass classification, we have a single true label for each instance and the predictions are constrained to a single prediction per instance. As a consequence, the measures accuracy and Micro-F1 coincide to the same formula.

Micro F1 aggregates true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) globally. It can be expressed as:

$$\text{Micro} - \text{F}_1 = \frac{2 \sum_c \text{TP}_c}{2 \sum_c \text{TP}_c + \sum_c \text{FP}_c + \sum_c \text{FN}_c},$$

where c iterates over all classes.

While the accuracy can be expressed as:

$$\text{Acc} = \frac{\sum_c \text{TP}_c + \sum_c \text{TN}_c}{\sum_c \text{TP}_c + \sum_c \text{TN}_c + \sum_c \text{FP}_c + \sum_c \text{FN}_c}$$

In multiclass classification, every true positive is also a true negative for another class. When summing those up over the entire dataset, we obtain

$$\sum_c \text{TP}_c = \sum_c \text{TN}_c.$$

Thus, we can rewrite

$$2\sum_c \mathrm{TP}_c = \sum_c \mathrm{TP}_c + \sum_c \mathrm{TN}_c$$

and see that the Micro-F1 and accuracy are equivalent in the multiclass (a.k.a. single-label) case.

Appendix E

Supplementary Material: Recommender Systems

E.1 Extended Results on the PubMed Dataset

The PubMed dataset was a special case in Chapter 6. This is because it was the only dataset that came with subject labels *and* citations. Therefore, we also included the subject labels as input to predict citations in the "all-metadata" case. Intuitively, the subject label input helps models to provide recommendations more easily, because the known topic of the publication is indicative for citation candidates.

In Figure E.1 , we provide more results to isolate the effect of using MeSH labels as input on PubMed. While using MeSH labels improves the performance, the models are similarly affected by the drop threshold in both cases.

We also studied the effect of a larger partial set of items as input on the results. In our experiments, MLP and VAE show a comparatively high performance when only few items are available as input (higher values of drop ratio). In order to decide if the reason for these results is because of larger training or because these methods are more useful in separating relevant documents from noise, we split the documents based on the median number of references as input. This allows us to compare results with longer or shorter lists of references as training input. As Table E.1 shows, all compared models benefit from having a longer input list without any other metadata. Still, better scores are achieved when using metadata. As expected, using metadata is particularly helpful when a smaller item set is provided as input.

E.2 Mean Average Precision Results

In addition to the mean reciprocal rank (MRR), we also assess the mean average precision (MAP) for the experiments investigating the effect of the number of dropped elements. Results are provided in Figure E.2 (citations), Figure E.3 (subject

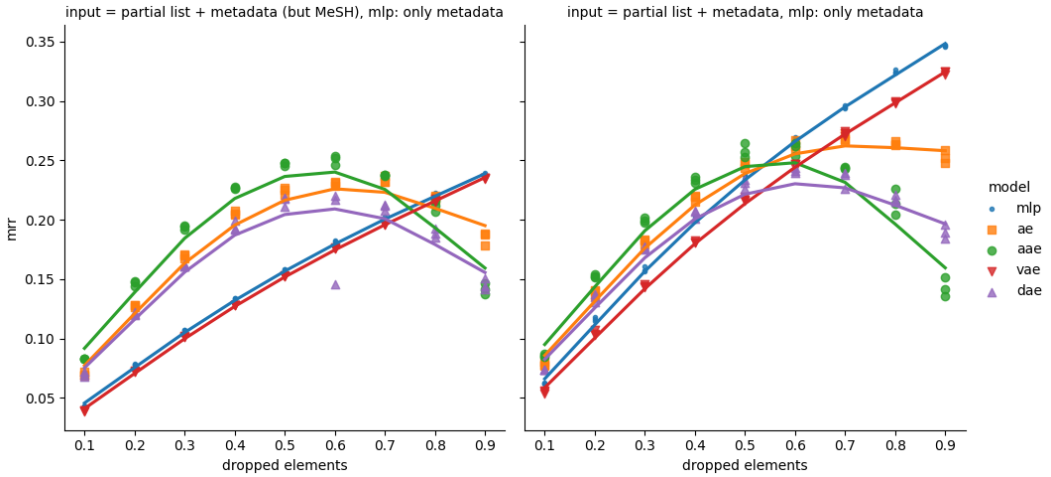


Figure E.1. MRR of predicted citations on the test set with varying number dropped elements for the PubMed citation dataset. The minimum item occurrence threshold is set to 55. *Left*: The partial set of items is given along with the document title, the authors, and the journal title. *Right*: The partial set of items is given along with the document title, the authors, the journal title, and the MeSH labels.

Table E.1. MRR of predicted citations for PubMed using either only documents with fewer references than the median number of references or with more references than the median number of references.

Model	<30 references		⩾30 references	
	partial list	metadata	partial list	metadata
IC	.1958 (.0019)	-	.2136 (.0004)	-
SVD	.1409 (.0004)	-	.1503 (.0002)	-
AE	.1734 (.0007)	.2946 (.0013)	.1996 (.0076)	.2520 (.0015)
AAE	.1485 (.0023)	.2756 (.0013)	.1957 (.0084)	.2256 (.0196)
VAE	.1337 (.0004)	.3580 (.0010)	.0907 (.0002)	.2750 (.0009)
DAE	.1311 (.0002)	.2710 (.0024)	.2022 (.0100)	.2049 (.0011)
MLP	**-**	**.3633 (.0014)**	-	**.2926 (.0011)**

labels), and Table E.2. The models show substantially the same behavior with MAP as with MRR. The only exception is that the VAE demonstrates an overall lower increase of performance with an increasingly higher number of dropped elements for the subject indexing datasets.

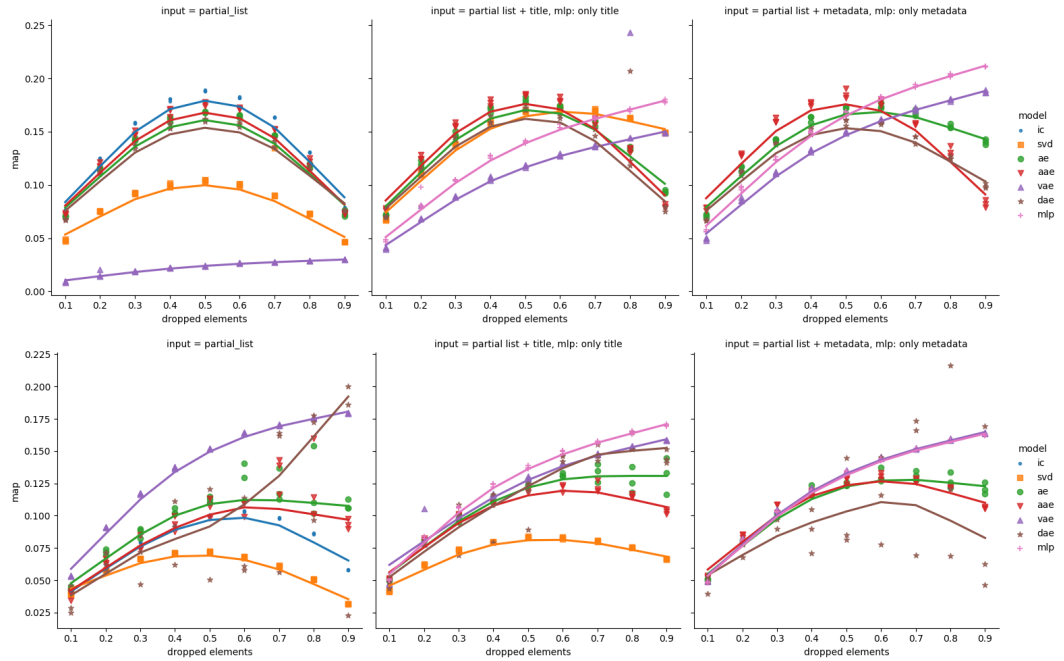


Figure E.2. MAP of predicted citations on the test set with varying number dropped elements for the PubMed (top row) and ACM (bottom row) citation datasets. The minimum item occurrence threshold is set to 55. *Left*: Only the partial set of items is given. *Center*: The partial set of items along with the document title is given. *Right*: The partial set of items is given along with the document title, the authors, the journal title, and the MeSH labels. MLP can only make use of either titles or titles, authors, journal titles, and MeSH labels.

Table E.2. MAP of predicted citations for PubMed using either only documents with fewer references than the median number of references (left column) or with more references than the median number of references (right column).

Model	**<30 references**		**⩾30 references**	
	partial list	**metadata**	**partial list**	**metadata**
IC	.1641 (.0015)	-	.1321 (.0006)	-
SVD	.1165 (.0009)	-	.0838 (.0004)	-
AE	.1205 (.0006)	.2299 (.0011)	.1199 (.0022)	.1503 (.0007)
AAE	.0902 (.0011)	.2250 (.0007)	.1187 (.0019)	.1574 (.0237)
VAE	.0795 (.0001)	.2767 (.0009)	.0384 (.0002)	.1670 (.0003)
DAE	.0808 (.0001)	.2076 (.0018)	.1190 (.0053)	.1200 (.0011)
MLP	**-**	**.2818 (.0017)**	-	**.1818 (.0009)**

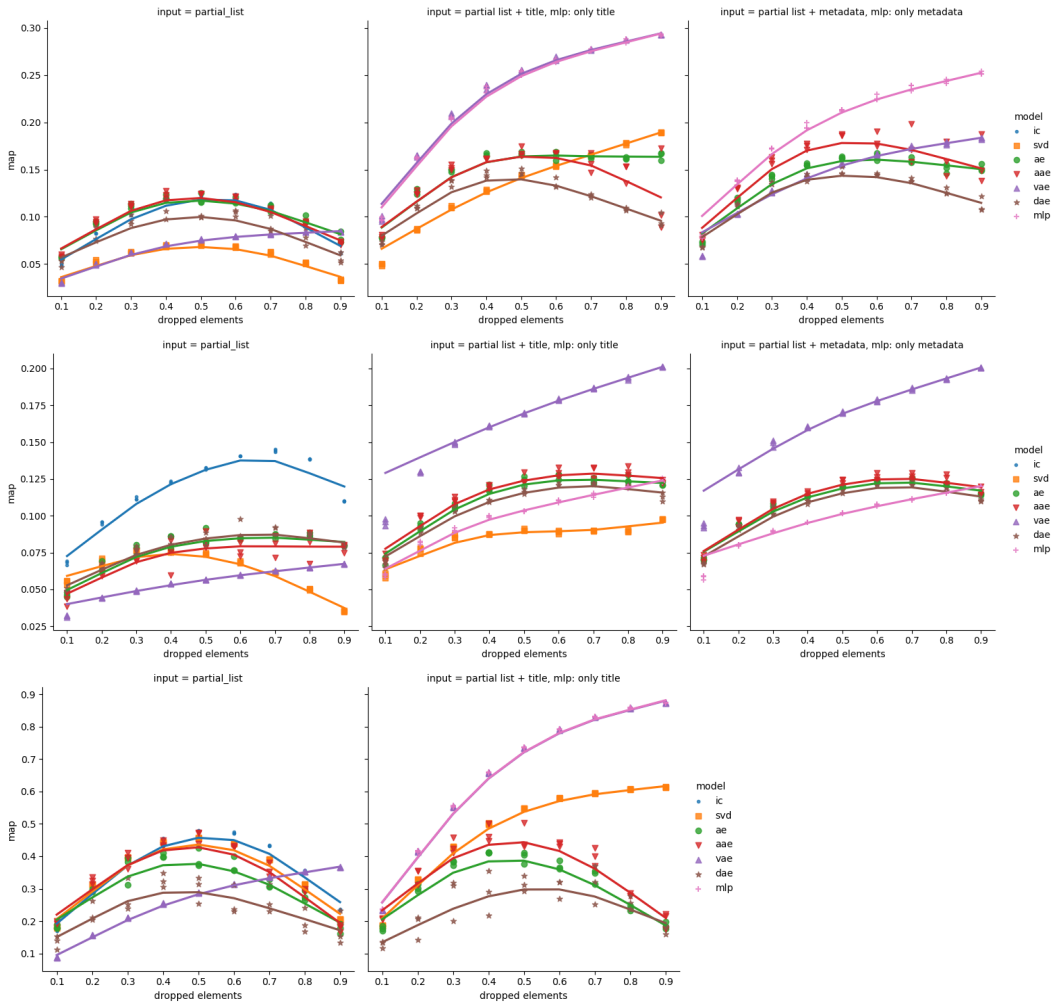


Figure E.3. MAP of predicted subject labels on the test set with varying number of dropped elements for the EconBiz (top row), IREON (middle row), and Reuters (bottom row) datasets. The minimum item occurrence threshold is set to 20. *Left*: Only the partial set of items is given. *Center*: The partial set of items along with the title is given. *Right*: The partial set of items along with the document title and authors is given. MLP can only use either titles or titles and authors.

Appendix F

Supplementary Material: Lifelong Learning on Graphs

F.1 Proof: k-Neighborhood Time Differences tdiff_k is Equivariant to Temporal Granularity

We show that our k-neighborhood time difference measure tdiff_k is equivariant to temporal granularity: We assume without loss of generality that t_m is more fine-grained than t_y, i. e., $a > 1$. Because tdiff_k is a multiset of time differences from which we take percentiles to determine history sizes, it is sufficient to show that the time difference $t(u) - t(v)$ between two nodes $u, v \in V$ is equivariant to the temporal granularity factor a, or more precisely: $\forall u, v \in V : a \cdot |t_y(u) - t_y(v)| \in]|t_m(u) - t_m(v)| - a; |t_m(u) - t_m(v)| + a[$.

