TermPicker:
Recommending Vocabulary Terms for Reuse When Modeling Linked Open Data

Dissertation

zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften
(Dr. rer. nat.)

der Technischen Fakultät
der Christian-Albrechts-Universität zu Kiel

vorgelegt von

Johann Schaible

Kiel, 2017

Referent: . . . . . Prof. Dr. Ansgar Scherp

Koreferent: . . . . . Prof. Dr. York Sure-Vetter

Tag der mündlichen Prüfung: . . . . . 17. Februar 2017

Zum Druck genehmigt: . . . . . 17. Februar 2017

Zusammenfassung

Linked Open Data (LOD) bezeichnet im World Wide Web publizierte und maschinenlesbare Daten, deren Bedeutung explizit definiert ist. Zur Repräsentation von LOD werden sogenannte *Resource Description Framework* (RDF)-Vokabulare verwendet. Ein RDF-Vokabular ist eine Kollektion von (eindeutigen) Vokabulartermen wie *Klassen*, die den Typ einer Datenentität beschreiben oder *Properties*, die eine Beziehung zwischen Entitäten darstellen. Beim Modellieren von LOD ist es gute Praxis, bestehende RDF-Vokabularterme wiederzuverwenden bevor man neue Terme definiert. Dies reduziert die Heterogenität von Datenrepräsentationen und erleichtert Dritten die Daten zu verstehen und zu verwenden.

Die schlichte Anweisung bestehende RDF-Vokabularterme wiederzuverwenden ist jedoch unzureichend für viele Datenmodellierer, um zu entscheiden welche Terme genau sie wiederverwenden sollen. Zum einen existieren immer mehr wiederverwendbare RDF-Vokabulare, sogar wenn die Suche auf eine Domäne eingeschränkt wird. Zum anderen kann ein RDF-Vokabular mehrere ähnliche Terme enthalten, so dass Datenmodellierer unsicher sind, welcher Term der passendere ist. Ohne weitere Orientierungshilfen ist es demnach schwer zu entscheiden welcher Term wiederverwendet werden sollte. Unpassende Verwendungen können hierbei die Datenqualität erheblich mindern und somit dazu führen, dass Dritte den Datensatz womöglich nicht verwenden werden.

In dieser Arbeit wird das neuartige Empfehlungssystem für Vokabularterme *TermPicker* vorgestellt und evaluiert. Das System sammelt Informationen, wie andere LOD-Datenanbieter ihre Daten modelliert haben und repräsentiert diese Informationen in sogenannten *Schema-Level Patterns*, kurz SLPs. Auf Basis der SLPs kalkuliert TermPicker eine geordnete Liste von vorgeschlagenen Vokabulartermen. Die Rangordnung der Empfehlungen wird hierbei entweder mit Hilfe der Machine Learning Methode "Learning To Rank" (L2R) oder der Data Mining Methode "Association Rule" mining (AR) ermittelt. TermPicker wird auf zwei Arten evaluiert. Zunächst werden TermPickers Vorschläge mit Empfehlungen, die auf aktuell gängigen Strategien für die Wiederverwendung von Vokabulartermen basieren, in einem automatisierten Kreuzvalidierungsverfahren verglichen. Die aktuell gängigen Strategien für die Wiederverwendung von Vokabulartermen werden hierbei mit Hilfe einer Umfrage unter LOD-Experten vor dem Validierungsverfahren ermittelt. Beim Kreuzvalidierungsverfahren selbst werden die *Mean Average Precision* (MAP) und der *Mean Reciprocal Rank* bis zur fünften Position (MRR@5) als Evaluationsmetriken verwendet, da sie die Qualität einer geordneten Liste bewerten können. Als zweite Evaluation überprüft eine Nutzerstudie welche der Empfehlungsmethoden (L2R vs. AR) echte Nutzer bei der Wiederverwendung von Vokabulartermen in der Praxis am meisten unterstützt. Hierfür wird TermPicker in das Datenmodellierungstool *Karma* integriert. Die Teilnehmer der Nutzerstudie, sprich TermPickers potentielle Nutzer, müssen beim Modellieren dreier LOD-Datensätze Vokabularterme wiederverwenden und erhalten hierfür Empfehlungen, entweder

basierend auf Learning To Rank, Association Rule mining, oder sie erhalten keine Empfehlungen.

Die Resultate des Kreuzvalidierungsverfahren zeigen, dass die Verwendung von SLPs TermPicker befähigt etwa 35% höhere MAP- und MRR@5-Werte bei der Ordnung von Vokabulartermen zu erreichen. Sowohl die L2R-basierte als auch die AR-basierte Methode erlangen MAP- und MRR@5-Werte im Bereich von MAP ≈ 0.75 und MRR@$5 \approx 0.80$. Die Ergebnisse der Nutzerstudie zeigen, dass Empfehlungen basierend auf AR von den Nutzern klar präferiert werden. Bei AR-basierten Empfehlungen werden empfohlene Terme häufiger akzeptiert, die Bearbeitungszeit der Modellierungsaufgaben ist geringer und die Qualität der erstellten LOD-Repräsentation sowie die allgemeine Zufriedenheit ist deutlich höher ausgefallen.

Die Ergebnisse dieser Arbeit zeigen eine neue und validierte Möglichkeit, wie man RDF-Vokabularterme für die Wiederverwendung suchen und in das LOD Modell integrieren kann. TermPicker vereinfacht das Auffinden von RDF Klassen und Properties, die von anderen Datenanbietern in der LOD-Cloud zum Modellieren ähnlicher Daten verwendet werden. Neben Karma kann TermPicker auch in andere LOD Modellierungstools, wie Neologism, integriert werden. Somit ist TermPicker ein möglicher Vorreiter einer nächsten Generation der Wiederverwendung von RDF-Vokabulartermen.

Abstract

Linked Open Data (LOD) refers to data published on the Web in a way that it is machine-readable, its meaning is explicitly defined, and it is linked to other data sets. So-called *Resource Description Framework* (RDF) vocabularies are employed for LOD modeling. An RDF vocabulary is a collection of unique vocabulary terms comprising *classes*, which describe the type of a data entity, and *properties*, which describe relationships between data entities. When modeling data as LOD, it is considered best practice to reuse such vocabulary terms from already existing RDF vocabularies before inventing new ones. This reduces the heterogeneity in data representation and makes it easier for others to understand and consume the data. Unfortunately, this simple guidance is too unspecific, as deciding which vocabulary terms to reuse is far from trivial. First, even when data engineers are focused on a specific domain, there are more and more vocabularies to choose from. Second, various vocabularies contain multiple terms that seem to represent similar semantics. Data engineers are thereby uncertain about which term is the more appropriate option. Thus, without additional guidance, it is very difficult to reuse appropriate vocabulary terms for representing data as LOD. Using inappropriate vocabulary terms decreases the data quality. As a consequence, others will less likely use the data.

This work proposes and evaluates *TermPicker*: a novel approach alleviating this situation by recommending vocabulary terms based on the information how other data providers modeled their data as LOD. TermPicker gathers such information and represents it via so-called *schema-level patterns* (SLPs), which are used to calculate a ranked list of RDF vocabulary term recommendations. The ranking of the recommendations is based either on the machine learning approach "Learning To Rank" (L2R) or on the data mining approach "Association Rule" mining (AR). TermPicker is evaluated in a two-fold way. First, an automated cross-validation evaluates the prediction accuracy of TermPicker's recommendations by comparing them to term recommendations based on the most typical strategies (currently used by LOD engineers) to reuse vocabularies. These most typical reuse strategies are elaborated with the help of an online survey, in which LOD experts are asked to rank given LOD models from "best" to "worst" regarding the utilized vocabulary terms. For the cross-validation, the *Mean Average Precision* (MAP) as well as the *Mean Reciprocal Rank* at the first five positions (MRR@5) are used as evaluation metrics, since these measures assess the quality of an ordered list. Second, a user study examines which of the recommendation methods (L2R vs. AR) aids real users more to reuse RDF vocabulary terms in a practical setting. To this end, both approaches are integrated into the UI-based data modeling tool *Karma*. The participants, i.e., TermPicker's potential users, are asked to reuse vocabulary terms while modeling three data sets as LOD, but they receive either L2R-based recommendations, AR-based recommendation, or no recommendations.

The results of the cross-validation show that using SLPs, TermPicker achieves 35% higher MAP and MRR@5 values compared to using solely the features based on the typical reuse strategies. Both the L2R-based and the AR-based recommendation methods were able to calculate lists of recommendations with MAP ≈ 0.75 and MRR@$5 \approx 0.80$. However, the results of the user study show that the majority of the participants favor the AR-based recommendations. When using Association Rule mining, the participants accept significantly more recommendations, they need less time to complete the modeling tasks, and the quality of the LOD representations as well as the participants' satisfaction is significantly higher compared to using recommendations based on Learning To Rank.

The outcome of this work demonstrates a novel and validated method to find vocabulary terms for reuse. TermPicker alleviates the situation of searching for classes and properties used by other data providers on the LOD cloud for representing similar data. Besides Karma, the recommendation approach can also be integrated in further LOD modeling tools, such as Neologism, allowing for next generation vocabulary reuse when modeling Linked Open Data.

Contents

List of Figures

List of Tables

Listings

1 Introduction

Linked Open Data (LOD) constitutes data published on the Web such that it is machine readable, its meaning is explicitly defined, and it is linked to other (external) data sets [11]. To this end, Resource Description Framework (RDF) [54] vocabularies are employed to describe the data. The data entities are referred to as *resources* and the entire data set is represented via so-called *RDF triples* [44]. LOD aims at connecting resources across different data providers, i.e., moving from data islands to a global data space [42]. Figure 1.1 shows the so-called *Linked Open Data cloud* that contains a large number of the data sets published as LOD.[1] Every circle denotes a data set published by a data provider (identified by the name in the circle) within different domains (identified by the circle's color). The size of a circle specifies the size of the data set while the arrows between the circles specify which data sets are connected to one another. For example, the DBpedia [12] data set in the middle of the cloud is one of the largest data sets that is connected to and from other data sets.

Four basic rules are defined to publish Linked (Open) Data, and these rules have become known as the *Linked Data principles* [11]. These four rules are:

- Use Uniform Resource Identifiers (URIs) as names for things, i.e., as names for resources
- Use HTTP URIs as names for resources, so that people, i.e., third parties, can look up those names
- When others look up a URI, provide useful information, using Semantic Web standards, e.g., RDF, SPARQL[2])
- Include links to other URIs, so that more things can be discovered

Using HTTP URIs makes it possible to look up resources on the Web, such as the DBpedia resource http://dbpedia.org/page/The_Godfather, and gain further information about it. RDF vocabularies are collections of unique *vocabulary terms* comprising *classes*, which describe the type of a resource, and *properties*, which describe relationships between resources, or between a resource and a literal.[3] For example, the class http://dbpedia.org/ontology/Film from the DBpedia vocabulary[4] can specify that the type of the DBpedia resource http://dbpedia.org/page/The_Godfather is a movie and not a book, or a screenplay.

[1] The Linked Open Data cloud diagram 2014 is made by Max Schmachtenberg, Christian Bizer, Anja Jentzsch and Richard Cyganiak. `http://lod-cloud.net/`, last accessed July 15th, 2016

[2] `https://www.w3.org/TR/rdf-sparql-query/`, last accessed February 14th, 2017

[3] In this work, classes are instances of `http://www.w3.org/2000/01/rdf-schema#Class` and properties are instances of `http://www.w3.org/1999/02/22-rdf-syntax-ns#Property`

[4] `http://dbpedia.org/ontology/`, last accessed February 14th, 2017

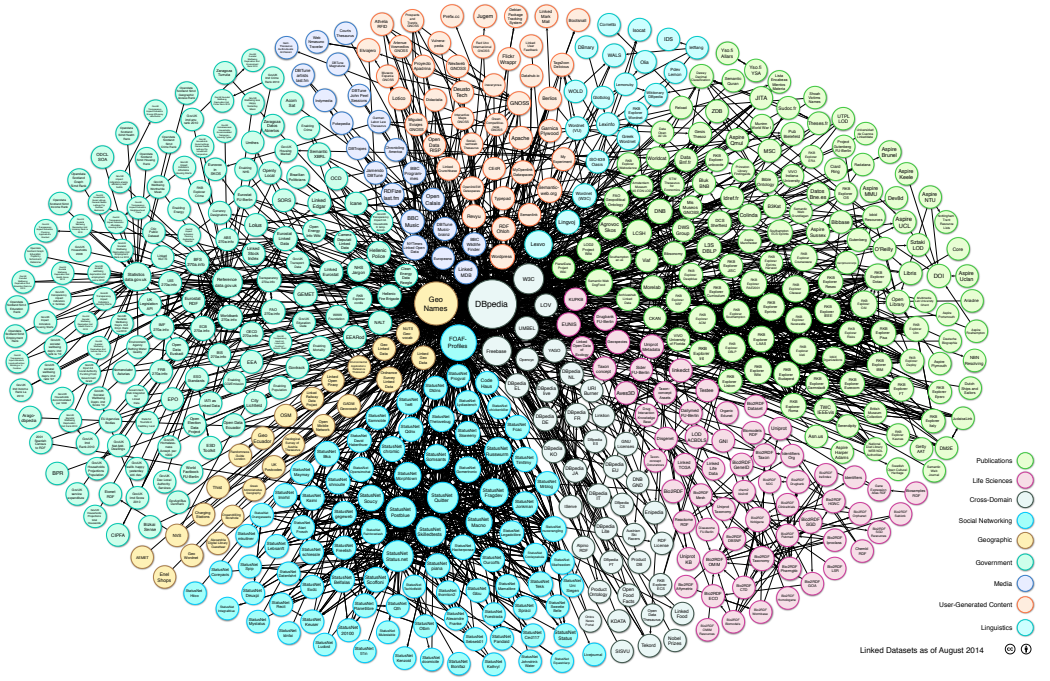


Figure 1.1: **The Linked Open Data Cloud 2014.** The Figure shows data sets from different domains (indicated by the color) that are connected to each other via typed links, thereby creating a global space of structured data.

A property can link the resource to another resource, e.g., to the director of the movie, or to a literal, e.g., the name of the movie. Encoding the data with such RDF vocabulary terms exploits the semantics of the resources and their connections, thereby allowing to query the data via the query language *SPARQL*. Including links to other (external) data sets makes it even possible to query across different data sets, and is thus a big step forward towards evolving from data islands to a global data space [42].

Providing open data and following the Linked Data principles, data engineers publish data as *5-star* Linked Open Data.[5] Besides being open and machine-readable data, this means that the data is modeled using RDF and is linked to other data sets. Additionally, there are best practices for modeling and publishing data as Linked Open Data [10, 42]. The practices include instructions how to choose URIs, which RDF vocabularies to use for representing specific information, how to provide the shared data, and how to set RDF links to other data sets [48].

This thesis focuses on the best practice stating that a LOD engineer should *reuse existing RDF vocabularies before inventing new ones*. By reusing existing RDF vocabulary terms, LOD engineers decrease the heterogeneity in data representation and thereby make the data easier to be consumed by LOD applications, like Watson [28] or Falcons [22, 24], or by querying the data using SPARQL [10, 42, 47]. However, as the amount of data published as

[5] `https://www.w3.org/DesignIssues/LinkedData.html`, last accessed February 14th, 2017

LOD increases, the diversity of how RDF vocabularies are used to describe the data is increasing as well [48]. In addition, different RDF vocabularies contain classes and properties that can be used to describe the same type of data. The simple advise to reuse RDF vocabulary terms is not sufficient and far from trivial, because there is already a vast amount of RDF vocabulary terms to choose from. Without additional guidance, LOD engineers may not know which vocabulary terms to select for representing a specific semantic meaning of a resource or a relation between resources.

The main contribution of this work is the introduction and the evaluation of *TermPicker*: a novel approach for alleviating the situation by recommending the LOD engineer RDF classes and properties for reuse. TermPicker induces structural information from published data sets on the LOD cloud how other data providers have combined vocabulary terms to represent their data. This structural information is used to recommend LOD engineers vocabulary terms that others have used to model data similar to the engineers' data. So-called *schema-level-patterns* (SLPs), which are tuples describing the connection between two sets of RDF classes via a set of properties, are used to represent the induced structural information. For example, if a data set on the LOD cloud uses the *Semantic Web for Research Communities* (SWRC)[6] vocabulary, the *Friend Of A Friend* (FOAF)[7] vocabulary, and the *Dublin Core Elements* (DC)[8] vocabulary for representing data on publications and their authors, the SLP

$$slp = (\{\text{swrc:Publication}\}, \{\text{dc:creator}\}, \{\text{foaf:Person}\}) \tag{1.1}$$

describes those RDF triples that have subject resources of type swrc:Publication and that are connected to resources of type foaf:Person via a dc:creator relation.[9] On the contrary to other methods for inducing schema from LOD data sets (cf. Section 3.2), SLPs describe only a small part of the entire LOD model representing the data. This makes them more useful for calculating term recommendations, as it is not needed to traverse the entire LOD model. TermPicker's workflow is as follows: TermPicker is provided an SLP as input, i.e., the query-SLP slp_q. The approach subsequently finds further vocabulary terms (in SLPs calculated from data on the LOD cloud) that others have used together with the terms in slp_q. For example, $slp_q = (\{\text{swrc:Publication}\}, \varnothing, \varnothing)$ is the input, and TermPicker recommends further vocabulary terms from other SLPs that also contain swrc:Publication in the first set of classes, e.g., the terms dc:creator and foaf:Person, if the SLP in equation (1.1) is calculated from an existing LOD data set. The quality of TermPicker's recommendations is evaluated in a two-fold way: First, it is compared to the quality of recommendations based on typical approaches to reuse vocabulary terms (which are currently employed by LOD engineers). To this end, 79 LOD experts and practitioners participated in a survey, which resulted in identifying the direct reuse of popular or domain-specific vocabulary terms as the most typical reuse strategies. The comparison is performed by measuring the Mean Average Precision (MAP) and the Mean Reciprocal Rank at the first five positions (MRR@5) that

[6] `http://ontoware.org/swrc/`, last accessed February 14th, 2017
[7] `http://xmlns.com/foaf/spec/`, last accessed February 14th, 2017
[8] `http://dublincore.org/documents/dces/`, last accessed February 14th, 2017
[9] Detailed information on the use of prefixes can be found in Section 2.1

asses the quality of a ranked list. The results of the evaluation showed that using SLPs to recommend vocabulary terms significantly improve the MAP and the MRR@5 values by about 35% (MAP ≈ 0.75 and MRR@5 ≈ 0.8). In a second evaluation, TermPicker is subject to a user study, in which LOD engineers received tasks to model data as LOD with the request to reuse RDF vocabulary terms wherever possible. Two recommendation algorithms, i.e., *Learning To Rank* (L2R) and *Association Rule* (AR) mining, (both having similar MAP and the MRR@5 values) are compared to each other based on task completion time, the number of accepted recommendations, the quality of the resulting LOD representations, and the user satisfaction regarding the recommendations. The results show that recommendations based on Association Rule mining performed significantly better regarding each measure.

In the following, the motivation for this work is outlined in Section 1.1. The investigated research questions and contributions are constituted in Section 1.2. Subsequently, Section 1.3 lists all accompanying publications before Section 1.4 outlines the remainder of this work.

1.1 Motivation

RDF vocabularies play a central role when modeling data as Linked Open Data. By reusing existing classes and properties from commonly known and widely adopted RDF vocabularies, data engineers increase the general understanding of the published data [10, 42].

However, reusing RDF vocabulary terms is far from trivial. This section provides a detailed insight on three major challenges when reusing RDF vocabulary terms. First, data engineers have to search for appropriate vocabulary terms that can be reused. Second, they have to choose between terms from different vocabularies that seem appropriate for reuse. Finally, even when focusing on one RDF vocabulary, it might contain several terms that appear to have a similar semantic meaning. Selecting improper vocabulary terms leads to a wrong representation of the semantics of the data. As a result, the data becomes more difficult to understand and to be consumed by LOD applications, e.g., finding resources to link to. This is likely to decrease the feasibility of constructing queries across data sets, which has a negative effect on moving towards a global data space.

Finding RDF Vocabulary Terms

As more and more data is published as LOD, an increasing amount of RDF vocabularies is being published on the Web as well. Optimally, the published vocabularies include specifications of the contained classes and properties as well as their documentation. Search engines, like Google[10], index the published vocabularies and their documentation [15], such that data engineers can easily find and access the vocabularies. A prominent example for an RDF vocabulary published in such a manner is the commonly known and widely accepted FOAF vocabulary.

However, even if the vocabulary and its documentation is published on the Web, first there is still a huge variety of RDF vocabularies to choose from, and second the documen-

[10] `http:www.google.com`, last accessed February 14th, 2017

tation does not specify how other data providers combine vocabulary terms. This includes combining terms from one vocabulary as well as across vocabularies. It also occurs that vocabularies get published without any documentation [47]. In the end, engineers have to examine existing data sets published on the LOD cloud, in order to manually infer how the vocabulary terms were combined to model the data as LOD.

To alleviate the situation, there are Web services that aggregate various RDF vocabularies and their documentations. Most popular examples are the Linked Open Vocabulary index (LOV)[11] [82], vocab.cc[12] [78], and LODstats[13] [3]. They provide a keyword-based search for vocabulary terms using string similarity to find appropriate matches. LOD engineers can find classes and properties that they able to describe with words. However, this might be very challenging. For example, "has colleague" could be a keyword-based query for finding a relationship between two resources of type foaf:Person specifying that they are colleagues. If no results are provided, the engineer must think of another phrasing of the query, such as "is colleague of", or "works with". This is likely to be very time-consuming or error-prone, as the engineer might not know how to describe a desired vocabulary term with words and tends to reuse a more general term, e.g., foaf:knows. Such a generalization does not express the actual relationship and the data loses a huge part of its semantic meaning. Furthermore, such keyword-based search services do not provide any information on how other data providers combine vocabulary terms to model data as LOD.

There is a clear need for RDF vocabulary term recommendation services. Such services could aid data engineers in finding reusable vocabulary terms without the need to phrase a string-based query, i.e., the recommender can suggest previously unknown vocabulary terms. For example, by examining how others specified that two persons are colleagues, a recommender could suggest the properties rel:colleagueOf and rel:worksWith from the relationship (REL)[14] vocabulary.

Equivalent Terms from Different Vocabularies

The increasing number of RDF vocabularies appearing on the Web makes it possible to combine a greater variety of different vocabulary terms for LOD modeling [48]. On the one hand, some vocabularies are domain-specific vocabularies and contain classes and properties that are not present in other vocabularies. On the other hand, there are more and more RDF vocabularies describing the same kind of data. As a result, many vocabularies contain classes and properties with the same semantic meaning. For example, the *schema.org*[15] vocabulary contains a class schema:Person representing a person in the same way as the class foaf:Person from the FOAF vocabulary.

The essential problem is that there are many vocabularies to choose from. LOD engineers face the challenge to investigate which vocabulary contains the most appropriate term. For instance, is it preferred to reuse schema:Person or foaf:Person? Figure 1.2 depicts a more

[11] `http://lov.okfn.org/dataset/lov`, last accessed February 14th, 2017
[12] `http://vocab.cc/`, last accessed February 14th, 2017
[13] `http://stats.lod2.eu/`, last accessed February 14th, 2017
[14] `http://vocab.org/relationship/`, last accessed February 14th, 2017
[15] `http://schema.org/`, last accessed February 14th, 2017

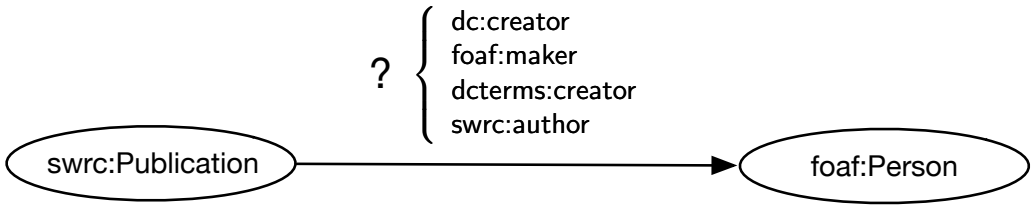


Figure 1.2: **Equivalent Terms from Different Vocabularies.** The engineer searches for a property that is most commonly used by other data providers to express the intended semantics of the relationship.

concrete example from the domain of scientific publications. Let us assume, that the class swrc:Publication from the SWRC vocabulary and the class foaf:Person have already been selected to represent some resources as publications and as persons. Subsequently, the data engineer wants to employ a property connecting these two classes to specify that a person is the author of a publication, but has several options. The Dublin Core Elements (DC), the Dublin Core Terms[16] (DCTERMS), which is an extension of Dublin Core Elements, the FOAF, and the SWRC vocabulary provide appropriate properties to describe the connection. On the one hand, by using swrc:author or foaf:maker the engineer reuses a term from an already used vocabulary. This reduces the complexity of using too many different vocabularies. On the other hand, the Dublin Core vocabularies provide terms for general meta information on library items such as publications. Thus, using dc:creator might represent the semantics of the relation in a more specific manner. As the DCTERMS vocabulary is used by more data sets than the DC vocabulary,[17] it might be even more appropriate to use dcterms:creator. There is also the possibility to change the classes from swrc:Publication to dcterms:BibliographicResource and from foaf:Person to dcterms:Agent. By doing so and using dcterms:creator as property, the engineer reuses terms from the most popular vocabulary in the community of bibliographic references and also reduces the complexity by reusing only terms from one vocabulary. In the end, the LOD engineer has the problem to decide which property of which vocabulary should be reused.

Vocabulary term recommendations based on how other data providers modeled their data could aid engineers in their decision which term to select. They can provide statistics not only how often a vocabulary term has been used by others, but also how often it has been used with other terms. For example, dcterms:creator is used more often than foaf:maker,[18] but as a property between the specific classes swrc:Publication and foaf:Person it could be the other way around. Choosing the property that is most commonly used in conjunction with the already used terms is more likely to reduce the heterogeneity in data representation.

[16] `http://dublincore.org/documents/dcmi-terms/`, last accessed February 14th, 2017

[17] 327 data sets use DCTERMS (`http://lov.okfn.org/dataset/lov/vocabs/dcterms`); 178 data sets use DC (`http://lov.okfn.org/dataset/lov/vocabs/dce`), last accessed February 15th, 2017

[18] 1.1 million overall usages (`http://stats.lod2.eu/properties?search=creator`) vs. 11k usages (`http://stats.lod2.eu/properties?search=maker`), last accessed February 15th, 2017

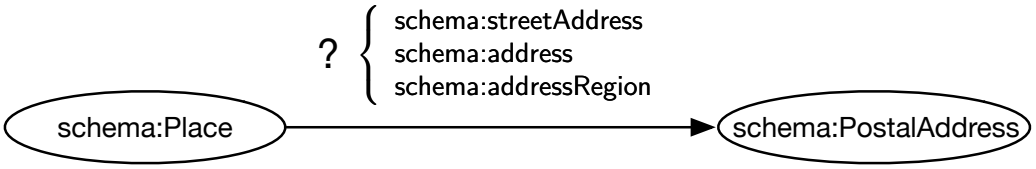


Figure 1.3: **Similar Terms within a Vocabulary.** The engineer searches for a property that connects two RDF classes in a semantically correct way considering the property's domain and range information.

After choosing a property, the LOD engineer can also repeat the process for the classes and exchange swrc:Publication and foaf:Person with better fitting ones.

Similar Terms within One Vocabulary

At some point of the modeling process, LOD engineers decide which vocabularies they want to reuse. But even if they have selected only one (domain-specific) vocabulary, it can be difficult to decide which terms to choose from that vocabulary. The reason is that various vocabularies contain classes and properties which seem to represent similar semantics. For instance, the classes mo:Recording and mo:RecordingSession from the Music Ontology[19] (for publishing music-related data) describe an event in which a song is recorded. However, they have different outgoing properties, so that the engineer has the challenge to decide which class to use.

In detail, an RDF property has a domain and range information that specifies which classes it is allowed to connect.[20] However, as classes and properties are described via human readable names, in many cases it is difficult to distinguish between the actual semantic meaning of terms. This specifically applies to cases in which classes and properties are encoded via non-self-explaining strings, such as "p_134" or "c-45". Figure 1.3 shows an example of the difficulty to select a property from one vocabulary to represent a specific relationship. The classes schema:Place and schema:PostalAddress from the schema.org vocabulary have already been chosen. The property connecting these classes should express that a place has a specific postal address. If choosing to reuse properties from the schema.org vocabulary, there are three possible options, i.e., schema:streetAddress, schema:address, and schema:addressRegion. LOD engineers must examine the vocabulary's documentation, in order to make a decision which of the properties can be used to represent the desired relation. This can be quite cumbersome, especially if the vocabulary does not have any documentation.

Recommending vocabulary terms based on how others modeled similar data can alleviate the situation and help the engineer to make a decision which term to reuse. As the domain and range information is encoded within the vocabularies, a recommender based on exploiting existing LOD cannot guarantee a correct suggestion. However, the majority of data sets on the LOD cloud use vocabulary terms correctly regarding the domain and range informa-

[19] `http://musicontology.com/`, last accessed February 14th, 2017

[20] Detailed information on the rdfs:domain and rdfs:range information of properties can be found in Section 2.1

tion of properties [48]. By investigating a huge amount of different data sets on the LOD cloud, the possibility of recommending a correct term is quite high. For example, when searching for a property connecting the classes schema:Place and schema:PostalAddress, the top recommendation would be schema:address, as it is the only property with the appropriate domain and range information. Even if there are data providers that use other properties, still the majority uses schema:address. Thus, schema:address is presented as the top recommendation and helps engineers in their decision to select the correct property. The same applies when choosing classes.

1.2 Research Questions

The challenges described in Section 1.1 motivate the use of a vocabulary term recommendation approach when modeling LOD. TermPicker alleviates the current situation by aiding engineers in finding previously unknown vocabulary terms and in deciding which vocabulary terms to select. On the one hand, the recommendations are based on features representing common strategies for reusing vocabulary terms. On the other hand, TermPicker uses schema-level patterns to compute a feature representing how vocabulary terms are combined in other data sets on the LOD cloud. Using that novel feature, TermPicker helps engineers to find classes and properties that have been used by other data providers for modeling data similar to the engineers' data. However, the central research question, which is investigated in this work, is: *When modeling Linked Open Data, to which extend does TermPicker aid LOD engineers in reusing RDF classes and properties compared to recommendations based on solely the (current) typical strategies for reusing vocabulary terms?*

To answer the main research question and thus to asses the overall benefit of using TermPicker, it is necessary to split this research question into three research questions (RQ1) to (RQ3) and answer them individually. The following paragraphs describe research questions (RQ1) to (RQ3) in more detail.

Research Question 1 (RQ1) – Typical Strategies for Reusing Vocabulary Terms

In LOD modeling, there seem to be several ways to reuse existing vocabulary terms, in order to make the published data as easy as possible to be consumed. Generally, the best practices suggest to reuse terms from *well-known* and *widely accepted* vocabularies wherever possible [10, 42]. Defining new classes and properties is considered appropriate only if existing vocabularies did not contain the required terms. Another strategy is to find rather domain-specific vocabularies and reuse as many terms as possible from these vocabularies [61]. An empirical study on Linked Data conformance [48] shows the diversity of these and further strategies how data providers reuse vocabulary terms. Some data providers use as many terms from existing vocabularies as possible whereas other data providers try minimize the number of different vocabularies to represent the data. The study also reveals that there is an equal share between data providers who reuse vocabulary terms directly and data providers who define their own terms and subsequently link these terms to existing vocabularies.

The diversity in different reuse strategies shows it remains unclear which strategies are currently most favored by active LOD engineers, although there are specific strategies mentioned in existing best practices. Thus, research question (RQ1) that is investigated in this work is defined as follows:

> **Research Question 1 (RQ1)**
>
> When modeling Linked Open Data, what are an LOD engineer's most typical strategies to reuse vocabulary terms?

Contributions to (RQ1). To gain insights on how LOD engineers approach the problem of reusing vocabulary terms, it is necessary to collect their experience and their most influencing factors whether to reuse a vocabulary term or not. Neither the set of best practices [42] nor the study on Linked Data conformance [48] provide such information on the engineers' experience and knowledge. In this work, such information is gathered by asking LOD engineers with the help of an online survey which examines the engineers' favorite strategies.

The survey consists of the following two parts: First, the survey's participants are asked to rate three *aspects* why they reuse vocabulary terms on a 5-point Likert scale from "not at all important" to "very important". These aspects comprise that engineers try to provide a clear data structure, make the data easier to be consumed by LOD applications, and/or establish an ontological agreement in data representation. Second, the participants are asked to perform three ranking tasks, in each ranking LOD models from "good" to "bad" in descending order with respect to the quality how vocabulary terms are reused. The data described by the LOD models is the same within one ranking task, but they differ across the tasks. Each model represents one specific reuse strategy, which are *minimizing the number of used vocabularies*, *interlinking proprietary terms with existing ones*, *reusing terms from popular vocabularies*, and *maximizing the number of used vocabularies*. The last strategy is used as a lower boundary.

Key insights: Providing a clear structure of the data and making it easier to be consumed by LOD applications are considered the most important aspects for reusing RDF vocabulary terms. In order to achieve this goals, most participants regard (i) reusing vocabulary terms from popular vocabularies directly and (ii) minizing the number of different vocabularies, i.e., reusing as many terms as possible from already used vocabularies, as most important reuse strategies.

Research Question 2 (RQ2) – Comparing Different Methods for Vocabulary Term Recommendations

For computing vocabulary term recommendations, TermPicker uses five features representing each recommended vocabulary term. Four of the five features are calculated based on the most typical strategies to reuse terms from popular vocabularies and minimize the number of different vocabularies, which were identified in the contribution of (RQ1). In addition, TermPicker uses schema-level patterns (SLPs) to calculate the so-called *SLP-feature*.

It counts the number of data providers on the LOD cloud that have used a recommended vocabulary term to represent data which is similar to the engineer's data.

Utilizing the SLP-feature for representing vocabulary terms describes a novel vocabulary reuse strategy, i.e., reuse vocabulary terms that other data providers on the LOD cloud have used to model data which is similar to your data. It is of interest to evaluate the benefit of using the SLP-feature for calculating vocabulary term recommendations. It is therefore necessary to compare recommendations based on this feature to recommendations based on features representing the current reuse strategies. To this end, it is also important to investigate which recommendation method can use the different features in the best way to produce valuable term suggestions. As a result, the second research question is as follows:

Research Question 2 (RQ2)

How do TermPicker's recommendations based on the SLP-feature compare to recommendations using features based on solely the typical strategies to reuse vocabulary terms?

Contributions to (RQ2). To calculate the prediction accuracy of the recommendation methods (including different sets of the five utilized features), the evaluation metrics *Mean Average Precision* (MAP) and *Mean Reciprocal Rank* to the 5-th position (MRR@5) are used, as they asses the quality of an ordered list. As recommendation methods, the machine learning approach "Learning To Rank" (L2R) [60, 40] and the data mining approach "Association Rule" mining (AR) [2, 1] are used. Learning To Rank allows for establishing a ranked set of RDF vocabulary terms based on different features. Which feature is most important for ranking the relevant terms to the top of the list is elaborated by the L2R approach automatically. This makes it possible to directly measure the impact of the SLP-feature on the ranking of RDF vocabulary terms. In addition, Association Rule mining uses solely the SLP-feature to define rules, such as "if other data providers user terms x and y, they also use term z". Therefore, the rules can be used to generate RDF vocabulary term recommendations by simply ranking the recommendations based on the SLP-feature in descending order. This allows for a comparison between the L2R-based approach and the AR-based approach.

Key insights: Using the SLP-feature increases both the MAP and the MRR@5 values by 35% compared to using features based only on the current typical reuse strategies. On average, the MAP value is MAP ≈ 0.75 and the MRR@5 value is MRR@5 ≈ 0.8 for the best performing L2R-based approach. However, the AR-based approach achieved very similar results, such that no significant differences were detected when comparing it to the L2R-based approach.

Research Question 3 (RQ3) – User Study to Compare the L2R-based Approach and the AR-based Approach in a Practical Setting

When evaluating recommender systems, it is accustomed to perform an evaluation without users, i.e., an offline evaluation, and an evaluation with real user interactions in a practical setting, i.e., an online evaluation [8, 50, 77]. This specifically applies, if the investigated recommendation methods achieve similar results in an offline evaluation.

Even if one recommendation method is able to produce a very qualitative list of suggested terms, it does not necessarily mean that users will be satisfied with the recommendations. For example, if five out of the first ten recommended vocabulary terms are appropriate for reuse and the top recommendation is a relevant recommendation, the Mean Reciprocal Rank value is MRR $= 1$ (best possible value). However, the relevant terms might be separated by irrelevant terms, e.g., only every second recommendation is a relevant term. This is likely confuse the LOD engineer, as it is difficult to distinguish between relevant and irrelevant recommendations [8]. Such recommendations are not likely to aid engineers in reusing vocabulary terms. Therefore, the third research question in this work investigates which of the two recommendation methods, i.e, Learning To Rank vs. Association Rule mining, is most suitable to aid LOD engineers in a practical setting with real user interaction. This research question is constituted as follows:

Research Question 3 (RQ3)

Which recommendation method (L2R vs. AR) aids LOD engineers most in reusing vocabulary terms for modeling data as LOD of higher quality with less effort in a practical setting?

Contributions to (RQ3). To compare the L2R-based approach to the AR-based approach in a practical setting, TermPicker is integrated into the data modeling tool Karma[21] [55]. Based on Karma, a user study is conducted in which participants, i.e., invited LOD engineers, make use of TermPicker's recommendations to reuse vocabulary terms for LOD modeling.

In detail, the participants have to perform three tasks in which they are asked to model data from three different domains. In each task, the participants shall reuse existing RDF vocabulary terms. TermPicker's recommendation approach is different for each of the three task, i.e., the participants receive L2R-based recommendation for one task, AR-based recommendations for another task, an no recommendation for the third task. The quality of the recommendations is assessed by measuring the *task completion time*, the *recommendation acceptance score* (number of times participants chose a recommended term vs. manual search), the *data quality* of the resulting LOD representation, and the participants' satisfaction.

Key insights: With recommendations based on Association Rule mining, participants were able to complete the modeling tasks in less time compared to the task completion time

[21] `http://usc-isi-i2.github.io/karma/`, last accessed February 14th, 2017

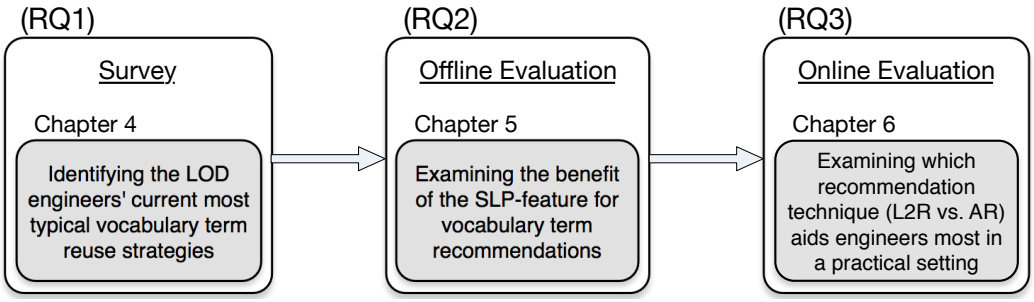


Figure 1.4: **Overview of the Contributions.** The Chapters 4, 5, and 6 provide the main contribution for examining research questions (RQ1) to (RQ3) and thereby for answering the main research question. Chapters 2 (Fundamentals) and 3 (Related Work) provide the basis for the contribution.

when using recommendations based on Learning To Rank. The differences are significant among all pairs. The same applies for the recommendation acceptance score and the quality of the resulting LOD representation. Both were also significantly higher for the AR-based recommendations. In addition, most participants (19 out of 20) rated the AR-based recommendations as more valuable and stated that AR-based recommendations helped them much more in reusing RDF vocabulary terms compared to the L2R-based recommendations.

Summary

This thesis comprises the description and the evaluation of the novel RDF vocabulary term recommendation approach *TermPicker*. Figure 1.4 depicts an overview of the three research questions (RQ1) to (RQ3) and the corresponding experiments. The experiments investigate the benefit of TermPicker for LOD engineers when using the approach to reuse vocabulary terms.

1.3 Publications

This work is based on four publications. The experiments and their raw results for three out the four publications have been published via *GitHub*[22] or the GESIS data repository *datorium*[23]. These publications and the accompanying data are as follows:

(P1) Johann Schaible, Thomas Gottron, Stefan Scheglmann, and Ansgar Scherp: *LOVER: Support for Modeling Data Using Linked Open Vocabularies.* In the 3rd International Workshop on Linked Web Data Management (LWDM) 2013 collocated with the 16th conference on Extending Database Technologies (EDBT). Genova, March 22nd 2013.

[22] `https://github.com/`, last accessed February 14th, 2017

[23] `https://datorium.gesis.org/xmlui/?locale-attribute=en`, last accessed February 14th, 2017

(P2) Johann Schaible, Thomas Gottron, and Ansgar Scherp: *Survey on Common Strategies of Vocabulary Reuse in Linked Open Data Modeling.* In the Proceedings of the 11th Extended Semantic Web Conference (ESWC) 2014. Crete, May 25th - 29th 2014.

- The survey and obtained data in SPSS format can be found under the following DOI: `http://dx.doi.org/10.7802/64`

(P3) Johann Schaible, Thomas Gottron, and Ansgar Scherp: *TermPicker: Enabling the Reuse of Vocabulary Terms by Exploiting Data from the Linked Open Data Cloud.* In the Proceedings of the 13th Extended Semantic Web Conference (ESWC) 2016. Crete, May 29th – June 2nd 2016.

- The evaluation data sets and the results are available at: `https://github.com/WanjaSchaible/l2r_eval_material`, last accessed February 14th, 2017.

(P4) Johann Schaible, Pedro Szekely, and Ansgar Scherp: *Comparing Vocabulary Term Recommendations using Association Rules and Learning To Rank: A User Study.* In the Proceedings of the 13th Extended Semantic Web Conference (ESWC) 2016. Crete, May 29th - June 2nd 2016.

- The user study material including the modeling tasks and results are available at: `https://github.com/WanjaSchaible/termpicker_karmaeval_material`, last accessed February 14th, 2017.
- The accompanying survey and the obtained results can be found under the following DOI: `http://dx.doi.org/10.7802/1206`.

1.4 Outline of the Thesis

In this chapter, an overview of the research questions and conducted experiments was shown in Figure 1.4. Before describing its contents in more detail, Chapter 2 introduces the fundamentals of this work. This includes detailed descriptions of RDF vocabularies, Linked Open Data, recommendation systems, and the evaluation of recommendation systems.

Subsequently, Chapter 3 comprises the related work to this thesis. First, currently used vocabulary term search and recommendation services as well as their basic components are discussed and compared to TermPicker. Second, existing concepts that are used to induce the schema from LOD sources are described and compared to SLPs utilized by TermPicker.

Chapter 4 outlines the survey that was performed to investigate the LOD engineers' most typical strategies to reuse RDF classes and properties. It contains the survey description including the ranking and rating tasks that the participants were asked to complete, as well as its results and the discussion of the results.

The evaluation of the SLP-feature is conducted in Chapter 5. First, the schema-level patterns as well as the utilized features, including the SLP-feature, are formalized and explained. Subsequently, the evaluation setup is illustrated. This comprises the data that was used and how the recommendations based on different sets of features are compared to

each other. Eight different L2R algorithms are evaluated and their results are shown. The three best performing L2R algorithms are discussed in more detail. Additionally, the best performing L2R algorithm is compared to the AR-based recommendation method.

Chapter 6 describes the user study, in which participants have to model three data sets from different domains as LOD. First, the need for a user study is indicated. Second, the data and the single steps of the three modeling tasks are explained. Subsequently, it is illustrated how the different recommendation methods are integrated into the data modeling tool Karma. Finally, the results of the user study are presented and thoroughly discussed.

This work is concluded in Chapter 7. It comprises the lessons learned with vocabulary term reuse based on term recommendations as well as with the design of the experiments to evaluate TermPicker's prediction accuracy. Finally, an outlook to future work is provided.

2 Fundamentals and Terminology

Section 2.1 begins by offering a short overview on Resource Description Framework vocabularies. Subsequently, Section 2.2 describes the best practice that a Linked Open Data engineer should reuse vocabularies before inventing new ones. Detailed information on recommender systems is outlined in Section 2.3. Finally, Section 2.4 illustrates how recommender systems are evaluated as well as how to draw reliable conclusions from the evaluations.

2.1 Ressource Description Framework (RDF) Vocabularies

The Resource Description Framework (RDF) is a framework for representing structured data on the Web [54, 44]. This graph-based data model is used for making statements about data entities, called *resources*, and their relationships. In other words, RDF essentially enables engineers to define the semantics of resources and their relationships. To this end, engineers encode the information in *RDF triples* using *RDF vocabulary terms*.

RDF Triples

An RDF triple, consisting of a *subject*, a *predicate*, and an *object* can be expressed as follows:

<subject> <predicate> <object>. (2.1)

Such a triple pattern makes it possible to encode data in a way that resembles a simple sentence structure (the period at the end of a triple indicates the triple's end). For example, one could encode a statement that "The Godfather" was directed by Francis Ford Coppola in the following manner:

subject: The Godfather (2.2)

predicate: is directed by (2.3)

object: Francis Ford Coppola (2.4)

RDF triple: The Godfather is directed by Francis Ford Coppola. (2.5)

Typically, the subject of a triple is a resource that is identified by a HTTP URI (indicated by brackets "<" and ">"). The object can be a resource as well, but it can also be a literal such as a string, an integer, or a date value.[24] The predicate specifies how the subject and the

[24] Technically, the subject and the object can also be *blank nodes*, but as differentiating them is irrelevant for this thesis, they are not explained in detail. The interested reader is directed to the following link for detailed information on blank nodes provided by the W3C: `https://www.w3.org/TR/rdf11-mt/#blank-nodes`, last accessed September 15th, 2016

object are related and is also identified by an URI. Continuing from the previous example, the subject can be represented with http://dbpedia.org/resource/The_Godfather, as did by DBpedia. The same applies for the predicate and the object which results in the following representation:

subject: http://dbpedia.org/resource/The_Godfather (2.6)

predicate: http://dbpedia.org/ontology/director (2.7)

object: http://dbpedia.org/resource/Francis_Ford_Coppola (2.8)

The corresponding RDF triple would therefore be:

<http://dbpedia.org/resource/The_Godfather>
<http://dbpedia.org/ontology/director>
<http://dbpedia.org/resource/Francis_Ford_Coppola> . (2.9)

RDF Classes and Properties

RDF vocabularies are sets of (unique) well-defined terms that are used to describe information about resources [44]. Such terms are referred to as *vocabulary terms* which are either *classes* (also called RDF types) or *properties* (these are the predicates in an RDF triple). Generally, RDF classes are used to describe the semantic type of a resource, and properties are used to describe the semantic meaning of a relationship between resources or between a resource and a literal. For example, the property http://dbpedia.org/ontology/director in (2.9) describes the following semantic meaning of a relationship: "The subject has the object as director".[25] Of course, such a relationship can lead to the assumption that the object is a person. However, this does not apply for the subject. It could be a movie, a musical, or a play. Even the subject's self-speaking URI in (2.9) does not indicate whether the subject is a movie, a musical, a play, or something else. To represent the semantics of the subject and the object, it is therefore necessary to define their semantic type, each in a separate RDF triple. To this end, the property http://www.w3.org/1999/02/22-rdf-syntax-ns#type from the RDF Schema (RDFS)[26] is used. For example, to specify that the resource http://dbpedia.org/resource/The_Godfather is a film, one can use the class Film from the DBpedia[27] vocabulary.[28] The corresponding RDF triple would be as follows:

<http://dbpedia.org/resource/The_Godfather>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://dbpedia.org/ontology/Film> . (2.10)

In other words, such a triple means that the resource http://dbpedia.org/resource/The_Godfather is an instance of the class http://dbpedia.org/ontology/Film. The same procedure can be applied to the resource http://dbpedia.org/resource/Francis_Ford_Coppola. To specify that it

[25] The semantic meaning is encoded in the rdfs:domain and rdfs:range information of a property (explained on page 18).
[26] https://www.w3.org/TR/rdf-schema/, last accessed February 14th, 2017
[27] http://dbpedia.org/ontology/, last accessed February 14th, 2017
[28] The convention is is to capitalize classes, and begin properties in the lower case

is a person, one can use the Person class from the same DBpedia vocabulary. The corresponding triple is

<http://dbpedia.org/resource/Francis_Ford_Coppola>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://dbpedia.org/ontology/Person> . (2.11)

and specifies that http://dbpedia.org/resource/Francis_Ford_Coppola is an instance of the class http://dbpedia.org/ontology/Person. This way, RDF vocabulary terms enrich resources and their relationships with semantics. Triples such as those in equation (2.10) and (2.11) define the semantic type of resources, and properties like in equation (2.9) define the semantics of a relationship. Using these triples in one data set makes it possible for a machine to interpret the modeled information in the following manner: "The resource http://dbpedia.org/resource/The_Godfather, which is a film, has the director http://dbpedia.org/resource/Francis_Ford_Coppola, which is a person".

In order to enable a machine to understand which vocabulary term is a class and which term is a property, it is necessary to define this via the http://www.w3.org/1999/02/22-rdf-syntax-ns#type as well. Throughout this work, mentioning *RDF class* or simply *class* refers to an instance of http://www.w3.org/2000/01/rdf-schema#Class[29] and mentioning *RDF property* or simply *property* refers to an instance of http://www.w3.org/1999/02/22-rdf-syntax-ns#Property[30].

RDF Serializations

There are various ways to encode data in RDF [44]. Writing one RDF triple after another, as exemplified when writing the triples in (2.9), (2.10), and (2.11) after one another, is the so-called *N3*[31] notation. Over the years, this notation of RDF triples has evolved into the *N-Triples*[32] serialization that became standard due to its popularity as an exchange format through the RDF Working Group. The W3C has also developed the RDF serialization *Turtle*[33] which allows for expressing RDF triples in an abbreviated form. Turtle makes it possible to use namespace declarations. This allows for using *prefixes*, such that a class or a property does not have to be specified by the entire URI. It is rather specified in the form prefix:suffix, where the prefix defines the vocabulary and a suffix defines the term from that vocabulary. Naturally, such prefixes can also be used for specifying resources. For example, with the definition of the namespace "dbr: <http://dbpedia.org/resource/>", one can simply use dbr:The_Godfather instead of http://dbpedia.org/resource/The_Godfather. Moreover, Turtle gathers a subject's outgoing predicates without repeating the subject (separated by a ";"). This way, RDF encoded data is presented in a more human-readable manner [44]. For example, the Turtle representation of the RDF triples illustrated in (2.9), (2.10), and (2.11) is shown in Listing 2.1. Lines 1 to 3 specify the namespaces, i.e., the prefixes. Line

[29] https://www.w3.org/TR/rdf-schema/#ch_class, last accessed February 14th, 2017
[30] https://www.w3.org/TR/rdf-schema/#ch_property, last accessed February 14th, 2017
[31] https://www.w3.org/TeamSubmission/n3/, last accessed February 14th, 2017
[32] https://www.w3.org/TR/n-triples/, last accessed February 14th, 2017
[33] https://www.w3.org/TR/turtle/, last accessed February 14th, 2017

```
1 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2 @prefix dbr: <http://dbpedia.org/resource/>
3 @prefix dbo: <http://dbpedia.org/ontology/>
4
5 dbr:The_Godfather
6   rdf:type dbo:Film;
7   dbo:director dbr:Francis_Ford_Coppola.
8
9 dbr:Francis_Ford_Coppola
10  rdf:type dbo:Person.
```

Listing 2.1: **RDF Triples (2.9), (2.10), and (2.11) in Turtle Syntax.** The listing contains namespace declarations (lines 1 to 3), the semantic types of resources The_Godfather (line 6) and Francis_Ford_Coppola (line 10), as well as their relationship (line 7).

5 specifies the subject dbr:The_Godfather and line 6 its semantic type. Analogously, lines 9 and 10 comprise the resource dbr:Francis_Ford_Coppola and its semantic type. Line 7 specifies the connection from dbr:The_Godfather to dbr:Francis_Ford_Coppola via the dbo:director property.

Domain and Range Information of Properties

An accurately defined RDF property has a so-called *domain* and *range* information that specifies which classes it is allowed to connect. Like the rdf:type property, both are themselves properties from the RDFS vocabulary, which is used for modeling other vocabularies. The rdfs:domain property is used to specify that any resource having a given property is an instance of one or more classes. For example, the DBpedia vocabulary encodes the triple

dbo:director rdfs:domain dbo:Film. (2.12)

This means that any resource having the outgoing property dbo:director should be an instance of the class dbo:Film. The rdfs:range property is used to state that the values of a property are instances of one or more classes. Again, the DBpedia vocabulary encodes the triple

dbo:director rdfs:range dbo:Person. (2.13)

This triple states that resources having the incoming property dbo:director should be instances of the class dbo:Person.

Properties with given domain and range information should not be used between classes that are not contained in the domain and range information. Otherwise, third party users are less likely to understand the data [47], or it can lead to false inferences when reasoning over the data [4, 31]. For example, the property foaf:knows has the class foaf:Person for both the domain and range. A second property foaf:maker has the class foaf:Document as domain and foaf:Person as range. Using foaf:knows as property to connect an instance of dbo:Film to an instance of dbo:Person is not only wrong considering the domain and range information, but it also does not make any sense, as a movie cannot know a person. Such a model

would irritate third party users and the data is less likely to be consumed. Using foaf:maker as property to connect an instance of dbo:Film to an instance of dbo:Person would make sense, as a person can be a maker of a movie. However, when reasoning over the data, the machine can understand the triples in a two-fold way: first, an instance of dbo:Film is also an instance of foaf:Document, due to the domain information of the property; second, the reasoner breaks, as it assumes that an instance of dbo:Film cannot also be an instance of foaf:Document. Which of the two possibilities will apply depends on the *assumption* of the logic, i.e., open-world assumption (the former) or closed-world assumption (the latter) [4].

2.2 Reuse of RDF Vocabularies: A Best Practice

Besides the Linked Data principles [11], there is a set of best practices for modeling and publishing data as Linked Open Data [10, 42]. This thesis focuses on the best practice that instructs Linked Data engineers to reuse existing RDF vocabularies before inventing a new one.

Why Reuse RDF vocabularies?

Reusing vocabulary terms is considered important, as it makes it easier for others to consume the published data [47]. In this work, consuming Linked Open Data refers to establishing links between LOD data sets, browsing through a LOD data set manually, as well as querying the data via SPARQL. Establishing such links, i.e., connecting data sets on the LOD cloud, requires bridging between the *schemata* that are used by the different data sources [42]. The schema of a LOD data set defines the mixture of distinct RDF classes and properties from different RDF vocabularies that are used by a data source to model the data as LOD. It is very likely that different data sources use dissimilar vocabulary terms to represent the same resources, e.g., different data sources describe the film "The Godfather" with different vocabulary terms. Thus, in order to connect data sets, browse through a data set, or generate a SPARQL query, it is necessary to be familiar with the utilized schema.

To deal with heterogeneous representations and to facilitate the consumption of data published as LOD, it is customary to reuse RDF classes as well as properties. First, LOD applications provide tailored support for well-known vocabularies, but not for proprietary ones [42]. For example, such support is included in tools like LIMES [67] or SILK [83] that suggest links between resources which are instances of well-known classes. Second, third party users are more familiar with well-known vocabularies, making it easier for them to understand the data and its semantics, if well-known vocabularies are reused. For example, specifying that a resource http://ex.org/FrncsFrdCppl is related to the movie "The Godfather" via the RDF property ex:dctr makes it challenging for others to understand the semantics of the relationship. Furthermore, using self-defined vocabulary terms will likely result in an ill-defined data representation, as the process of developing a vocabulary can be quite complicated [68]. As a result, ill-defined vocabularies, e.g., ill-defined domain and range information of properties, lead to inconsistencies when inferring information based on reasoning [4, 31]. Well-known and widely used vocabularies are likely to be well-defined,

so that an engineer does not have to worry about such consequence when reusing terms from these vocabularies.

Prominent examples of vocabularies that the best practice advocates for reuse [42] are, e.g., the Dublin Core[34] vocabulary providing general metadata attributes such as the title, creator, or date of an bibliographic Resource; the previously introduced Friend of a Friend (FOAF) vocabulary for describing persons; the Good Relations Ontology[35] providing terms to describe products and services; the schema.org[36] vocabulary that has a broad range of terms for resources from various domains, among others. It is worth noting that different communities, naturally, have different preferences regarding the vocabularies they prefer to use. For example, modeling library data utilizes RDF vocabulary terms that represent different types of literature, their genres, authors, and additional metadata [81]. Modeling cultural heritage data, which comprises the legacy of physical objects, traditions, and knowledge of a society inherited from the past, goes beyond library data and requires a larger variety of RDF vocabulary terms in order to define a semantically reich representation of the data [49]. For instance, the vocabularies must include terms for representing descriptive metadata on organizations and groups, as well as on places, time periods, events, and domain terminologies.

Defining a proprietary vocabulary is recommended only if well-known vocabularies (for general metadata or for domain-specific data) are unlikely to provide the needed vocabulary terms for describing the intended semantics [10, 42]. Adhering to this best practice is likely to decrease heterogeneity in data representation on the LOD cloud, thereby increasing the general understanding of which semantics are described by which vocabulary terms. Moreover, additional LOD applications can be designed to provide support for well-known or established vocabularies, facilitating the processing of the published data. For example, ELLIS [38] is a tool for browsing LOD on schema level, i.e., it illustrates graphically how RDF classes within a data set are connected. To this end, ELLIS shows the URIs of the classes and properties at each node. However, it could be easier to read for humans, if it supported well-known vocabularies, such that each term from a well-known vocabulary is displayed via its literal string and not via its URI. This facilitates reading the entire data set.

How to Reuse RDF Vocabularies

There are two general strategies to reuse vocabulary terms [9, 42, 47]. LOD engineers may reuse existing vocabulary terms directly, or they can define proprietary terms and subsequently link these terms to the corresponding ones which already exist, formalized using RDF and standards from the Web Ontology Language (OWL)[37] [46].

Considering the example LOD representation in Listing 2.1, it can be observed that all vocabulary terms used to describe the semantic types and the relationship come from the DBpedia vocabulary. Let us assume this is a proprietary vocabulary. Furthermore, let us

[34] `http://dublincore.org/documents/dcmi-terms/`, last accessed February 14th, 2017
[35] `http://www.heppnetz.de/ontologies/goodrelations/v1`, last accessed February 14th, 2017
[36] `http://schema.org/`, last accessed February 14th, 2017
[37] `https://www.w3.org/TR/owl-ref/`, last accessed February 14th, 2017

```
1 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2 @prefix dbr: <http://dbpedia.org/resource/>
3 @prefix dbo: <http://dbpedia.org/ontology/>
4 @prefix schema: <http://schema.org/>
5 @prefix foaf: <http://xmlns.com/foaf/0.1/>
6
7 dbr:The_Godfather
8   rdf:type schema:Movie;
9   dbo:director dbr:Francis_Ford_Coppola.
10
11 dbr:Francis_Ford_Coppola
12   rdf:type foaf:Person.
```

Listing 2.2: **Example of Reusing RDF Vocabulary Terms Directly.** The classes are exchanged with terms from the schema.org and FOAF vocabularies.

```
1 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2 @prefix dbr: <http://dbpedia.org/resource/>
3 @prefix dbo: <http://dbpedia.org/ontology/>
4 @prefix schema: <http://schema.org/>
5 @prefix foaf: <http://xmlns.com/foaf/0.1/>
6 @prefix owl: <http://www.w3.org/2002/07/owl#>
7
8 dbr:The_Godfather
9   rdf:type dbo:Film;
10  dbo:director dbr:Francis_Ford_Coppola.
11
12 dbr:Francis_Ford_Coppola
13  rdf:type dbo:Person.
14
15 dbo:Film
16  owl:equivalentClass schema:Movie.
17
18 dbo:Person
19  owl:equivalentClass foaf:Person.
```

Listing 2.3: **Example of Interlinking Vocabulary Terms.** The classes dbo:Film and dbo:Person are interlinked with existing terms from well-known vocabularies.

assume that there are existing classes to represent a film and a person, but there is no existing property to represent that a person is a director of a film. Reusing vocabulary terms directly means that LOD engineers should implement terms from other vocabularies directly into the LOD representation. Listing 2.2 illustrates the exchange of the two classes from the DBpedia vocabulary with classes from the schema.org and the FOAF vocabularies. Line 8 now comprises the class schema:Movie to describe the semantic type of the resource dbr:The_Godfather. In line 12, one can observe that the reused class foaf:Person describes the semantic type of the resource dbr:Francis_Ford_Coppola. On the contrary, reusing vocabulary terms by setting links to them from proprietary terms means that both are included in the LOD representation. Listing 2.3 shows that the proprietary terms are not exchanged. They are rather linked to equivalent classes from well-known vocabularies, specified in two additional triples (lines 15 to 19). To this end, OWL provides specific properties, such as

owl:equivalentClass for connecting equivalent classes and owl:equivalentProperty for connecting equivalent properties, respectively. For example, in Listing 2.3 line 9 the class dbo:Film is still used, but in lines 15/16 it is linked to the class schema:Person via the owl:equivalentClass property.