Prerequisite (PRE) With $t_y(u) = \left\lfloor \frac{t_m(u)}{a} \right\rfloor$ we have $\frac{t_m(u)}{a} \leqslant t_y(u) < \frac{t_m(u)+a}{a} \Rightarrow t_m(u) \leqslant a \cdot t_y(u) < t_m(u) + a$. Note that the left-hand side is less than or equal due to rounding down, while the right-hand side adds a time step a, which makes it a true inequality.

Proof. Using the prerequisite, we now show that

(i) $a \cdot |t_y(u) - t_y(v)| > |t_m(u) - t_m(v)| - a$, and
(ii) $a \cdot |t_y(u) - t_y(v)| < |t_m(u) - t_m(v)| + a$

via case differentiation.

Case (i-a): $t_y(u) = t_y(v)$ The left-hand side of inequality (i) becomes zero and it remains to show that $|t_m(u) - t_m(v)| < a$. We apply (PRE) to find the highest possible value for the term $|t_m(u) - t_m(v)|$ with respect to t_y such that the term is still smaller than a. The highest possible value for $t_m(u)$, expressed in terms of t_y,

is $a \cdot t_y(u) + \epsilon$ with $0 < \epsilon < a$. This is because $a \cdot t_y(u)$ is the upper bound of $t_m(u)$ in the inequality of the prerequisite (PRE) and we insert a small but positive ϵ to account for being "truly lower". The smallest possible value for $t_m(v)$ is $a \cdot t_y(u)$. Again, we take this value $a \cdot t_y(u)$ from the prerequisite, where it is the lower bound of $t_y(u)$. Then, we have $|a \cdot t_y(u) + a - \epsilon - a \cdot t_y(v)| < a$. Now, as $t_y(u) = t_y(v)$, we obtain $|a - \epsilon| < a$.

Case (i-b): $t_y(u) \neq t_y(v)$ We transform the left-hand side of (i) to $|a \cdot t_y(u) - a \cdot t_y(v)|$, while recalling that $t_y \in \mathbb{N}_{>0}$. We use (PRE) to obtain $|t_m(u) - t_m(v)|$ as the smallest possible value of the left-hand side. Now, the left and right sides are the same except for $-a$ on the right. As $a > 1$ (is positive), inequality (i) is valid.

Case (ii-a): $t_y(u) = t_y(v)$ The left-hand side of (ii) becomes zero and it remains to show that $0 < |t_m(u) - t_m(v)| + a$, which is true because $a > 1$.

Case (ii-b): $t_y(u) \neq t_y(v)$ Again, we transform the left-hand side of (ii) to $|a \cdot t_y(u) - a \cdot t_y(v)|$. This time, we are interested in the highest possible value with respect to (PRE), which is $|t_m(u) + a - \epsilon - t_m(v)|$ with $0 < \epsilon < a$. This is because the highest possible difference is between the upper bound $t_m + a - \epsilon$ and the lower bound t_m. With the triangle inequality, we obtain $|t_m(u) + a - \epsilon - t_m(v)| \leqslant |t_m(u) - t_m(v)| + |a - \epsilon| < |t_m(u) - t_m(v)| + a$, which is true because $|a - \epsilon| < a$. This concludes the proof. □

F.2 Details on Changes in the Class Sets

In this section, we provide details of the changes within the class set during the evaluation period for our newly compiled datasets: PharmaBio, DBLP-easy, and DBLP-hard. This is a major factor of the distribution shift since new classes appear over time in the DBLP datasets.

F.2.1 PharmaBio

All seven journal categories are present during the evaluation time span 1999–2016.

F.2.2 DBLP-Easy

On DBLP-Easy has 12 venues in total within the evaluation timespan 2004–2015. During that timespan, we have one bi-annual conference: "international conference on document analysis and recognition" and four new venues

- Journal of Cognitive Neuroscience — starts in 2005
- IEEE Transactions on Audio, Speech, and Language Processing — starts in 2006
- Iet Communications — starts in 2007
- IEEE Wireless Communication Letters — starts in 2012

F.2.3 DBLP-Hard

DBLP-Hard has 73 venues in total over the evaluation timespan 2004–2015. In the following we give the details, which venues were newly added during that timespan, discontinued, were bi-annual, or were irregular.

New venues DBLP-hard has 23 new venues in total:

- PLOS Computational Biology — starts in 2005
- Journal of Cognitive Neuroscience — starts in 2005
- wireless and mobile computing, networking and communications — starts in 2005
- IEEE Transactions on Audio, Speech, and Language Processing — starts in 2006
- BMC Systems Biology — starts in 2007
- Iet Communications — starts in 2007
- Marketing Science — starts in 2008
- International Conference on Intelligent Robotics and Applications — started in 2008
- Biomedical Engineering and Informatics — starts in 2008, not held in 2010
- Workshop on Applications of Signal Processing to Audio and Acoustics — starts in 2009 (and is biannual)
- Journal of Zhejiang University Science C — starts in 2010

- Broadband and Wireless Computing, Communication and Applications — starts in 2010 (not present in 2015)
- International Conference on Multimedia Retrieval — starts in 2011
- Innovative Mobile and Internet Services in Ubiquitous Computing — starts in 2011
- Conference on Computer as a tool — starts in 2011 (and is biannual)
- International Conference on Signal and Image Processing Applications — starts in 2011 (and is biannual)
- IEEE Wireless Communication Letters — starts in 2012
- IEEE Transactions on Human-Machine Systems — starts in 2013
- IEEE Journal of Biomedical and Health Informatics — starts in 2013
- IEEE Internet of Things Journal — starts in 2014
- Picture Coding Symposium — present in 2009, 2010, 2012, 2013, 2015
- International Convention on Information and Communication Technology, Electronics and Microelectronics — present in 2011, 2012, 2014
- International Journal of Central Banking — present in 2011, 2014

Discontinued venues The venue "Computational Intelligence for Modelling, Control and Automation" was present in 2005, 2006, 2008, and was then discontinued.

Bi-annual venues Nine venues occur every second year.

- International Conference on Control, Automation, Robotics and Vision
- Affective Computing and Intelligent Interaction
- World of Wireless Mobile and Multimedia Networks
- World Haptics Conference
- International Conference on Document Analysis and Recognition
- Information Sciences, Signal Processing and their Applications
- Workshop on Applications of Signal Processing to Audio and Acoustics
- Conference on Computer as a Tool
- International Conference on Signal and Image Processing Applications

Irregular venues The following 6 venues had some irregularities:

- International Symposium on Biomedical Imaging — not present in 2005
- Sensor Mesh and ad hoc Communications and Networks — not present in 2006, 2014, 2015
- Robotics and Biomimetics — not present in 2008
- Picture Coding Symposium — present in 2009, 2010, 2012, 2013, 2015
- International Convention on Information and Communication Technology, Electronics and Microelectronics — present in 2011, 2012, 2014
- International Journal of Central Banking — present in 2011, 2014

F.3 Extended Results for Unseen Class Detection

As in the main part of this thesis, the measures we report are MCC of the reject decision and Macro-F1 with a virtual OOD class. To recall, the MCC score specifically reflects the capabilities to detect unseen classes and the Open Macro-F1 score reflects the performance on all classes including a special class for instances from unseen classes.

In Table F.1, we report the results for the development dataset DBLP-easy, on which the hyperparameters were tuned for maximum accuracy, before the hyperparameters have been transferred to the test dataset DBLP-hard.

In Table F.2 and Table F.3, we report the full results of our experiments on unseen class detection on the DBLP-hard dataset.

When comparing DBLP-easy and DBLP-hard, we see that the absolute F1 score and the MCC score attained on DBLP-easy are higher than the absolute scores on DBLP-hard, which is expected since DBLP-hard has more classes and DBLP-easy is the subset of data on which hyperparameters were tuned.

Table F.1. Results for unseen class detection on the development set DBLP-easy with GraphSAGE as base model (average of 5 repetitions). α indicates that risk reduction is used with the respective factor for the standard deviation, τ is the minimum threshold. Runs named gDOC are trained with weighted cross entropy. DOC is our baseline.

		MCC		**Open Macro-F1**	
		cold	warm	cold	warm
c	**Open Learning Method**				
1	DOC ($\tau = 0.50$)	.05	.07	.25	.25
	DOC ($\tau = 0.50, \alpha = 3.0$)	.05	.07	.25	.25
	gDOC ($\tau = 0.50$)	.04	.08	.30	**.33**
	gDOC ($\tau = 0.50, \alpha = 3.0$)	.04	.05	.30	.32
	gDOC ($\tau = 0.75$)	.04	**.07**	.30	.30
3	DOC ($\tau = 0.50$)	.05	.05	.28	.30
	DOC ($\tau = 0.50, \alpha = 3.0$)	.05	.05	.28	.30
	gDOC ($\tau = 0.50$)	.05	.08	.34	.34
	gDOC ($\tau = 0.50, \alpha = 3.0$)	.06	.08	.34	.34
	gDOC ($\tau = 0.75$)	.07	**.09**	.34	.34
6	DOC ($\tau = 0.50$)	.06	.06	.31	.32
	DOC ($\tau = 0.50, \alpha = 3.0$)	.06	.06	.31	.32
	gDOC ($\tau = 0.50$)	.07	.07	.35	.35
	gDOC ($\tau = 0.50, \alpha = 3.0$)	.07	.07	.35	.35
	gDOC ($\tau = 0.75$)	.09	**.10**	.35	.35
∞	DOC ($\tau = 0.50$)	.07	.07	.32	.33
	DOC ($\tau = 0.50, \alpha = 3.0$)	.07	.07	.32	.33
	gDOC ($\tau = 0.50$)	.06	.06	.35	.35
	gDOC ($\tau = 0.50, \alpha = 3.0$)	.06	.06	.35	.35
	gDOC ($\tau = 0.75$)	.08	**.10**	.35	.35

Table F.2. Extended results for unseen class detection on DBLP-hard with GraphSAGE as base model (average of 5 repetitions). α indicates that risk reduction is used with the respective factor for the standard deviation, τ is the minimum threshold. The gDOC models are trained with weighted binary-cross entropy. History sizes 1 and 3.

		MCC		**Open Macro-F1**	
		cold	warm	cold	warm
c	**Open Learning Method**				
1	DOC ($\tau = 0.50$)	.01	.04	.01	.01
	DOC ($\tau = 0.25$)	.01	.02	**.04**	**.04**
	DOC ($\tau = 0.50, \alpha = 3.0$)	.01	.02	.01	.01
	DOC ($\tau = 0.50, \alpha = 1.5$)	.01	.02	.01	.01
	DOC ($\tau = 0.25, \alpha = 1.5$)	.01	.04	**.04**	**.04**
	gDOC ($\tau = 0.75$)	.04	**.09**	**.13**	**.13**
	gDOC ($\tau = 0.50$)	.04	**.05**	**.13**	**.13**
	gDOC ($\tau = 0.25$)	.01	-	**.13**	**.13**
	gDOC ($\tau = 0.50, \alpha = 3.0$)	.04	**.05**	**.13**	**.13**
	gDOC ($\tau = 0.50, \alpha = 1.5$)	**.05**	**.05**	**.13**	**.13**
	gDOC ($\tau = 0.25, \alpha = 1.5$)	**.05**	.00	**.13**	**.13**
	gDOC ($\tau = 0.75, \alpha = 3.0$)	.04	**.09**	**.13**	**.13**
	gDOC ($\tau = 0.75, \alpha = 1.5$)	.04	**.09**	**.13**	**.13**
3	DOC ($\tau = 0.50$)	.02	.03	.02	.05
	DOC ($\tau = 0.25$)	.02	.04	.06	.12
	DOC ($\tau = 0.50, \alpha = 3.0$)	.02	.03	.02	.05
	DOC ($\tau = 0.50, \alpha = 1.5$)	.02	.03	.02	.05
	DOC ($\tau = 0.25, \alpha = 1.5$)	.02	.04	.06	.12
	gDOC ($\tau = 0.75$)	.05	**.08**	**.15**	**.15**
	gDOC ($\tau = 0.50$)	.05	.06	**.15**	**.15**
	gDOC ($\tau = 0.25$)	.01	.00	**.15**	**.15**
	gDOC ($\tau = 0.50, \alpha = 3.0$)	.05	.06	**.15**	**.15**
	gDOC ($\tau = 0.50, \alpha = 1.5$)	.06	.06	**.15**	**.15**
	gDOC ($\tau = 0.25, \alpha = 1.5$)	.06	.04	**.15**	**.15**
	gDOC ($\tau = 0.75, \alpha = 3.0$)	.05	**.08**	**.15**	**.15**
	gDOC ($\tau = 0.75, \alpha = 1.5$)	.05	**.08**	**.15**	**.15**

Table F.3. Extended results for unseen class detection on DBLP-hard with GraphSAGE as base model (average of 5 repetitions). α indicates that risk reduction is used with the respective factor for the standard deviation, τ is the minimum threshold. The gDOC models are trained with weighted binary-cross entropy. History sizes 6 and 3.