2.3 Recommender Systems

Recommender systems are information filtering techniques designed to predict the rating an *active* user, i.e., the user receiving recommendations, would give to suggested items [70, 50]. Recommender systems are primarily directed towards users who lack sufficient personal experience to evaluate each potentially useful item offered. In other words, recommenders try to predict the most suitable (new) items for the active user.

In some cases recommender systems predict the rating of an item based on the active user's preferences and constraints [70, 50]. These preferences and constraints can be gathered explicitly, e.g., the active user provides a search query or states which item genres are preferred. The preferences and constraints can also be gathered implicitly, e.g., the user rates items that she previously used/selected. Recommenders utilizing such preferences and constraints are defined as *personalized* recommendation systems. For example, if the active user likes movies of the genre "action" (via explicitly stating such in a user profile or via rating previously watched movies), the recommender considers this information and suggests more films of that genre. Recommenders not considering user preferences and constraints are defined as *unpersonalized* recommender systems. Unpersonalized approaches recommend items regardless of the user and the provided query, e.g., the system recommends the current top ten movies to watch. This work focuses on personalized recommender systems, as the set of recommendations depends on a query provided by an LOD eningeer.

To identify useful items, a recommender must predict that an item is worth suggesting. To this end, it uses some information on the active user, or on the recommended items [70, 50]. One kind of information can be gathered by analyzing which items other users with similar/opposite preferences and constraints have chosen. Such an approach is called *collaborative filtering* and the corresponding recommendation system is known as a *collaborative recommendation system* (cf. Section 2.3.1). The key idea is: if the active user consciously agreed in the past with some other users, then items that are unknown to the active user and that are liked by these other users are very likely to be interesting to the active user as well. Another kind of recommender system suggests the active user items that have a similar content to the items that the active user has liked or chosen in the past. Such an approach is called *content-based information filtering* and the corresponding recommender system is known as a *content-based recommendation system* (cf. Section 2.3.2). Other users are not needed for this type of recommendations. The similarity of items is calculated based on features associated with the compared items, such as the genre of a movie. A third option is to compute item recommendations based solely on detailed metadata about the items and the active user's profile/preferences. Such *knowledge-based recommendation systems* (cf. Section 2.3.3) disregard other users, or the active user's previous likes/choices. Basically, only metadata on items such as the resolution of a TV or a car's top speed are used

for calculating the recommendations. In order to compute personalized recommendations, knowledge-based approaches use the active user's query as well as the active user's characteristics, e.g., the user's level of expertise, brand preferences, etc. [33]. Such approaches are typically chosen if users perform a one-time action (or very infrequent actions), e.g., buying digital cameras, TV's, cars, and other occasionally acquired items, such that recommendations cannot rely on previous actions like a purchase history [50].

In the following, these three types of recommendation approaches are illustrated in more detail.

2.3.1 Collaborative Recommender Systems

As mentioned, collaborative filtering approaches suggest to the active user items that other users with similar or opposite tastes have liked or disliked in the past [50, 70, 56]. Usually, the similarity in taste of two users is calculated based on the similarity in the rating history of the users, such as movie or product ratings. The recommendations follow the paradigm: "Other users that liked the same items as you, have also liked: item_1, item_2, ...". For example, if user A and user B liked the same movies, and if user B has liked another movie that user A has not watched yet, then this movie is considered a good recommendation for user A.

One possible way to perform collaborative filtering is to identify so-called *nearest neighbors* and then calculate (heuristic) models to predict relevant items [29]. Some other user can be considered a neighbor to the active user, if the other user has similar preferences and constraints. The more preferences and constraints are shared, the nearer the neighbor. Users that are nearest neighbors to the active user probably share the same interests and are very likely to rate items in a similar way. Thus, the fist step of a collaborative recommender system is to identify the nearest neighbors, followed in the next step by the application of rule-based approaches to specify: "If your nearest neighbors like that item, you might like it as well". To this end, collaborative filtering uses not only explicit ratings of an item, but also implicit feedback [56]. Such feedback includes purchase history, browsing history, search patterns, or even mouse movements. This procedure can also be applied to neighbors that are far away from the active user. If they dislike the items the active user likes, then further disliked items are likely to be favored by the active user and are thereby an appropriate recommendation.

By comparing the users' behavior and not the items' content, collaborative filtering enables so-called "out of the box"-thinking [50]. It means that the active user is able to receive recommendations of items that have different characteristics than other previously liked items. For example, it is possible to suggest a movie to the active user that is very different from the user's usual taste, or a movie that is not well known, if one of the user's nearest neighbors has given it a positive rating. This way, collaborative filtering can produce a wide range of rather unexpected recommendations.

However, collaborative filtering needs ratings or usage statistics from other users in order to find the nearest neighbors of the active usern [70]. Only then can it recommend items that are most liked by the neighbors of the active user. This can be very challenging, as the ratings from other users are usable only, if the profiles of the other users fit the profile of the

active user. In other words, there are not very many nearest neighbors in practice [70, 29]. For instance, even if there are 500.000 user ratings of a movie, only the ratings of users with similar or totally opposite profiles can be used. This reduces the amount of appropriate user ratings to a small fraction, which can have a negative impact on the recommendations. In addition, it can happen that there is nothing known about a recommended item. The only explanation for an item recommendation is that unknown users with similar tastes liked that item.

2.3.2 Content-based Recommender Systems

Content-based recommender systems suggest items to the active user that are similar to those the user has liked in the past [50, 70]. The basic principle consists of matching the preferences and interests of a user with the content attributes of an item, in order to recommend new items that might be interesting to the user.

Specifically, a content-based recommender system analyzes a set of items that were previously used/rated by the active user, and subsequently builds a model or profile of user interests based on the features of the used/rated items [62]. The result is a relevance judgment that represents the user's level of interest in a recommended item. This way, the system is able to either filter search results by deciding whether a user is interested in a specific item, or rank the most interesting items to the top of the results list. For example, the active user likes all films in the "The Godfather" trilogy. The system analyzes the attributes of these three movies and of all other movies, such as the genre of the movie, its actors, director, and others. Subsequently, it recommends the user other movies having attributes similar to the "The Godfather" movies. The recommendations are likely to contain movies such as "Scarface" and "Goodfellas" based on the attributes that they are from the same genre and have actors that also starred in the Godfather movies. Movies such as "The Terminator" are less likely to be recommended, as they share little to no common attribute values with the three "The Godfather" movies.

Content-based information filtering is usually used, when the number of an item's different attributes is relatively high, but the number of other user ratings is low [50, 70]. Even in the domain of movies, music, books, and products, it is common to use content-based recommendation approaches, as it is very likely to happen that the number of nearest neighbors is rather small for collaborative filtering.

The main benefit of content-based information filtering is thereby that it is independent from other users [50, 70, 62]. All previous actions of the active user, e.g., the user's purchase history, are the foundation for finding items that can be of interest to the user. This way, even items that were recently added to a repository, e.g., new products that can be bought, and that do not have any ratings by other users can be recommended to the user. The ability to list all features of items makes the recommended items very transparent, as there is a lot known about the item compared to an item recommended via collaborative filtering [62].

Nevertheless, content-based recommendations depend on the number and the quality of features describing an item [50, 70]. If such features are not sufficiently provided, the recommendations are likely to be less useful. For example, movie recommendations based on solely on the title and the release date of a movie are less likely to be useful compared

to recommendations based on features including the movie's genre, actors, director, and ratings. In addition, content-based information filtering does not enable any "out of the box" thinking, as all recommendations are likely to be similar to the active user's previous choices [62].

2.3.3 Knowledge-based Recommender Systems

Knowledge-based recommendation approaches do not rely on numerous item ratings from other users, nor do they encounter old item preferences by the active user. They rather suggest items by exploiting the active user's requirements and by using a wide-ranging knowledge about the recommended items and their domain [17].

To provide such recommendations, the user requirements can be gathered in a user profile specifying the user's preferences and needs, or directly within a recommendation session as part of the user's query [50, 70]. Recommendations are based either on a *constraint-based* approach [33] or on a case-based approach [17, 14]. Both approaches are similar with respect to the recommendation process, i.e., the active user specifies a set of requirements and the system attempts to find items that meet the requirements. If no items are found, the user must change the requirements, e.g., increase the maximum price that she is willing to spend on an item. The main difference between constraint-based recommendations and case-based recommendations is how they use the requirements and the metadata on the items to calculate the item recommendations. Constraint-based recommender systems use *recommendation rules* that need to be specified by the system's developer. Such rules include statements like "If the customer is interested in printing large pictures, recommendations of high-resolution cameras are preferred" [50]. To this end, it is typical for constraint-based approaches to maintain user profiles, in which the users specify their general interests and needs. On the contrary, content-based recommender systems aim at finding similarities between the active user's requirements and an item's metadata. The approach manages this by comparing the requirements to an item's metadata directly and uses a scoring function to calculate whether or not the item matches the requirements. For example, if a user specifies that she is interested in high-resolution cameras, the recommender will preferably suggest cameras with a high resolution. If a user specifies that she is interested in printing large pictures, the recommender is more likely to suggest high-quality printers instead of high-resolution cameras [50, 17].

Knowledge-based recommendation approaches aim at recommending items infrequently used and/or bought, e.g., cars, houses, apartments [33]. Comparing the active user's rating of a bought house to the other users' rating of their houses is rather insufficient, as one typically buys a house once or twice in a life time. There would not be enough information for a collaborative filtering [17]. The same applies for content-based filtering [50]. Furthermore, even if ratings exist, they are likely to be very old (more than five years old) and most probable no longer useful [33, 17]. Knowledge-based recommendations alleviate this situation and are able to support the user in finding items without a user's history of actions, e.g., purchase history, or comparing ratings with ratings from other users.

However, knowledge-based recommenders have one major downside in that they have to conquer a so-called *knowledge acquisition bottleneck*. Simply put, this means that knowl-

edge on items, i.e., the items' metadata, must first be acquired before it can be used. In many cases this produces a lot of work, as such knowledge has to be gathered by domain experts and often one must also transform it into formal and executable representations [33]. For example, if a new apartment for sale is included into the system such that it can be recommended, all its metadata, like number of square meters, number of rooms, location, and further information, has to be included as well. This can be cumbersome, as it might be necessary to manually incorporate the data and transform it into the desired format.

2.4 Evaluation of Recommender Systems

As mentioned, collaborative, content-based, and knowledge-based filtering methods seek to improve the quality of recommendations. This raises the question how is it possible to measure such an improvement.

Evaluations of recommender systems are classified into two major categories: *offline* and *online* evaluations [70, 50, 77]. Offline evaluations (cf. Section 2.4.1) are performed without actual users whereas online evaluations (cf. Section 2.4.2) examine the benefit of the recommendations by undertaking experiments with users in real-life scenarios. Generally, both the offline and the online evaluations are conducted, as they differ in complexity and needed costs, but also in their results, i.e., recommendation approaches performing better in offline evaluation do not necessarily show an improved performance in online evaluations [8].

When carrying out evaluations based on user data, the primary investigator, i.e., the experimenter who designs and conducts the experiment, must choose between the *between-subjects* and the *within-subjects* design [8, 77] (cf. Section 2.4.3). Both designs differ in the manner how to split the users, i.e., between-subjects designed studies use one group of users per recommender, whereas within-subjects designed studies ask each user to evaluate all recommenders.

When comparing two or more recommendation approaches, the results of offline and online experiments illustrate whether one recommender provides more appropriate suggestions than the other recommenders. To demonstrate that such a difference is not coincidental, the experimenter needs to perform significance tests that illustrate whether the best recommender has a *significantly* higher prediction accuracy than the other examined recommender [86] (cf. Section 2.4.4). If the difference is not significant, all recommenders are considered to have the same prediction accuracy.

2.4.1 Offline Evaluation

In the design phase of recommender systems, the evaluation should be offline, as it does not require any interaction with real users and is thereby the easiest to conduct [77]. Offline evaluations consist of running the different recommendation approaches on the same previously collected data sets, which can also contain user interactions like ratings, and comparing the performance of the approaches [50, 70]. It is possible to simulate the behavior of users that interact with a recommendation system using such data. The assumption

is that the user behavior when the data was collected will be similar enough to the user behavior when the recommender system is deployed. However, care must be taken when choosing data for offline evaluations, as it, e.g., could contain bias towards specific items that can be recommended [50]. The performance of recommendation approaches in offline experiments can be calculated via standard measures of relevance, such as precision and recall. However, using metrics like the Mean Average Precision (MAP) or the Mean Reciprocal Rank (MRR) is known to be more useful, as they also consider the ranking position of a relevant item in the list of all recommendations [77].

2.4.1.1 Data Sets for Offline Experiments

As the goal of offline evaluations is to filter out inappropriate recommendation algorithms, the utilized data should match the data the designer expects the recommender system to face when deployed as closely as possible [70, 50]. It is therefore necessary to choose data that has no obvious bias towards specific items that can be recommended, towards a specific type of users, or towards one recommendation approach being evaluated [50, 77]. For example, in cases where data from an existing system is available, the experimenter may be tempted to pre-filter the data by excluding items or users with low counts, e.g., filter out items that have been rated only a few times. This reduces the costs of experimentation, but it might also introduce a systematic bias in the data. As a result, recommendation approaches that can deal with sparse data do not perform as well as approaches that are designed to deal with rich data. Without pre-filtering the data, it could be the other way around. Therefore, random sampling is generally a preferred method for reducing data and not having any bias towards specific items [50]. Only if random sampling cannot be used for providing meaningful results, it is customary to use predefined sampling or data preparation [77]. For example, if a knowledge-based recommender has its strengths in recommending cars based on a significant amount of metadata on a car, and if such data exists for only 5% of the cars, then random sampling would not provide meaningful results.

It is also crucial to use the same data for each recommendation approach, in order to eliminate every factor that is not explicitly evaluated [70, 50, 77]. This eliminates variables that are not investigated and helps in confirming or rejecting a hypothesis regarding which recommender performs better. Otherwise it is not possible to say whether one recommender outperforms the other, or whether the data was more suitable for one particular recommender.

2.4.1.2 Simulating User Behavior

To evaluate recommendation approaches offline, it is necessary to simulate the online process where the system makes recommendations and the user interacts with the recommendations [70, 50, 77]. Such a simulation of user behavior is usually done by recording historical user data and then hiding some of the user interactions in order to simulate how a user will choose/rate an item. There are many ways to choose the items to be hidden, but to avoid any biases, it is considered best practice to choose the items and their number randomly.

This way, the choice can simulate the target application as closely as possible without the high costs associated with analyzing the current real-world situation.

The preferred way to simulate user behavior, is to assume that the recommender was in place when users made their decision which items to use/select [77]. The data utilized for the offline experiment should contain all decisions made by users. For each query that simulates a user's search for an item, the experimenter randomly chooses one of the user's decisions in favor of an item. Subsequently, the experimenter removes this chosen item from the list of the user's decisions and *hides* it to simulate that the user has not chosen this item before. All recommended items are then compared to the hidden item. If the hidden item is among the recommended ones, one can state that the recommender suggested the item that was used/chosen by the user. The amount of hidden items can be defined by the experimenter, or it can be a different amount for each query. For example, let us assume that a user has bought the following movies in one transaction:

$$\{\text{The Godfather, Scarface, The Godfather Part II, Goodfellas, Blow}\} \tag{2.14}$$

The assignment is to evaluate a movie recommender based on the movies that are already in the user's basket, i.e., "users who have bought these movies, have also bought the following movies:...". The experimenter randomly hides the user interactions of buying the movies "Scarface" and "Blow". The examined recommender receives all movies from the set (2.14) except the movies "Scarface" and "Blow" as input to calculate suggestion for additional movies. The result might be the following ranked list of items:

$$<\text{The Godfather Part III, } \mathbf{Scarface}\text{, Casino, Mafia, Layer Cake, } \mathbf{Blow}\text{, ...}> \tag{2.15}$$

The assumption is that, in the end, the user will be most interested in buying the movies "Scarface" and "Blow", so it is crucial to observe the positions of these two movies in the list of recommendations. The further up the list they appear, the better the recommender's prediction accuracy.

2.4.1.3 Measuring the Prediction Accuracy

Providing recommendations is also referred to as making predictions on further decisions or actions, and the prediction accuracy basically denotes the quality of recommendations [50, 70]. The general assumption in a recommender system is that a system which provides more accurate predictions will be preferred by the user. In order to measure the prediction accuracy of a recommender in an offline experiment, two types of information is required [77]: (i) data that can be used as input for the recommender (e.g. a query provided by the user, a user profile, etc.), and (ii) the information whether a recommended item is relevant or not. Both are provided by using historical data including user interactions (as seen in equations (2.14) and (2.15)).

Evaluating the prediction accuracy is typically done by measuring the *precision* and *recall* of the provided recommendations [63, 5, 45].

$$\text{Precision} = \frac{\text{\#True-Positive}}{\text{\#True-Positive} + \text{\#False-Positive}} \tag{2.16}$$

$$\text{Recall} = \frac{\text{\#True-Positive}}{\text{\#True-Positive} + \text{\#False-Negative}} \tag{2.17}$$

Precision is the fraction of retrieved instances that are relevant, while recall is the fraction of all relevant instances that are retrieved. In other words, they both measure the quality of the calculated recommendations and indicate how many relevant items can be recommended.

However, users are likely to browse only the top-k (where k is a relatively small number such as 5 or 10) results of the list of all recommended items [70]. Precision and recall do not take the ranking position of relevant items into consideration. For example, the list of movie recommendations in equation (2.15) contains two relevant recommendations. The precision is $2/6 = 0.33$ regardless of whether the relevant movies are at the top or the bottom of the recommendation. However, the user is rather concerned about the position of relevant items in the list of recommendations [77]. Therefore, it is typical to use measurements which also consider the ranking position of a recommended item. Furthermore, it is typical that users will not browse the entire list until they find an appropriate term to reuse [77]. Thus, a metric having a fast discount with respect to the position needs to be used. A metric having a fast discount means that the measured prediction accuracy decreases more drastically if relevant items have a lower rank at the top of the recommendations list. For example, there is a large difference in the measured prediction accuracy between a relevant item being at the first and the second position of the list. On the contrary, there is only a little difference in the measured prediction accuracy between a relevant item being at the 99th and the 100th position of the list. Two frequently used measures combining these requirements are the *Mean Average Precision* (MAP) and the *Mean Reciprocal Rank* (MRR) [50, 77]. These two measures are also used in this work to assess the quality TermPicker.

Mean Average Precision (MAP)

When using MAP to evaluate the prediction accuracy of a recommender, the key assumption is that the active user is likely to be interested in multiple items from the list of recommendations. The Mean Average Precision presents the quality of a ranked list of recommendations as an aggregation of the precision of the relevant items [63]. It therefore provides a measure of quality across recall levels. The higher the MAP value, the more relevant vocabulary terms are ranked to the top positions of the recommendation list.

Formally, the set of queries is defined as $Q = \{q_1, ..., q_n\}$, where each query is represented by a list of recommendations. The MAP value for this set of queries Q is basically the arithmetic mean across all queries of the average precision values at different top-k result levels. If the set of relevant items for a query $q_j \in Q$ is $\{r_1, \ldots, r_{m_j}\}$ and R_{jh}

($1 \leq h \leq m_j$) is the set of ranked retrieval results from the top result until one gets to the relevant item r_h, then the Mean Average Precision of Q is defined as

$$\text{MAP}(Q) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} \frac{1}{m_j} \sum_{h=1}^{m_j} \text{Precision}(R_{jh}) \tag{2.18}$$

Calculated MAP scores can, however, vary across information needs when measured within a single system, for instance, between 0.1 and 0.7 [77]. Therefore, the data used for the evaluation must be both large enough and diverse enough to be representative of system effectiveness across different queries.

Mean Reciprocal Rank (MRR)

The Mean Reciprocal Rank to the k-th position (MRR@k) investigates the results list only to the rank position of the first relevant vocabulary term or at maximum to the k-th position [63, 26]. It either returns the reciprocal of the ranking position of the first relevant term, or zero, if no relevant term is contained in the first k results. MRR@k is associated with a user who wishes to see only a single and most relevant item. Naturally, the higher the MRR@k value, the better the recommendation approach.

Formally, given the set of queries $Q = \{q_1, ..., q_n\}$ and the set R_j containing the ranked retrieval results from the top until one gets to the first relevant item for query q_j ($1 \leq q \leq n$), the Mean Reciprocal Rank to the k-th position is defined as

$$\text{MRR}(Q) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} \frac{1}{|R_j|} \quad , \quad |R_j| \leq k \tag{2.19}$$

One can observe that the Mean Reciprocal Rank is a measure that changes its value whenever the first relevant item is moved. It changes rather drastically, if the first relevant item is moved at the top of the results list and only a little, if the first relevant item is moved at the bottom of the list. For example, the change in the MRR value is much larger when the relevant item moves from the first rank (MRR $= 1$) to the second rank (MRR $= \frac{1}{2} = 0.5$) compared to moving the relevant item from rank 100 (MRR $= \frac{1}{100} = 0.01$) to 110 (MRR $= \frac{1}{110} = 0.009$). Therefore, the Mean Reciprocal Rank is of special interesest, as it resembles the fast discount with respect to the position of the first relevant item very well [26]. If the MRR value is MRR $= 0.75$, it means the first relevant item is ranked first in the recommendation list for at least half of all queries.

MAP and MRR Example

Let us recall equations (2.14) and (2.15). The set of movies in equation (2.14) is the input for a recommender and the ranked list of recommendations is denoted by (2.15). To demonstrate the calculation of MAP and MRR, let us take two additional sets of movies with two corresponding lists of recommendations. Recall that the movies in bold font mark

the relevant items, which are extracted and hidden from the set of movies prior to the recommendation process. The second set of movies and its corresponding list of recommendations are

$$\{\text{Star Trek, Star Wars, The Empire Strikes Back, Blade Runner, Matrix}\} \tag{2.20}$$

$$\text{<}\textbf{Matrix}\text{, Inception, Avatar, }\textbf{Star Wars}\text{, Metropolis, Alien, ...>} \tag{2.21}$$

and the third set of movies and its corresponding list of recommendations are

$$\{\text{Jason Bourne, BatmanBegins, The Avengers, Predator, SWAT, }\} \tag{2.22}$$

$$\text{<Mad Max, }\textbf{The Avengers}\text{, }\textbf{SWAT}\text{, Bad Boys, Casino Royale, Gladiator, ...>} \tag{2.23}$$

For an easier overview, let us transform the lists of recommendations for the corresponding queries into vectors. The set of queries Q contains query results q_1 (the set of movie recommendations in equation (2.15)), query q_2 (the set of movie recommendations in (2.21)), and query q_3 (the set of movie recommendations in (2.23)). The vector values are either 0 or 1, where 0 denotes an irrelevant recommendation item and 1 denotes a relevant one.

$$q_1 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}; \qquad q_2 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}; \qquad q_3 = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{2.24}$$

In total, there are three queries, i.e., $|Q| = 3$, such that the Mean Reciprocal Rank is

$$\text{MRR}(Q) = \frac{1}{3}\left(\frac{1}{2} + \frac{1}{1} + \frac{1}{2}\right) = 0.67 \tag{2.25}$$

and the Mean Average Precision is

$$\text{MAP}(Q) = \frac{1}{3}\left(\frac{1}{2}\left(\frac{1}{2} + \frac{2}{6}\right) + \frac{1}{2}\left(\frac{1}{1} + \frac{2}{4}\right) + \frac{1}{2}\left(\frac{1}{2} + \frac{2}{3}\right)\right) = 0.58 \tag{2.26}$$

For all three queries, there were two relevant recommendations out the first six items. However, one can observe that both measures have a fast discount with respect to the position of all items or the first relevant item. Thus, a recommender with a Mean Average Precision and a Mean Reciprocal Rank above 0.7 can be considered to have a very good prediction accuracy [50, 77].

2.4.1.4 Benefits and Drawbacks of Offline Experiments

The main benefit of using an offline evaluation is that many scenarios can be tested, in which users receive recommendations [50]. In addition, most recommendation algorithms include parameters, such as weights, thresholds, the number of neighbors, etc., requiring constant adjustment and calibration [70]. This can be performed automatically in offline evaluations.

The drawback of offline evaluations is that they can answer only a narrow set of questions, typically providing just a metric value about the prediction accuracy of a recommendation approach [77]. No information about how real users might interact with the recommender is elaborated. The results of offline evaluations point towards the assumption that such real users interact with the examined recommender in the same way, as it was simulated using historical data. This is not often the case [77]. Offline evaluations cannot directly measure the recommender's influence on actual user behavior. In other words, obtaining a very good prediction accuracy does not necessarily mean that actual users will be satisfied with the recommendations in a real-world scenario, despite the choice of appropriate evaluation data [77].

2.4.2 Online Evaluation

As recommender systems rely on the interaction with active users, it is quite difficult to create reliable simulations of such user interactions via offline evaluations [50]. Therefore, one needs to evaluate a recommendation approach in such a way that it allows for observing and collecting real user interactions [70]. This is performed via *online evaluations*. Potential users of developed recommenders interact with the recommender in a practical setting, in which they can also provide additional feedback on the recommender's performance. Such online evaluations can be conducted by *explicitly* or *implicitly* collecting data on user interactions.

2.4.2.1 Collecting Implicit Interactions

Implicit collections are observations of users interacting with the recommender system in real-life and not in a controlled space [77]. Typically, users are redirected to alternative recommendation approaches, where the users' interactions with the different systems are recorded. For example, the so-called A/B-Testing splits the set of active users into two groups [72]. One group receives recommendations from the one recommendation approach, and the other group receives recommendations from another recommender. The logs of both groups are analyzed to ascertain which recommendations were preferred, accepted, and used.

When collecting user interactions implicitly, it is important to sample (redirect) users randomly, to ensure that the comparisons between alternatives are fair [50, 77]. Even then it may happen that one recommendation approach gets more experienced users than the others. It is also necessary to single out the different aspects of the recommenders [70]. For example, if the algorithmic accuracy is evaluated, it is important to keep the user interface fixed. Following these requirements, the evaluation results based on implicit interaction collection are the most meaningful, as they illustrate the actual real-life scenario [77]. The main reason is that the users do not know that they are part of an experiment and behave and interact naturally.

Such experiments are, however, quite risky. First, the system in which the recommender approach is evaluated could have not enough users for performing an A/B-Test [50]. With only 10 users per group, the results are not expressive enough to draw any conclusions. Sec-

ond, the utilized recommendation approach as well as the user interface around the recommender must be fully implemented. This could be too time-consuming for a first evaluation with real users [70]. If the user interface or the recommendation approach contain major flaws, it is likely that the recommender's aspects are not singled out and the experiment must then be repeated. Additionally, it must be taken into consideration that a system which provides many irrelevant recommendations, may discourage the users from using the real system ever again. As a result, the risk lies in losing potential users/customers [50, 77].

In conclusion, an experiment collecting interactions implicitly can have a negative effect on the entire system providing the recommendations, and this is unacceptable, especially in commercial applications. For these reasons, it is best to run an implicit online evaluation last [70]. After an extensive offline study has provided evidence that the candidate approaches are reasonable, it is best to collect user interactions explicitly first. This allows for measuring the users' attitude towards the system with less effort, before observing and collecting user interactions implicitly. Such a gradual process reduces the risk of creating significant user dissatisfaction [50, 70].

2.4.2.2 Collecting Explicit Interactions: User Studies

Experiments in which user interactions with a recommender are observed and collected explicitly are called *user studies* [77]. Typically, they are performed in a laboratory setting, which allows for collecting data by explicitly observing and/or questioning users while they interact with the recommender.

A user study is conducted by recruiting a set of participants, i.e., the recommender's potential users, and asking them to perform several tasks in a controlled environment requiring an interaction with the recommendation system [70, 77]. While the users perform the tasks, the principal investigator (the experimenter) observes and records the users' behavior and interactions. This includes collecting any number of quantitative measurements, such as what portion of the task was completed, the accuracy of the task results, the time taken to complete the task, or the recommendation acceptance score which specifies how often a participant selected a recommendation [77]. In many cases, it is also possible to ask qualitative questions before, during, and after the task is completed. Such questions can collect data that is not directly observable, such as the users' satisfaction with the recommendations or whether the users perceived the tasks as easy to complete [70].

The main benefit of user studies is that it is the only setting allowing the experimenter to collect qualitative data that is often crucial for interpreting the quantitative results [77]. Another benefit is that it is a close approximation a possible real-life situation, as it relies on users' feedback (in comparison to offline experiments), while simultaneously having the benefit of lowering the risk of upsetting a user (in comparison to implicitly collection user interactions) [70]. This makes user studies a crucial part of experiments on the quality of recommendations.

However, user studies are expensive to conduct [85]. Collecting a large set of users and asking them to perform a large enough set of tasks is costly in terms of either the number of users, the users' time, or the experimenter's time. The tasks and questions for gaining the quantitative and qualitative results must be well-designed and tested in several iterations.

Otherwise, there is a risk that the differences of the analyzed recommendation approaches is not highlighted, and the experimenter must redo the user study [77]. Therefore, a user study must be restricted to a relatively small set of users and a relatively small set of tasks, and cannot test all possible scenarios. Furthermore, each scenario has to be repeated several times in order to make reliable conclusions, further limiting the range of distinct tasks that can be tested [77].

In conclusion, it is best practice to perform an offline evaluation first (this eliminates poor performing recommendation algorithms) [50, 70, 77]. Subsequently, one can perform a user study (this provides further information of the algorithms that performed best in offline evaluations), before finally performing an evaluation, in which all user interactions are collected implicitly, e.g., A/B-Testing.

2.4.3 Between-Subjects vs. Within-Subjects Experiments

When evaluating a recommender system based on observing and collecting user interaction data, the experimenter can compare the different recommender systems within the same group of users or between different groups of users [77, 21]. Both approaches can be used in offline and online evaluations, but these principles are mostly discussed when conducting a user study.

As a user study typically compares a few candidate recommenders, each candidate must be tested across the same tasks. While it is possible to compare the candidates *between-subjects* (the *subjects* are the participants of the user study and potential users of the recommender) where each participant is assigned to only one candidate method and experiments with it [21], the alternative comparison is *within-subjects*, which compares the candidates by letting each participant test all candidates [39]. Both have benefits and drawbacks, and it is crucial to discuss which one to use before conducting the user study.

2.4.3.1 Between-Subjects Experiments

The between-subjects design is one of the most common experiment types in user studies [21]. The idea is that participants can only be part of one group each investigating a different recommendation approach. For example, one group of participants is asked to use recommender A, another group of participants is asked to use recommender B, and the control group is asked to use no recommender. The members of each group are different, i.e., each participant can be part of one group only. In the end, the experimenter compares the results from one group to the results of the other group.

Between-subjects experiments have the main advantage that only one task is needed to investigate which recommender is most suitable. Compared to within-subjects experiments, they allow for conducting only one assignment that each user has to complete. The experimenter merely uses different recommender approaches for the same task for each group. This avoids so-called *carryover effects* that are a problem in within-subjects designs [21] (cf. Section 2.4.3.2). Such carryover effects occur when users become accustomed to the tasks that are performed for the user study, e.g., they get accustomed to the recommendation approach, or the type of assignment.

However, there are many disadvantages of between-subject experiments. First, they require a large number of participants to generate any useful and analyzable data [77]. Per group there should be between 30 to 40 participants, which can be quite difficult to acquire, if three or more recommendation approaches are evaluated [21]. Second, it is impossible to maintain homogeneity across the groups [77]. The evaluation of recommenders and the results obtained could depend too much on the users and their subtle differences. For example, one group can contain mainly participants with a higher level degree, whereas another group may consist of students. Finally, the investigated recommenders cannot be directly compared, as one group uses only one recommender. It is not possible to ask the participants which recommender they liked more. This is a huge drawback, as much of the potential information estimating what users need is lost.

Between-subject experiments are usually used to test long-term effects of using a specific system [77, 21]. The user is not required to switch systems, which makes it easy for the experimenter to provide such a system. An A/B-Testing is a typical example of such an experiment [72].

2.4.3.2 Within-Subjects Experiments

Contrary to a between-subjects experiment, a within-subjects design allows every single participant of the user study to interact with every investigated recommendation approach [39]. For example, the usefulness of the three recommendation approaches A, B, and C is investigated. Each participant is asked to complete three tasks in which they interact with the recommendation approaches A, B, and C. To complete the initial task, the participants receive recommendations from recommendation approach A. For the second task, they receive recommendation based on approach B and for the third task, recommendations are based on approach C.

In general, within-subjects experiments are more informative, as they eliminate the assumption that a biased split of participants between two (or more) recommendation approaches is responsible for a superiority of one approach [21]. As each participant is able to experience each recommendation approach, it is also possible to ask comparative questions about the approaches, such as the participant's preference towards one specific approach. Another advantage is that within-subjects experiments require fewer participants, making it easier for the experimenter to conduct the user study, i.e., inviting participants to the lab and observing them while they complete the tasks. For example, if four recommenders have to be tested, using four groups each of 30 participants means conducting 120 experiments. Using 30 participants who test all four recommenders means conducting and supervising only 30 experiments, which is much more feasible.

The main disadvantage of within-subjects experiments lies in the carryover effects [39, 77]. The experimenter must use different data for each task, in order to create different sets of recommendations. Otherwise, the participants can memorize the recommended items from the first task, which makes it easier to complete the second and third task. This will improve, e.g., the task completion time, regardless of the examined recommendation approaches. Using data from different domains for each task reduces this learning effect. However, changing the data for each task can introduce a new bias towards one specific data

domain [21]. For example, participants that are experts in one of the domains are likely to complete the task that uses items from that domain faster compared to the other tasks, again regardless of the recommendation approaches. It can also happen the other way around. If one of the chosen domains is too difficult for the participants to understand, they might not recognize relevant item recommendations. Therefore, the experimenter has to make sure that the chosen domain of items "fit" at least the majority of the participants [39, 77]. Furthermore, participants might experience the carryover effect with the utilized *surrounding* [21]. This includes the specification of the tasks, as they are typically the same, as well as the system, in which the examined recommendation approaches are integrated [77]. Participants can feel more confident after the first task, not because they memorize the recommended items, but because they have gained experience with the task's scope and/or the utilized system. The opposite effect appears if the participants need a long time (uninterrupted) to complete tasks [21]. For example, if a user study comprises three tasks, each lasting half an hour, then it is very likely that the participants get tired towards the end. This decreases their performance considering the second and third tasks, which corrupts the results as well.

Despite the disadvantages related to carryover effects, within-subjects experiments are still used more commonly for short-term user studies, as it is quite possible to overcome these disadvantages [39, 21, 77]. To deal with the practice effect, the experimenter should follow the counterbalance design, where the order of the examined recommendation approaches is varied. It is also possible to let the participants get accustomed to the surrounding via a small "warm-up" task. To overcome the problem that the participants might get tired, each task should be limited in the maximum time required. For example, if participants do not complete a task within 5 minutes, the task is considered incomplete and the participants are asked to move on to the next task.

2.4.4 Drawing Reliable Conclusions

For statistical analyses of experiments, such as within-subjects or between-subjects experiments, it is necessary to formulate a *hypothesis* to be examined [86]. Based on the hypothesis, the experimenter defines the *dependent variable*, i.e., the variable that is being measured, and the *independent variable*, i.e., the variable that is being manipulated. For example, the hypothesis states that recommendation approach A has a higher prediction accuracy than approach B. The dependent variable is the prediction accuracy, as it is being measured, and the independent variable is the recommendation approach, as it is being manipulated. A *null hypothesis* states that there is no difference in the prediction accuracy between approach A and approach B. As a result of the statistical analysis, the hypothesis is either rejected, e.g., approach A does not have a higher prediction accuracy than approach B, or accepted, e.g., approach A has indeed a higher prediction accuracy compared to approach B. In the following, the independent variable is referred to as *condition*, e.g., three investigated recommender systems means that there are three conditions.

The choice of a statistical test depends on the nature of the dependent variable [86]. Is the dependent variable some data on a nominal (also called categorical) scale, an ordinal scale, or an interval scale? Nominal scales are used for labeling variables, without any ranking,

Table 2.1: **Non-Parametric Significance Tests.** The table illustrates which non-parametric significance test to use depending on the experiment design and the number of conditions.

Experiment Design	Two Conditions	Three or more Conditions
Between-Subjects	Mann-Whitney U Test	Kruskal-Wallis Test
Within-Subjects	Wilcoxon Signed-Rank Test	Friedman Test

e.g., hair color. Ordinal scales are used to describe the order of the data values, but the differences between each value are not measurable, e.g., a 5-point Likert scale indicating how a person feels from "very unhappy" to "very happy". Interval scales are numeric scales in which not only the order, but also the exact differences between the values is known, e.g., the temperature in Celsius. It is also differentiated whether or not the dependent variable's data has a normal distribution [86]. If the data has a normal distribution, the statistical analysis should be performed via parametric tests. If the data does not have a normal distribution, the experimenter needs to perform a non-parametric test [86, 27].

Jumping ahead with the dependent variables in this work, the data measured has no normal distribution and it is either some ordinal data, such as percentages[38] and 5-point Likert scales, or some interval data, such as time taken to complete some assignment[39]. To investigate whether the differences between two or more conditions are significant for such data, one uses non-parametric significance tests [59, 86]. Table 2.1 illustrates four non-parametric tests. When conducting between-subjects experiments, it is accustomed to use the Kruskal-Wallis [58] test for three or more conditions as well as the Mann-Whitney U test [64] for two conditions [59, 86]. On the contrary, when conducting within-subjects experiments, one should use the Friedman test [37] for three or more conditions and the Wilcoxon Signed-Rank test [84] for two conditions.[40]

[38] Percentages do not fall into either of the categories, but it is accustomed to treat them as ordinal data [86, 27]

[39] Time is typically considered as *ratio* data, but it is customary to use the same statistical tests as for interval data [86, 27]

[40] The interested reader can find a more extensive table showing which statistical analysis to use at: `http://www.ats.ucla.edu/stat/mult_pkg/whatstat/`, last accessed February 14th, 2017

3 Related Work

This Chapter presents the related work focusing on existing services and tools that support an engineer in reusing vocabularies (cf. Section 3.1). In addition, Section 3.2 comprises various approaches that are similar to the schema-level patterns utilized for inducing schema from data sets on the LOD cloud.

3.1 Tools and Services for Finding RDF Vocabulary Terms

Existing Tools and Services aiding data engineers in reusing RDF vocabulary terms for LOD modeling are primarily vocabulary term search engines which enable for a string-based search for RDF classes and properties. Such Services and the ways in which they differ from TermPicker are elaborated in Section 3.1.1. There are also several tools and approaches for recommending RDF vocabulary terms to the LOD engineer, such as CORE [34] or various approaches built in the data modeling tool Karma[41] [55]. These tools and approaches are described in Section 3.1.2 and also distinguished from TermPicker.

3.1.1 RDF Vocabulary Term Search Systems

As mentioned in Section 1.1, it is far from trivial to follow the advice to reuse existing vocabulary terms before defining proprietary ones. To alleviate this situation there are many helpful RDF vocabulary term search services that aid Linked Data engineers in finding appropriate vocabulary terms. The most prominent examples are the Linked Open Vocabulary index (LOV) [82], LODStats [3], and vocab.cc [78]. All these services aid data engineers in finding classes and properties based on a keyword-based approach. Input for these is a string describing the desired vocabulary term, e.g., a search-string "Person" to find vocabulary terms describing a person. The list of results is a set of classes and/or properties that are similar to the search-string based on some string similarity measure, such as the Levenshtein distance [66]. In addition to providing search results on a web page, LOV also offers an API[42] that enables retrieving vocabulary terms by providing a query (e.g. "Person") and various other parameters, such as a *type* (e.g. "class") or even a *tag* specifying a category for a term (e.g. "people").

Each of the services mentioned above also provide further meta-information on vocabulary terms and their vocabularies, such as the term's number of usages on the LOD cloud or which data sets use them. For example, vocab.cc uses data from the Billion Triple Challenge data set 2012 [41] and counts how often a vocabulary term appears in that data set, i.e., it illustrates the number of a term's total usages. LODStats provides similar information, but it

[41] `http://usc-isi-i2.github.io/karma/`, last accessed February 14th, 2017
[42] `http://lov.okfn.org/dataset/lov/api`, last accessed February 14th, 2017

```
1 PREFIX owl: <http://www.w3.org/2002/07/owl#>
2 PREFIX foaf: <http://xmlns.com/foaf/0.1/>
3
4 SELECT DISTINCT ?t {
5   GRAPH ?src{
6     ?t owl:equivalentClass foaf:Person.
7 }} ORDER BY ?t
```

Listing 3.1: **SPARQL Query in LOV.** Querying for RDF classes (`?t`) from all vocabularies/graphs in LOV (`?src`) that are equivalent to the class foaf:Person. This enables exploitation of structural information encoded in the RDF vocabularies

uses data from the Comprehensive Knowledge Archive (CKAN, "The Data Hub")[43] metadata registry to obtain a comprehensive picture of the usage of RDF vocabulary terms. This comprises not only the number of total occurrences of a term but also how many data sets use a term and/or its vocabulary. LOV uses LODStats to provide such kind of metadata as well, while also exploiting information from RDF vocabularies directly. To this end, LOV collects RDF vocabularies in a manually curated repository and validates each vocabulary such that it contains hierarchical structures like sub-class, sub-property, or equivalence relations between vocabulary terms [82]. It also provides an opportunity to use SPARQL queries for exploiting information on vocabularies and their terms.[44] This information represents the specifications of classes as well as properties within the vocabularies. Based on this information Linked Data engineers can search for equivalent classes or properties via owl:equivalentClass or owl:equivalentProperty, for sub-classes and sub-properties via rdfs:subClass or rdfs:subProperty, or for other relations between vocabulary terms which are defined within the vocabularies. Listing 3.1 illustrates an example query to retrieve classes from all vocabularies contained in LOV that are equivalent to foaf:Person. SPARQL queries for selecting RDF types that are a rdfs:subClassOf another RDF type, or properties having a specific domain and range can be designed analogously.

However, engineers must first be aware of the limitation that they cannot always express the needed vocabulary term via keywords. Recall the example from Section 1.1. Finding a relationship that two persons are colleagues at work has several possibilities to specify a search string, like "has colleague", "is colleague of", or "works with". Finding such synonymous strings is likely to be very time-consuming or even error-prone, as the engineer might reuse a more general term, e.g., foaf:knows. Additionally, not many vocabularies contain a large number of explicitly defined relationships between vocabulary terms, especially across vocabularies [48, 82]. This means that many vocabularies do not contain statements specifying equivalent classes or properties via owl:equivalentClass or owl:equivalentProperty, or sub-classes and sub-properties via rdfs:subClass or rdfs:subProperty. The results of SPARQL queries such as in Listing 3.1 however depend on these explicitly defined connections. Equivalent vocabulary terms cannot be retrieved if vocabulary terms are not con-

[43] `https://datahub.io/`, last accessed February 14th, 2017
[44] `http://lov.okfn.org/dataset/lov/sparql`, last accessed February 14th, 2017

nected via links like owl:equivalentClass, rdfs:subClassOf, owl:equivalentProperty, or others.

Comparison to TermPicker

In contrast to the tools and services stated above, TermPicker does not use a keyword-based approach for searching reusable RDF vocabulary terms. It uses a rather structure-based approach. This enables TermPicker to recommend specific vocabulary terms unknown to the engineer based on other usages of the term on the LOD cloud. For example, providing the query-SLP $slp_q = (\{\text{foaf:Person}\}, \varnothing, \{\text{foaf:Person}\})$ as input prompts TermPicker to recommend properties used by other data providers to connect their resources of type foaf:Person. This can include rel:worksWith from the Relationship vocabulary (cf. example in Section 1.1) without using "works with" as a search string. Furthermore, the additional meta information on RDF vocabulary terms provided by LOV, LODStats, and vocab.cc does not comprise how other data sets on the LOD cloud describe their data. For exploiting such schema-information, LOV relies on the information explicitly stated in the vocabularies themselves. TermPicker, however, does not rely on such defined information in vocabularies, but rather induces the schema-information directly from data sets on the LOD cloud. This way, TermPicker is able to retrieve connections between vocabulary terms that are not explicitly stated via owl:equivalentClass, owl:equivalentProperty or others.

3.1.2 RDF Vocabulary Term Recommendation Services

Apart from the services allowing LOD engineers to search for RDF classes and properties, there is a set of recommendation tools and services that suggest engineers vocabulary terms based on other provided vocabulary terms. Some of these recommenders are integrated into Semantic Web search engines or data modeling frameworks, and some others are standalone tools.

Prominent examples of vocabulary term recommendation approaches that are embedded in Semantic Web search engines are Watson [28] as well as Falcons concept and Falcons object search [22, 24]. Falcons concept search recommends additional vocabularies/ontologies once the user selects an RDF class or a property from the results list of an initial keyword-based search. To this end, it uses specific relatedness features between vocabularies [23], in order to identify which terms could be used together to represent some data. Compared to traditional ontology matching approaches, which align ontologies based on *similarity*, the inventors and authors of Falcons use *relatedness* in order to identify which vocabulary terms might express similar semantics. On the contrary, Falcons object search as well as Watson let the user search for specific entities, such as "Barack Obama". Subsequently, it retrieves resources that match the search string from data sets on the LOD cloud. The results also comprise the RDF classes along with outgoing and incoming properties for the retrieved resources. This way, Falcons Object Search and Watson are able to recommend vocabulary terms that others have used to represent specific resources. In addition to

offering the results via a web page, Watson uses the described approach as a plug-in for the NeOn ontology engineering toolkit[45] supporting engineers in reusing vocabularies.

There are also various data modeling tools and services that transform data from various formats, such as CSV data, into RDF, but only a few provide support for reusing vocabulary terms by integrating a vocabulary recommendation service. The "data2Ontology" module of the Datalift platform [75] provides suggestions to match data entities to a vocabulary term based on linguistic proximity between the data entity and the vocabulary term. Among other criteria, the quality of the recommendations use various features from LOV, such as the popularity of a vocabulary and its terms. The data integration tool Karma [55] contains two different types of vocabulary term recommendations. Let us assume, the data to be modeled is represented as a table. One approach suggests so-called *semantic types* for a column, such as suggesting the term foaf:firstName for a column containing first names of persons [57]. The approach analyzes the content of the column using natural language processing (NLP) techniques [6] and recommends an RDF class in conjunction with a datatype property containing the literal value of a column's cell. The other recommendation approach is based on what the Karma user has previously modeled [79]. For example, if she has already modeled data entities and relationship about museum items, and the next data collection contains data on other museum items, the system is likely to recognize this and then recommends the vocabulary terms that were used to model the previous data collection.

Finally, there are the stand-alone vocabulary term recommendation services. The most prominent example is the collaborative system CORE (short for: Collaborative Ontology Reuse and Evaluation) for ontology engineering [34]. A set of initial keywords defines CORE's input. Starting from this, CORE determines a ranked list of domain-specific vocabulary recommendations based on a syntactic match between the input strings and the classes and properties within a vocbulary. The approach also uses WordNet[46] to expand the initial set of terms and performs a search for each of the defined keywords on an index of contained vocabularies. Besides syntactic and semantic similarity measures, CORE uses manual user evaluations of suggested vocabularies to raise the recommendation quality. A similar system was developed by Romero et al. [71]. However, it differs in that it measures the popularity of an ontology by the number of appearances in Wikipedia or bookmarks on Del.icio.us.[47]

Comparison to TermPicker

Watson can only show RDF classes and incoming and outgoing properties for a data entity. The recommendations do not include information about other resources the data entity links to, or from which resources it is linked. Essentially, it is one single node in a graph with outgoing and incoming labeled edges, but no other node at the end of these labeled edges. This makes it hard to nearby impossible to identify how a set of classes is linked to another set of classes. Falcons also incorporates information from vocabularies to illustrate which properties can link to which classes. But Falcons' approach is mainly

[45] `http://www.neon-project.org/`, last accessed February 14th, 2017
[46] `http://wordnet.princeton.edu/`, last accessed February 14th, 2017
[47] `http://delicious.com/`, last accessed February 14th, 2017

designed to establish a general relatedness between vocabularies specifying that different vocabularies contain terms describing similar data. Thus, it does not actually illustrate which classes the data links to, but they illustrate the classes to which a property could link to, based on the calculated relatedness between vocabularies. On the contrary, TermPicker does not make use of any data entities. It rather uses the induced schematic information how other data providers modeled their data. This enables TermPicker to illustrate not only a set of classes and their outgoing properties, but it also shows further classes that can be connected to. It is basically a graph, whereby the nodes are the sets of classes representing a set of resources and labeled edges (representing properties) linking to other nodes. To provide such recommendations, TermPicker's input contains not only vocabulary terms, but it also contains a structure represented via the SLPs. Such a structure is not part of Watson or Falcons queries.

Built-in services like the data2Ontology module for the DataLift platform or Karma's recommenders use either string similarity, analyze the modeled data entities themselves, or rely on data previously modeled by the user. Such services also do not consider what other data providers on the LOD cloud have used to model their data, which is the major difference to TermPicker's recommendation approach.

Finally, stand-alone recommenders like CORE or Romero's approach use a single string or a set of strings specifying a vocabulary term or a domain of interest. As for Watson and Falcons, these services provide recommendations based on string analyses. Unlike TermPicker, they do not exploit any structural information on how vocabulary terms are connected to each other.

3.2 Exploiting Linked Open Data (LOD)

For recommending RDF vocabulary terms based on how other data providers combined RDF classes and properties to represent their data, it is necessary to efficiently induce this information from data published on the LOD cloud. Some approaches induce solely the utilized vocabularies, others even contain information on instance level, i.e., such approaches comprise information on which classes are used to describe resources and which properties are used to interlink resources.

The following describes the most prominent approaches in more detail, whereby they are compared to the schema-level patterns introduced in this work. In addition, when using information on schema level, one might instinctively think of comparing schemata and thereby also of *ontology matching* [32]. Therefore, the utilized schema-level patterns are also differentiated from the concept of ontology matching.

3.2.1 Inducing Schema from Data on the LOD Cloud

The notion of schema-level patterns can be compared to the notion of so-called *triple patterns* [80], which essentially describe which property is in between a certain subject and a certain object. They can also be used to identify the RDF classes of the resources in subject and object position of an RDF triple, leading to the possibility of constructing a tuple that

specifies which RDF type is connected to another type via a specific property. The tool *Loupe*[48] for inspecting and exploring data sets, makes use of these triple patterns to explore the RDF triples in a data set. Such results can also be achieved using a SPARQL query that retrieves the RDF classes of resources as well as the connecting property between resources. However, both Loupe and SPARQL queries contain only a single RDF class for the resource in subject position of a triple, one RDF class for the resource in object position, and one property connecting the resources.

Other prominent approaches to induce schema information from Linked Open Data are Knowledge Patterns (KPs) [69], Statistical Knowledge Patterns (SKPs) [90], or the RDF graph summary [19]. KPs identify *PathElements* between all RDF classes in the data. Statistical Knowledge Patterns (SKPs) find synonymous properties from different RDF classes. And the RDF graph summary describes a Linked Data set via several layers, of which the Node-Collection Layer represents the schema-information. However, KPs do not contain object properties to resources that do not have an RDF class, nor do they contain data type properties to some literal values. SKPs only find synonymous vocabulary terms for a given term and not additional terms that other LOD providers have used together with the given term. Finally, besides the Node-Collection Layer, the graph summary also contains an Entity Layer describing the data on instance level. SPARQL queries over such a graph summary might be too costly, if the information on the RDF instance level is not needed.

On the contrary, the notion of schema-level patterns (SLPs) for inducing schema information from data sets on the LOD cloud have a rather flat structure and do not contain any information from the Entity Layer, thus making them useful for fast queries. This is a huge benefit compared to the SPARQL queries of the RDF graph summary. SLPs also enable recommending data type properties and object properties that do not have an RDF class as rdfs:range encoded in the data. Approaches like the Statistical Knowledge Patterns can rather be used in addition to SLPs to suggest further synonymous terms. Comparing schema-level patterns to Loupe's triples patterns, it was observed that SLPs may include more RDF classes to describe a resource and more than one property to describe a relationship. It is a more condensed form of representation of the triple patterns and makes it easier to understand the data and faster to compute vocabulary terms recommendation. For example, the single SLP

$$slp = (\{\text{foaf:Person}, \text{dbo:SoccerPlayer}\}, \{\text{foaf:knows}, \text{schema:colleague}\}, \{\text{schema:Person}, \text{dbo:Coach}\}) \tag{3.1}$$

is enough to specify that resources types foaf:Person and dbo:SoccerPlayer are connected to resources of types schema:Person and dbo:Coach via the properties foaf:knows and schema:colleague. Based on the calculation of triple patterns, one would need eight triple patterns, i.e., every combination between the RDF class and the two properties, in order to specify the relationship. With each additional vocabulary term, the number of triples patterns needed to represent the relationship rises drastically. This could make it quite difficult to understand the data, and it could make it more costly to calculate recommendations based on triple patterns.

[48] `http://loupe.linkeddata.es/loupe/`, last accessed February 14th, 2017

Finally, tools like LODSight [30] or ELLIS [38] (the latter is based on SLPs) induce the schema from data sets on the LOD cloud and provide visualizations of a data set's schema information. However, they are very useful for exploring relations within and across data sets, but such tools cannot be used to calculatex RDF vocabulary term recommendations.

3.2.2 Ontology Matching and Alignment

Given that TermPicker compares the query-SLP with other SLPs calculated from existing data sets on the LOD cloud in order to calculate vocabulary term recommendations, one might consider TermPicker's approach as being related to ontology matching [32]. However, typical ontology matching techniques attempt to find correspondences between semantically related vocabulary terms of two or more different ontologies by applying (semi-)automatic alignment algorithms. In contrast, SLPs solely represent the connection between resources of specific RDF classes via a set of properties. The comparison of two SLPs is done solely syntactically, i.e., if the two sets of RDF classes and the set of properties of two SLPs contain the same vocabulary terms, these two SLPs are considered the same. Thus, SLPs do not discover and identify any correspondences between semantically related vocabulary terms and are therefore not some type of ontology matching technique, nor can SLPs be directly compared to such.

4 Survey on Common Strategies for Reusing RDF Vocabulary Terms

As mentioned in Section 1, it is advised by the best practices to reuse RDF vocabulary terms when modeling data as Linked Open Data [10, 42]. However, this simple advice is not sufficient, as studies on Linked Data conformance illustrate that different data providers reuse RDF classes and properties in a different, and not at all homogeneous, way [48]. For example, some data providers reuse RDF vocabularies directly and others define their own vocabulary terms and link them explicitly to existing terms. Generally, LOD engineers seem to decide which vocabulary terms to reuse based on their experience and knowledge about vocabularies. Such experience and knowledge lead to various reuse strategies for reusing RDF vocabulary terms. For example, reuse strategies comprise, among others, the reuse of terms from only one vocabulary to increase a clear data structure, or the reuse of terms from popular vocabularies, as many LOD applications provide support for popular vocabularies [42].

This chapter provides more insights on the experience and knowledge of several LOD engineers and illustrates the engineers' typical strategies for reusing RDF vocabulary terms. However, besides reusing "well-known" vocabularies, there are no established recommendations formulated on *how to choose* vocabulary terms to reuse. Such information is gathered via an online survey in which LOD experts and practitioners worldwide are invited to participate. The participants are asked to rank several data representations exemplifying different vocabulary reuse strategies, from *most preferred* to *least preferred* with respect to the reuse of vocabularies. In addition, the participants are asked to answer different questions regarding their preferences when reusing vocabularies.

Feedback was obtained from 79 participants acquired through public mailing lists. The main findings of the survey indicate that reusing vocabularies directly is considered significantly favorable than defining proprietary terms and establishing links on schema level to other vocabulary terms. In addition, a trade-off should be made between reusing popular and domain-specific vocabularies. Additional meta-information on the domain of a vocabulary and on the number of LOD data sets using a vocabulary are considered the most helpful information for deciding which vocabulary to reuse. Overall, the results provide very valuable insights into how data engineers decide which vocabulary terms to reuse when modeling Linked Open Data.

Section 4.1 comprises the design of the survey, covering specification of how vocabulary terms are represented, an explanation how the ranking tasks are conducted, and the tasks' accompanying questions intended to explain the participants' ranking decisions. Demographics on the participants are illustrated in Section 4.3. The outcome of the survey containing the results of the ranking tests as well as the qualitative questions appears in

Section 4.4, followed by discussion of these results in Section 4.5. A summary of the chapter is found in the concluding Section 4.6.

4.1 Survey Design

The survey asks LOD experts and practitioners to share their knowledge and experience in choosing appropriate vocabulary terms for reuse.[49] To gather this information, first *ranking tasks* are employed, whereby the participants are provided several different RDF data models of certain data – the models are the LOD representations of the data – and they are asked to decide which of these models reuses vocabulary terms the best way. The survey then asks the participants to explain their choices, i.e., participants rate different aspects related to why they ranked the models the way they did.

Each model used for the ranking tasks represents a specific vocabulary reuse strategy such as reusing only popular or domain-specific vocabularies. Section 4.1.1 comprises a set of features used to describe a model and its underlying vocabulary reuse strategy. Section 4.1.2 illustrates details on the ranking tasks and qualitative questions.

4.1.1 Features Representing Vocabulary Reuse Strategies

Each model describes the same example data, but with different RDF vocabulary terms. The vocabulary terms differ in their popularity (number of data sets on the LOD cloud that use a term) or whether they are from an already used vocabulary. They can also be self-defined but linked to other classes and properties via the owl:equivalentClass or owl:equivalentProperty property. These differences are defined via specified features that represent each vocabulary term.

Let $\mathbb{V} = \{V_1, V_2, ..., V_n\}$ with $n \in \mathbb{N}$ be the set of all vocabularies used on the LOD cloud. Each vocabulary $V \in \mathbb{V}$ consists of properties and classes such that $V = P_V \cup T_V$. P_V is the set of all properties, and T_V is the set of all classes in vocabulary V. Furthermore, let $\mathbb{DS} = \{DS_1, DS_2, ..., DS_m\}$ with $m \in \mathbb{N}$ be the set of all data sets that are currently published on the LOD cloud. Each $DS \in \mathbb{DS}$ is a tuple $DS = (G, c)$ consisting of a context URI c of DS, where an RDF graph G can be found. G is a set of triples with

$$G = \{(s, p, o) \mid p \in URI, s \in URI, o \in (URI \cup LIT)\} \tag{4.1}$$

where (s, p, o) denotes an RDF triple (cf. Section 2.1 for more details on RDF triples), URI is a set of URI's, and LIT a set of literals.

The function $\phi : \mathbb{DS} \rightarrow \mathcal{P}(W)$ maps each data set on the LOD cloud to the set of vocabularies used by the data set. $\mathcal{P}(W)$ denotes the power set of all vocabularies, and $\phi((G, c))$ (given that $DS = (G, c)$) specifies which vocabularies are used by a data set DS to specify all triples in graph G. The function ϕ is defined with

$$\phi((G, c)) = \{V \mid (\exists\,(s, p, o) \in G : p \in V) \vee (\exists\,(s, \text{rdf:type}, o) \in G : o \in V)\} \tag{4.2}$$

[49] The survey was implemented via the software *QuestBack Unipark* (`http://www.unipark.com/`). It is archived at the GESIS data repository service *datorium* including the raw result data in SPSS format under the following URL: `http://dx.doi.org/10.7802/64`

Accordingly, $|\phi(G, c)|$ is the number of all utilized vocabularies in the data set.