		MCC		**Open Macro-F1**	
		cold	warm	cold	warm
c	**Open Learning Method**				
6	DOC ($\tau = 0.50$)	.02	.03	.05	.08
	DOC ($\tau = 0.25$)	.02	.05	.11	.14
	DOC ($\tau = 0.50, \alpha = 3.0$)	.02	.03	.05	.08
	DOC ($\tau = 0.50, \alpha = 1.5$)	.02	.03	.05	.08
	DOC ($\tau = 0.25, \alpha = 1.5$)	.02	.05	.11	.14
	gDOC ($\tau = 0.75$)	.05	**.07**	**.16**	**.16**
	gDOC ($\tau = 0.50$)	.05	.06	**.16**	**.16**
	gDOC ($\tau = 0.25$)	.02	.02	**.16**	.15
	gDOC ($\tau = 0.50, \alpha = 3.0$)	.05	.06	**.16**	**.16**
	gDOC ($\tau = 0.50, \alpha = 1.5$)	.06	.06	**.16**	**.16**
	gDOC ($\tau = 0.25, \alpha = 1.5$)	.06	**.07**	**.16**	**.16**
	gDOC ($\tau = 0.75, \alpha = 3.0$)	.05	**.07**	**.16**	**.16**
	gDOC ($\tau = 0.75, \alpha = 1.5$)	.05	**.07**	**.16**	**.16**
∞	DOC ($\tau = 0.50$)	.02	.04	.08	.12
	DOC ($\tau = 0.25$)	.03	.06	.13	**.16**
	DOC ($\tau = 0.50, \alpha = 3.0$)	.02	.04	.08	.12
	DOC ($\tau = 0.50, \alpha = 1.5$)	.02	.04	.08	.12
	DOC ($\tau = 0.25, \alpha = 1.5$)	.04	**.06**	.13	**.16**
	gDOC ($\tau = 0.75$)	.05	.07	**.16**	**.16**
	gDOC ($\tau = 0.50$)	.04	.05	**.16**	**.16**
	gDOC ($\tau = 0.25$)	.02	.01	**.16**	**.16**
	gDOC ($\tau = 0.50, \alpha = 3.0$)	.05	.05	**.16**	**.16**
	gDOC ($\tau = 0.50, \alpha = 1.5$)	.06	.06	**.16**	**.16**
	gDOC ($\tau = 0.25, \alpha = 1.5$)	.06	.06	**.16**	**.16**
	gDOC ($\tau = 0.75, \alpha = 3.0$)	.05	**.07**	**.16**	**.16**
	gDOC ($\tau = 0.75, \alpha = 1.5$)	.05	**.07**	**.16**	**.16**

Bibliography

[AAC+19] Yuri Sousa Aurelio, Gustavo Matheus de Almeida, Cristiano Leite Castro, and Antônio de Pádua Braga. "Learning from imbalanced data sets with weighted cross-entropy function". In: *Neural Process. Lett.* 50.2 (2019), pp. 1937–1949. DOI: 10.1007/s11063-018-09977-1. URL: https://doi.org/10.1007/s11063-018-09977-1.

[AKM+20] Zafar Ali, Pavlos Kefalas, Khan Muhammad, Bahadar Ali, and Muhammad Imran. "Deep learning in citation recommendation models survey". In: *Expert Syst. Appl.* 162 (2020), p. 113790. DOI: 10.1016/j.eswa.2020.113790.

[ALM17] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. "A simple but tough-to-beat baseline for sentence embeddings". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL: https://openreview.net/forum?id=SyK00v5xx.

[AR17] Shima Asaadi and Sebastian Rudolph. "Gradual learning of matrix-space models of language for sentiment analysis". In: *Rep4NLP@ACL*. Association for Computational Linguistics, 2017, pp. 178–185.

[AS14] Charu Aggarwal and Karthik Subbian. "Evolutionary network analysis: a survey". In: *ACM Comput. Surv.* 47.1 (May 2014), 10:1–10:36. ISSN: 0360-0300. DOI: 10.1145/2601412.

[AUK+21] Zafar Ali, Irfan Ullah, Amin Khan, Asim Ullah Jan, and Khan Muhammad. "An overview and evaluation of citation recommendation models". In: *Scientometrics* (Mar. 2021). ISSN: 1588-2861. DOI: 10.1007/s11192-021-03909-y.

[BAB+17] Julio Barbieri, Leandro G. M. Alvim, Filipe Braida, and Geraldo Zimbrão. "Autoencoders and recommender systems: COFILS approach". In: *Expert Syst. Appl.* 89 (2017), pp. 81–90.

[BB16] Abhijit Bendale and Terrance E. Boult. "Towards open set deep networks". In: *CVPR*. IEEE, 2016. DOI: 10.1109/CVPR.2016.173.

[BB19] Jinxin Bai and Zhijie Ban. "Collaborative multi-auxiliary information variational autoencoder for recommender systems". In: *Proceedings of the 2019 11th International Conference on Machine Learning and Computing, ICMLC '19, Zhuhai, China, February 22-24, 2019*. ACM, 2019, pp. 501–505. DOI: 10.1145/3318299.3318336.

[BBC+21] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. *Geometric deep learning: grids, groups, graphs, geodesics, and gauges*. 2021. arXiv: 2104.13478 [cs.LG].

[BCB15] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015. URL: http://arxiv.org/abs/1409.0473.

[BCN06] Cristian Bucila, Rich Caruana, and Alexandru Niculescu-Mizil. "Model compression". In: *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, August 20-23, 2006*. ACM, 2006, pp. 535–541. DOI: 10.1145/1150402.1150464.

[BCV13] Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. "Representation learning: A review and new perspectives". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 35.8 (2013), pp. 1798–1828. DOI: 10.1109/TPAMI.2013.50.

[BGJ+17] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomás Mikolov. "Enriching word vectors with subword information". In: *Trans. Assoc. Comput. Linguistics* 5 (2017), pp. 135–146. URL: https://transacl.org/ojs/index.php/tacl/article/view/999.

[BGL+16] Jőran Beel, Bela Gipp, Stefan Langer, and Corinna Breitinger. "Research-paper recommender systems: a literature survey". In: *Int. J. Digit. Libr.* 17.4 (2016), pp. 305–338. DOI: 10.1007/s00799-015-0156-0.

[BGM+21] Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. "On the dangers of stochastic parrots: can language models be too big?" In: *FAccT '21: 2021 ACM Conference on Fairness, Accountability, and Transparency, Virtual Event / Toronto, Canada, March 3-10, 2021*. ACM, 2021, pp. 610–623. DOI: 10.1145/3442188.3445922.

[BHA+21] Rishi Bommasani et al. *On the opportunities and risks of foundation models*. 2021. arXiv: 2108.07258 [cs.LG].

[BHB+18] Peter W. Battaglia et al. *Relational inductive biases, deep learning, and graph networks*. 2018. arXiv: 1806.01261 [cs.LG].

[BJN+02] Albert-Laszlo Barabâsi, Hawoong Jeong, Zoltan Néda, Erzsebet Ravasz, Andras Schubert, and Tamas Vicsek. "Evolution of the social network of scientific collaborations". In: *Physica A: Statistical mechanics and its applications* 311.3-4 (2002), pp. 590–614.

[BKB+20] Djalila Boughareb, Abdennour Khobizi, Rima Boughareb, Nadir Farah, and Hamid Seridi. "A graph-based tag recommendation for just abstracted scientific articles tagging". In: *Int. J. Cooperative Inf. Syst.* 29.3 (2020), 2050004:1–2050004:30. DOI: 10.1142/S0218843020500045.

[BKH16] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. *Layer normalization*. 2016. arXiv: 1607.06450 [stat.ML].

[BKP+20] Aleksandar Bojchevski, Johannes Klicpera, Bryan Perozzi, Amol Kapoor, Martin Blais, Benedek Rózemberczki, Michal Lukasik, and Stephan Günnemann. "Scaling graph neural networks with approximate pagerank". In: *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*. ACM, 2020, pp. 2464–2473. DOI: 10.1145/3394486.3403296.

[BKR21] Markus Bayer, Marc-André Kaufhold, and Christian Reuter. *A survey on data augmentation for text classification*. 2021. arXiv: 2107.03158 [cs.CL].

[BMR+20] Tom B. Brown et al. "Language models are few-shot learners". In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. 2020.

[BNJ03] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. "Latent dirichlet allocation". In: *J. Mach. Learn. Res.* 3 (2003), pp. 993–1022. URL: http://jmlr.org/papers/v3/blei03a.html.

[Bor17] Gábor Borbély. "Language modeling with matrix embeddings". In: *K + K = 120* (2017). Papers dedicated to László Kálmán and András Kornai on the occasion of their 60th birthdays.

[Bot19] Léon Bottou. *Learning representations using causal invariance*. Invited Talk at the International Conference for Learning Representations (ICLR 2019). 2019. URL: https://youtube.videoken.com/embed/8UxS4ls6g1g?tocitem=2.

[BP98] Sergey Brin and Lawrence Page. "The anatomy of a large-scale hypertextual web search engine". In: *Comput. Networks* 30.1-7 (1998), pp. 107–117. DOI: 10.1016/S0169-7552(98)00110-X. URL: https://doi.org/10.1016/S0169-7552(98)00110-X.

[CDA+17] Daniel M. Cer, Mona T. Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. "Semeval-2017 task 1: semantic textual similarity multilingual and crosslingual focused evaluation". In: *Proceedings of the 11th International Workshop on Semantic Evaluation, SemEval@ACL 2017, Vancouver, Canada, August 3-4, 2017*. Association for Computational Linguistics, 2017, pp. 1–14. DOI: 10.18653/v1/S17-2001.

[Che21] Peng Chen. "Permuteformer: efficient relative position encoding for long sequences". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*. Association for Computational Linguistics, 2021, pp. 10606–10618. URL: https://aclanthology.org/2021.emnlp-main.828.

[CJ20] Davide Chicco and Giuseppe Jurman. "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation". In: *BMC genomics* 21.1 (2020), pp. 1–13.

[CK18] Alexis Conneau and Douwe Kiela. "SentEval: An evaluation toolkit for universal sentence representations". In: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018*. European Language Resources Association (ELRA), 2018. URL: http://www.lrec-conf.org/proceedings/lrec2018/summaries/757.html.

[CKL+18] Alexis Conneau, Germán Kruszewski, Guillaume Lample, Loic Barrault, and Marco Baroni. "What you can cram into a single vector: probing sentence embeddings for linguistic properties". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018,*

Volume 1: Long Papers. Association for Computational Linguistics, 2018, pp. 2126–2136. DOI: 10.18653/v1/P18-1198.

[CKS+17] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. "Supervised learning of universal sentence representations from natural language inference data". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*. Association for Computational Linguistics, 2017, pp. 670–680. DOI: 10.18653/v1/d17-1070.

[CL18] Zhiyuan Chen and Bing Liu. *Lifelong machine learning, second edition*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2018.

[CLS+19] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. "Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*. ACM, 2019, pp. 257–266. DOI: 10.1145/3292500.3330925.

[ÇM17] Erion Çano and Maurizio Morisio. "Hybrid recommender systems: A systematic literature review". In: *Intell. Data Anal.* 21.6 (2017), pp. 1487–1524. DOI: 10.3233/IDA-163209.

[CMB+14] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. "On the properties of neural machine translation: encoder-decoder approaches". In: *Proceedings of SSST@EMNLP 2014, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, Doha, Qatar, 25 October 2014*. Association for Computational Linguistics, 2014, pp. 103–111. DOI: 10.3115/v1/W14-4012. URL: https://aclanthology.org/W14-4012/.

[CR18] Yifan Chen and Maarten de Rijke. "A collective variational autoencoder for top-n recommendation with side information". In: *Proceedings of the 3rd Workshop on Deep Learning for Recommender Systems, DLRS@RecSys 2018, Vancouver, BC, Canada, October 6, 2018*. ACM, 2018, pp. 3–9. DOI: 10.1145/3270323.3270326.

[Cra09] Nick Craswell. "Mean reciprocal rank". In: *Encyclopedia of Database Systems*. Springer, 2009, pp. 1703–1703. ISBN: 978-0-387-39940-9.

[CSM+13] Cornelia Caragea, Adrian Silvescu, Prasenjit Mitra, and C. Lee Giles. "Can't see the forest for the trees?: a citation recommendation system". In: *13th ACM/IEEE-CS Joint Conference on Digital Libraries, JCDL '13, Indianapolis, IN, USA, July 22 - 26, 2013*. ACM, 2013, pp. 111–114. DOI: 10.1145/2467696.2467743.

[CT+91] Thomas M Cover, Joy A Thomas, et al. "Entropy, relative entropy and mutual information". In: *Elements of information theory* 2.1 (1991), pp. 12–13.

[CT06] Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. Wiley, 2006. ISBN: 0471241954.

[CW08] Ronan Collobert and Jason Weston. "A unified architecture for natural language processing: deep neural networks with multitask learning". In: *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*. Vol. 307. ACM International Conference Proceeding Series. ACM, 2008, pp. 160–167. DOI: 10.1145/1390156.1390177.

[CWB18] Woojin Chung, Sheng-Fu Wang, and Samuel R. Bowman. "The lifted matrix-space model for semantic composition". In: *Proceedings of the 22nd Conference on Computational Natural Language Learning, CoNLL 2018, Brussels, Belgium, October 31 - November 1, 2018*. Association for Computational Linguistics, 2018, pp. 508–518. DOI: 10.18653/v1/k18-1049.

[CWG+22] Jie Cai, Xin Wang, Chaoyu Guan, Yateng Tang, Jin Xu, Bin Zhong, and Wenwu Zhu. "Multimodal continual graph learning with neural architecture search". In: *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*. ACM, 2022, pp. 1292–1300. DOI: 10.1145/3485447.3512176. URL: https://doi.org/10.1145/3485447.3512176.

[Cyb89] George Cybenko. "Approximation by superpositions of a sigmoidal function". In: *Math. Control. Signals Syst.* 2.4 (1989), pp. 303–314. DOI: 10.1007/BF02551274.

[CYL+20] Haihua Chen, Yunhan Yang, Wei Lu, and Jiangping Chen. "Exploring multiple diversification strategies for academic citation contexts recommendation". In: *Electron. Libr.* 38.4 (2020), pp. 821–842. DOI: 10.1108/EL-02-2020-0046.

[CYL17] Sanxing Cao, Nan Yang, and Zhengzheng Liu. "Online news recommender based on stacked auto-encoder". In: *16th IEEE/ACIS International Conference on Computer and Information Science, ICIS 2017, Wuhan, China, May 24-26, 2017*. IEEE Computer Society, 2017, pp. 721–726. DOI: 10.1109/ICIS.2017.7960088.

[CZS18] Jianfei Chen, Jun Zhu, and Le Song. "Stochastic training of graph convolutional networks with variance reduction". In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 941–949. URL: http://proceedings.mlr.press/v80/chen18p.html.