The function $\Phi : W \rightarrow \mathcal{P}(\mathbb{DS})$ specifies which data sets on the LOD cloud use a vocabulary V. Given a vocabulary V, it identifies each data set $DS = (G, c)$ that uses at least one class or property from vocabulary V. It is defined with

$$\Phi(V) = \{(G, c) \mid (\exists\,(s, p, o) \in G : p \in V) \vee (\exists\,(s, \text{rdf:type}, o) \in G : o \in V)\} \quad (4.3)$$

Accordingly, $|\Phi(V)|$ is the number of data sets on the LOD cloud that use vocabulary V.

To identify how often a vocabulary term $v \in V$ has occurred on the LOD cloud, an auxiliary function is first defined with $\psi : (V, \mathbb{DS}) \rightarrow \mathbb{N}$. It calculates the cardinality of the set of all triples $(s, p, o) \in G$ including a vocabulary term $v \in V$. In other words, ψ counts all triples from the graph of one data set (G, c) that contain the given vocabulary term v. It is defined as follows:

$$\psi(v, (G, c)) = |\{(s, p, o) \in G | v = p \vee (v = o \wedge p = \text{rdf:type})\}| \quad (4.4)$$

Using this auxiliary function, the overall occurrences of a vocabulary term v on the LOD cloud can be calculated via summing up the values of $\psi(v, (G, c))$ over all $DS \in \mathbb{DS}$. To this end, the $\Psi : V \rightarrow \mathbb{N}$ is used. It is is defined with

$$\Psi(v) = \sum_{(G,c) \in \mathbb{DS}} \psi(v, (G, c)) \quad (4.5)$$

For the online survey, the values for $|\Phi|$ and Ψ have been retrieved from LODStats [3].

4.1.2 Ranking of Vocabulary Reuse Strategies

To reiterate, the survey consists of several ranking tasks and rating of aspects that specify how much they influenced the ranking decision. The ranking tasks provide several alternative models that show the LOD representations of the data. Each model contains the same data that is described via different vocabulary reuse strategies. The participants are asked to rank the models from *preferred* to *least preferred*. Ranking the models directly instead of the underlying reuse strategies is useful in order to circumvent answers that are simply influenced by the theory of vocabulary reuse [10, 42].

The differences between the models, and thus differences between the varying vocabulary term reuse strategies is illustrated by using the previously defined features $\phi(G, c)$, $|\Phi(V)|$, and $\Psi(v)$. The vocabularies in $\phi(G, c)$ provide information on which vocabularies have been used in a model, e.g., only two domain-specific vocabularies. The values of $|\Phi(V)|$ and $\Psi(v)$ provide information on the popularity of a vocabulary V, and a vocabulary term v, respectively. The modeling examples and thus the underlying reuse strategies are considered as different if there is a difference in the features values of $\phi(G, c)$, $|\Phi(V)|$, and $\Psi(v)$. For example, strategies like *minimize the number of different vocabularies* or *maximize the number of different vocabularies* are reflected by $|\phi(G, c)|$.

Listing 4.1 and Listing 4.2 show examples of two models M_a and M_b that are given the online survey participants for ranking from most to least preferred considering the RDF vocabulary term reuse strategy. Both models describe the same example resources and a

```
1 <http://ex1.org/pub/001>
2   rdf:type swrc:Publication;
3   swrc:title "Title";
4   swrc:author <http://ex1.org/xyz>.
5
6 <http://ex1.org/xyz>
7   rdf:type swrc:Person;
8   swrc:name "xyz".
```

Listing 4.1: **Example Model M_a.** Reusing only one vocabulary to represent the data

```
1 <http://ex1.org/pub/001>
2   rdf:type swrc:Publication;
3   dc:title "Title";
4   dc:creator <http://ex1.org/xyz>.
5
6 <http://ex1.org/xyz>
7   rdf:type foaf:Person;
8   foaf:name "xyz".
```

Listing 4.2: **Example Model M_b.** Reusing popular vocabularies to represent the data

relation between the resources with different classes and properties from different vocabularies.[50] They illustrate the fictive resource http://ex1.org/publ/01 that is of type Publication with the title "Title". It is connected to the other fictive resource http://ex1.org/xyz of type Person, which has also a name, via a property that specifies that http://ex1.org/xyz is the author of http://ex1.org/publ/01. Listing 4.1 uses primarily classes and properties from the Semantic Web for Research Communities ontology (SWRC), which is a known domain-specific vocabulary in the domain of scientific publications. If third party users are unfamiliar with the SWRC vocabulary, they might have to look up the exact semantic meaning of the SWRC vocabulary terms. However, as they have to look up terms from only one vocabulary, it is not too time consuming, given that it simplifies understanding and consuming the data. On the contrary, besides the SWRC vocabulary, Listing 4.2 uses additional terms from various other vocabularies, such as FOAF and Dublin Core (DC). The assumption is that FOAF and DC are well-known and wide-spread vocabularies, and third party users usually do not need to look up the exact semantic meaning of their terms. This too can make it easy for them to understand and consume the data. The main question is then: which of these two possibilities to reuse vocabulary terms is actually the more favorable reuse strategy?

The participants are also provided further information on the vocabularies and the utilized classes and properties. This information is used to *invoke* the participants' knowledge and experience about which model to choose. Table 4.1 illustrates such information for the models M_a and M_b. It contains the models' underlying vocabulary reuse strategy, and their features regarding $|\Phi(V)|$ and $\Psi(v)$. A "✓" specifies that the according vocabulary or vocabulary term is used in the model, whereas a "−" specifies that it is not used.

Table 4.1 illustrates more clearly that model M_a uses only a minimum amount of vocabularies (reuse strategy *minV* with $|\phi(M_a)| = 1$) and M_b reuses primarily popular vocabularies (reuse strategy "*pop*" with $|\phi(M_b)| = 3$). For model M_a it is $\phi(M_a) = \{\text{swrc}\}$, whereas for model M_b it is $\phi(M_b) = \{\text{foaf, dc}\}$. The vocabularies FOAF and DC are considered more popular as SWRC, as the values of $|\Phi(V)|$ are indicating ($|\Phi(\text{foaf})| = 232 > 6 = |\Phi(\text{swrc})|$ and $|\Phi(\text{dc})| = 287 > 6 = |\Phi(\text{swrc})|$). They mean that far more data sets are using FOAF and DC. In addition, M_b also makes use of more popular vocabulary terms than M_a as indicated by the various values of Ψ. Nonetheless, the total numbers of occurrences of the SWRC vocabulary terms such as $\Psi(\text{swrc:title}) = 10,487$,

[50] For improved readability, the prefix declaration of each vocabulary has been removed from the Listings.

Table 4.1: **Further Information on the Models in Listings 4.1 and 4.2.** The table denotes the models' reuse strategy, how many vocabularies are used ($|\phi|$), the number of data sets using a vocabulary (Φ), and the number of total occurrences of a vocabulary term (Ψ).

	M_a	M_b	$\|\Phi(V)\|$	$\Psi(v)$
Reuse Strategy	(*minV*)	(*pop*)		
$\|\phi(M)\|$	1	3		
$V =$ foaf	–	✓	232	
$V =$ dc	–	✓	287	
$V =$ swrc	✓	✓	10	
$v =$ swrc:Publication	✓	✓		30
$v =$ swrc:title	✓	–		10,487
$v =$ swrc:author	✓	–		16,754
$v =$ swrc:Person	✓	–		30,510
$v =$ swrc:name	✓	–		35,756
$v =$ dc:title	–	✓		17,120,348
$v =$ dc:creator	–	✓		7,372,111
$v =$ foaf:Person	–	✓		2,333,589
$v =$ foaf:name	–	✓		3,287,920

are still quite high and thus indicate a highly domain-specific use by a few, but large data sets.

Generally, the participants are provided just the models, without the actual reuse strategy or the statistics regarding the used vocabularies and vocabulary terms. This ensures that they do not select the model that reuses vocabularies considered primarily popular as stated by the commonly accepted best practices. The participants are asked to rank the models specifying their preferred reuse strategy based on their experience with the utilized vocabularies and vocabulary terms. The research question underlying these tasks seeks to answer which vocabulary reuse strategies are considered to be more appropriate in a real-world scenario. Only for one ranking tasks, the participants are provided with the additional information how many data sets use the utilized vocabularies and their classes and properties. It is interesting to investigate whether such further information is able to influence the participants' decisions which model to rank to the top.

4.1.3 Utilized Data for the Ranking Tasks

The survey comprises three ranking tasks, each investigating which reuse strategies the participants prefer. Taken individually, the separate tasks comprises up to four models that represent some given data with different RDF vocabulary terms. The differences between the utilized vocabularies and their terms are represented by the number of utilized vocabularies ($|\phi|$), the number of data sets on the LOD cloud using a vocabulary ($|\Phi(V)|$), and the number of total occurrences of a vocabulary term ($\Psi(v)$). The resources in one task are the same, but they differ across tasks, to counter any potential carryover effects, should participants become familiar with one specific model.

4.1.3.1 Information on the Data Models

The additional (meta)-information on the vocabularies and their terms, i.e., the values of functions $|\phi|$, $|\Phi(V)|$, and $\Psi(v)$, are not provided to the participants. They receive this information solely for the final ranking task. This is done for two reasons: (I) for the first two ranking tasks the goal is to aggregate and condense the participants' knowledge and experience without having these numbers at hand, and (II) the third ranking task investigates how such additional meta-information influences the participants' ranking decision.

The assumption is that participants will rank the model most easily understood by them as the most highly preferred, i.e., list the model at the top of their ranking. This is likely to be due to their knowledge about the utilized vocabulary terms or the preferred manner of modeling some data as LOD.

4.1.3.2 Data Size and Contents

All models within a ranking task describe the same data entities. This allows for comparing solely the utilized vocabulary terms in each model exclusively in order to draw reliable and meaningful conclusions.

However, between the ranking tasks, the models represent different data entities from different domains. This is important to avoid domain-specific bias and carryover effects. For example, if a participants was already familiar with the domain of scientific publishing, it is likely that she could also have a profound knowledge about which vocabulary terms are more suitable. Such a domain-specific bias could affect the results, if each ranking task contained data from the same domain. The participants may also get familiar with the data from the first ranking task and use this newly acquired knowledge to rank the models in the subsequent task. This specifies an unwanted carryover effect, making it difficult to gain insights about the participants' real knowledge and experience.

As the data for each ranking task is different, the ranking tasks are not linked to one another. Each task answers a different research question such as which amount of mixed vocabularies is considered best. This way, it is possible to investigate various aspects of the best practice to reuse RDF vocabularies and identify which reasons drive which decisions.

Only three to four different models per task are used, as only some strategies were important for each task (and its goal), e.g., one ranking task covers reusing vocabularies vs. establishing links. This singles out each investigated aspect, e.g., investigating whether the participants prefer to reuse vocabulary directly, or to establish links to existing terms. Otherwise, it would not be possible to differentiate between different reuse strategies for one specific use case. For example, differentiating between reusing minimal number of vocabularies and minimal number of vocabularies per concept is not important, if it is investigated whether the participants prefer to reuse vocabulary directly, or to establish links to existing terms. It also keeps the survey understandable and manageable for the participants. Ranking tasks containing 10 different models would overwhelm the participants, such that they get tired and perform poorly, or, even worse, simply reject to participate in the survey.

4.1.4 Questions on Influencing Factors for RDF Vocabulary Reuse

The different models and their underlying vocabulary term reuse strategies are based on several aspects of *preference* that have been identified from the state of the art about how to publish Linked Data [10, 42]. In detail, they are:

(A1) provide a clear structure of the data

(A2) make the data easier to be consumed

(A3) establish an ontological agreement in data representation

As part of the survey, participants were provided a questionnaire directing them to specify their aspects of preference why they ranked the models that represent the data in the ranking tasks the way they did. Each participant had to rate these aspects on a 5-point-Likert scale before and after the first two ranking tasks, to investigate whether the aspects have influenced the participant's ranking decision. Besides the insights about the participant's answers, it allows for making a correlation between the user ratings of the aspects and the rankings of the models. For example, if aspect (A1) is significantly considered the most important aspect and the ranking of the strategy which reuses only a minimum number of vocabularies is significantly the best, then this would suggest that in order to provide a clear data structure, one has to minimize the number of reused vocabularies instead of maximizing them.

4.1.5 Gathering Demographic Information

Along with the typical demographic information on the participants, such as gender and age, the profession and the highest academic degree was also collected. More concrete to the field of Linked Open Data, the participants were also asked to provide information on how long they have worked with Linked Open Data, whether it was more publishing or consuming LOD, and how they consider their knowledge level in LOD. For various demographic questions the participants had to indicate their answers on a 5-point-Likert scale from *none at all* to *expert*. Finally, they had to provide the domain of the data they work with on a regular basis.

All this information allows for a proper separation of the results, such as whether or not – and how – the ranking decisions differ regarding the participants' academic degree. Most interesting though is the difference between how experts completed the ranking tasks (and explained their ranking decision) in comparison to LOD practitioners that do not consider themselves experts.

4.2 The Three Ranking Tasks of the Survey

Each of the three ranking tasks covers a different use case of the engineer's decision making process [48, 47]. For example, should the LOD engineer focus on reusing vocabularies directly, or on defining proprietary classes and properties and afterwards establishing links on schema level to external vocabularies, e.g., via owl:equivalentClass or

owl:equivalentProperty? The latter makes it possible to infer the semantics via reasoning [31].

In detail, the different vocabulary reuse strategies investigated are as follows:

- interlink proprietary terms with existing ones (*link*)
- minimize total number of vocabularies (*minV*)
- minimize number of vocabularies per concept (class of a resource) (*minC*)
- confine to domain-specific vocabularies (*minD*)
- reuse popular vocabularies (*pop*)
- maximize number of vocabularies (*max*)

The data representations for each task are illustrated in Turtle[51] syntax. Accordingly, their reuse strategies as well as values considering the features $|\phi|$, $|\Phi(V)|$, and $\Psi(v)$ are illustrated in tables such as Table 4.1.

In the following, the three different tasks and their motivation along with the utilized models are described. The models that are used in the tasks are fictive and prototypical for the different strategies. They do not not contain real data to prevent biased rankings as real-world data might be known to some participants.

4.2.1 Ranking Task T_1: Reuse vs. Interlink

The first ranking task centers on reusing vocabulary terms directly vs. establishing links between proprietary terms to existing ones on schema level. The participants are provided three models that represent the same resources with different vocabulary terms. The three models are shown in Listing 4.3, Listing 4.4, and Listing 4.5. Model M_{1a} is based on the strategy to reuse popular vocabulary terms, such as FOAF and DC, directly. On the contrary, model M_{1c} is based on the strategy to reuse a maximum of different vocabularies directly. Last but not least, model M_{1b} uses the strategy to first define proprietary classes and properties, and subsequently link them to existing terms via the properties owl:equivalentClass and owl:equivalentProperty.

Each model expresses the same example resources using a different strategy, i.e., with dissimilar vocabulary terms form different vocabularies, to represent that an Actor played in a certain Movie. The vocabulary exemplified by the namespace myMov is the proprietary (self-defined) vocabulary. Table 4.2 illustrates the reuse strategies of these models and their features. Model M_{1a} reuses vocabulary terms from the FOAF and Dublin Core vocabularies directly, which seem to be quite popular as indicated by the values $|\Phi(V)|$ and $\Psi(v)$, i.e., it follows the strategy (*pop*). In detail, it has a fair amount of reused vocabularies ($|\phi(M_{1a})| = 2$) and the popularity of the vocabularies ($|\Phi(\text{foaf})| = 232$, $|\Phi(\text{dc})| = 287$) as well as the total occurrence of their vocabulary terms, such as foaf:Person ($\Psi(\text{foaf:Person}) = 18,477,533$) and dc:title ($\Psi(\text{dc:title}) = 3,605,629$) is very high. On

[51] Details on the Turtle serialization can be found in Section 2.1

```
<http://ex1.org/actor/1661/>
  rdf:type foaf:Person;
  foaf:name "Jack Nickolson";
  foaf:made <http://ex1.org/6354/>.

<http://ex1.org/6354/>
  rdf:type myMov:Film;
  dc:title "Batman".
```

Listing (4.3) **Model M_{1a}.** Direct reuse of popular vocabulary terms

```
<http://ex1.org/actor/1661/>
  rdf:type foaf:Person;
  awol:name "Jack Nickolson";
  movie:performance
    <http://ex1.org/6354/>.

<http://ex1.org/6354/>
  rdf:type movie:Film;
  awol:title "Batman".
```

Listing (4.4) **Model M_{1c}.** Direct reuse of terms from unknown vocabularies

```
<http://ex1.org/actor/1661/>
  rdf:type myMov:Actor;
  myMov:name "Jack Nickolson";
  myMov:made <http://ex1.org/6354/>.

<http://ex1.org/6354/>
  rdf:type myMov:Film;
  myMov:title "Batman".

myMov:Actor a rdf:Class;
  rdfs:subClassOf foaf:Person.

myMov:name a rdf:Property;
  owl:equivalentProperty foaf:name.

myMov:made a owl:ObjectProperty;
  owl:equivalentProperty foaf:made.

myMov:title a rdf:Property;
  owl:equivalentProperty dc:title.
```

Listing (4.5) **Model M_{1b}.** Interlink proprietary terms to popular terms

the other hand, model M_{1b} uses a self-defined vocabulary but links its classes and properties to the FOAF and Dublin Core vocabularies via rdfs:subClassOf and owl:equivalentProperty properties. However, with $\Psi(\text{rdfs:subClassOf}) = 12,107$ and $\Psi(\text{owl:equivalentProperty}) = 127$ it can be observed that this strategy, namely strategy (*link*), is not used very often. It is arguable whether M_{1a} or M_{1b} is more likely to achieve the goals provided by the aspects (A1), (A2), and (A3) from Section 4.1.4. Whereas M_{1a} reuses vocabulary terms directly and makes the data easier to read for humans, M_{1b} might be easier to be processed by Linked Data applications. Strategy (*max*), exemplified by M_{1c}, is similar to M_{1a}, but instead of reusing well-known vocabularies it maximizes the number of different vocabularies within one data set by also using the MOVIE[52] and AWOL[53] vocabulary. It is used as a *lower boundary* indicated by $|\Phi(\text{movie})| = 0$ and $|\Phi(\text{awol})| = 0$. The hypothesis is that this strategy will be considered as least preferred option, and that

[52] `http://data.linkedmdb.org/all`, last accessed February 14th, 2017

[53] `http://bblfish.net/work/atom-owl/2006-06-06/`, last accessed February 14th, 2017

Table 4.2: **Ranking Task T_1.** The models $M_{1a} - M_{1c}$, their reuse strategy, and feature values.

	M_{1a}	M_{1b}	M_{1c}	$\lvert\Phi(V)\rvert$	$\Psi(v)$
Reuse Strategy	(*pop*)	(*link*)	(*max*)		
$\lvert\phi(M)\rvert$	2	4	3		
$V =$ foaf	✓	✓	✓	232	
$V =$ dc	✓	✓	–	287	
$V =$ owl	–	✓	–	277	
$V =$ rdfs	–	✓	–	533	
$V =$ awol	–	–	✓	0	
$V =$ movie	–	–	✓	0	
$v =$ foaf:Person	✓	✓	✓		$18,477,53$
$v =$ foaf:name	✓	✓	–		$9,235,251$
$v =$ foaf:made	✓	✓	–		$57,791$
$v =$ dc:title	✓	✓	–		$3,605,629$
$v =$ rdfs:subClassOf	–	✓	–		$12,207$
$v =$ owl:equivalentProperty	–	✓	–		127
$v =$ awol:name	–	–	✓		0
$v =$ movie:performance	–	–	✓		0
$v =$ movie:Film	–	–	✓		$12,494$
$v =$ awol:title	–	–	✓		0

the other two strategies are considered as the significantly more appropriate options with respect to the quality of modeling and publishing Linked Open Data.

In total, the ranking of these models is based on the participants' decision whether it is preferred to reuse vocabulary terms directly or actually use self-defined (i.e. proprietary) terms and establish links on schema level to external vocabulary terms.

4.2.2 Ranking Task T_2: Appropriate Mix of Vocabularies

The second ranking task covers the topic of mixing an appropriate amount of different vocabularies. The participants were provided with the four models M_{2a} , M_{2b}, M_{2c}, and M_{2d}. The feature values as well as the underlying reuse strategies are described in Table 4.3. The participants were asked to decide which of the models that represent the data contains the most appropriate number of different vocabularies.

The models express the same data about a Publication and a Person. The publication includes a title, a creation, and a publication date. Furthermore, the publication resource is connected to a person resource specifying that the person is the author of the publication. The author has a name and working place as properties.

With the SWRC vocabulary, model M_{2a} minimizes the number of different vocabularies, i.e., it reuses only one vocabulary (strategy (*minV*)), which is neither used in very many data set ($\lvert\Phi(\text{swrc})\rvert = 10$) nor do its vocabulary terms occur very frequently; $\Psi(\text{swrc:author}) = 16,754$ is the vocabulary term with the highest number of occurrences in this model. However, it is highly domain-specific and the entire data can be described by

```
<http://ex1.org/lod/publ/001>
  rdf:type swrc:Publication;
  swrc:title "Example Title";
  swrc:creationDate
    "Example Issued Date";
  swrc:startDate
    "Example Available Date";
  swrc:author
    <http://ex1.org/pers/xyz>.

<http://ex1.org/pers/xyz>
  rdf:type swrc:Person;
  swrc:name "xyz";
  swrc:institution
    "Example Institution".
```

Listing (4.6) **Model M_{2a}.** Reuse minimum amount of different vocabularies

```
<http://ex1.org/lod/publ/001>
  rdf:type swrc:Publication;
  dcterms:title "Example Title";
  xfoaf:issueDate
    "Example Issued Date";
  dcterms:available
    "Example Available Date";
  swrc:author
    <http://ex1.org/pers/xyz>.

<http://ex1.org/pers/xyz>
  rdf:type npg:Contributor;
  foaf:name "xyz";
  umbc:institution
    "Example Institution".
```

Listing (4.7) **Model M_{2b}.** Reuse maximum amount of different vocabularies

```
<http://ex1.org/lod/publ/001>
  rdf:type swrc:Publication;
  dcterms:title "Example Title";
  dcterms:issued
    "Example Issued Date";
  dcterms:available
    "Example Available Date";
  dcterms:creator
    <http://ex1.org/pers/xyz>.

<http://ex1.org/pers/xyz>
  rdf:type foaf:Person;
  foaf:name "xyz";
  swrc:institution
    "Example Institution".
```

Listing (4.8) **Model M_{2c}.** Reuse terms from popular vocabularies

```
<http://ex1.org/lod/publ/001>
  rdf:type dcterms:
      BibliographicResource;
  dcterms:title "Example Title";
  dcterms:issued
    "Example Issued Date";
  dcterms:available
    "Example Available Date";
  dcterms:creator
    <http://ex1.org/pers/xyz>.

<http://ex1.org/pers/xyz>
  rdf:type swrc:Person;
  swrc:name "xyz";
  swrc:institution
    "Example Institution".
```

Listing (4.9) **Model M_{2d}.** Reuse minimum amount of different vocabularies per concept

using classes and properties from this vocabulary such as Publication, Person, author and institution.

Model M_{2b} reuses a maximum set of different vocabularies (strategy (*max*)) and is again used as the *lower boundary* in this ranking task. Most vocabularies are not used by many data sets, and with the exception of foaf:name and dcterms:title the total occurrences of the remaining vocabulary terms is also quite low.

Strategy (*pop*), exemplified by M_{2c}, reuses the most popular vocabulary terms and vocabularies as it can be observed in the metrics. The vocabularies FOAF and Dublin core are very popular as the metrics indicate: $|\Phi(\text{foaf})| = 232$, $|\Phi(\text{dc})| = 287$). The total number of their vocabulary terms is very high as well: $\Psi(\text{dc:creator} = 7,372,111)$, $\Psi(\text{foaf:Person}) = 2,333,589)$.

The strategy (*minC*), exemplified by M_{2d}, reuses one vocabulary per concept, i.e., the publication resource is described via the popular Dublin Core vocabulary and the person

Table 4.3: **Ranking Task T_2**: The models $M_{2a} - M_{2d}$, their reuse strategy, and features values

	M_{2a}	M_{2b}	M_{2c}	M_{2d}	$\lvert\Phi(V)\rvert$	$\Psi(v)$
Reuse Strategy	*minV*	*max*	*pop*	*minC*		
$\lvert\phi(M)\rvert$	1	6	3	2		
$V =$ swrc	✓	✓	✓	✓	10	
$V =$ dc	–	✓	✓	✓	287	
$V =$ foaf	–	✓	✓	–	232	
$V =$ xfoaf	–	✓	–	–	0	
$V =$ npg	–	✓	–	–	5	
$V =$ umbc	–	✓	–	–	1	
$v =$ swrc:Publication	✓	✓	✓	–		30
$v =$ swrc:title	✓	–	–	–		10,487
$v =$ swrc:creationDate	✓	–	–	–		0
$v =$ swrc:startdate	✓	–	–	–		0
$v =$ swrc:author	✓	✓	–	–		16,754
$v =$ swrc:Person	✓	–	–	✓		30,510
$v =$ swrc:name	✓	–	–	✓		35,756
$v =$ swrc:institution	✓	–	✓	✓		241
$v =$ dc:title	–	✓	✓	✓		17,120,348
$v =$ xfoaf:issueDate	–	✓	–	–		0
$v =$ dc:available	–	✓	✓	✓		1,308
$v =$ npg:Contributor	–	✓	–	–		0
$v =$ foaf:name	–	✓	✓	–		3,287,920
$v =$ umbc:institution	–	✓	–	–		0
$v =$ dc:issued	–	–	✓	✓		232,329
$v =$ dc:creator	–	–	✓	✓		7,372,111
$v =$ foaf:Person	–	–	✓	–		2,333,589
$v =$ dc:BibliographicResource	–	–	–	✓		0

resource is described via the domain-specific SWRC vocabulary. This strategy is considered quite common, as it provides a good mix between reusing many popular vocabularies and a minimum amount of different vocabularies. First, with SWRC it contains a domain-specific vocabulary and with Dublin Core also a popular vocabulary ($|\Phi(\mathsf{dc})| = 287$ and the Ψ values of DC's terms).

Apart from M_{2b}, every model and their underlying vocabulary reuse strategies in this ranking task is likely to comply with aspects (A1) to (A3) (cf. Section 4.1.4). Reusing a minimum amount of vocabularies might provide a clear data structure, but it might also fail to capture all the semantics of the data. Reusing mainly popular vocabularies might also fail to capture some domain-specific semantics, but it is easy to understand by humans. In such a case, M_{2d} might provide a well defined trade-off between M_{2a} and M_{2c}.

This second ranking task is designed to investigate whether it is preferred to use as few vocabularies as possible or to use several different vocabularies. Reusing as few vocabularies as possible is more likely to increase the readability of the data, but there is also a risk of fitting an entity into a less suitable vocabulary term. Reusing several vocabularies is more

```
<http://ex1.org/artist/artist_01>
  rdf:type mo:MusicArtist;
  rdfs:label "Joe␣Somebody";
  mo:published
    <http://ex1.org/record_01/>;
  mo:homepage <http://www.jsb.org>;
  mo:activity_start "2002-09-24";
  mo:label
    <http://ex1.org/label_xyz/>.

<http://ex1.org/record_01/>
  rdf:type mo:Record;
  rdfs:label "Example␣Record␣Title";
  mo:track_count 20;
  mo:image <http://ex1.org/image01>.
```

Listing (4.10) **Model M_{3a}.** Reuse terms from domain-specific vocabularies

```
<http://ex1.org/artist/artist_01>
  rdf:type schema:Person;
  schema:name "Joe␣Somebody";
  schema:album
    <http://ex1.org/record_01/>;
  schema:url <http://www.jsb.org>;
  schema:foundingDate "2002-09-24";
  schema:acountablePerson
    <http://ex1.org/label_xyz/>.

<http://ex1.org/record_01/>
  rdf:type schema:MusicAlbum;
  schema:name "Example␣Record␣Title";
  schema:numTracks 20;
  schema:image <http://ex1.org/image01
      >.
```

Listing (4.11) **Model M_{3b}.** Reuse a minimum amount of different vocabularies

```
<http://ex1.org/artist/artist_01>
  rdf:type mo:MusicArtist;
  foaf:name "Joe␣Somebody";
  mo:published
    <http://ex1.org/record_01/>;
  foaf:homepage <http://www.jsb.org>;
  mo:activity_start "2002-09-24";
  mo:label
    <http://ex1.org/label_xyz/>.

<http://ex1.org/record_01/>
  rdf:type mo:Record;
  dc:title "Example␣Record␣Title";
  mo:track_count 20;
  foaf:img <http://ex1.org/image01>.
```

Listing (4.12) **Model M_{3c}.** Reuse terms from popular vocabularies

likely to provide better fitting vocabulary terms, but it might also lead to a decrease in the readability of the data.

4.2.3 Ranking Task T_3: Vocabulary Reuse with Additional Information

The third ranking task differs from the other two tasks in that it examines how much the factors, i.e., the feature values from $|\phi|$, $|\Phi|$, and ψ, influence the participants in their decision regarding vocabulary reuse. Here the participants are provided additional information about the vocabularies and vocabulary terms. In other words, this ranking task presents to the participants the information provided by the functions $|\phi|$, $|\Phi|$, and ψ. Additionally, the participants are also asked to rank this given meta-information from *most helpful* to *least helpful*. This tasks investigates whether some information such as a documentation on the semantics of a vocabulary term can aid LOD engineers in their decision.