[DAB+21] Mostafa Dehghani, Anurag Arnab, Lucas Beyer, Ashish Vaswani, and Yi Tay. *The efficiency misnomer*. 2021. arXiv: 2110.12894 [cs.LG].

[DB21] Vijay Prakash Dwivedi and Xavier Bresson. *A generalization of transformer networks to graphs*. 2021. arXiv: 2012.09699 [cs.LG].

[DBK+21] Alexey Dosovitskiy et al. "An image is worth 16x16 words: transformers for image recognition at scale". In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL: https://openreview.net/forum?id=YicbFdNTTy.

[DCJ19] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. "Are we really making much progress? A worrying analysis of recent neural recommendation approaches". In: *Proceedings of the 13th ACM Conference on Recommender Systems, RecSys 2019, Copenhagen, Denmark, September 16-20, 2019*. ACM, 2019, pp. 101–109. DOI: 10.1145/3298689.3347058.

[DCL+19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: pre-training of deep bidirectional transformers for language understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, 2019, pp. 4171–4186. DOI: 10.18653/v1/n19-1423.

[DGB18] Akshay Raj Dhamija, Manuel Günther, and Terrance E. Boult. "Reducing network agnostophobia". In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Process-*

ing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada. 2018, pp. 9175–9186. URL: https://proceedings.neurips.cc/paper/2018/hash/48db71587df6c7c442e5b76cc723169a-Abstract.html.

[DJL+20] Vijay Prakash Dwivedi, Chaitanya K. Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. *Benchmarking graph neural networks*. 2020. arXiv: 2003.00982 [cs.LG].

[DQB04] Bill Dolan, Chris Quirk, and Chris Brockett. "Unsupervised construction of large paraphrase corpora: exploiting massively parallel news sources". In: *COLING 2004, 20th International Conference on Computational Linguistics, Proceedings of the Conference, 23-27 August 2004, Geneva, Switzerland*. 2004. URL: https://aclanthology.org/C04-1051/.

[DS12] Klaas Dellschaft and Steffen Staab. "Measuring the influence of tag recommenders on the indexing quality in tagging systems". In: *23rd ACM Conference on Hypertext and Social Media, HT '12, Milwaukee, WI, USA, June 25-28, 2012*. ACM, 2012, pp. 73–82. DOI: 10.1145/2309996.2310009.

[DWL+20] Kaize Ding, Jianling Wang, Jundong Li, Dingcheng Li, and Huan Liu. "Be more with less: Hypergraph attention networks for inductive text classification". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*. Association for Computational Linguistics, 2020, pp. 4927–4936. DOI: 10.18653/v1/2020.emnlp-main.399.

[DZS+16] Stephan Doerfel, Daniel Zoller, Philipp Singer, Thomas Niebler, Andreas Hotho, and Markus Strohmaier. "What users actually do in a social tagging system: a study of user behavior in bibsonomy". In: *ACM Trans. Web* 10.2 (May 2016). ISSN: 1559-1131. DOI: 10.1145/2896821.

[EF17] Travis Ebesu and Yi Fang. "Neural citation network for context-aware citation recommendation". In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*. ACM, 2017, pp. 1093–1096. DOI: 10.1145/3077136.3080730.

[Elm91] Jeffrey L. Elman. "Distributed representations, simple recurrent networks, and grammatical structure". In: *Mach. Learn.* 7 (1991), pp. 195–225. DOI: 10.1007/BF00114844.

[FCA21] Quentin Fournier, Gaétan Marceau Caron, and Daniel Aloise. *A practical survey on faster and lighter transformers*. 2021. arXiv: 2103.14636 [cs.LG].

[FJ20] Michael Färber and Adam Jatowt. "Citation recommendation: approaches and datasets". In: *Int. J. Digit. Libr.* 21.4 (2020), pp. 375–405. DOI: 10.1007/s00799-020-00288-2.

[FJN+13] Alexander Felfernig, Michael Jeran, Gerald Ninaus, Florian Reinfrank, and Stefan Reiterer. "Toward the next generation of recommender systems: applications and research challenges". In: *Multimedia Services in Intelligent Environments: Advances in Recommender Systems*. Springer, 2013, pp. 81–98. ISBN: 978-3-319-00372-6. DOI: 10.1007/978-3-319-00372-6_5.

[FL19] Matthias Fey and Jan E. Lenssen. "Fast graph representation learning with PyTorch Geometric". In: *ICLR Workshop on Representation Learning on Graphs and Manifolds*. 2019.

[FLW+21] Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Jure Leskovec. "GNNAutoScale: scalable and expressive graph neural networks via historical embeddings". In: *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 3294–3304. URL: http://proceedings.mlr.press/v139/fey21a.html.

[FRE+20] Fabrizio Frasca, Emanuele Rossi, Davide Eynard, Ben Chamberlain, Michael Bronstein, and Federico Monti. "SIGN: Scalable inception graph neural networks". In: (2020). arXiv: 2004.11198 [cs.LG].

[FWL16] Geli Fei, Shuai Wang, and Bing Liu. "Learning cumulatively to become more knowledgeable". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. ACM, 2016, pp. 1565–1574. DOI: 10.1145/2939672.2939835.

[FXM+22] Falih Gozi Febrinanto, Feng Xia, Kristen Moore, Chandra Thapa, and Charu Aggarwal. *Graph Lifelong Learning: A Survey*. 2022. arXiv: 2202.10688 [cs].

[FZM+20] Xinyu Fu, Jiani Zhang, Ziqiao Meng, and Irwin King. "MAGNN: metapath aggregated graph neural network for heterogeneous graph embedding". In: *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*. ACM, 2020, pp. 2331–2341. DOI: 10.1145/3366423.3380297.

[FZS21] William Fedus, Barret Zoph, and Noam Shazeer. *Switch transformers: scaling to trillion parameter models with simple and efficient sparsity*. 2021. arXiv: 2101.03961 [cs.LG].

[GAG+15] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. "Deep learning with limited numerical precision". In: *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*. Vol. 37. JMLR Workshop and Conference Proceedings. JMLR.org, 2015, pp. 1737–1746. URL: http://proceedings.mlr.press/v37/gupta15.html.

[GB10] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*. Vol. 9. JMLR Proceedings. JMLR.org, 2010, pp. 249–256. URL: http://proceedings.mlr.press/v9/glorot10a.html.

[GBC16] Ian J. Goodfellow, Yoshua Bengio, and Aaron C. Courville. *Deep learning*. Adaptive computation and machine learning. MIT Press, 2016. ISBN: 978-0-262-03561-3. URL: http://www.deeplearningbook.org/.

[GCC20] Palash Goyal, Sujit Rokka Chhetri, and Arquimedes Canedo. "Dyngraph2vec: capturing network dynamics using dynamic graph representation learning". In: *Knowl.-Based Syst.* 187 (2020).

[GCM+22] Lukas Galke, Isabelle Cuber, Christoph Meyer, Henrik Ferdinand Nölscher, Angelina Sonderecker, and Ansgar Scherp. "General cross-architecture distillation of pretrained language models into matrix embeddings". In: *International Joint Conference on Neural Networks, IJCNN 2022, Padua, Italy, July 18-23, 2022*. IEEE, 2022, pp. 1–10. DOI: 10.1109/IJCNN55064.2022.9892144. URL: https://doi.org/10.1109/IJCNN55064.2022.9892144.

[GFZ+21] Lukas Galke, Benedikt Franke, Tobias Zielke, and Ansgar Scherp. "Lifelong learning of graph neural networks for open-world node classification". In: *International Joint Conference on Neural Networks, IJCNN 2021, Shenzhen, China, July 18-22, 2021*. IEEE, 2021, pp. 1–8. DOI: 10.1109/IJCNN52387.2021.9533412.

[GG21] Santiago González-Carvajal and Eduardo C. Garrido-Merchán. *Comparing bert against traditional machine learning text classification*. 2021. arXiv: 2005.13012 [cs.CL].

[GJM+20] Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard S. Zemel, Wieland Brendel, Matthias Bethge, and Felix A. Wichmann. "Shortcut learning in deep neural networks". In: *Nat Mach Intell* 2 (2020), pp. 665–683. DOI: 10.1038/s42256-020-00257-z.

[GKH+18] Palash Goyal, Nitin Kamra, Xinran He, and Yan Liu. *DynGEM: deep embedding method for dynamic graphs*. 2018. arXiv: 1805.11273 [cs.SI].

[GL16] Aditya Grover and Jure Leskovec. "Node2vec: scalable feature learning for networks". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. ACM, 2016, pp. 855–864. DOI: 10.1145/2939672.2939754.

[GML15] Bela Gipp, Norman Meuschke, and Mario Lipinski. "CITREC: An Evaluation Framework for Citation-Based Similarity Measures based on TREC Genomics and PubMed Central". In: *iConference*. Mar. 2015.

[GMS+17] Lukas Galke, Florian Mai, Alan Schelten, Dennis Brunsch, and Ansgar Scherp. "Using titles vs. full-text as source for automated semantic document annotation". In: *Proceedings of the Knowledge Capture Conference, K-CAP 2017, Austin, TX, USA, December 4-6, 2017*. ACM, 2017, 20:1–20:4. DOI: 10.1145/3148011.3148039.

[GMS19] Lukas Galke, Florian Mai, and Ansgar Scherp. "What if we encoded words as matrices and used matrix multiplication as composition function?" In: *49. Jahrestagung der Gesellschaft für Informatik, 50 Jahre Gesellschaft für Informatik - Informatik für Gesellschaft, INFORMATIK 2019, Kassel, Germany, September 23-26, 2019*. Vol. P-294. LNI. GI, 2019, pp. 287–288. DOI: 10.18420/inf2019_47.

[GMV+18] Lukas Galke, Florian Mai, Iacopo Vagliano, and Ansgar Scherp. "Multi-modal adversarial autoencoders for recommendations of citations and subject labels". In: *Proceedings of the 26th Conference on User Modeling, Adaptation and Personalization, UMAP 2018, Singapore, July 08-11, 2018*. ACM, 2018, pp. 197–205. DOI: 10.1145/3209219.3209236.

[GNS15] Gregor Große-Bölting, Chifumi Nishioka, and Ansgar Scherp. "A comparison of different strategies for automated semantic document annotation". In: *Proceedings of the 8th International Conference on Knowledge Capture, K-CAP 2015, Palisades, NY, USA, October 7-10, 2015*. ACM, 2015, 8:1–8:8. DOI: 10.1145/2815833.2815838.

[Gol17] Yoav Goldberg. *Neural network methods for natural language processing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2017. DOI: 10.2200/S00762ED1V01Y201703HLT037.

[Got16] Gregory Goth. "Deep or shallow, NLP is breaking out". In: *Commun. ACM* 59.3 (2016), pp. 13–16. DOI: 10.1145/2874915.

[GPH+17] Zhe Gan, Yunchen Pu, Ricardo Henao, Chunyuan Li, Xiaodong He, and Lawrence Carin. "Learning generic sentence representations using convolutional neural networks". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*. Association for Computational Linguistics, 2017, pp. 2390–2400. DOI: 10.18653/v1/d17-1254.

[GPM+14] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. "Generative adversarial nets". In: *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*. 2014, pp. 2672–2680. URL: https://proceedings.neurips.cc/paper/2014/hash/5ca3e9b122f61f8f06494c97b1afccf3-Abstract.html.

[GS11] Edward Grefenstette and Mehrnoosh Sadrzadeh. "Experimental support for a categorical compositional distributional model of meaning". In: *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL*. ACL, 2011, pp. 1394–1404. URL: https://aclanthology.org/D11-1129/.

[GS22] Lukas Galke and Ansgar Scherp. "Bag-of-words vs. graph vs. sequence in text classification: Questioning the necessity of text-graphs and the surprising strength of a wide MLP". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Dublin, Ireland: Association for Computational Lin-

guistics, May 2022, pp. 4038–4051. DOI: 10.18653/v1/2022.acl-long.279. URL: https://aclanthology.org/2022.acl-long.279.

[GSS17] Lukas Galke, Ahmed Saleh, and Ansgar Scherp. "Word embeddings for practical information retrieval". In: *47. Jahrestagung der Gesellschaft für Informatik, Digitale Kulturen, INFORMATIK 2017, Chemnitz, Germany, September 25-29, 2017*. Vol. P-275. LNI. GI, 2017, pp. 2155–2167. DOI: 10.18420/in2017_215. URL: https://doi.org/10.18420/in2017_215.

[GVF+21] Lukas Galke, Iacopo Vagliano, Benedikt Franke, Tobias Zielke, and Ansgar Scherp. *Lifelong learning in evolving graphs with limited labeled data and unseen class detection*. Under review. 2021. arXiv: 2112.10558 [cs.LG].

[GVS19] Lukas Galke, Iacopo Vagliano, and Ansgar Scherp. "Can graph neural networks go "online"? An analysis of pretraining and inference". In: *Representation Learning on Graphs and Manifolds, ICLR Workshop*. 2019. URL: https://rlgm.github.io/papers/21.pdf.

[Ham20] William L. Hamilton. *Graph representation learning*. Vol. 14. Synthesis Lectures on Artificial Intelligence and Machine Learning 3. Morgan & Claypool Publishers, 2020, pp. 1–159. URL: https://www.cs.mcgill.ca/~wlh/grl_book/.

[Har54] Zellig S Harris. "Distributional structure". In: *Word* 10.2-3 (1954), pp. 146–162.

[HCK16] Felix Hill, Kyunghyun Cho, and Anna Korhonen. "Learning distributed representations of sentences from unlabelled data". In: *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*. The Association for Computational Linguistics, 2016, pp. 1367–1377. DOI: 10.18653/v1/n16-1162.