Table 4.4: **Ranking Task T_3.** The models $M_{3a} - M_{3c}$, their reuse strategy, and feature values

Model	M_{3a}	M_{3b}	M_{3c}	$\|\Phi(V)\|$	$\Psi(v)$
Reuse Strategy	*minD*	*minV*	*pop*		
$\|\phi(M)\|$	3	1	3		
$V =$ foaf	✓	✓	✓	232	
$V =$ mo	✓	–	✓	4	
$V =$ rdfs	✓	–	–	533	
$V =$ schema	–	✓	–	3	
$V =$ dc	–	–	✓	287	
$v =$ mo:MusicArtist	✓	–	✓		1,713,860
$v =$ schema:Person	–	✓	–		375,277
$v =$ rdfs:label	✓	–	–		91,521,315
$v =$ schema:name	–	✓	–		0
$v =$ foaf:name	–	–	✓		9,235,251
$v =$ mo:published	✓	–	✓		0
$v =$ schema:album	–	✓	–		0
$v =$ mo:homepage	✓	–	–		0
$v =$ schema:url	–	✓	–		0
$v =$ foaf:homepage	–	–	✓		8,244,952
$v =$ mo:activity_start	✓	–	✓		0
$v =$ schema:foundingDate	–	✓	–		0
$v =$ mo:label	✓	–	✓		0
$v =$ schema:acountablePerson	–	✓	–		0
$v =$ mo:Record	✓	–	✓		5,770
$v =$ schema:MusicAlbum	–	✓	–		59,248
$v =$ dc:title	–	–	✓		3,605,629
$v =$ mo:track_count	✓	–	✓		0
$v =$ schema:numTracks	–	✓	–		0
$v =$ mo:image	✓	–	–		23,065
$v =$ schema:image	–	✓	–		3
$v =$ foaf:img	–	–	✓		11,004,064

First, the participants are given an initial data model (IM), which represents an example instance of a Music_Artist, who has a specific name and published an Album having a title. The initial model uses three vocabularies $\phi(DS) = \{$foaf, mo, rdfs$\}$, of which the MO[54] vocabulary is a domain-specific vocabulary for representing data on music and musical artists. Subsequently, the participants are provided three more models each extending the IM with added properties such as the artist's homepage, the image of the record's cover, and others. These models are illustrated in Listing 4.10, Listing 4.11, and Listing 4.12. The feature values of the utilized vocabularies and their terms are denoted in Table 4.4.

Some vocabulary terms used in IM were updated with other classes and properties. For example, for one model the vocabulary term foaf:Agent has been updated with the term mo:MusicArtist to describe the type of the musician resource.

Model M_{3a} extends the schema in IM with supplementary properties from the MO ontology, while it also updates the other terms such as foaf:Agent with mo:MusicArtist or

[54] `http://purl.org/ontology/mo/`, last accessed February 14th, 2017

foaf:name with rdfs:label. Thus, the (*minD*) strategy tries to express the data with as few domain-specific vocabularies as possible along with utilizing generic vocabulary terms such as rdfs:label for entities that cannot be expressed with the domain-specific vocabulary. The number of data sets using the MO ontology is not very high ($|\Phi(\text{mo})| = 4$) but the total occurrences of mo:MusicArtist ($\Psi(\text{mo:MusicArtist}) = 1,713,860$) indicates that there is a large data set on musical artists.

The strategy (*minV*), exemplified by M_{3b}, uses only one vocabulary, but the schema.org[55] vocabulary covers a broad range of different domains, including music artists. Therefore, it possesses some specific classes and properties for music artists such as schema:MusicAlbum or schema:album, but also general vocabulary terms such as schema:Person or schema:name to cover general data entities. Thus, it is possible to express the entire data set with this one vocabulary, although it is not really as popular as indicated by the features $|\Phi|$ and Ψ.

Model M_{3c} again follows the strategy to reuse popular vocabularies, such as FOAF and Dublin Core, even though some vocabulary terms are not quite domain specific for describing the data. For example, the property dc:title describes the name of an album, but is not considered specific for the music domain. Thus, such vocabulary terms are not very concise in representing some specific semantics, but their popularity is very high.

The additional meta-information, which is also referred to as "support types", on the provided models M_{3a} to M_{3c} contain the following information:

ST_1 *Domain of a vocabulary:* domain of FOAF is people and relationships; domain of MO is musical work and artists.

ST_2 *Statistics about vocabulary usage:* number of data providers in LOD cloud using FOAF: 500; number of data providers using MO: 50.

ST_3 *Statistics about vocabulary term usage:* number of uses of foaf:homepage: 800; number of uses of mo:homepage: 200.

ST_4 *Semantic information on vocabulary term:* foaf:homepage is used for the web page of a person, while mo:homepage is used for a fan/band page of an artist.

ST_5 *Statistics about vocabulary terms in triple context:* Most common object property between mo:MusicArtist and mo:Record is mo:published.

The data for ST_2, ST_3, and ST_5 is fictive and not retrieved from any real web service, but it represents the numbers calculated by the functions $|\Phi|$ and Ψ.

The assumption is that using the additional information provided by ST_1 to ST_5, participants will rather prefer to use model M_{3a}, as it uses a domain-specific vocabulary which is also quite popular for modeling data about music and musicians.

4.3 Participants

Overall, a total of $N = 79$ participants (16 female) took part in the survey. However, it was possible to skip questions and to withdraw from participating in the survey at any moment.

[55] `http://schema.org/`, last accessed February 14th, 2017

This resulted in $N = 67$ participants who finished the survey (including providing demographic information), but only $N = 59$ participants also answered all questions. About 67% of the 59 participants work in academia, 23% work in industry, and 10% in both. The largest groups of participants are research associates (22), post doctoral researchers (14), and professors (8). On average, the participants were $M = 34.6$ years ($SD = 8.6$) and have worked for 4 years with Linked Open Data ($M = 4.07, SD = 2.64$). They rated their own expertise consuming and publishing LOD quite high on a 5-point-Likert scale from 1 (*none at all experienced*) to 5 (*expert*). Considering consuming Linked Data, the expertise was about $M = 3.67$ ($SD = 0.99$) whereas for publishing Linked Data it was $M = 3.61$ ($SD = 1.1$). About $59, 7\%$ of the participants consider themselves to be high experienced or above (4 or 5 on the Likert-scale) and $40, 3\%$ consider themselves to have moderate knowledge or less. On the whole, the participants are quite experienced in the field of Linked Data. This makes the results of the survey very promising with respect to their validity for identifying the best strategy in choosing appropriate vocabulary terms.

The participants were assembled using the following mailing lists: (a) public LOD mailing list,[56] (b) public Semantic Web mailing list,[57] (c) EuropeanaTech-Community.[58] In addition, various data engineers and data maintainers of LOD data sets on CKAN[59] as well as participants and lecturers from the Summer School for Ontological Engineering and Semantic Web (SSSW[60] were contacted in person and asked to participate in the survey and share their expertise.

4.4 Results of the Survey

The results of the survey show that the majority of participants prefer to reuse vocabulary terms directly, instead of defining proprietary ones and establishing links to existing terms on schema level. It is also observed that the majority of the participants regard the strategies to reuse terms from popular and domain-specific vocabularies as the most favored strategy. This is underpinned by the ranking of the usefulness of the additional meta-information. Especially the domain of a vocabulary as well as the number of LOD data sets using a vocabulary are considered most relevant information.

The following sections illustrate the results of the ranking tasks (cf. Section 4.4.1) and the participants' most influencing reasons why they ranked the models the way they did (cf. Section 4.4.2)

[56] `http://lists.w3.org/Archives/Public/public-lod/2013Apr/0120.html`, last accessed February 14th, 2017

[57] `http://lists.w3.org/Archives/Public/semantic-web/`, last accessed February 14th, 2017

[58] `http://pro.europeana.eu/web/network/europeana-tech`, last accessed February 14th, 2017

[59] `http://datahub.io/group/lodcloud`, last accessed February 14th, 2017

[60] `http://sssw.org/2013/`, last accessed February 14th, 2017

Table 4.5: **Results of the Three Ranking Tasks T_1 to T_3.** The table shows each model per ranking task, the model's strategy, its median rank, as well as the Chi-square and the p-value for each task.

Ranking Task	Model	Strategy	Median Rank	Friedman test
T_1	M_{1a}	*pop*	1	$\chi^2(2, 78) = 11.521, p = .003$
	M_{1b}	*link*	2	
	M_{1c}	*max*	2	
T_2	M_{2a}	*minV*	3	$\chi^2(3, 63) = 40.536, p < .001$
	M_{2b}	*max*	4	
	M_{2c}	*pop*	1	
	M_{2d}	*minC*	2	
T_3	M_{3a}	*minD*	2	$\chi^2(2, 61) = 3.1$, **n.s.**, $p = .211$
	M_{3b}	*minV*	2	
	M_{3c}	*pop*	2	

4.4.1 Results of the Three Ranking Tasks

The ranking position of the models obtained from the ranking tasks are encoded with a score starting at 1, 2, and so on, i.e., the lower the ranking score the better the rank position of a response option, i.e., of a model representing the data. For each ranking task, a Friedman test was performed to detect significant differences between the strategies (with a threshold set to $\alpha = .05$), as the answers are provided on an ordinal scale. Subsequently, pair-wise Wilcoxon signed-rank tests with Bonferroni correction were applied, if significant differences have been detected.

Table 4.5 summarizes the results of all three ranking tasks. Besides the models and their underlying vocabulary reuse strategy, it contains the median ranks for each model and the outcome of the Friedman test. Table 4.5 thereby provides a first insight into how the different models have been perceived by the participants and how they have ranked the models.

4.4.1.1 Ranking Task T_1

The first ranking task, completed by $N = 78$ respondents, evinced a significant difference between the three models with respect to an appropriate reuse of vocabularies, $\chi^2(2, 78) = 11.521$, $p = .003$. The Median (Mdn) ranks show that M_{1a} with the underlying strategy of reusing popular vocabulary terms directly, i.e., strategy *pop*, has a lower ranking score ($Mdn = 1$) than the other two models and their strategy ($Mdn = 2$). Wilcoxon signed-rank tests with a Bonferroni correction applied (with $\alpha' = .017$), show that strategy *pop* is considered as the significantly favorable reuse strategy compared to strategy *link* ($Z = -3.214, p < .001$) and compared to strategy *max* ($Z = -3.197, p < .001$). In detail, the tests showed that strategy *pop* compared to strategy *link* was considered more appropriate by 48 respondents and as worse by 25 respondents (48 respondents ranked the model M_{1a} before model M_{1b} and 25 had these in inverse order). Compared to strategy *max*, strategy *pop* shows similar numbers. 49 participants ranked the model M_{1a} before model M_{1c}

and 24 did it the other way around. The ranking between strategy *link* and strategy *max* was evenly spread. Only 35 respondents considered to rank model M_{1b} before the model lower boundary model M_{1c}, and a total of 34 did the contrary. In the end, there was no significant difference between strategy *max* and strategy *link* ($Z = -0.181$, **n.s.**, $p = .856$).

4.4.1.2 Ranking Task T_2

The second ranking task, which was completed by $N = 63$ respondents, once again shows that the model with the strategy of reusing mainly popular vocabularies, i.e., strategy *pop* for model M_{2c} is ranked first ($Mdn = 1$). A mix between reusing popular and a minimum set of different vocabularies (strategy *minC*) was considered to be second best, whereas reusing only a minimum amount of vocabularies (strategy *micV*) was seen to be the third best option. Model M_{2b} which reuses a maximum amount of different vocabularies and is the lower boundary, was ranked as least preferred option. The Friedman test shows that these differences between the four models are significant ($\chi^2(3, 63) = 40.536, p < .001$). A pair-wise Wilcoxon signed-rank test was conducted with a Bonferroni correction applied (now $\alpha' = .008$). There was no significant difference between strategy *minV* and strategy *minC* ($Z = -0.602$, **n.s.**, $p = .551$) as well as no significant difference between strategy *pop* and strategy *minC* ($Z = -2.292$, **n.s.**, $p = .021$), despite the different median ranks. However, the other comparisons show significant differences. Strategy *pop* is considered better than strategy *minV* ($Z = -2.616, p < .008$) and better than strategy *max* ($Z = -5.632, p < .001$). But, both strategy *minV* and strategy *minC* are significantly preferred compared to strategy *max* ($Z = -3.902, p < .001$, and $Z = -3.926, p < .001$, respectively).

4.4.1.3 Ranking Task T_3

The last ranking task comprises two parts and included a total of $N = 59$ respondents who completed both. In the first part, as shown in Table 4.5, the median ranks for the three models and their strategies are the same (*Mdn = 2*). Indeed, the results of the Friedman test to detect significant differences show that there is no significant difference between the strategies whatsoever ($\chi^2(2, 61) = 3.1$, **n.s.**, $p = .211$).

In the second part, the participants had to rank which support type (the additional meta-information) was most helpful for making their ranking decision. The median ranks for the five support types and whether there was a significant difference detected is displayed in Table 4.6. The results of the Friedman test show that there is a significant difference between the five different types of support ($\chi^2(3, 59) = 36.165, p < .001$). ST_1 and ST_2 are considered more helpful for making the right choice considering vocabulary reuse, whereas ST_3 and ST_4 seems less helpful. ST_5 was considered least helpful. Pair-wise Wilcoxon signed-rank tests with a Bonferroni correction applied (now $\alpha' = .005$) show that ST_1 is not significantly different to ST_2 ($Z = -0.586$, **n.s.**, $p = .564$), but differs significantly from all other support types: ST_1 better than ST_3 ($Z = -3.788, p < .001$), ST_1 better than ST_4 ($Z = -3.493, p < .001$), and ST_1 better than ST_5 ($Z = -4.333, p < .001$). The second support type ST_2 is preferred more compared to ST_3 ($Z = -4.547, p < .001$) and

Table 4.6: **Ranking of the Support Types.** The table shows the ranking of the five support types that were provided to the participants to complete Ranking Task T_3 and the result of the Friedman test.

Support Type	Support	*Mdn*	Friedman test
ST_1	Information on domain of vocabulary	2	$\chi^2(2, 78) = 11.521, p = .003$
ST_2	Number of LOD data sets using a vocabulary	2	
ST_3	Number of all occurrences of a vocabulary term in LOD cloud	3	
ST_4	Documentation of a vocabulary term	3	
ST_5	Information on most common use of an object property	4	

to ST_5 ($Z = -3.804, p < .001$), but not to ST_4 ($Z = -2.555$, **n.s.**, $p = .01$). Finally, there are no significant differences between ST_3 and ST_4 ($Z = -0.581$, **n.s.**, $p = .565$), ST_3 and ST_5 ($Z = -1.289$, **n.s.**, $p = .199$), and ST_4 and ST_5 ($Z = -2.118$, **n.s.**, $p = .034$).

4.4.2 Results of the Main Aspects for RDF Vocabulary Reuse

The participants were asked to evaluate the different aspects (A1) to (A3) (cf. Section 4.1.4) regarding why they ranked the models in the tasks the way they did. They had to provide this information at three points in time, namely at the beginning of the survey and after the first and second ranking task.

Basically, the majority of the respondents rated each aspect quite high. The median rating for the three aspects (A1) provide a clear structure of the data, (A2) make the data easier to be consumed, and (A3) establish an ontological agreement was $Mdn \geq 4$. Applying a Friedman test to measure whether there are significant differences to the second and third rating, shows that in each case, the respondents ranked the three aspects at the beginning of the survey significantly higher than after the two ranking tests ((A1): $\chi^2(2, 63) = 6.881, p = .031$; (A2): $\chi^2(2, 63) = 34.889, p < .001$; (A3): $\chi^2(1, 63) = 6.429, p = .017$). Post hoc analysis with a Bonferroni correction applied (now for (A1) and (A2): $\alpha = .017$), showed that regarding (A1) there is a significant difference between the first and the second rating ($Z = -2.523, p = .011$), as well as between the first and the third rating ($Z = -2.511, p = .011$). However, the second rating was not significantly different to the third one ($Z = -.146$, **n.s.**, $p = .909$). Regarding (A2), post hoc analysis showed the first rating is significantly different to the second ($Z = -3.778, p < .001$) and third rating ($Z = -4.805, p < .001$), though no differences were found between the second and the third rating ($Z = -1.937$, **n.s.**, $p = .065$). The median ratings dropped for both (A1) and (A2) from $Mdn = 5$ to $Mdn = 4$. The aspect (A3) was asked only twice and the post hoc analysis with a Bonferroni correction applied (now $\alpha = .025$) showed that the first rating was significantly higher that the second one ($Z = -2.155$, $p = .032$), despite the fact that the median rating for this aspect is $Mdn = 4$. Furthermore, splitting the ratings into two groups with one group having a level of LOD experience graded as < 4

(moderate and below) and the other group being ≥ 4 (high to expert knowledge), shows that both groups have decreased the ratings of aspects (A1) to (A3). There are no significant differences in the rating before and after the ranking tasks between these two groups.

4.5 Discussion

Section 4.5.1 opens with a discussion of the results, subsequently followed by a discussion of the threats to the validity of the results in Section 4.5.2.

4.5.1 Discussion of the Results

The results of analyzing the most important aspects to reuse vocabulary terms show that most participants, in theory, prefer to publish Linked Open Data in an easy to process way, i.e., provide a clear structure of the data and make it as easy as possible to consume. However, it is very interesting to see that the theoretical intention to follow these best practices ((A1) to (A3)) seems to be higher than the intention to actually follow them in a real-life scenario. This is indicated by the ratings of (A1) to (A3) being high at the beginning ($Mdn = 5$) but not as high after asking the participants whether these aspects influenced their ranking decision ($Mdn = 4$). Nonetheless, each of these aspects was still rated with a median of $Mdn = 4$ on a 5-point-Likert scale, which still shows that these aspects are considered as "somewhat important". Therefore, the participants' goals to provide a clear structure and thereby increase the readability of the data set can be considered as relatively consistent throughout the survey. Furthermore, there were no significant differences between the group of participants who have high to expert level knowledge when set against the group with moderate LOD knowledge and below. This indicates that these goals are very genuine ones.

Having these goals in mind, it is worthwhile to examine more closely the rankings of the three tasks. For **Ranking Task T_1**, the *pop* strategy is the definite preferred choice. Although, this strategy is preferred by the best practices, it is somewhat interesting, as it is also considered important to establish links between vocabulary terms on schema level. However, the *link* strategy was not even favored compared to the *max* strategy (reusing a maximum of vocabulary terms), which was defined as the lower boundary. Furthermore, looking at the quite small total occurrence of properties such as owl:equivalentProperty indicates that other data providers do not follow this good practice either. In fact, looking at the total occurrence of the term owl:sameAs ($|\Phi(\text{owl:sameAs})| = 18,678,552$) indicates that for data providers it is more important to link Linked Open Data on instance level.

In **Ranking Task T_2**, the results showed that reusing well known and widely-used vocabulary terms from widely-used vocabularies is considered as the best option. This time, all strategies were considered significantly more appropriate than the lower boundary of reusing a maximum amount of different vocabularies (the *max* strategy). The *pop* strategy, which reuses popular (in the meaning of widely-used) vocabulary terms, was regarded significantly better compared to the *minV* strategy that used only one domain-specific vocabulary. This is also somewhat noteworthy, as selecting the domain vocabulary first and

using as many of its terms as possible is also considered best practice. In fact, if the data can be described with one domain vocabulary, which can be considered well-known by users working in this domain, it may seem odd to reuse another (popular) vocabulary. Apparently, this was not considered as helpful in providing a clear data structure. Correlating the ranking of the various aspects why vocabularies should be reused and the results of this ranking task, it seems that preferring widely-used vocabulary terms from widely used vocabularies serves the purpose more than reusing mainly the domain-specific vocabulary. Despite this, both of these strategies were not more appropriate than the strategy that uses a minimum amount of vocabularies per concept (*minC*). This *minC* strategy indeed seems to provide a good trade-off between reusing popular and domain-specific vocabularies.

For **Ranking Task T_3**, no significant differences between the strategies were found in the first part of this ranking task. The second part showed that the information on how many data sets use a specific vocabulary and the information on the domain of a vocabulary seem to be the most preferred additional meta-information. The results are interesting in a two-fold way: First, ranking task T_3 was very similar to ranking task T_2. Despite this similarity, the obtained results are very different. The additional information in part one of T_3 states that the MO vocabulary covers the domain of musical artists and their work as well as that the MO vocabulary is used by 50 data sets (fictive number; real number is $|\Phi(\text{MO})| = 3$). This might lead to the belief that the MO vocabulary is a suitable candidate for expressing musical data, as it is used by many other data providers. Therefore, other vocabularies such as FOAF or Dublin Core are not needed, as MO is well-known and widely-used. Second, regarding the different support types (the additional meta-information), it is interesting to observe that the number of data sets using vocabulary V was considered more informative than the number of the total occurrences of vocabulary term $v \in V$. It seems to be preferred though to reuse vocabulary terms from a vocabulary that is used by many, probably smaller data sets, particularly for establishing ontological agreement in data representation. One might be more familiar with these vocabularies and Linked Data applications might have tailored support for these vocabulary terms.

4.5.2 Threat to Validity

The results of the survey might have been influenced by several factors, such as the specific use cases regarding the models, as well as the format in which the models were presented to the participants.

Regarding different use cases, one might primarily use LOD for publishing the data on the web for automated consumption, but one might also define a LOD vocabulary to represent the domain knowledge for an own application. For example, the proprietary class myMov:Actor represents an actor. When modeling Linked Open Data and trying to provide a clear schema structure as well as make the data easier to be consumed, the use of foaf:Person might be adequate. Whereas when defining an ontology, defining the proprietary vocabulary term and specifying a rdfs:subClassOf relationship might be considered more appropriate and more correct. As the actual use case for which the LOD representation is created was not specified, there are several factors that might have influenced the

results in a similar way. However, the survey did not focus on these factors as they are very difficult to grasp in a structured way and to simplify the study.

Considering the format in which the models were presented, i.e. in Turtle syntax, the point could be made that participants did not quite understand the represented Linked Open data. The survey is addressing LOD practitioners who work with Linked Open Data on a regular basis, therefore, the modeling examples appeared in Turtle syntax as this is the most common way of representing data in a relatively easily readable fashion. By using Turtle some participants who might not be comfortable with Turtle syntax could have been excluded, but this was considered the best possible way to present the data in as simple and comprehensible manner on both the instance and schema level.

4.6 Conclusion and Summary of the Chapter

This chapter presented a study investigating which vocabulary reuse strategy is followed most by Linked Open Data experts and practitioners in various real-life scenarios. A survey examined the most influential factors via tasks, in which the participants were asked to rank various modeling examples according to their understanding of appropriate reuse of vocabularies, and rating assignments to explain which aspects most influenced the ranking decisions. The results of the ranking tasks illustrate that reusing vocabulary terms from widely-used as well as domain-specific vocabularies directly is the more favored approach than defining proprietary terms and the to interlink them with external classes and properties. Furthermore, reusing popular vocabulary terms from frequently used vocabularies is more important than frequently used vocabulary terms from vocabularies that are not used by many data providers. To balance vocabulary terms from popular and domain-specific vocabularies, it is considered to be important to maintain an appropriate mix, in order to provide a clear structure of the data and make it easier to be consumed.

These findings can be used for future vocabulary term recommendation systems such as the LOVER approach [73].[61] To this end, the factors that most influence an engineer's decision whether to reuse a vocabulary term or not are derived into *features* that can be used to represent a recommended vocabulary term. Based on such a representation, a recommender system can calculate RDF vocabulary terms for reuse. In detail, the features for representing a vocabulary term are based on the two most influencing reuse strategies, i.e., strategy *pop* for reusing popular vocabularies and *minC* for reusing terms from a vocabulary that has already been used.

[61] LOVER represents the first conceptual ideas of TermPicker

5 Vocabulary Term Recommendations Based on Schema-Level Patterns

As mentioned in Section 3.1, there are tools and services like LOV, Falcons, or CORE that aid LOD engineers in finding RDF vocabulary terms for reuse based on string search. However, none of these services provide information on how existing data providers on the LOD cloud combine the classes and properties from the different vocabularies to model their data. Without such schematic information, using string based search to find vocabulary terms that other data providers have used to describe their data is still time-consuming. The LOD engineer has to identify which data sets use specific vocabulary terms and subsequently examine the data on instance-level manually, i.e., browse through resources of each identified data set.

This chapter provides a detailed description of the novel vocabulary term recommendation approach *TermPicker* and an accompanying offline evaluation. TermPicker enables the reuse of vocabulary terms by exploiting already published data sets on the LOD cloud. This approach subsequently recommends terms that others have used together with the terms the engineer already uses. The prediction accuracy of TermPicker is evaluated via simulates the modeling process of already existing LOD data sets. One or more vocabulary terms from such data sets are randomly hidden, and it is then investigated at which position of the term recommendation list these hidden terms appear.

TermPicker uses schema-level patterns (SLPs) to capture structural information specifying how other data providers modeled their data. An SLP describes the connection between two sets of classes via a set of properties (cf. Section 5.1). For example, the SLP

$$slp_{ex} = (\{\text{mo:SoloMusicArtist}\}, \{\text{foaf:made}\}, \{\text{mo:Record}\}) \tag{5.1}$$

specifies that resources of type mo:SoloMusicArtist (from the Music Ontology[62]) are connected to other resources of type mo:Record via the property foaf:made (from the FOAF vocabulary). TermPicker's input is such an SLP denoting the part of the model the engineer is currently focused on. This SLP is called *query-SLP* and is specified with slp_q. TermPicker uses slp_q and calculates various *features values* for each recommendation candidate (a recommended vocabulary term), such as the popularity of a candidate, or if a recommended term is from an already used vocabulary (cf. Section 5.2). The set of features also includes the so-called *SLP-feature* that computes how often a recommendation candidate is used in other SLPs that are similar to slp_q. These "other" SLPs are calculated from existing data sets on the LOD cloud and form some sort ot "knowledge base". TermPicker's output comprises classes and properties which the engineer can use in a two-fold way: first,

[62] `http://musicontology.com/`, last accessed February 14th, 2017

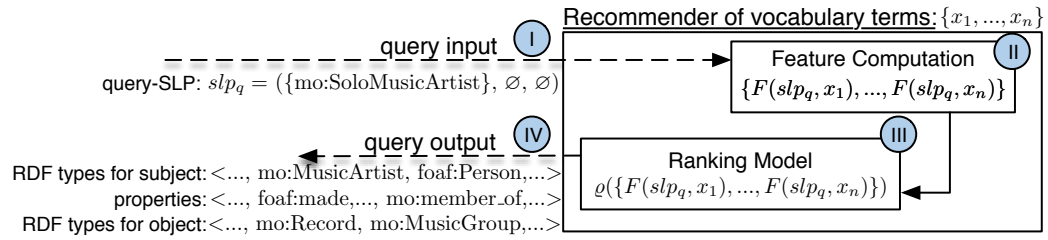


Figure 5.1: **TermPicker Workflow by Example.** The engineer models data as LOD and has already chosen to reuse the class mo:SoloMusicArtist. TermPicker uses the corresponding query-SLP (slp_q) (step (I)) and calculates feature values for each recommendation candidate x_i from the set of all terms published on the LOD cloud ($\{x_1, ..., x_n\}$) (step (II)). The ranking model ϱ uses the feature values $F(slp_q, x_i)$ (step (III)) to provide ranked lists of vocabulary term recommendations to the engineer (step (IV)).

the engineer can extend the query-SLP with additional terms, or second, she can replace existing terms with better fitting ones. Figure 5.1 shows TermPicker's workflow by example: The query-SLP describing the part of the model that the engineer is currently focused on is $slp_q = (\{\text{mo:SoloMusicArtist}\}, \varnothing, \varnothing)$. TermPicker calculates the feature values for each recommendation candidate $x_i \in \{x_1, ..., x_n\}$ based on slp_q and presents the engineer three ranked lists of class and property recommendations.

The ranking of recommendation candidates from most to least appropriate is performed by the ranking model ϱ. In this work, it is calculated based on *Learning To Rank* (L2R) or *Association Rule* (AR) mining (cf. Section 5.3). On one hand, Learning To Rank constitutes a family of supervised machine learning algorithms which establish a ranking over a set of items by observing a general correlation between the utilized features and the relevance of candidates in the training data. On the other hand, Association Rule mining denotes a data mining method for discovering relations between items in large data sets, in this case relations between vocabulary terms, such as which terms are often used together.

TermPicker and the quality of its recommendations is evaluated with the help of a 10-fold leave-one-out evaluation for different situations where engineers need to select a vocabulary term for reuse (cf. Section 5.4). The prediction accuracy is assessed via Mean Average Precision (MAP) and Mean Reciprocal Rank at the first five positions (MRR@5). The evaluation mainly investigates the SLP-feature's impact on the prediction accuracy, but it also examines which L2R algorithm is most suitable for computing the best ranking model. Naturally, the prediction accuracy of L2R-based recommendations are compared to term recommendations computed with Association Rule mining. Results of the evaluation are illustrated in Section 5.5 and then discussed in Section 5.6, directly followed by conclusions and a summary of the main findings to close the chapter.

5.1 Aggregating LOD Schema Using Schema-Level Patterns

The main objective of schema-level patterns is to aid LOD engineers in investigating how other data providers combined RDF vocabulary terms for LOD modeling. SLPs comprise this information based on inducing it directly from data sets published on the LOD cloud. This aids in demonstrating to engineers how resources from such data sets are connected with one another.

Section 5.1.1 provides a formal definition of SLPs and their operations, i.e, adding or removing terms to/from SLPs as well as comparing two SLPs with each other. Subsequently, Section 5.1.2 offers a detailed description how SLPs are computed from data published on the LOD cloud. For a clearer overview of all variables in this section, Table 5.1 enlists the most important ones for defining and calculating SLPs.

5.1.1 Formal Definition of Schema-Level Patterns

Schema-level patterns are tuples specifying which sets of classes are connected via which set of properties. For example, the schema-level pattern

$$slp = (\{\text{foaf:Person}, \text{dbo:ChessPlayer}\}, \{\text{foaf:knows}\}, \{\text{foaf:Person}, \text{dbo:Coach}\}) \quad (5.2)$$

illustrates that resources of types foaf:Person and dbo:ChessPlayer (from the DBpedia ontology[63]) are connected to resources of types foaf:Person and dbo:Coach via the property foaf:knows. In comparison to SLP slp_{ex} in equation (5.1), one can observe that SLPs can also contain several classes (or properties) in one set specifying the types of resources or a relationship.

Formally, as in Section 4.1.1, let $\mathbb{V} = \{V_1, V_2, ..., V_n\}$ be the set of all vocabularies used by data sets on the LOD cloud. Each vocabulary $V \in \mathbb{V}$ consists of vocabulary terms that are either an instance of rdfs:Class or rdfs:Property, such that $V = P_V \cup T_V$, where P_V is the set of all properties p and T_V is the set of all RDF classes t in vocabulary V. Accordingly, $\mathbb{T} = \bigcup_{V \in \mathbb{V}} T_V$ is the set of all RDF classes and $\mathbb{P} = \bigcup_{V \in \mathbb{V}} P_V$ the set of all properties on the LOD cloud. Based on this, the formal definition of an SLP is

$$slp \in \mathcal{P}(\mathbb{T}) \times \mathcal{P}(\mathbb{P}) \times \mathcal{P}(\mathbb{T}) \quad (5.3)$$

where $\mathcal{P}(\mathbb{T})$ is the power set of all classes and $\mathcal{P}(\mathbb{P})$ the power set of all properties on the LOD cloud. Therefore, an SLP is defined as a tuple

$$slp = (sts, ps, ots) \quad (5.4)$$

where $sts \in \mathcal{P}(\mathbb{T})$ is the set of classes describing resources in subject position of an RDF triple, $ots \in \mathcal{P}(\mathbb{T})$ is the set of classes describing resources in object position of a triple, and $ps \in \mathcal{P}(\mathbb{P})$ is the set of properties interlinking these resources.