[HFC+19] Ziniu Hu, Changjun Fan, Ting Chen, Kai-Wei Chang, and Yizhou Sun. "Unsupervised pre-training of graph convolutional networks". In: *Representation Learning on Graphs and Manifolds workshop at ICLR*. 2019. URL: https://rlgm.github.io/papers/73.pdf.

[HFZ+20] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. "Open graph benchmark: datasets for machine learning on graphs". In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural*

Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual. 2020. URL: https://proceedings.neurips.cc/paper/2020/hash/fb60d411a5c5b72b2e7d3527cfc84fd0-Abstract.html.

[HHY+19] Lu Haonan, Seth H. Huang, Tian Ye, and Guo Xiuyan. *Graph star net for generalized multi-task learning*. 2019. arXiv: 1906.12330 [cs.SI].

[HKC+12] Wenyi Huang, Saurabh Kataria, Cornelia Caragea, Prasenjit Mitra, C. Lee Giles, and Lior Rokach. "Recommending citations: translating papers into references". In: *21st ACM International Conference on Information and Knowledge Management, CIKM'12, Maui, HI, USA, October 29 - November 02, 2012*. ACM, 2012, pp. 1910–1914. DOI: 10.1145/2396761.2398542.

[HKF+13] Lushan Han, Abhay L. Kashyap, Tim Finin, James Mayfield, and Jonathan Weese. "Umbc ebiquity-core: semantic textual similarity systems". In: *Proceedings of the Second Joint Conference on Lexical and Computational Semantics, *SEM 2013, June 13-14, 2013, Atlanta, Georgia, USA*. Association for Computational Linguistics, 2013, pp. 44–52. URL: https://aclanthology.org/S13-1005/.

[HL04] Minqing Hu and Bing Liu. "Mining and summarizing customer reviews". In: *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, Washington, USA, August 22-25, 2004*. ACM, 2004, pp. 168–177. DOI: 10.1145/1014052.1014073.

[HLG+20] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay S. Pande, and Jure Leskovec. "Strategies for pre-training graph neural networks". In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL: https://openreview.net/forum?id=HJlWWJSFDH.

[HMD19] Dan Hendrycks, Mantas Mazeika, and Thomas G. Dietterich. "Deep anomaly detection with outlier exposure". In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL: https://openreview.net/forum?id=HyxCxhRcY7.

[HMZ19] Ming He, Qian Meng, and Shaozong Zhang. "Collaborative additional variational autoencoder for top-n recommender systems". In: *IEEE Access* 7 (2019), pp. 5707–5713. DOI: 10.1109/ACCESS.2018.2890293.

[HPW05] Mark Herbster, Massimiliano Pontil, and Lisa Wainer. "Online learning over graphs". In: *ICML*. Vol. 119. ACM, 2005.

[HR18] Jeremy Howard and Sebastian Ruder. "Universal language model fine-tuning for text classification". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*. Association for Computational Linguistics, 2018, pp. 328–339. DOI: 10.18653/v1/P18-1031.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural Comput.* 9.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

[HVD15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. *Distilling the knowledge in a neural network*. 2015. arXiv: 1503.02531 [stat.ML].

[HWL+15] Wenyi Huang, Zhaohui Wu, Chen Liang, Prasenjit Mitra, and C. Lee Giles. "A neural probabilistic model for context based citation recommendation". In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*. AAAI Press, 2015, pp. 2404–2410. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9737.

[HYL17] William L. Hamilton, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. 2017, pp. 1024–1034. URL: https://proceedings.neurips.cc/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html.

[HZR+15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Delving deep into rectifiers: surpassing human-level performance on imagenet classification". In: *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*. IEEE Computer Society, 2015, pp. 1026–1034. DOI: 10.1109/ICCV.2015.123.

[HZR+16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90. URL: https://doi.org/10.1109/CVPR.2016.90.

[HZR+18] Wen-bing Huang, Tong Zhang, Yu Rong, and Junzhou Huang. "Adaptive sampling towards fast graph representation learning". In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*. 2018, pp. 4563–4572. URL: https://proceedings.neurips.cc/paper/2018/hash/01eee509ee2f68dc6014898c309e86bf-Abstract.html.

[IMB+15] Mohit Iyyer, Varun Manjunatha, Jordan L. Boyd-Graber, and Hal Daumé III. "Deep unordered composition rivals syntactic methods for text classification". In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*. The Association for Computer Linguistics, 2015, pp. 1681–1691. DOI: 10.3115/v1/p15-1162.

[ISO96] ISO 999. *Information and documentation — guidelines for the content, organization and presentation of indexes*. 1996.

[JGB+17] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. "Bag of tricks for efficient text classification". In: *EACL (2)*. ACL, 2017, pp. 427–431. URL: https://www.aclweb.org/anthology/E17-2068/.

[JYS+20] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. "Tinybert: distilling BERT for natural language understanding". In: *EMNLP (Findings)*. Vol. EMNLP 2020. Findings of ACL. Association for Computational Linguistics, 2020, pp. 4163–4174. DOI: 10.18653/v1/2020.findings-emnlp.372.

[Kad19] Ammar Ismael Kadhim. "Survey on supervised machine learning techniques for automatic text classification". In: *Artif. Intell. Rev.* 52.1 (2019), pp. 273–292.

[Kat87] Slava M. Katz. "Estimation of probabilities from sparse data for the language model component of a speech recognizer". In: *IEEE Trans. Acoust. Speech Signal Process.* 35.3 (1987), pp. 400–401. DOI: 10.1109/TASSP.1987.1165125. URL: https://doi.org/10.1109/TASSP.1987.1165125.

[KB15] Diederik P. Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015. URL: http://arxiv.org/abs/1412.6980.

[KBG19] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. "Predict then propagate: graph neural networks meet personalized pagerank". In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL: https://openreview.net/forum?id=H1gL-2A9Ym.

[KBH+21] Devin Kreuzer, Dominique Beaini, William L. Hamilton, Vincent Létourneau, and Prudencio Tossou. *Rethinking graph transformers with spectral attention*. 2021. arXiv: 2106.03893 [cs.LG].

[KFN09] Ralf Krestel, Peter Fankhauser, and Wolfgang Nejdl. "Latent Dirichlet allocation for tag recommendation". In: *Proceedings of the 2009 ACM Conference on Recommender Systems, RecSys 2009, New York, NY, USA, October 23-25, 2009*. ACM, 2009, pp. 61–68. DOI: 10.1145/1639714.1639726.

[KGB14] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. "A convolutional neural network for modelling sentences". In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*. The Association for Computer Linguistics, 2014, pp. 655–665. DOI: 10.3115/v1/p14-1062.

[KMH+19] Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura E. Barnes, and Donald E. Brown. "Text classification algorithms: A survey". In: *Inf.* 10.4 (2019), p. 150.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*. 2012, pp. 1106–1114. URL: https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html.

[KSH17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks". In: *Commun. ACM* 60.6 (2017), pp. 84–90. DOI: 10.1145/3065386.

[KUM+17] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. "Self-normalizing neural networks". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach,*

CA, USA. 2017, pp. 971–980. URL: https://proceedings.neurips.cc/paper/2017/hash/5d44ee6f2c3f71b73125876103c8f6c4-Abstract.html.

[KW14] Diederik P. Kingma and Max Welling. "Auto-encoding variational bayes". In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. 2014. URL: http://arxiv.org/abs/1312.6114.

[KW16] Thomas N. Kipf and Max Welling. "Variational graph auto-encoders". In: *Bayesian Deep Learning NIPS 2016 Workshop*. 2016.

[KW17] Thomas N. Kipf and Max Welling. "Semi-supervised classification with graph convolutional networks". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL: https://openreview.net/forum?id=SJU4ayYgl.

[KZL19] Srijan Kumar, Xikun Zhang, and Jure Leskovec. "Predicting dynamic embedding trajectory in temporal interaction networks". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*. ACM, 2019, pp. 1269–1278. DOI: 10.1145/3292500.3330895.

[KZS+15] Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. "Skip-thought vectors". In: *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*. 2015, pp. 3294–3302. URL: https://proceedings.neurips.cc/paper/2015/hash/f442d33fa06832082290ad8544a8da27-Abstract.html.

[LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey E. Hinton. "Deep learning". In: *Nat.* 521.7553 (2015), pp. 436–444. DOI: 10.1038/nature14539.

[LDL+21] Qingsong Lv, Ming Ding, Qiang Liu, Yuxiang Chen, Wenzheng Feng, Siming He, Chang Zhou, Jianguo Jiang, Yuxiao Dong, and Jie Tang. "Are we really making much progress?: revisiting, benchmarking and refining heterogeneous graph neural networks". In: *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*. ACM, 2021, pp. 1150–1160. DOI: 10.1145/3447548.3467350.

[LDS+21] Hanxiao Liu, Zihang Dai, David R. So, and Quoc V. Le. *Pay attention to MLPs*. 2021. arXiv: 2105.08050 [cs.LG].

[LDS89] Yann LeCun, John S. Denker, and Sara A. Solla. "Optimal brain damage". In: *Advances in Neural Information Processing Systems 2, [NIPS Conference, Denver, Colorado, USA, November 27-30, 1989]*. Morgan Kaufmann, 1989, pp. 598–605. URL: http://papers.nips.cc/paper/250-optimal-brain-damage.

[LFY+20] Kai Lei, Qiuai Fu, Min Yang, and Yuzhi Liang. "Tag recommendation by text classification with attention-based capsule network". In: *Neurocomputing* 391 (2020), pp. 65–73. DOI: 10.1016/j.neucom.2020.01.091.

[LG14] Omer Levy and Yoav Goldberg. "Neural word embedding as implicit matrix factorization". In: *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*. 2014, pp. 2177–2185. URL: https://proceedings.neurips.cc/paper/2014/hash/feab05aa91085b7a8012516bc3533958-Abstract.html.

[LGG+21] Yonghao Liu, Renchu Guan, Fausto Giunchiglia, Yanchun Liang, and Xiaoyue Feng. "Deep attention diffusion graph neural networks for text classification". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*. Association for Computational Linguistics, 2021, pp. 8142–8152. URL: https://aclanthology.org/2021.emnlp-main.642.

[LGS11] Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. "Content-based recommender systems: state of the art and trends". In: *Recommender Systems Handbook*. Springer, 2011, pp. 73–105. ISBN: 978-0-387-85820-3. DOI: 10.1007/978-0-387-85820-3_3.

[LH19] Ilya Loshchilov and Frank Hutter. "Decoupled weight decay regularization". In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL: https://openreview.net/forum?id=Bkg6RiCqY7.

[Lin91] Jianhua Lin. "Divergence measures based on the Shannon entropy". In: *IEEE Trans. Information Theory* 37.1 (1991), pp. 145–151. DOI: 10.1109/18.61115.

[Liu17] Bing Liu. "Lifelong machine learning: a paradigm for continuous learning". In: *Frontiers of Computer Science* 11.3 (2017).

[LKF15] Sheng Li, Jaya Kawale, and Yun Fu. "Deep collaborative filtering via marginalized denoising auto-encoder". In: *Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia, October 19 - 23, 2015*. ACM, 2015, pp. 811–820. DOI: 10.1145/2806416.2806527.

[LKH+18] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. "Variational autoencoders for collaborative filtering". In: *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*. ACM, 2018, pp. 689–698. DOI: 10.1145/3178876.3186150.

[LL18] Lajanugen Logeswaran and Honglak Lee. "An efficient framework for learning sentence representations". In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL: https://openreview.net/forum?id=rJvJXZb0W.

[LL20] Shengfei Lyu and Jiaqi Liu. *Combine convolution with recurrent networks for text classification*. 2020. arXiv: 2006.15795 [cs.CL].

[LLL+18] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. "A simple unified framework for detecting out-of-distribution samples and adversarial attacks". In: *NeurIPS*. 2018. URL: https://proceedings.neurips.cc/paper/2018/hash/abdeb6f575ac5c6676b747bca8d09cc2-Abstract.html.

[LLR18] Shiyu Liang, Yixuan Li, and Srikant R. "Enhancing the reliability of out-of-distribution image detection in neural networks". In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL: https://openreview.net/forum?id=H1VGkIxRZ.

[LNR+20] John Boaz Lee, Giang Nguyen, Ryan A Rossi, Nesreen K Ahmed, Eunyee Koh, and Sungchul Kim. "Dynamic node embeddings from edge streams". In: *IEEE Transactions on Emerging Topics in Computational Intelligence* (2020).

[LOG+19] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. *Roberta: a robustly optimized bert pretraining approach*. 2019. arXiv: 1907.11692 [cs.CL].

[LPL+21] Qian Li, Hao Peng, Jianxin Li, Congying Xia, Renyu Yang, Lichao Sun, Philip S. Yu, and Lifang He. *A survey on text classification: from shallow to deep learning*. 2021. arXiv: 2008.00364 [cs.CL].

[LR02] Xin Li and Dan Roth. "Learning question classifiers". In: *19th International Conference on Computational Linguistics, COLING 2002, Howard International House and Academia Sinica, Taipei, Taiwan, August 24 - September 1, 2002*. 2002. URL: https://aclanthology.org/C02-1150/.

[LR17] David Lopez-Paz and Marc'Aurelio Ranzato. "Gradient episodic memory for continual learning". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. 2017, pp. 6467–6476. URL: https://proceedings.neurips.cc/paper/2017/hash/f87522788a2be2d171666752f97ddebb-Abstract.html.

[LS17] Xiaopeng Li and James She. "Collaborative variational autoencoder for recommender systems". In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*. ACM, 2017, pp. 305–314. DOI: 10.1145/3097983.3098077.

[LWK+18] Yu Liu, Shuai Wang, M. Shahrukh Khan, and Jieyu He. "A novel deep hybrid recommender system based on auto-encoder with neural collaborative filtering". In: *Big Data Min. Anal.* 1.3 (2018), pp. 211–221.