[63] `http://dbpedia.org/ontology/`, last accessed February 14th, 2017

Table 5.1: **Overview of Utilized Variables.** The table shows an overview of the notation that is used for the formal definition of SLPs and how they are computed.

Variable	Definition
$\mathbb{V}$	Set of all vocabularies on the LOD cloud
$\mathbb{T}$	Set of all RDF types (classes) from all vocabularies in $\mathbb{V}$
$\mathbb{P}$	Set of all properties all vocabularies in $\mathbb{V}$
slp	A schema-level pattern with $slp = (sts, ps, ots)$
sts	Subject type set with $sts \in \mathcal{P}(\mathbb{T})$: RDF types describing a resource in subject position of a triple
ots	Object type set with $ots \in \mathcal{P}(\mathbb{T})$: RDF types describing a resource in object position of a triple
ps	Property set with $ps \in \mathcal{P}(\mathbb{P})$: properties interlinking resources of types in sts and ots
$\mathbb{DS}$	The set of data sets that are published on the LOD cloud
G	A graph representing a data set such that $G \in \mathbb{DS}$
(s, p, o, c)	An RDF quadruple consisting of a subject, property, object, and a context URI where G can be found

Using the two defined operators $\oplus$ and $\ominus$ it is possible to add and remove vocabulary terms to/from SLPs. The commutative $\oplus$ operator combines two SLPs:

$$slp_i \oplus slp_j := (sts_i \cup sts_j, ps_i \cup ps_j, ots_i \cup ots_j) \tag{5.5}$$

It can also be used for extending an SLP with a further vocabulary term by adding it either to the sets sts, ps, or ots. To this end, the operator $\oplus_{sts}$ adds an RDF class to the set sts, operator $\oplus_{ots}$ adds a class to ots, and the operator $\oplus_{ps}$ adds a property to the set of properties ps. This is specifically useful for examining whether a query-SLP is used in combination with a recommendation candidate by other data providers on the LOD cloud. The operation to remove terms from an SLP via the $\ominus$ operator is defined accordingly,

$$slp_i \ominus slp_j := (sts_i \backslash sts_j\ ,\ ps_i \backslash ps_j\ ,\ ots_i \backslash ots_j) \tag{5.6}$$

such that the operator $\ominus_{sts}$ removes a class from the set sts, operator $\ominus_{ots}$ removes a class from ots and the operator $\ominus_{ps}$ removes a property from the set of properties ps. For example, first removing a property from an SLP and subsequently extending the same SLP with a class for resources in object position is demonstrated in the following:

$$slp = (\{\text{foaf:Person}\}, \{\text{dc:date}\}, \varnothing) \ominus_{ps} \text{dc:date} \tag{5.7}$$
$$= (\{\text{foaf:Person}\}, \varnothing, \varnothing) \tag{5.8}$$
$$slp = (\{\text{foaf:Person}\}, \varnothing, \varnothing) \oplus_{ots} \text{foaf:Image} \tag{5.9}$$
$$= (\{\text{foaf:Person}\}, \varnothing, \{\text{foaf:Image}\}) \tag{5.10}$$

To facilitate readability, the operators $\oplus_{sts}$, $\oplus_{ps}$, and $\oplus_{ots}$ are generalized with

$$slp \oplus x := (sts \cup \{x\}) \veebar (ps \cup \{x\}) \veebar (ots \cup \{x\}) \tag{5.11}$$

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
@prefix foaf: <http://xmlns.com/foaf/0.1/>
@prefix dbo: <http://dbpedia.org/ontology/>

<http://ex1.org/sports_001>
rdf:type foaf:Person;
rdf:type dbo:ChessPlayer;
foaf:knows <http://ex1.org/sports_002>.

<http://ex1.org/sports_002>
rdf:type foaf:Person;
rdf:type dbo:Coach.
```

Listing 5.1: **Fictive RDF Triples in Turtle Syntax.** The RDF triples specify that a resource of types Person and ChessPlayer knows a resource of types Person and Coach

that specifies: the SLP slp is extended with a vocabulary term x by adding x either to the set sts, ps, or ots. Accordingly, the operators $\ominus_{sts}$, $\ominus_{ots}$, and $\ominus_{ps}$ are generalized with

$$slp \ominus x := (sts \backslash \{x\}) \veebar (ps \backslash \{x\}) \veebar (ots \backslash \{x\}) \tag{5.12}$$

that specifies: term x is removed either from the set sts, ps, or from ots. The logical operator $\veebar$ implements an exclusive disjunction, such that only one of the three operations can occur.

To compare two or more SLPs, the relationship "$\leq$" (*inclusion*) between two schema-level patterns slp_i and slp_j is defined. It specifies whether an SLP is a subset of, or equal to another SLP. It is denoted as

$$slp_i \leq slp_j, \quad \text{iff} \quad (sts_i \subseteq sts_j) \wedge (ps_i \subseteq ps_j) \wedge (ots_i \subseteq ots_j) \tag{5.13}$$

and specifies that slp_j contains more, or at least as many vocabulary terms as slp_i. This relationship is especially useful for identifying SLPs that can be used to calculate vocabulary term recommendations.

5.1.2 Computing Schema-Level Patterns from Linked Data

Schema-level patterns can be generated in more than one way. They can be created from scratch by adding terms using the $\oplus$ operator, or they can be computed, or induced, from existing Linked Data. For example, Listing 5.1 represents two resources from the DBpedia data set on the LOD cloud, and equation (5.2) shows the SLP that is calculated based on these RDF triples.

In detail, let $\mathbb{DS} = \{G_1, G_2, ..., G_m\}$ be the set of all data sources on the LOD cloud. G denotes the RDF graph of the data source and can be considered as a set of quadruples with

$$\begin{aligned} G = \{(s, p, o, c) \,|\, & s \in \mathit{URI} \cup \mathit{BN},\ p \in \mathit{URI}, \\ & o \in \mathit{URI} \cup \mathit{BN} \cup \mathit{LIT},\ c = \text{contextURI}\} \end{aligned} \tag{5.14}$$

where *URI* is a set of URIs, *BN* a set of blank nodes, *LIT* a set of literals, and the context URI c specifies the location, where the RDF triple can be found. The Function

$$\lambda(G) = \{slp_1, slp_2, ..., slp_k\} \tag{5.15}$$

defines the set of SLPs computed from graph G. Calculating SLPs from all data sets in $\mathbb{DS}$ results in the set $\mathbb{SLP}$. Function $\lambda : \mathbb{DS} \rightarrow \mathbb{SLP}$ is defined as

$$\begin{aligned} \lambda(G) = \{(sts, ps, ots) \mid \exists\, s, o : \\ (\forall\, t_s \in sts : (s, \text{rdf:type}, t_s) \in G) \wedge \\ (\forall\, p \in ps : (s, p, o) \in G) \wedge \\ (\forall\, t_o \in ots : (o, \text{rdf:type}, t_o) \in G)\} \end{aligned} \tag{5.16}$$

The set $\mathbb{SLP}$ is thereby denoted as the joint set of all schema-level patterns that are computed from each graph $G \in \mathbb{DS}$, i.e.,

$$\mathbb{SLP} = \bigcup_{G \in \mathbb{DS}} \lambda(G) \tag{5.17}$$

Preferably the RDF triples used for SLP computation are encoded in N3 or NQUAD syntax (cf. Section 2.1), such that RDF triples can be processed one after another. Iterating over all triples in a data set G, SLPs are computed using two hash maps. One hash map contains all resources and their RDF classes, and the other comprises a pair of resources (the subject along with the object of an RDF triple) and the properties connecting these resources. Listing 5.2 illustrates how the hash maps are used to generate SLPs. The first map, referred to as $classMap$, contains a resource as key and the set of the resource's classes as value. For example, after iterating over the triples in Listing 5.1, $classMap$ includes the entries

$$\big[\,\text{http://ex1.org/sports\_001}\,,\,\{\text{foaf:Person, dbo:ChessPlayer}\}\,\big] \tag{5.18}$$

$$\big[\,\text{http://ex1.org/sports\_002}\,,\,\{\text{foaf:Person, Coach}\}\,\big]. \tag{5.19}$$

The second map, referred to as $propertyMap$, contains a pair of the subject and object resources as key, and the set of all properties between these resources as value. After iterating over the triples in Listing 5.1, $propertyMap$ contains the entry

$$\big[\,(\text{http://ex1.org/sports\_001, http://ex1.org/sports\_002})\,,\,\{\text{foaf:knows}\}\,\big]. \tag{5.20}$$

Subsequently, iterating over the $propertyMap$, the SLPs are constructed using the RDF class information of every resource in $classMap$. Equation (5.2) illustrates the computed SLP from the data listed in Listing 5.1.

5.2 Recommending Vocabulary Terms Using Schema-Level Patterns

Typically, an LOD engineer provides TermPicker some input information, and in return the engineer receives a set of desired vocabulary term suggestions as output. TermPicker's input