[LXL+15] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. "Recurrent convolutional neural networks for text classification". In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*. AAAI Press, 2015, pp. 2267–2273. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9745.

[LYR+04] David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. "RCV1: a new benchmark collection for text categorization research". In: *Machine Learning Research* 5 (2004).

[LYZ+20] Xien Liu, Xinxin You, Xiao Zhang, Ji Wu, and Ping Lv. "Tensor graph convolutional networks for text classification". In: *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, 2020, pp. 8409–8416. URL: https://aaai.org/ojs/index.php/AAAI/article/view/6359.

[MAC+02] Sean M. McNee, István Albert, Dan Cosley, Prateep Gopalkrishnan, Shyong K. Lam, Al Mamunur Rashid, Joseph A. Konstan, and John Riedl. "On the recommending of citations for research papers". In: *CSCW 2002, Proceeding on the ACM 2002 Conference on Computer Supported Cooperative Work, New Orleans, Louisiana, USA, November 16-20, 2002*. ACM, 2002, pp. 116–125. DOI: 10.1145/587078.587096.

[MBM+17] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodolà, Jan Svoboda, and Michael M. Bronstein. "Geometric deep learning on graphs and manifolds using mixture model cnns". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 2017, pp. 5425–5434. DOI: 10.1109/CVPR.2017.576.

[MBX+17] Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. "Learned in translation: contextualized word vectors". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. 2017, pp. 6294–6305. URL: https://proceedings.neurips.cc/paper/2017/hash/20c86a628232a67e7bd46f76fba7ce12-Abstract.html.

[Mel21] Luke Melas-Kyriazi. *Do you even need attention? a stack of feed-forward layers does surprisingly well on imagenet*. 2021. arXiv: 2105.02723 [cs.CV].

[MGK+11] Mohammad M. Masud, Jing Gao, Latifur Khan, Jiawei Han, and Bhavani M. Thuraisingham. "Classification and novel class detection in concept-drifting data streams under time constraints". In: *IEEE Trans. Knowl. Data Eng.* (2011).

[MGS18] Florian Mai, Lukas Galke, and Ansgar Scherp. "Using deep learning for title-based semantic subject indexing to reach competitive performance to full-text". In: *Proceedings of the 18th ACM/IEEE on Joint*

Conference on Digital Libraries, JCDL 2018, Fort Worth, TX, USA, June 03-07, 2018. ACM, 2018, pp. 169–178. DOI: 10.1145/3197026.3197039.

[MGS19] Florian Mai, Lukas Galke, and Ansgar Scherp. "CBOW is not all you need: Combining CBOW with the compositional matrix space model". In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL: https://openreview.net/forum?id=H1MgjoR9tQ.

[MJ17] Angshul Majumdar and Anant Jain. "Cold-start, warm-start and everything in between: an autoencoder based approach to recommendation". In: *2017 International Joint Conference on Neural Networks, IJCNN 2017, Anchorage, AK, USA, May 14-19, 2017*. IEEE, 2017, pp. 3656–3663. DOI: 10.1109/IJCNN.2017.7966316.

[ML21] David Macêdo and Teresa Ludermir. *A seamless and high-performance out-of-distribution detection approach simply replacing the softmax loss*. 2021. arXiv: 2105.14399 [cs.LG].

[MML+16] Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. "Natural language inference by tree-based convolution and heuristic matching". In: *ACL (2)*. The Association for Computer Linguistics, 2016. URL: https://aclanthology.org/P16-2022/.

[MQD+09] Elena Montañés, José Ramón Quevedo, Irene Díaz, and José Ranilla. "Collaborative tag recommendation system based on logistic regression". In: *Proceedings of ECML PKDD (The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases) Discovery Challenge 2009, Bled, Slovenia, September 7, 2009*. Vol. 497. CEUR Workshop Proceedings. CEUR-WS.org, 2009. URL: http://ceur-ws.org/Vol-497/paper%5C_20.pdf.

[MRM20] Franco Manessi, Alessandro Rozza, and Mario Manzo. "Dynamic graph convolutional networks". In: *Pattern Recognition* 97 (2020).

[MRZ+21] David Macêdo, Tsang Ing Ren, Cleber Zanchettin, Adriano L. I. Oliveira, and Teresa Bernarda Ludermir. "Entropic out-of-distribution detection". In: *International Joint Conference on Neural Networks, IJCNN 2021, Shenzhen, China, July 18-22, 2021*. IEEE, 2021, pp. 1–8. DOI: 10.1109/IJCNN52387.2021.9533899.

[MSC+13] Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. "Distributed representations of words and phrases and their compositionality". In: *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*. 2013, pp. 3111–3119. URL: https://proceedings.neurips.cc/paper/2013/hash/9aa42b31882ec039965f3c4923ce901b-Abstract.html.

[MSJ+16] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. *Adversarial autoencoders*. 2016. arXiv: 1511.05644 [cs.LG].

[MT04] Rada Mihalcea and Paul Tarau. "Textrank: bringing order into text". In: *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing , EMNLP 2004, A meeting of SIGDAT, a Special Interest Group of the ACL, held in conjunction with ACL 2004, 25-26 July 2004, Barcelona, Spain*. ACL, 2004, pp. 404–411. URL: https://aclanthology.org/W04-3252/.

[MWW+20] Yihuan Mao, Yujing Wang, Chufan Wu, Chen Zhang, Yang Wang, Quanlu Zhang, Yaming Yang, Yunhai Tong, and Jing Bai. "Ladabert: lightweight adaptation of BERT through hybrid model compression". In: *COLING*. International Committee on Computational Linguistics, 2020, pp. 3225–3234. DOI: 10.18653/v1/2020.coling-main.287.

[MZL20] Shutian Ma, Chengzhi Zhang, and Xiaozhong Liu. "A review of citation recommendation: from textual content to enriched context". In: *Scientometrics* 122.3 (2020), pp. 1445–1472. DOI: 10.1007/s11192-019-03336-0.

[NBG19] Allen Nie, Erin Bennett, and Noah D. Goodman. "Dissent: learning sentence representations from explicit discourse relations". In: *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*. Association for Computational Linguistics, 2019, pp. 4497–4510. DOI: 10.18653/v1/p19-1442.

[New01] M. E. J. Newman. "Clustering and preferential attachment in growing networks". In: *Phys. Rev. E* 64 (2 July 2001), p. 025102. DOI: 10.1103/PhysRevE.64.025102.

[New05] Mark EJ Newman. "Power laws, Pareto distributions and Zipf's law". In: *Contemporary physics* 46.5 (2005).

[NH10] Vinod Nair and Geoffrey E. Hinton. "Rectified linear units improve restricted boltzmann machines". In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*. Omnipress, 2010, pp. 807–814. URL: https://icml.cc/Conferences/2010/papers/432.pdf.

[NIK10] Tetsuji Nakagawa, Kentaro Inui, and Sadao Kurohashi. "Dependency tree-based sentiment classification using crfs with hidden variables". In: *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 2-4, 2010, Los Angeles, California, USA*. The Association for Computational Linguistics, 2010, pp. 786–794. URL: https://aclanthology.org/N10-1120/.

[NKB+20] Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. "Deep double descent: where bigger models and more data hurt". In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL: https://openreview.net/forum?id=B1g5sA4twr.

[NKM+19] Jinseok Nam, Young-Bum Kim, Eneldo Loza Mencía, Sunghyun Park, Ruhi Sarikaya, and Johannes Fürnkranz. "Learning context-dependent label permutations for multi-label classification". In: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 4733–4742. URL: http://proceedings.mlr.press/v97/nam19a.html.

[NLB+18] Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. *Towards understanding the role of over-parametrization in generalization of neural networks*. 2018. arXiv: 1805.12076 [cs.LG].

[NLR+18] Giang Hoang Nguyen, John Boaz Lee, Ryan A. Rossi, Nesreen K. Ahmed, Eunyee Koh, and Sungchul Kim. "Continuous-time dynamic network embeddings". In: *WWW*. 2018.

[NMK+17] Jinseok Nam, Eneldo Loza Mencía, Hyunwoo J. Kim, and Johannes Fürnkranz. "Maximizing subset accuracy with recurrent neural networks in multi-label classification". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Pro-*

cessing Systems 2017, December 4-9, 2017, Long Beach, CA, USA. 2017, pp. 5413–5423. URL: https://proceedings.neurips.cc/paper/2017/hash/2eb5657d37f474e4c4cf01e4882b8962-Abstract.html.

[OBY98] Yukio Ohsawa, Nels E. Benson, and Masahiko Yachida. "Keygraph: Automatic indexing by co-occurrence graph based on building construction metaphor". In: *Proceedings of the IEEE Forum on Research and Technology Advances in Digital Libraries, IEEE ADL '98, Santa Barbara, California, USA, April 22-24, 1998*. IEEE Computer Society, 1998, pp. 12–18. DOI: 10.1109/ADL.1998.670375. URL: https://doi.org/10.1109/ADL.1998.670375.

[PAS14] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. "Deepwalk: online learning of social representations". In: *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*. ACM, 2014, pp. 701–710. DOI: 10.1145/2623330.2623732.

[PDC+20] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao B. Schardl, and Charles E. Leiserson. "EvolveGCN: Evolving graph convolutional networks for dynamic graphs". In: *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, 2020, pp. 5363–5370. URL: https://aaai.org/ojs/index.php/AAAI/article/view/5984.

[PGJ18] Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. "Unsupervised learning of sentence embeddings using compositional n-gram features". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*. Association for Computational Linguistics, 2018, pp. 528–540. DOI: 10.18653/v1/n18-1049.

[PHY20] Yiteng Pan, Fazhi He, and Haiping Yu. "Learning social representations with deep autoencoder for recommender system". In: *World Wide Web* 23.4 (2020), pp. 2259–2279. DOI: 10.1007/s11280-020-00793-z.

[PL04] Bo Pang and Lillian Lee. "A sentimental education: sentiment analysis using subjectivity summarization based on minimum cuts". In: *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, 21-26 July, 2004, Barcelona, Spain*. ACL, 2004, pp. 271–278. DOI: 10.3115/1218955.1218990. URL: https://aclanthology.org/P04-1035/.

[PL05] Bo Pang and Lillian Lee. "Seeing stars: exploiting class relationships for sentiment categorization with respect to rating scales". In: *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*. Ann Arbor, Michigan: ACL, June 2005, pp. 115–124. DOI: 10.3115/1219840.1219855. URL: https://aclanthology.org/P05-1015.

[PLP21] Gary Phua, Shaowei Lin, and Dario Poletti. *Word2rate: training and evaluating multiple word embeddings as statistical transitions*. 2021. arXiv: 2104.08173 [cs.CL].

[PNI+18] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. "Deep contextualized word representations". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*. Association for Computational Linguistics, 2018, pp. 2227–2237. DOI: 10.18653/v1/n18-1202.

[PSC+21] Guansong Pang, Chunhua Shen, Longbing Cao, and Anton van den Hengel. "Deep learning for anomaly detection: A review". In: *ACM Comput. Surv.* 54.2 (2021), 38:1–38:38. DOI: 10.1145/3439950.

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. "Glove: global vectors for word representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. ACL, 2014, pp. 1532–1543. DOI: 10.3115/v1/d14-1162.

[PSP18] Christian S. Perone, Roberto Silveira, and Thomas S. Paula. *Evaluation of sentence embeddings in downstream and linguistic probing tasks*. 2018. arXiv: 1806.06259 [cs.CL].

[PWS+13] Lisa Posch, Claudia Wagner, Philipp Singer, and Markus Strohmaier. "Meaning as collective use: predicting semantic hashtag categories on twitter". In: *22nd International World Wide Web Conference, WWW '13,*

Rio de Janeiro, Brazil, May 13-17, 2013, Companion Volume. International World Wide Web Conferences Steering Committee / ACM, 2013, pp. 621–628. DOI: 10.1145/2487788.2488008.

[RCF+20] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. *Temporal graph networks for deep learning on dynamic graphs*. 2020. arXiv: 2006.10637 [cs.LG].

[RE13] Paul Ruvolo and Eric Eaton. "ELLA: an efficient lifelong learning algorithm". In: *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*. Vol. 28. JMLR Workshop and Conference Proceedings. JMLR.org, 2013, pp. 507–515. URL: http://proceedings.mlr.press/v28/ruvolo13.html.

[REC+10] Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. "Automatic keyword extraction from individual documents". In: *Text Mining. Applications and Theory*. John Wiley and Sons, Ltd, 2010, pp. 1–20. ISBN: 9780470689646. DOI: 10.1002/9780470689646.ch1.

[RFP17] Aravind Sesagiri Raamkumar, Schubert Foo, and Natalie Pang. "Using author-specified keywords in building an initial reading list of research papers in scientific paper retrieval and recommender systems". In: *Inf. Process. Manag.* 53.3 (2017), pp. 577–594. DOI: 10.1016/j.ipm.2016.12.006.

[RG10] Sebastian Rudolph and Eugenie Giesbrecht. "Compositional matrix-space models of language". In: *ACL*. The Association for Computer Linguistics, 2010, pp. 907–916. URL: https://aclanthology.org/P10-1093/.

[RG19] Nils Reimers and Iryna Gurevych. "Sentence-bert: sentence embeddings using siamese bert-networks". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*. Association for Computational Linguistics, 2019, pp. 3980–3990. DOI: 10.18653/v1/D19-1410.

[RHW86] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *nature* 323.6088 (1986), pp. 533–536.

[RKS+17] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. "Icarl: incremental classifier and representation learning". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 2017, pp. 5533–5542. DOI: 10.1109/CVPR.2017.587. URL: https://doi.org/10.1109/CVPR.2017.587.

[Rob95] Anthony V. Robins. "Catastrophic forgetting, rehearsal and pseudorehearsal". In: *Connect. Sci.* 7.2 (1995).

[RPH+11] Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. "Classifier chains for multi-label classification". In: *Mach. Learn.* 85.3 (2011), pp. 333–359.

[RSI+21] Rahul Ragesh, Sundararajan Sellamanickam, Arun Iyer, Ramakrishna Bairi, and Vijay Lingam. "Hetegcn: heterogeneous graph convolutional networks for text classification". In: *WSDM '21, The Fourteenth ACM International Conference on Web Search and Data Mining, Virtual Event, Israel, March 8-12, 2021*. ACM, 2021, pp. 860–868. DOI: 10.1145/3437963.3441746.

[RSR+20] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. "Exploring the limits of transfer learning with a unified text-to-text transformer". In: *J. Mach. Learn. Res.* 21 (2020), 140:1–140:67. URL: http://jmlr.org/papers/v21/20-074.html.

[Rud19] Sebastian Ruder. "Neural transfer learning for natural language processing". PhD thesis. National University of Ireland, Galway, 2019.

[RZL+16] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. "Squad: 100, 000+ questions for machine comprehension of text". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*. The Association for Computational Linguistics, 2016, pp. 2383–2392. DOI: 10.18653/v1/d16-1264.

[SAH20] Nazmus Sakib, Rodina Binti Ahmad, and Khalid Haruna. "A collaborative approach toward scientific paper recommendation using citation context". In: *IEEE Access* 8 (2020), pp. 51246–51255. DOI: 10.1109/ACCESS.2020.2980589.

[SB88] Gerard Salton and Christopher Buckley. "Term-weighting approaches in automatic text retrieval". In: *Information processing & management* 24.5 (1988).

[SCG+19] Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. "Patient knowledge distillation for BERT model compression". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*. Association for Computational Linguistics, 2019, pp. 4322–4331. DOI: 10.18653/v1/D19-1441.

[SCP+19] Nícollas Silva, Diego Carvalho, Adriano C. M. Pereira, Fernando Mourão, and Leonardo C. da Rocha. "The pure cold-start problem: A deep study about how to conquer first-time users in recommendations domains". In: *Inf. Syst.* 80 (2019), pp. 1–12.

[SDC+20] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. *Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter*. 2020. arXiv: 1910.01108 [cs.CL].

[SDV+18] Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. "Structured sequence modeling with graph convolutional recurrent networks". In: *Neural Information Processing - 25th International Conference, ICONIP 2018, Siem Reap, Cambodia, December 13-16, 2018, Proceedings, Part I*. Vol. 11301. Lecture Notes in Computer Science. Springer, 2018, pp. 362–373. DOI: 10.1007/978-3-030-04167-0_33.

[SGM16] Florian Strub, Romaric Gaudel, and Jérémie Mary. "Hybrid recommender system based on autoencoders". In: *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, DLRS@RecSys 2016, Boston, MA, USA, September 15, 2016*. ACM, 2016, pp. 11–16. DOI: 10.1145/2988450.2988456.

[SGM19] Emma Strubell, Ananya Ganesh, and Andrew McCallum. "Energy and policy considerations for deep learning in NLP". In: *ACL (1)*. Association for Computational Linguistics, 2019, pp. 3645–3650. URL: 10.18653/v1/p19-1355.

[SGT+09] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. "The graph neural network model". In: *IEEE Trans. Neural Networks* 20.1 (2009).

[SHK+14] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting". In: *J. Mach. Learn. Res.* 15.1 (2014), pp. 1929–1958. URL: http://dl.acm.org/citation.cfm?id=2670313.

[SHM+12] Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. "Semantic compositionality through recursive matrix-vector spaces". In: *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL 2012, July 12-14, 2012, Jeju Island, Korea*. ACL, 2012, pp. 1201–1211. URL: https://aclanthology.org/D12-1110/.

[SJH+21] Koustuv Sinha, Robin Jia, Dieuwke Hupkes, Joelle Pineau, Adina Williams, and Douwe Kiela. "Masked language modeling and the distributional hypothesis: order word matters pre-training for little". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*. Association for Computational Linguistics, 2021, pp. 2888–2913. URL: https://aclanthology.org/2021.emnlp-main.230.

[Sma73] Henry Small. "Co-citation in the scientific literature: A new measure of the relationship between two documents". In: *J. Am. Soc. Inf. Sci.* 24.4 (1973), pp. 265–269. DOI: 10.1002/asi.4630240406.

[SMB+19] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. *Pitfalls of graph neural network evaluation*. 2019. arXiv: 1811.05868 [cs.LG].

[SMD+13] Ilya Sutskever, James Martens, George E. Dahl, and Geoffrey E. Hinton. "On the importance of initialization and momentum in deep learning". In: *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*. Vol. 28. JMLR Workshop and Conference Proceedings. JMLR.org, 2013, pp. 1139–1147. URL: http://proceedings.mlr.press/v28/sutskever13.html.

[SMS+15] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. "AutoRec: Autoencoders meet collaborative filtering". In: *Proceedings of the 24th International Conference on World Wide Web Companion, WWW 2015, Florence, Italy, May 18-22, 2015 - Companion Volume*. ACM, 2015, pp. 111–112. DOI: 10.1145/2740908.2742726.

[SNB+09] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. "Collective classification in network data". In: *AI Mag.* 29.3 (2009), pp. 93–106. DOI: 10.1609/aimag.v29i3.2157.

[SP97] Mike Schuster and Kuldip K. Paliwal. "Bidirectional recurrent neural networks". In: *IEEE Trans. Signal Process.* 45.11 (1997), pp. 2673–2681. DOI: 10.1109/78.650093.

[SPW+13] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. "Recursive deep models for semantic compositionality over a sentiment treebank". In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*. ACL, 2013, pp. 1631–1642. URL: https://aclanthology.org/D13-1170/.

[STB+18] Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J. Pal. "Learning general purpose distributed sentence representations via large scale multi-task learning". In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL: https://openreview.net/forum?id=B18WgG-CZ.

[Ste19] Harald Steck. "Embarrassingly shallow autoencoders for sparse data". In: *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*. ACM, 2019, pp. 3251–3257. DOI: 10.1145/3308558.3313710.

[SVL14] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. "Sequence to sequence learning with neural networks". In: *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*. 2014, pp. 3104–3112. URL: https://proceedings.neurips.cc/paper/2014/hash/a14ac55a4f27472c5d894ec1c3c743d2-Abstract.html.

[SVR09] Shilad Sen, Jesse Vig, and John Riedl. "Tagommenders: Connecting users to items through tags". In: *Proceedings of the 18th International Conference on World Wide Web, WWW 2009, Madrid, Spain, April 20-24, 2009*. ACM, 2009, pp. 671–680. DOI: 10.1145/1526709.1526800. URL: https://doi.org/10.1145/1526709.1526800.

[SWG+20] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, and Hao Yang. "DySAT: Deep neural representation learning on dynamic graphs via self-attention networks". In: *WSDM '20: The Thirteenth ACM International Conference on Web Search and Data Mining, Houston, TX, USA, February 3-7, 2020*. ACM, 2020, pp. 519–527. DOI: 10.1145/3336191.3371845.

[SWR20] Victor Sanh, Thomas Wolf, and Alexander M. Rush. "Movement pruning: adaptive sparsity by fine-tuning". In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. 2020. URL: https://proceedings.neurips.cc/paper/2020/hash/eae15aabaa768ae4a5993a8a4f4fa6e4-Abstract.html.

[SWW+18] Dinghan Shen, Guoyin Wang, Wenlin Wang, Martin Renqiang Min, Qinliang Su, et al. "Baseline needs more love: on simple word-embedding-based models and associated pooling mechanisms". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*. Association for Computational Linguistics, 2018, pp. 440–450. DOI: 10.18653/v1/P18-1041. URL: https://aclanthology.org/P18-1041/.

[SXL17] Lei Shu, Hu Xu, and Bing Liu. "DOC: Deep open classification of text documents". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*. Association for Computational Linguistics, 2017, pp. 2911–2916. DOI: 10.18653/v1/d17-1314.

[SYL13] Daniel L. Silver, Qiang Yang, and Lianghao Li. "Lifelong machine learning systems: beyond learning algorithms". In: *AAAI Spring Symposium: Lifelong Machine Learning*. Vol. SS-13-05. AAAI, 2013.

[SYS+20] Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. "Mobilebert: a compact task-agnostic BERT for resource-limited devices". In: *ACL*. Association for Computational Linguistics, 2020, pp. 2158–2170. DOI: 10.18653/v1/2020.acl-main.195.

[SZ08] Börkur Sigurbjörnsson and Roelof van Zwol. "Flickr tag recommendation based on collective knowledge". In: *Proceedings of the 17th*

International Conference on World Wide Web, WWW 2008, Beijing, China, April 21-25, 2008. ACM, 2008, pp. 327–336. DOI: 10.1145/1367497.1367542.

[SZJ+21] Jianshan Sun, Mingyue Zhu, Yuanchun Jiang, Ye-Zheng Liu, and Le Wu. "Hierarchical attention model for personalized tag recommendation". In: *J. Assoc. Inf. Sci. Technol.* 72.2 (2021), pp. 173–189. DOI: 10.1002/asi.24400.

[TBG+18] Ilya O. Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schölkopf. "Wasserstein auto-encoders". In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL: https://openreview.net/forum?id=HkL7n1-0b.

[TCL+19] Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. *Well-read students learn better: on the importance of pre-training compact models*. 2019. arXiv: 1908.08962 [cs.CL].

[TDA+21] Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. "Long range arena : A benchmark for efficient transformers". In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL: https://openreview.net/forum?id=qVyeW-grC2k.

[TDB+20] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. *Efficient transformers: a survey*. 2020. arXiv: 2009.06732 [cs.LG].

[TDW+17] Rakshit Trivedi, Hanjun Dai, Yichen Wang, and Le Song. "Know-evolve: deep temporal reasoning for dynamic knowledge graphs". In: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 3462–3471. URL: http://proceedings.mlr.press/v70/trivedi17a.html.

[TFB+19] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. "DyRep: Learning representations over dynamic graphs". In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL: https://openreview.net/forum?id=HyePrhR5KX.

[THK+21] Ilya Tolstikhin et al. *Mlp-mixer: an all-mlp architecture for vision*. 2021. arXiv: 2105.01601 [cs.CV].

[Thr98] Sebastian Thrun. "Lifelong learning algorithms". In: *Learning to learn*. Springer, 1998.

[TJF+17] Shuai Tang, Hailin Jin, Chen Fang, Zhaowen Wang, and Virginia R. de Sa. "Rethinking skip-thought: A neighborhood based approach". In: *Proceedings of the 2nd Workshop on Representation Learning for NLP, Rep4NLP@ACL 2017, Vancouver, Canada, August 3, 2017*. Association for Computational Linguistics, 2017, pp. 211–218. DOI: 10.18653/v1/w17-2625.

[TK07] Grigorios Tsoumakas and Ioannis Katakis. "Multi-label classification: an overview". In: *Int. J. Data Warehous. Min.* 3.3 (2007), pp. 1–13.

[TKV11] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis P. Vlahavas. "Random k-labelsets for multilabel classification". In: *IEEE Trans. Knowl. Data Eng.* 23.7 (2011), pp. 1079–1089.

[TLL+19] Raphael Tang, Yao Lu, Linqing Liu, Lili Mou, Olga Vechtomova, and Jimmy Lin. *Distilling task-specific knowledge from bert into simple neural networks*. 2019. arXiv: 1903.12136 [cs.CL].

[TM95] Sebastian Thrun and Tom M. Mitchell. "Learning one more thing". In: *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI 95, Montréal Québec, Canada, August 20-25 1995, 2 Volumes*. Morgan Kaufmann, 1995, pp. 1217–1225. URL: http://ijcai.org/Proceedings/95-2/Papers/026.pdf.

[TQM15] Jian Tang, Meng Qu, and Qiaozhu Mei. "PTE: predictive text embedding through large-scale heterogeneous text networks". In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*. ACM, 2015, pp. 1165–1174. DOI: 10.1145/2783258.2783307.

[TRJ+19] Henry Tsai, Jason Riesa, Melvin Johnson, Naveen Arivazhagan, Xin Li, and Amelia Archer. "Small and practical BERT models for sequence labeling". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*. Association for Computational Linguistics, 2019, pp. 3630–3634. DOI: 10.18653/v1/D19-1374.

[TS20] Martin Toepfer and Christin Seifert. "Fusion architectures for automatic subject indexing under concept drift". In: *Int. J. Digit. Libr.* 21.2 (2020), pp. 169–189. DOI: 10.1007/s00799-018-0240-3. URL: https://doi.org/10.1007/s00799-018-0240-3.

[TSZ+20] Shaoyu Tao, Chaoyuan Shen, Li Zhu, and Tao Dai. "SVD-CNN: A convolutional neural network model with orthogonal constraints based on SVD for context-aware citation recommendation". In: *Comput. Intell. Neurosci.* 2020 (2020), 5343214:1–5343214:12. DOI: 10.1155/2020/5343214.

[TZY+08] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. "Arnetminer: extraction and mining of academic social networks". In: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008*. ACM, 2008, pp. 990–998. DOI: 10.1145/1401890.1402008.

[VBK16] Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. "Order matters: sequence to sequence for sets". In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. 2016. URL: http://arxiv.org/abs/1511.06391.

[VCC+18] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. "Graph attention networks". In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL: https://openreview.net/forum?id=rJXMpikCZ.

[VFH+19] Petar Velickovic, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. "Deep graph infomax". In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL: https://openreview.net/forum?id=rklz9iAcKQ.

[VGM+18] Iacopo Vagliano, Lukas Galke, Florian Mai, and Ansgar Scherp. "Using adversarial autoencoders for multi-modal automatic playlist continuation". In: *Proceedings of the ACM Recommender Systems Challenge, RecSys Challenge 2018, Vancouver, BC, Canada, October 2, 2018*. ACM, 2018, 5:1–5:6. DOI: 10.1145/3267471.3267476.

[VGS22] I. Vagliano, L. Galke, and A. Scherp. "Recommendations for item set completion: On the semantics of item co-occurrence with data sparsity, input size, and input modalities". In: *Information Retrieval Journal* (Apr. 2022). ISSN: 1573-7659. DOI: 10.1007/s10791-022-09408-9.

[VLB+08] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. "Extracting and composing robust features with denoising autoencoders". In: *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*. Vol. 307. ACM International Conference Proceeding Series. ACM, 2008, pp. 1096–1103. DOI: 10.1145/1390156.1390294. URL: https://doi.org/10.1145/1390156.1390294.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention is all you need". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. 2017, pp. 5998–6008. URL: https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

[VTF+21] Ishwar Venugopal, Jessica Töllich, Michael Fairbank, and Ansgar Scherp. "A comparison of deep-learning methods for analysing and predicting business processes". In: *IJCNN*. IEEE, 2021. URL: https://arxiv.org/abs/2102.07838.

[WBG+16] John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. "Towards universal paraphrastic sentence embeddings". In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. 2016. URL: http://arxiv.org/abs/1511.08198.

[WCL+21] Bi Wang, Yang Chen, Xuelian Li, and Junfu Chen. "Lifelong classification in open world with limited storage requirements". In: *Neural Comput.* 33.7 (2021), pp. 1818–1852.

[WCS+21] Lingfei Wu, Yu Chen, Kai Shen, Xiaojie Guo, Hanning Gao, Shucheng Li, Jian Pei, and Bo Long. *Graph neural networks for natural language processing: a survey*. 2021. arXiv: 2106.06090 [cs.CL].

[WHC+16] Geoffrey I Webb, Roy Hyde, Hong Cao, Hai Long Nguyen, and Francois Petitjean. "Characterizing concept drift". In: *Data Mining and Knowledge Discovery* 30.4 (2016).

[WJS+19] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S. Yu. "Heterogeneous graph attention network". In: *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*. ACM, 2019, pp. 2022–2032. DOI: 10.1145/3308558.3313562.

[WJZ+19] Felix Wu, Amauri H. Souza Jr., Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. "Simplifying graph convolutional networks". In: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 6861–6871. URL: http://proceedings.mlr.press/v97/wu19e.html.

[WJZ+20] Hao Wu, Patrick Judd, Xiaojie Zhang, Mikhail Isaev, and Paulius Micikevicius. *Integer quantization for deep learning inference: principles and empirical evaluation*. 2020. arXiv: 2004.09602 [cs.LG].

[WK19] John Wieting and Douwe Kiela. "No training required: exploring random encoders for sentence classification". In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL: https://openreview.net/forum?id=BkgPajAcY7.

[WLC+19] Ruishuang Wang, Zhao Li, Jian Cao, Tong Chen, and Lei Wang. "Convolutional recurrent neural networks for text classification". In: *International Joint Conference on Neural Networks, IJCNN 2019 Budapest, Hungary, July 14-19, 2019*. IEEE, 2019, pp. 1–6. DOI: 10.1109/IJCNN.2019.8852406.

[WLG+18] Geoffrey I Webb, Loong Kuan Lee, Bart Goethals, and François Petitjean. "Analyzing concept drift and shift from sample data". In: *Data Mining and Knowledge Discovery* 32.5 (2018), pp. 1179–1199.

[WM12] Sida I. Wang and Christopher D. Manning. "Baselines and bigrams: simple, good sentiment and topic classification". In: *The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, July 8-14, 2012, Jeju Island, Korea - Volume 2: Short Papers*. The Association for Computer Linguistics, 2012, pp. 90–94. URL: https://aclanthology.org/P12-2018/.

[WPI20] Moshe Wasserblat, Oren Pereg, and Peter Izsak. "Exploring the boundaries of low-resource BERT distillation". In: *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*. Association for Computational Linguistics, Nov. 2020, pp. 35–40. DOI: 10.18653/v1/2020.sustainlp-1.5. URL: https://aclanthology.org/2020.sustainlp-1.5.

[WPN+19] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. "SuperGLUE: A stickier benchmark for general-purpose language understanding systems". In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. 2019, pp. 3261–3275. URL: https://proceedings.neurips.cc/paper/2019/hash/4496bf24afe7fab6f046bf4923da8de6-Abstract.html.

[WPZ20] Man Wu, Shirui Pan, and Xingquan Zhu. "Openwgl: open-world graph learning". In: *20th IEEE International Conference on Data Mining, ICDM 2020, Sorrento, Italy, November 17-20, 2020*. IEEE, 2020, pp. 681–690. DOI: 10.1109/ICDM50108.2020.00077.

[WQG+22] Chen Wang, Yuheng Qiu, Dasong Gao, and Sebastian Scherer. "Lifelong Graph Learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 13719–13728.

[WSC+16] Yonghui Wu et al. *Google's neural machine translation system: bridging the gap between human and machine translation*. 2016. arXiv: 1609.08144 [cs.CL].

[WSH+18] Yequan Wang, Aixin Sun, Jialong Han, Ying Liu, and Xiaoyan Zhu. "Sentiment analysis by capsules". In: *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*. ACM, 2018, pp. 1165–1174. DOI: 10.1145/3178876.3186015. URL: https://doi.org/10.1145/3178876.3186015.

[WSM+19] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. "GLUE: A multi-task benchmark and analysis platform for natural language understanding". In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL: https://openreview.net/forum?id=rJ4km2R5t7.

[WSS+14] Claudia Wagner, Philipp Singer, Markus Strohmaier, and Bernardo A. Huberman. "Semantic stability and implicit consensus in social tagging streams". In: *IEEE Trans. Comput. Soc. Syst.* 1.1 (2014), pp. 108–120. DOI: 10.1109/TCSS.2014.2307455. URL: https://doi.org/10.1109/TCSS.2014.2307455.

[WSW+20] Junshan Wang, Guojie Song, Yi Wu, and Liang Wang. "Streaming graph neural networks via continual learning". In: *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020*. ACM, 2020, pp. 1515–1524. DOI: 10.1145/3340531.3411963. URL: https://doi.org/10.1145/3340531.3411963.

[WWC05] Janyce Wiebe, Theresa Wilson, and Claire Cardie. "Annotating expressions of opinions and emotions in language". In: *Language resources and evaluation* 39.2 (2005), pp. 165–210.

[WWY15] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. "Collaborative deep learning for recommender systems". In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*. ACM, 2015, pp. 1235–1244. DOI: 10.1145/2783258.2783273.

[WYZ+17] Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. "IRGAN: A minimax game for unifying generative and discriminative information retrieval models". In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*. ACM, 2017, pp. 515–524. DOI: 10.1145/3077136.3080786.

[XHL+19] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. "How powerful are graph neural networks?" In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL: https://openreview.net/forum?id=ryGs6iA5Km.

[XLT+18] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. "Representation learning on graphs with jumping knowledge networks". In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. Vol. 80. Proceedings

of Machine Learning Research. PMLR, 2018, pp. 5449–5458. URL: http://proceedings.mlr.press/v80/xu18c.html.

[XRK+20] Da Xu, Chuanwei Ruan, Evren Körpeoglu, Sushant Kumar, and Kannan Achan. "Inductive representation learning on temporal graphs". In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL: https://openreview.net/forum?id=rJeW1yHYwH.

[YC11] Ainur Yessenalina and Claire Cardie. "Compositional matrix-space models for sentiment analysis". In: *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL*. ACL, 2011, pp. 172–182. URL: https://aclanthology.org/D11-1016/.

[YCL+21] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. "Do transformers really perform badly for graph representation?" In: *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*. 2021, pp. 28877–28888. URL: https://proceedings.neurips.cc/paper/2021/hash/f1c1592588411002af340cbaedd6fc33-Abstract.html.

[YCS16] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. "Revisiting semi-supervised learning with graph embeddings". In: *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*. Vol. 48. JMLR Workshop and Conference Proceedings. JMLR.org, 2016, pp. 40–48. URL: http://proceedings.mlr.press/v48/yanga16.html.

[YJK+19] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J. Kim. "Graph transformer networks". In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. 2019, pp. 11960–11970. URL: https://proceedings.neurips.cc/paper/2019/hash/9d63484abb477c97640154d40595a3bb-Abstract.html.

[YML19] Liang Yao, Chengsheng Mao, and Yuan Luo. "Graph convolutional networks for text classification". In: *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Appli-*

cations of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019. AAAI Press, 2019, pp. 7370–7377. DOI: 10.1609/aaai.v33i01.33017370.

[YSG+19] Ze Yang, Linjun Shou, Ming Gong, Wutao Lin, and Daxin Jiang. *Model compression with multi-task knowledge distillation for web-scale question answering system*. 2019. arXiv: 1904.09636 [cs.CL].

[YYM+18] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. "Hierarchical graph representation learning with differentiable pooling". In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*. 2018, pp. 4805–4815. URL: https://proceedings.neurips.cc/paper/2018/hash/e77dbaf6759253c7c6d0efc5690369c7-Abstract.html.

[YZL+21] Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. *Generalized out-of-distribution detection: A survey*. 2021. arXiv: 2110.11334.

[ZC18] Muhan Zhang and Yixin Chen. "Link prediction based on graph neural networks". In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*. 2018, pp. 5171–5181.

[ZC21] Fan Zhou and Chengtai Cao. "Overcoming catastrophic forgetting in graph neural networks with experience replay". In: *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*. AAAI Press, 2021, pp. 4714–4722. URL: https://ojs.aaai.org/index.php/AAAI/article/view/16602.

[ZDX+21] Lu Zhang, Jiandong Ding, Yi Xu, Yingyao Liu, and Shuigeng Zhou. "Weakly-supervised text classification based on keyword graph". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*. Association for Computational Lin-

guistics, 2021, pp. 2803–2813. URL: https://aclanthology.org/2021.emnlp-main.222.

[ZGL+20] Xujuan Zhou, Raj Gururajan, Yuefeng Li, Revathi Venkataraman, Xiaohui Tao, Ghazal Bargshady, Prabal Datta Barua, and Srinivas Kondalsamy-Chennakesavan. "A survey on text classification and its applications". In: *Web Intell.* 18.3 (2020), pp. 205–216.

[ZM20] Yang Zhang and Qiang Ma. "DocCit2Vec: citation recommendation via embedding of content and structural contexts". In: *IEEE Access* 8 (2020), pp. 115865–115875. DOI: 10.1109/ACCESS.2020.3004599.

[ZQZ+16] Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. "Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling". In: *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*. ACL, 2016, pp. 3485–3495. URL: https://aclanthology.org/C16-1329/.

[ZSH+19] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V. Chawla. "Heterogeneous graph neural network". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*. ACM, 2019, pp. 793–803. DOI: 10.1145/3292500.3330961.

[ZW15] Min-Ling Zhang and Lei Wu. "Lift: Multi-label learning with label-specific features". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 37.1 (2015), pp. 107–120. DOI: 10.1109/TPAMI.2014.2339815. URL: https://doi.org/10.1109/TPAMI.2014.2339815.

[ZWG+20] Da Zheng, Minjie Wang, Quan Gan, Zheng Zhang, and George Karypis. "Learning graph neural networks with deep graph library". In: *Companion of The 2020 Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*. ACM, 2020, pp. 305–306. DOI: 10.1145/3366424.3383111.

[ZWY+16] Dell Zhang, Jun Wang, Emine Yilmaz, Xiaoling Wang, and Yuxin Zhou. "Bayesian performance comparison of text classifiers". In: *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, SIGIR 2016, Pisa, Italy, July 17-21, 2016*. ACM, 2016, pp. 15–24. DOI: 10.1145/2911451.2911547.

[ZXW+20] Renjie Zhou, Dongchen Xia, Jian Wan, and Sanyuan Zhang. "An intelligent video tag recommendation method for improving video popularity in mobile computing environment". In: *IEEE Access* 8 (2020), pp. 6954–6967. DOI: 10.1109/ACCESS.2019.2961392.

[ZYW21] Weidong Zhao, Zhaoxin Yu, and Ran Wu. "A citation recommendation method based on context correlation". In: *Intell. Data Anal.* 25.1 (2021), pp. 225–243. DOI: 10.3233/IDA-195041.

[ZYX+17] Shuai Zhang, Lina Yao, Xiwei Xu, Sen Wang, and Liming Zhu. "Hybrid collaborative recommendation via semi-autoencoder". In: *Neural Information Processing - 24th International Conference, ICONIP 2017, Guangzhou, China, November 14-18, 2017, Proceedings, Part I*. Vol. 10634. Lecture Notes in Computer Science. Springer, 2017, pp. 185–193. DOI: 10.1007/978-3-319-70087-8_20.

[ZZ14] Min-Ling Zhang and Zhi-Hua Zhou. "A review on multi-label learning algorithms". In: *IEEE Trans. Knowl. Data Eng.* 26.8 (2014), pp. 1819–1837. DOI: 10.1109/TKDE.2013.39.

[ZZP+19] Shichao Zhu, Chuan Zhou, Shirui Pan, Xingquan Zhu, and Bin Wang. "Relation structure-aware heterogeneous graph neural network". In: *2019 IEEE International Conference on Data Mining, ICDM 2019, Beijing, China, November 8-11, 2019*. IEEE, 2019, pp. 1534–1539. DOI: 10.1109/ICDM.2019.00203.

[ZZQ+17] Fuzhen Zhuang, Zhiqiang Zhang, Mingda Qian, Chuan Shi, Xing Xie, and Qing He. "Representation learning via dual-autoencoder for recommendation". In: *Neural Networks* 90 (2017), pp. 83–89. DOI: 10.1016/j.neunet.2017.03.009.

[ZZS+20] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor K. Prasanna. "GraphSAINT: Graph sampling based inductive learning method". In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL: https://openreview.net/forum?id=BJe8pkHFwS.