```
1  CALCULATE-SLPs(𝔻𝕊)
2  {
3    HashMap<String, Set> classMap
4    HashMap<String, Set> propertyMap
5    Set<slp> 𝕊𝕃ℙ
6
7    for each G in 𝔻𝕊
8    {
9      for each (s,p,o,c) in G
10     {
11       if(p.equals(``rdf:type''))
12       {
13         addToClassMap(s, o)
14       }
15       else
16       {
17         addToPropertyMap((s,o), p)
18       }
19     }
20   }
21
22   for each (s,o) in propertyMap
23   {
24     sts = classMap.get(s)
25     ps = propertyMap.get((s,o))
26     ots = classMap.get(o)
27
28     slp = (sts,ps,ots)
29
30     𝕊𝕃ℙ.add(slp)
31   }
32 }
```

Listing 5.2: **Calculation of SLPs.** The Listing shows the algorithm how SLPs are calculated from the set $\mathbb{DS}$ containing RDF triples of all data sets on the LOD cloud

information is the query-SLP slp_q. It denotes an excerpt of the LOD model the engineer is currently focused on. Recommendation candidates, i.e., a ranked subset of vocabulary terms $\{x_1, ..., x_n\}$ from the set of classes and properties from all data sets on the LOD cloud $(x \in (\mathbb{T} \cup \mathbb{P}))$, are TermPicker's output. The work flow is as follows: First, the query-SLP is extended via the $\oplus$ operator with a vocabulary term x. In other words, the system performs the operation $slp_q \oplus x$, and adds x either to the set sts, ps, or ots. Subsequently, TermPicker calculates five *feature values* for candidate x and the combination of the query-SLP slp_q and x, i.e., $slp_q \oplus x$. These feature values are used by a ranking model to provide an ordered list of vocabulary term recommendations. If only ranking methods are applied, all terms in $\{x_1, ..., x_n\}$ are included in the list of recommendations, but in an ideal case, all relevant candidates are ranked at the top of the list. If filtering methods are applied, e.g., one specific feature value has to be greater than zero, then those terms that violate such a constraint are not included in the recommendation list.

The five features used to describe every recommendation candidate x and the combination $slp_q \oplus x$ comprise the popularity of x (identified by three different features), whether

Table 5.2: **Overview of the Features.** The table illustrates the five features f_1 to f_5 used to represent vocabulary terms $x \in (\mathbb{T} \cup \mathbb{P})$, i.e., the recommendation candidates.

Feature	Definition of the Feature
f_1	Number of data sets on the LOD cloud using the recommendation candidate x
f_2	Number of data sets on the LOD cloud using the vocabulary $V(x)$, i.e., the vocabulary of recommendation candidate x
f_3	Total number of occurrences of recommendation candidate x on the LOD cloud
f_4	Whether the recommendation candidate x is from a vocabulary that is already used in query-SLP slp_q
f_5	Number of SLPs in $\mathbb{SLP}$ that contain recommendation candidate x together with all terms in slp_q

candidate x is already used in slp_q, and the number of other SLPs in $\mathbb{SLP}$ that use all terms in slp_q and candidate x. Table 5.2 presents an overview of these five features.

These five features are formally defined in Sections 5.2 to 5.2. It is presented how their values are calculated and how they aid to calculate appropriate vocabulary term recommendations.

Vocabulary Term Popularity (Features f_1 to f_3)

Feature f_1 comprises the number of data sets $G \in \mathbb{DS}$ on the LOD cloud using a recommendation candidate x. It is calculated by examining whether the term x is contained in an RDF quadruple of a graph G.

$$f_1(x) = |\{\ G \mid (\exists\,(s,p,o,c) \in G : p = x) \vee (\exists\,(s, \mathsf{rdf{:}type}, o, c) \in G : o = x)\}| \tag{5.21}$$

Feature f_2 calculates the number of data sets on the LOD cloud using the vocabulary of the recommendation candidate x, which is computed by a function $V(x)$. In this context, a vocabulary is defined simply by its URI namespace, which is either a *hash namespace* or a *slash namespace* as specified by the W3C.[64] In other words, a term's vocabulary $V(x)$ is denoted by the URI without the string value after the last slash or the last hash of the URI. The feature value for f_2 is computed similar to feature f_1, but it examines whether the vocabulary of term x is used in a quadruple of graph G.

$$f_2(V(x)) = |\{\ G \mid (\exists\,(s,p,o,c) \in G : p \in V(x)) \vee (\exists\,(s, \mathsf{rdf{:}type}, o, c) \in G : o \in V(x))\}| \tag{5.22}$$

[64] `http://www.w3.org/2001/sw/BestPractices/VM/http-examples/2006-01-18/#naming`, last accessed February 14th, 2017

The total number of a candidate's occurrences on the LOD cloud is calculated by feature f_3. In contrast to the features f_1 and f_2, feature f_3 is calculated by counting each triple/quadruple across all data sets in $\mathbb{DS}$, in which the vocabulary term x is contained.

$$f_3(x) = \sum_{G \in \mathbb{DS}} |\{(s,p,o,c) \in G \mid (p = x) \vee (o = x \wedge p = \text{rdf:type})\}| \tag{5.23}$$

These three features define the popularity of a vocabulary term on a fine-grained level. Whereas the total number of occurrences of x show its overall usage, the amount of data sets on the LOD cloud using x and the amount of data sets using its vocabulary specify whether the usage of x is spread across many data sets on the LOD cloud or concentrates on only a few large ones.

In general, reusing popular vocabulary terms, i.e., recommendation candidates with relatively high values for features f_1 to f_3, facilitates an easier consumption of the data, since many Linked Data tools provide tailored support for popular vocabularies [42].[65] It also makes the data more understandable for LOD engineers, as they are likely to know terms from popular vocabularies. TermPicker makes use of these features and establishes a ranking model that ranks popular vocabulary terms high in the ordered list of recommendations.

Terms from an Already Used Vocabulary (Feature f_4)

Feature f_4 indicates whether the vocabulary of a recommendation candidate x is already contained in the query-SLP $slp_q = (sts_q, ps_q, ots_q)$. In detail, it is investigated whether the sets sts_q, ps_q, and ots_q contain a term from the vocabulary of candidate x. The calculation returns a binary value, where 1 denotes that the vocabulary of candidate x is already used in slp_q, and 0 if it is not contained in slp_q.

$$f_4(slp_q, x) = \begin{cases} 1 & \text{if } \exists\, V : x \in V \wedge (sts_q \cup ps_q \cup ots_q) \cap V \neq \emptyset \\ 0 & \text{else} \end{cases} \tag{5.24}$$

Reusing terms from the same vocabulary is considered as an important strategy, as illustrated in the survey on vocabulary reuse strategies described in Chapter 4. This strategy is used by various domains. For example, in the statistics domain [61], it is accustomed to reuse primarily vocabulary terms from SKOS[66] or XKOS[67]. The reason is that SKOS and XKOS are domain-specific vocabularies for the statistics domain and they are far more likely to contain many RDF classes and properties that can be reused for describing data from that domain. The same applies to other domains, such as the museum domain, or to the domain for describing geographic locations. The assumption is that an engineer prefers to search for vocabularies covering the domain of interest and subsequently adapt classes and properties from those vocabularies for particular needs [76]. Furthermore, reusing terms from the same vocabulary reduces the overload from too many different vocabularies. This makes the data easier to understand for others, especially if they are familiar with domain-specific vocabularies [74]. An example is the usage of the GEO vocabulary when modeling

[65] `http://www.w3.org/TR/ld-bp/#VOCABULARIES`, last accessed February 14th, 2017
[66] `http://www.w3.org/2004/02/skos/`, last accessed February 14th, 2017
[67] `http://rdf-vocabulary.ddialliance.org/xkos.html`, last accessed February 14th, 2017

Table 5.3: **Feature Values Example.** The table shows the feature values of features f_1 to f_5 for four recommendation candidates (x_1 to x_4) that are used to extend slp_q

	f_1	f_2	f_3	f_4	f_5 (SLP-feature)
(slp_q, x_1)	7	9	20	1	0
(slp_q, x_2)	3	3	5	0	6
(slp_q, x_3)	10	20	80	0	2
(slp_q, x_4)	4	20	29	1	1

museum data as LOD.[68] This vocabulary contains many terms on specifying geographical information. Using this vocabulary, it is possible not only to specify a place, e.g., a town or a country, where an artwork was issued, where it was made, where it was before it came to the museum, but it makes it also possible to describe the place with latitude, longitude, and altitude properties.

The Schema-Level Pattern Feature (Feature f_5)

The SLP-feature is calculated based on the extended query-SLP, i.e., on $slp_q \oplus x$. This extended query-SLP is compared to SLPs $slp_i \in \mathbb{SLP}$, in order to find SLPs with $slp_q \oplus x \leq slp_i$. The number of SLPs $slp_i \in \mathbb{SLP}$ with $slp_q \oplus x \leq slp_i$ represents how often other data providers use the vocabulary term x in conjunction with the terms in slp_q.

$$f_5(slp_q \oplus x\,,\, \mathbb{SLP}) = |\{slp_i \mid slp_q \oplus x \leq slp_i,\, slp_i \in \mathbb{SLP}\}| \tag{5.25}$$

The assumption is that recommendations based on this feature can result in reducing heterogeneity in data representation, as the recommended terms have already been used in a similar manner by other data providers. In essence, the more SLPs in $\mathbb{SLP}$ use candidate x together with all terms in the query-SLP, the more appropriate it seems to be to reuse candidate x. If the term x already exists in slp_q, then the query-SLP is not extended, and x is ignored as a recommendation candidate.

5.3 Utilized Recommendation Approaches

Recommender systems, in their simplest form, provide a list of items ranked from most to least relevant in descending order [50, 70]. However, establishing a ranking model that sorts recommendation candidates in such a way is not trivial when it is necessary to consider several features representing the recommended items. For example, features f_1 to f_5 describe each recommendation candidate x in a quite unique way. Table 5.3 illustrates an example, in which four recommendation candidates with different feature values have to be ranked from most to least appropriate in descending order. To this end, one must observe a general coherence between the features and the relevance of each recommendation candidate. One

[68] `https://www.w3.org/2003/01/geo/`, last accessed February 14th, 2017

could argue that candidate term x_3 should be ranked to the top as it is more popular than the other candidates. It could also be said that x_2 is the best choice, as it is used in more SLPs together with the terms in slp_q. Observing such a coherence manually is difficult and often not feasible. It must rather be observed in an automatic way, i.e., the recommender system must "learn" which features are most responsible for ranking the relevant candidates to the top of the recommendation list.

Learning To Rank (L2R) is a family of supervised machine learning algorithms that take a set of features and relevance information on the recommendation candidates as input, and return a generalized ranking model for candidates based on these features. It is a state-of-the-art method for examining how features correlate with the relevance of recommendation candidates. In other words, L2R establishes one general weighting function over all features and can apply it to new and previously unknown situations.

It is also of interest to investigate term recommendations based on solely the SLP-feature. The computation of SLPs is based on the idea for building *frequent item sets*, i.e., items that often occur together in a transaction. Frequent item sets are commonly known in the area of basket analysis, as it allows to identify products that have been bought together in one transaction, e.g., milk and cereal. Association Rule mining (AR) is the most common method in basket analysis to use frequent item sets for inferring rules that specify "Customers that have bought these products have also bought the following products:...". Such rules can also be inferred from the set of SLPs calculated from data on the LOD cloud as well, since this set of SLPs represents sets of terms frequently used together by other data providers. In detail, a transaction is represented by an SLP that states which vocabulary terms have been used together to represent a specific part of the LOD model. The AR approach uses such SLPs to infer rules specifying "Data providers on the LOD cloud that have used the terms in slp_q, have also used the following terms: $x_j, ..., x_k$". Contrary to L2R, AR uses solely the SLP-feature, thus, the list of recommendations is simply sorted by the SLP-feature value in descending order.

In the following, Sections 5.3.1 and 5.3.2 explain Learning To Rank and Association Rule mining in more detail, particularly how they are used to generate RDF vocabulary term recommendations.

5.3.1 Learning To Rank

Learning To Rank algorithms compute a ranking model ϱ from some training data by observing correlations between the utilized features and the relevance of a recommendation candidate [60]. Ideally, the derived model ranks all relevant vocabulary terms high and above less relevant vocabulary terms.

L2R algorithms are categorized into three different groups according to their method for learning a ranking model [60]: *point-wise* L2R algorithms, *pair-wise* L2R algorithms, and *list-wise* L2R algorithms. A point-wise approach ranks vocabulary terms directly by using a scoring function for allocating a ranking score to each recommendation candidate. This can be done either via regression-based, classification-based, or ordinal regression-based algorithms [60]. Subsequently, the candidates are sorted by the ranking score in descending order. Optimally, the ranking score is equal to or very close to the relevance value of the

candidate such that the relevant candidates are ranked highest. Pair-wise approaches do not focus on estimating the ranking score of a candidate directly. Instead, these approaches try to find the relative order between two recommendation candidates. Typically, this is done via classifying the candidates, i.e., the algorithms determine which candidate in a pair is preferred. The goal is to maximize the amount of correctly classified pairs, i.e., maximize the amount of pairs in which the relevant candidate is preferred. If all pairs are classified correctly, it results in a correct ranking. List-wise L2R approaches take the entire list of recommendations associated with a given query and try to optimize quality measures such as Mean Average Precision (MAP) or Mean Reciprocal Rank (MRR) directly. Such approaches perform multiple iterations (default used in this work is 1000), adjusting the weighting function with each iteration. This way, they are able to compare the measured MAP and/or MRR values. If positive impacts are detected, i.e., the MAP and/or MRR values increase, the weighting function continues to fine-tune the feature last adjusted. If negative impacts are detected, the weighting function attempts to adjust another feature. After all iterations and after the ranking model has been continuously adjusted, the final ranking model is defined by the best performing weighting function.

In particular, pair-wise and list-wise approaches have demonstrated good performance in generic ranking scenarios [18]. However, point-wise algorithms seem to outperform pair-wise and list-wise algorithms when recommendation candidates have a binary relevance value, i.e., the recommendation is either relevant or not, as they are based on regression and classification algorithms such as Decision Trees [60]. One problem with point-wise approaches is that they provide only a sub-optimal solution, as they do not compare recommendation candidates directly to each other [60, 18]. Pair-wise and list-wise approaches alleviate this situation by comparing either pairs of candidates or by evaluating the entire list of candidates. Pair-wise approaches establish an improved modeling of the relative order among recommendation candidates, while list-wise approaches make the information on the ranking positions of candidates more visible to the L2R algorithm.

In this work, the training data used for inferring an L2R-based ranking model for vocabulary term recommendations is a set of query-SLPs with *provided relevance information* on each recommendation candidate. Such relevance information can be gathered in various ways, such as using a gold standard data set for training, or hiding specific data and considering this hidden data as relevant (cf. Section 2.4.1). For example, some training data for an L2R method contains an SLP like $slp_q = (\{\text{mo:SoloMusicArtist}\}, \varnothing, \varnothing)$ with the relevance information that solely the terms foaf:made and mo:member_of are considered relevant when recommending properties. Using this information, L2R algorithms iterate over the training data multiple times — defined by the user — and adjusts the correlation between the features, such that the relevant recommendation candidates appear as far as possible towards the top of the results list. This way, the learned ranking model can be used for new and previously unknown query-SLPs. For instance, the trained ranking model has learned that most relevant recommendation candidates have a high SLP-feature value. Let us further assume that the set $\mathbb{SLP}$ contains a lot of SLPs with $(\{\text{cc:Work}\}, \{\text{w3:presenter}\}, \{\text{foaf:Person}, \text{dc:Agent}\})$ comprising terms from the Creative

Commons[69] ontology and from an ontology for managing presentations at W3C[70]. Given a query-SLP $slp_q = (\{\text{cc:Work}\}, \{\text{w3:presenter}\}, \varnothing)$, which was not part of the training set, as input for TermPicker. The approach calculates the SLP-feature for various recommendation candidates, and the terms foaf:Person and dc:Agent will have a high value, as according to the set $\mathbb{SLP}$ they are used by many data providers in conjunction with the terms in the query-SLP. Therefore, these two terms are likely to be listed at the top positions of the recommendation list.

5.3.2 Association Rule Mining

Association Rules (ARs) are *if/then* statements that help discover relationships between data in a data repository [89]. In this thesis, ARs are used to identify such relationships between vocabulary terms, i.e, the goal is to find vocabulary terms that are frequently used together. For example, a museum wants to publish data on artworks as Linked Open Data. The corresponding engineer identifies the ECRM[71] ontology, which is used by other museum for modeling data as LOD, and reuses the RDF class ecrm:E22_Man-Made_Object to specify the type of an artwork. Association Rule mining can now help in finding properties that were frequently used by others together with the class ecrm:E22_Man-Made_Object, such as ecrm:P138i_has_representation.

SLPs calculated from data sets on the LOD cloud, i.e., the elements in the set $\mathbb{SLP}$, represent sets of jointly used vocabulary terms. Association Rules utilized by TermPicker are computed from the set $\mathbb{SLP}$ by considering each $slp \in \mathbb{SLP}$ as one independent transaction. To assess the quality of a rule, two measures called *support* and *confidence* are used. Basically, both measures specify how often a set of vocabulary terms appear together in the set $\mathbb{SLP}$, and thereby denotes validity of a rule. There are manually defined minimum support and minimum confidence values, which define a threshold for generating an *if/then* statement. These are defined by the person, who generates the rules. In the event that the support and confidence values for a recommendation candidate are above the threshold, a rule is generated. More technical, let us assume there are two sets of different vocabulary terms X and Y, the support specifies how often the union of X and Y occurs in the set $\mathbb{SLP}$ divided by the total number of SLPs in $\mathbb{SLP}$. The confidence denotes how often the union of X and Y occurs in $\mathbb{SLP}$ divided by the number of occurrences of X in $\mathbb{SLP}$. If both the support and the confidence are higher than the self-defined threshold, then one can induce the rule $X \Rightarrow Y$ specifying "*if* the vocabulary terms in X are used, *then* the vocabulary terms in Y are used as well". For example, X comprises the vocabulary terms ecrm:E22_Man-Made_Object as well as ecrm:E38_Image and Y comprises the term ecrm:P138i_has_representation. If there are several SLPs in $\mathbb{SLP}$ containing the object property ecrm:P138i_has_representation together with ecrm:E22_Man-Made_Object and ecrm:E38_Image, then AR mining generates the rule $X \Rightarrow Y$ specifying: "*If* ecrm:E22_Man-Made_Object and ecrm:E38_Image are used together, *then* ecrm:P138i_has_representation is used as well".

[69] `http://creativecommons.org/ns#`, last accessed February 14th, 2017
[70] `http://www.w3.org/2004/08/Presentations.owl#`, last accessed February 14th, 2017
[71] `http://erlangen-crm.org/`, last accessed February 14th, 2017

Formally, the support $Supp$ and the confidence $Conf$ for a rule $X \Rightarrow Y$ are defined as follows:

$$Supp(X,Y) = \frac{frq(X \cup Y)}{N} \tag{5.26}$$

$$Conf(X,Y) = \frac{frq(X \cup Y)}{frq(X)} \tag{5.27}$$

where frq denotes the frequency how often the arguments occur together in $\mathbb{SLP}$ and N denotes the number of SLPs in $\mathbb{SLP}$, i.e., $N = |\mathbb{SLP}|$. Generally, the higher the self-defined thresholds, the more precise are the rules. But with higher thresholds, various rules that make sense will not be considered. Even worse, by setting the thresholds too high, it is likely to produce no rules at all [89]. In order to generate all possible rules, the thresholds are set as low as possible, i.e., only if the support and confidence values are zero ($Supp = 0$ and $Conf = 0$), a rule will not be generated. This way, Lift, phi-coefficient, or other figures measuring the quality and/or the correctness of a rule are not needed [25]. However, with such a low minimum support and minimum confidence, there is a chance of getting misleading rules, which results in recommending irrelevant vocabulary terms. This generates a problem to identify truly frequent item sets, but further investigations showed that this happened quite rarely. Thus, it is rather effective and manageable to generate all possible rules, and subsequently filter out the non-appropriate terms by hand.

To generate the rules, a state of the art algorithm called *Apriori* [1] is used. It is an iterative approach for finding frequent item sets, in this case frequent sets of vocabulary terms that are used together in SLPs. The algorithm is based on the assumption that a subset of a frequent set of terms must also be a frequent set of terms, and this makes it possible to define rules with

$$s_i \rightarrow (slp_i \ominus s_i) \tag{5.28}$$

where s_i is a non-empty subset of slp_i with $s_i \leq slp_i$ and $slp_i \in \mathbb{SLP}$. This means, every non-empty subset of slp_i except s_i contains vocabulary terms that can be recommended, if s_i is provided as input. For example, given the slp_i

$$slp_i = (\{\text{swrc:Publication}\}, \{\text{dc:creator}\}, \{\text{foaf:Person}\}) \tag{5.29}$$

and a non-empty subset s_i

$$s_i = (\{\text{swrc:Publication}\}, \{\}, \{\text{foaf:Person}\}). \tag{5.30}$$

Subtracting subset s_i from the SLP slp_i will result in

$$slp_i \ominus s_i = (\varnothing, \{\text{dc:creator}\}, \varnothing) \tag{5.31}$$

Therefore, if s_i is the input for TermPicker and if the set $\mathbb{SLP}$ contains solely slp_i, then the list of vocabulary term recommendations is

$$s_i \rightarrow (slp_i \ominus s_i) = \{\text{dc:creator}\} \tag{5.32}$$

5.4 Evaluation

TermPicker is evaluated using a 10-fold leave-one-out evaluation. Each fold comprises a *training set* for computing the ranking model, a *test set* for validating the ranking model, and a set which simulates the data sets that are already published on the LOD cloud to calculate features f_1 to f_5.[72] TermPicker is evaluated by investigating four aspects. The first three aspects compare different L2R algorithms to each other, and the fourth aspects compares the best performing L2R algorithm with the Association Rule mining approach. In detail, these aspects are:

(i) To what extent does the SLP-feature (using feature f_5 plus features f_1 to f_4) enhance the prediction accuracy compared to the baseline of reusing only popular vocabulary terms [42, 74] (using features f_1 to f_3) and to the baseline of reusing popular terms from the same vocabulary [61, 74] (using features f_1 to f_4)?

(ii) Is the prediction accuracy higher for recommending RDF classes for resources in subject position of a triple, for recommending classes describing resources in object position, or for recommending properties?

(iii) Which Learning To Rank algorithm provides the best vocabulary term suggestions, i.e., the highest prediction accuracy, when using features f_1 to f_5?

(iv) Learning To Rank vs. Association Rule mining: which method achieves the higher prediction accuracy?

To compare L2R algorithms with each other, the RankLib[73] library is used. This library contains eight different L2R algorithms, among them point-wise, pair-wise, as well as list-wise approaches, and provides an entire framework to train and evaluate an algorithm's ranking model. The prediction accuracy is measured using the Mean Average Precision (MAP) and the Mean Reciprocal Rank at the first five positions (MRR@5) (cf. Section 2.4.1.3).

The evaluation design is described in detail in Section 5.4.1 with the following Section 5.4.2 providing an illustration of the utilized data sets.

5.4.1 Evaluation Design

TermPicker's recommendations are evaluated by simulating a recommendation scenario, in which a LOD engineer is looking for a specific vocabulary term. For the evaluation, three sets of SLPs are used. As the input for TermPicker is an SLP, the training set and the test set for computing and evaluating the ranking model are disjoint sets of distinct SLPs. These sets of SLPs have been computed from different data sets on the LOD cloud. Another set of LOD data is used to compute the third set of SLPs, i.e., the set $\mathbb{SLP}$, in order to calculate the

[72] Evaluation data and results are available at: `https://github.com/WanjaSchaible/l2r_eval_material`, last accessed February 14th, 2017

[73] `http://sourceforge.net/p/lemur/wiki/RankLib/`, last accessed February 14th, 2017

SLP-feature. The set $\mathbb{SLP}$ must not be disjoint with the training or the test set, otherwise, the SLP-feature would be zero for each recommendation candidate.

Before providing TermPicker with a query-SLP, one or more vocabulary terms from the query-SLP are randomly selected for "hiding" (cf. Section 2.4.1 for more details on the held-out evaluation). To this end, the terms that are selected for hiding are extracted from the query-SLP using the $\ominus$ operator. They represent the set of *relevant* recommendation candidates, since they are the ones that have initially been used. All other recommendation candidates, which are not selected for hiding, are considered as irrelevant recommendations. This way, for each query-SLP, the algorithms are provided a set of recommendation candidates, a set of features categorizing each candidate, and the relevance information for each candidate. For example, given the query-SLP slp_q from the training or test set with

$$slp_q = (\{\text{mo:SoloMusicArtist}\}, \{\text{foaf:made}\}, \{\text{mo:Record}\}) \tag{5.33}$$

The term foaf:made is randomly selected our the three terms in slp_q and removed via the $\ominus_{ps}$ operator. The resulting SLP slp'_q is the one that is provided as input for TermPicker, and the extracted property foaf:made is hidden.

$$slp'_q = slp_q \ominus_{ps} \text{foaf:made} = (\{\text{mo:SoloMusicArtist}\}, \varnothing, \{\text{mo:Record}\}) \tag{5.34}$$

This means, when the SLP slp'_q is provided as input for TermPicker, the property foaf:made is the single relevant recommendation candidate. Such a setup makes it possible to compute and evaluate a ranking model by interpreting a ranked list of recommendations

$$< \text{foaf:name}, \text{mo:remixed}, \textbf{foaf:made}, ... > \tag{5.35}$$

in the following way: the first two recommendations are irrelevant, and the first relevant recommendation is at the third rank of the results list. Using this information, it is possible to calculate quality measures such as MAP and MRR@5, and thus, to demonstrate the prediction accuracy for L2R and AR algorithms.

5.4.2 Evaluation Data

TermPicker's prediction accuracy is examined by performing one evaluation based on the DyLDO [52][74] data set (snapshot from 2014-03-02) and a second evaluation based on the Billion Triple Challenge (BTC) 2014 data set [53][75] (crawl no. 1).

DyLDO comprises data from the LOD cloud with about 10.8 million triples from 382 different pay-level domains (PLDs).[76] In total there are approximately 2.3 million distinct vocabulary terms from about 600 vocabularies. The BTC 2014 data set contains about 1.4 billion triples, of which the first 34 millions are used in order to reduce the memory requirements for the SLP computation. These 34 million triples are provided by 3,493

[74] `http://swse.deri.org/dyldo/`, last accessed February 14th, 2017

[75] `http://km.aifb.kit.edu/projects/btc-2014/`, last accessed February 14th, 2017

[76] A pay-level domain (PLD) is a sub-domain of a top-level domain, such as *.org* or *.com*, or of a second-level country domain, such as *.de* or *.uk*. To calculate the PLD, the Google guava library is used in this work: `https://code.google.com/p/guava-libraries/`, last accessed February 14th, 2017

Table 5.4: **PLDs Selected as Training and Test Sets.** The selection is based on $(C1)$ - PLDs that provide the highest number of distinct vocabulary terms - and $(C2)$ - PLDs with the highest ratio between the reused vocabulary terms and all RDF types and properties.

DyLDO				BTC 2014			
PLD	$(C1)$	$(C2)$	# of SLPs	PLD	$(C1)$	$(C2)$	# of SLPs
bbc.co.uk	146	1.00	522	b4mad.net	291	1.00	393
bblfish.net	82	0.99	150	derby.ac.uk	137	1.00	197
bl.uk	102	0.46	246	heppnetz.de	121	1.00	199
data.gov.uk	258	0.93	920	ivan-herman.net	196	1.00	303
fundacionctic.org	110	0.97	390	jones.dk	164	1.00	155
kanzaki.com	176	0.99	294	ldodds.com	115	1.00	125
kasei.us	100	1.00	121	lmco.com	128	1.00	204
taxonconcept.org	139	0.92	424	mfd-consult.dk	192	1.00	315
thefigtrees.net	89	1.00	102	mit.edu	174	0.96	293
wikier.org	96	1.00	133	nickshanks.com	100	0.97	164

pay-level domains. Within these triples there are ca. 5.5 million distinct RDF classes and properties from about $1,530$ different vocabularies. The data sets differ by their size and the seed lists, i.e., the set of URIs that form the core of the data crawling, thus containing different data.

The evaluation data set is split by PLDs such that the data from ten PLDs is used as a training set as well as a test set, while the data from the remaining PLDs is used to simulate the data sets published on the LOD cloud (cf. Section 7.1 for detailed explanation why it is necessary to split the data by PLDs). The split could not be performed randomly, as it is necessary to make sure that the training and test data contained enough SLPs to train and evaluate the ranking model. Thus, to generate representative results, ten pay-level domains were selected based on two criteria:

$(C1)$ A high number of distinct vocabulary terms within a PLD, and

$(C2)$ A high ratio between reused vocabulary terms and all RDF types and properties within a PLD.

$(C1)$ indicates that resources of various RDF types are interlinked via several different properties. This way, it is very likely to calculate a high number of distinct SLPs from that data. $(C2)$ indicates that most resources and their connections via properties are described by reused and not self-defined vocabulary terms. A reused vocabulary term is defined as follows: if a vocabulary term does not contain the PLD in its namespace, then it is considered a reused vocabulary term. $(C2)$ ensures that the relevant recommendation candidates (the hidden terms from the SLPs in the training and test sets) are not self-defined terms. Otherwise, all feature values for the relevant candidates would be zero which makes computing a ranking model impossible. In summary, using $(C1)$ and $(C2)$ enables for calculating SLPs that most likely contain many reused terms, and this is most important for generating valuable recommendations.

Table 5.4 provides an overview of the selected PLDs used for the evaluations based on the DyLDO (left half of the table) and BTC 2014 (right half of the table) data set as well as the numbers considering $(C1)$ and $(C2)$. Furthermore, it displays the number of distinct SLPs that are calculated from the data of the selected pay-level domains. The data from the remaining PLDs that is used for calculating the feature values contains $117,776$ (DyLDO) and $227,010$ (BTC 2014) SLPs, respectively.

5.5 Results

The results are illustrated in the following manner: Section 5.5.1 presents the results for aspects (i) to (iii). It illustrates the impact of the SLP-feature on the prediction accuracy, the differences between recommending RDF classes and properties, and the three best performing L2R-algorithms from the RankLib library. Subsequently, Section 5.5.2 shows the results for aspect (iv). It is presented how the best performing L2R-algorithm compares to the Association Rule mining approach.

5.5.1 Prediction Accuracy of Recommendations based on Learning To Rank

The prediction accuracy of the three best performing L2R algorithms from the RankLib library are illustrated in Figure 5.2 (measured via the Mean Average Precision (MAP)) and in Figure 5.3 (measured via the Mean Reciprocal Rank at the first five positions (MRR@5)). Both depict the measurements of the recommendation quality considering the aspects (i), (ii), and (iii). The most competitive L2R algorithms in the RankLib library are: the list-wise algorithms *Coordinate Ascent* [65] and *LambdaMART* [87], as well as the point-wise algorithm *Random Forests* [13]. Reusing solely popular vocabulary terms (using features f_1 to f_3 and marked as POP) as well as reusing vocabulary terms from the same vocabulary (using features f_1 to f_4 and marked as SAME) resemble the baselines. The SLP-feature-based approach uses features f_1 to f_5, where f_5 is the SLP-feature. Each box plot displays the different recommendations of an RDF class for resources in subject position (abbreviated as *sts*), a class for resources in object position (abbreviated as *ots*), and a property (abbreviated as *ps*) for both the BTC 2014 and the DyLDO data set. The box plot comprises the measured values from each of the ten folds. The plot in bold font presents the best performing configuration.

In both Figures 5.2 and 5.3, one can observe that the three L2R-algorithms tend to perform a quite similarly. The MAP and the MRR@5 values increase when adding the SLP-feature, and they all achieve higher results using the BTC 2014 data. The most constant performance across the two data sets when using the SLP-feature shows the Random Forests algorithm. It can be seen that the median is higher when using the SLP-feature. In detail, the Mean Average Precision is MAP ≈ 0.4 when using solely features f_1 to f_3, when using features f_1 to f_4 it is MAP ≈ 0.45, and when also using the SLP-feature it is MAP ≈ 0.75. The MRR@5 values are slightly higher. These results are underpinned by the calculated mean values of all folds illustrated in Tables 5.5. Evidently, using the SLP-feature along

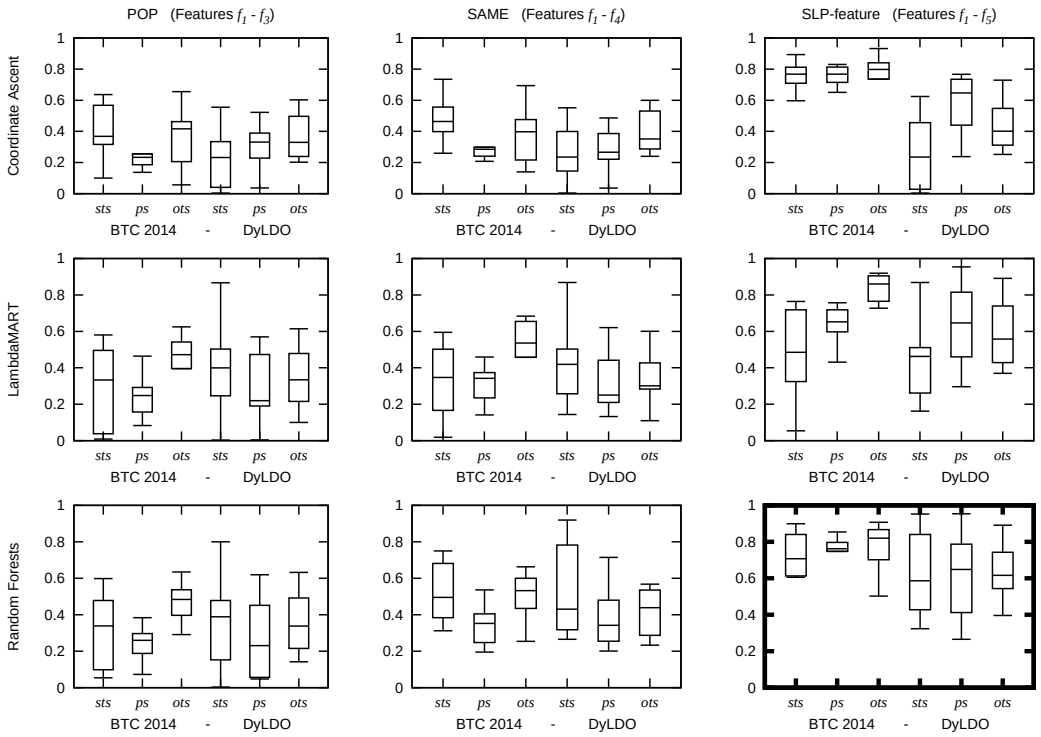


Figure 5.2: **Mean Average Precision (MAP) based on Learning To Rank.** The figure shows the prediction accuracy of the three best performing L2R-algorithms (Random Forests, LambdaMART, and Coordinate Ascent) for different sets of features (POP, SAME, and SLP-feature) Each box plot denotes a 10-fold cross-validation split by recommending terms for sts, ps, and ots based on the DyLDO and the BTC 2014 data sets. The plot marked bold depicts the overall best results.

with features f_1 to f_4 provides the best prediction accuracy. The *Coordinate Ascent* algorithm performs best on the BTC 2014 data set, but the prediction accuracy drops when applied on the DyLDO data set. Leveraging the quality measures over both data sets, it can be seen that the point-wise L2R algorithm Random Forests performs best. Regarding whether to recommend a class or a property, there seems to be no difference.

Sections 5.5.1.1 to 5.5.1.3 comprise the description of the values within the box plots and whether or not the differences in the recommendation qualities are significant or not.

5.5.1.1 Aspect (i): Impact of the SLP-feature

Most noticeable in Figures 5.2 and 5.3 is the influence of the SLP-feature when using the BTC 2014 data set as evaluation data. The median recommendation quality increases by about 30% compared to the baseline of reusing only popular vocabulary terms (POP) and by 20% compared to the SAME baseline. This can be observed for all depicted L2R al-

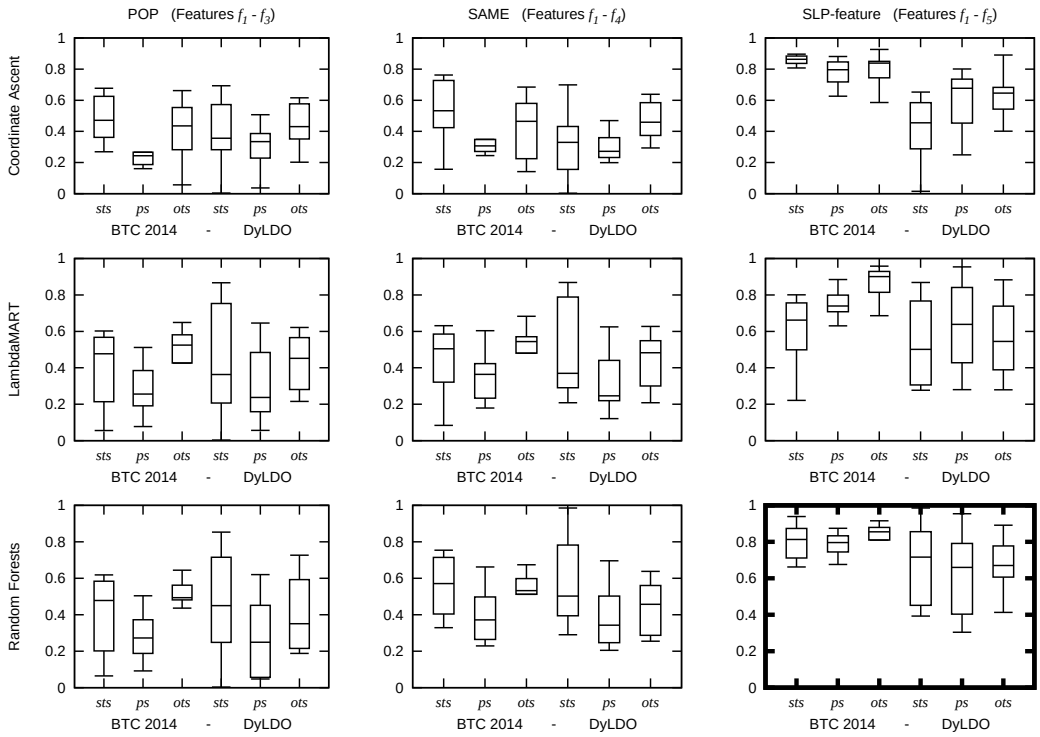


Figure 5.3: **Mean Reciprocal Rank (MRR@5) based on Learning To Rank.** The figure shows the prediction accuracy of the three best performing L2R-algorithms (Random Forests, LambdaMART, and Coordinate Ascent) for different sets of features (POP, SAME, and SLP-feature). Each box plot denotes a 10-fold cross-validation split by recommending terms for sts, ps, and ots based on the DyLDO and the BTC 2014 data sets. The plot marked bold depicts the overall best results.

gorithms. Such differences are not as noticeable when performing the evaluation on the DyLDO data set. However, they are still around $15 - 20\%$ compared to the baselines POP and SAME. In total, using the SLP-feature increases the average MAP and MRR@5 values up to MAP ≈ 0.75 and MRR@5 ≈ 0.8. Applying a Friedman test with $\alpha = .05$ illustrates that the differences between using the SLP-feature and the baselines are significant ($\mathcal{X}^2(3) = 51,667, p < .001$). A post hoc Wilcoxon signed-rank test with Bonferroni correction applied ($\alpha' = \alpha/3 = .017$) shows that using the SLP-feature provides significantly better recommendations ($Z = -4.782, p < .001$ compared to the SAME baseline and $Z = -5.832, p < 0.001$ compared to the POP baseline).

Such a result demonstrates the extent to which the SLP-feature is relevant for providing valuable vocabulary term recommendations. It makes it possible to recommend to the engineer vocabulary terms that might lead to reducing heterogeneity in data representation. Given that real-life data is used for the evaluation, and as the SLP-feature increases the prediction accuracy, it can be argued that the utilized real-life data was initially modeled by

Table 5.5: **Mean MAP and MRR@5 Values for Learning To Rank.** The table shows the mean MAP and MRR@5 values and their standard deviation for the three most competitive L2R algorithms and different sets of features.

		BTC 2014		DyLDO	
Model	Features	MAP	MRR@5	MAP	MRR@5
CoordinateAscent	POP	.34 (.16)	.39 (.15)	.30 (.16)	.37 (.17)
	SAME	.39 (.15)	.44 (.15)	.31 (.14)	.36 (.16)
	SLP	.76 (.11)	.81 (.09)	.43 (.19)	.55 (.18)
LambdaMART	POP	.33 (.19)	.37 (.18)	.43 (.26)	.47 (.25)
	SAME	.39 (.18)	.42 (.16)	.43 (.25)	.49 (.25)
	SLP	.64 (.16)	.73 (.13)	.57 (.23)	.58 (.24)
Random Forests	POP	.34 (.16)	.39 (.16)	.41 (.27)	.47 (.28)
	SAME	.46 (.15)	.48 (.15)	.51 (.24)	.54 (.23)
	SLP	.75 (.11)	.80 (.09)	.64 (.23)	.67 (.21)

investigating which vocabulary terms other data providers have used to model their data. This way, the SLP-features supports LOD engineers in their search regarding how other data providers modeled their data.

5.5.1.2 Aspect (ii): Differences Between the Recommendation types

Before modeling data as LOD, it is common to describe the type of data entities and then how the data entities are related.[77] Therefore, an engineer first defines a set of classes to describe resources and then defines which properties to use for connecting these classes. Resources are often described using more than one RDF class[78], whereas there is usually one property to describe a relationship. For that reason, and due to the experiment design, there is the chance that TermPicker provides more appropriate property recommendations. There are many possible recommendation candidates for RDF types, but only a fraction of these candidates are considered relevant in the evaluation design. Recommendation candidates for properties are not that numerous. This increases the chance that a property recommendation is a relevant recommendation. However, this was not an influencing factor in the experiment. Figures 5.2 and 5.3 illustrate that there are no differences between recommending RDF classes for resources in subject position (sts), classes for resources in object position (ots), or recommending properties (ps). Only a slight differences (between $5-10\%$) in the prediction accuracy can be perceived when using all features. The Friedman tests that were applied on the results when using the SLP-feature show that the differences are not significant ($\mathcal{X}^2(3) = 14,000$, **n.s.**, $p = .449$ calculated for the results based on the

[77] `https://www.w3.org/TR/ld-bp/#MODEL`, last accessed February 14th, 2017

[78] See the rdf:type information on Pelé: `http://dbpedia.org/page/Pele`, last accessed April 15th, 2016

Table 5.6: **Mean MAP and MRR@5 Values for Association Rule Mining.** The table shows the mean MAP and MRR@5 values and their standard deviation in brackets for the Apriori algorithm. For an easier comparison, the last row shows the average results of the Random Forests algorithm.

		BTC 2014		DyLDO	
Rec. Method	Rec. Position	MAP	MRR@5	MAP	MRR@5
AR mining	*sts*	.70 (.17)	.81 (.11)	.51 (.22)	.55 (.26)
	ps	.71 (.12)	.75 (.14)	.54 (.27)	.55 (.28)
	ots	.75 (.12)	.80 (.13)	.49 (.24)	.57 (.29)
	Total Average	.72 (.14)	.79 (.12)	.51 (.25)	.56 (.28)
L2R (Random Forests)	Total Average	.75 (.11)	.80 (.09)	.64 (.23)	.67 (.21)

DyLDO data and $\mathcal{X}^2(3) = 13,550$, **n.s.**, $p = .316$ calculated for the results based on the BTC 2014 data).

5.5.1.3 Aspect (iii): Differences between Learning To Rank Algorithms

Comparing the three most competitive L2R algorithms (only when also using the SLP-feature), it was observed that Coordinate Ascent and Random Forests outperform LamdaMART. Based on the BTC 2014 data set Coordinate Ascent and Random Forests achieved on average MAP ≈ 0.77 and MRR@5 ≈ 0.8. LamdaMART achieved only MAP ≈ 0.60 and MRR@5 ≈ 0.67). Using the DyLDO data set, the differences between the L2R algorithms are not as noticeable. However, a Friedman test ($\mathcal{X}^2(3) = 14,000, p = .001$) shows that these differences are still significant. A subsequent pair-wise Wilcoxon signed-rank test with Bonferroni correction shows that the Random Forests algorithm provides significantly better recommendations than the Coordinate Ascent algorithm ($Z = -2.492, p = .013$) as well as than LambdaMART algorithm ($Z = -4.237, p < .001$). Out of the other L2R algorithms in the RankLib library, only the MART [36] algorithm was able to achieve similar, but slightly inferior results. Three other algorithms, i.e., *RankNet* [16], *RankBoost* [35], and *ListNet* [20], resulted in a recommendation quality of MAP ≈ 0.4 and MRR@5 ≈ 0.5 when using all features. The L2R algorithm *AdaRank* [88] had the lowest recommendation quality with an MAP ≈ 0.1 and an MRR@5 ≈ 0.2 when using all features.

5.5.2 Prediction Accuracy of Recommendations based on Association Rule Mining

In contrast to Learning To Rank, the Association Rule mining approach uses solely feature f_5, i.e., the SLP-feature to calculate vocabulary term recommendations. Figure 5.4 illustrates the recommendation quality based on the Mean Average Precision (MAP) and the Mean Reciprocal Rank at the first five positions (MRR@5).

Again, it can bee seen that AR mining achieves higher results on the BTC 2014 data set (median MAP value MAP ≈ 0.77) than on the DyLDO data set (median MAP value

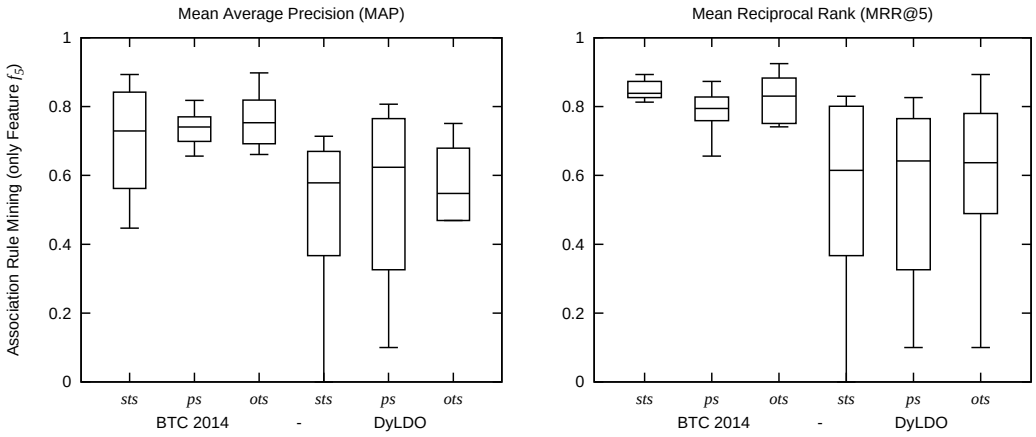


Figure 5.4: **MAP and MRR@5 Values based on Association Rule Mining.** The figure shows the prediction accuracy of Association Rule mining via the Apriori algorithm that uses solely the SLP-feature to compute vocabulary term recommendations.

MAP ≈ 0.57). This difference becomes even more visible when looking at the MRR@5 values. Whereas the BTC2014 box plots are short, i.e., the differences between the highest and the lowest values as well as the differences between the upper quartile and the lower quartile are quite small, the DyLDO box plots are comparatively large. This shows once more that using the BTC 2014, the recommender achieves a high level of agreements between the ten folds.

In addition, Table 5.6 denotes the average values regarding MAP and MRR@5 for both data sets and confirms this observation. The total average MAP and MRR@5 values (MAP $= 0.72$ and MRR@5 $= 0.79$) for the BTC 2014 data set are more than 20% higher than the average values for the DyLDO data (MAP $= 0.51$ and MRR@5 $= 0.56$).

Table 5.6 also shows the comparison in the MAP and MRR@5 values between AR mining and the best performing L2R algorithm Random Forests (using all five features). One can observe that the numbers are quite similar, when performing the evaluation using the BTC 2014 data set. For both recommendation approaches, the average MAP and MRR@5 values are about MAP ≈ 0.75 and MRR@5 ≈ 0.8. The difference in the results are more noticeable when performing the evaluation on the DyLDO data set. AR mining is able to achieve a prediction accuracy of MAP ≈ 0.51 and MRR@5 ≈ 0.56. On the contrary, L2R achieves a quality of MAP ≈ 0.64 and MRR@5 ≈ 0.67. The L2R-based approach performs 13% better based on the evaluation using the DyLDO data set. However, a Wilcoxon-signed rank test shows that the differences between both approaches are not significant when performing the evaluation on the BTC 2014 data set ($Z = -1.471$, **n.s.**, $p = .141$) as well as on the DyLDO data set ($Z = -1.183$, **n.s.**, $p = .237$).

5.6 Discussion

5.6.1 Discussion of the Results

The experiments show that using the SLP-feature significantly improves the quality of RDF vocabulary term recommendations when using the Learning To Rank. The reason why Random Forests outperforms the other L2R algorithms can be explained by the fact that only binary relevance values are used, i.e., a recommendation candidate is either relevant or irrelevant. Point-wise approaches outperform the other L2R approaches in such scenarios, especially if there are just one or a few relevant recommendation candidates for most queries [18]. Moreover, the improvement when using the SLP-feature is not as noticeable when performing the evaluation with the data from the DyLDO data set. Further investigations showed that the training sets based on BTC 2014 and DyLDO had differences regarding their query-SLPs. The data reveals that the training sets based on BTC 2014 contained 37% more relevant recommendation candidates with an SLP-feature value greater than zero ($f_5 > 0$) compared to the training sets based on DyLDO. This means that the ranking model computed from the DyLDO data does not consider the SLP-feature as much as the ranking model computed from the BTC 2014 data.

Such an observation is affirmed by the results based on calculating term recommendations with Association Rule mining, which uses exclusively the SLP-feature. For the evaluation based on DyLDO, The MAP and MRR@5 values are much lower compared to the values achieved by the L2R approach. A deeper analysis showed that using AR mining frequently resulted in empty recommendation lists. In other words, for many relevant recommendation candidates, the SLP-feature was $f_5 = 0$. In the AR mining case, this means these candidates are not included in the results list. Should this occur for all relevant candidates, the MAP and MRR@5 values are MAP $=$ MRR@$5 = 0$. The L2R-based approach benefits from the other four features, compared to the AR-based approach. Using features f_1 to f_4 in addition to feature f_5 helps to rank the relevant candidates in the upper middle area of the results list, as in various cases these candidates were either quite popular or from an already used vocabulary. The AR-based approach does not use these features. Therefore, the results lists contain zero vocabulary terms and the corresponding MAP and MRR@5 values are zero. As a result, the total MAP and MRR@5 values decreased. Using BTC 2014 as evaluation data, the number of SLPs simulating data sets on the LOD cloud is twice as high compared to the number of such SLPs using DyLDO. In general, the larger the data for calculating the feature values, the more representative are the generated results [18]. Therefore, it can be assumed that the results based on using BTC 2014 are more representative than the results of the evaluation based on DyLDO.

5.6.2 Threat to Validity

A potential threat to the validity of the experiments is that the relevant recommendation candidates that have been extracted from a query-SLP and hidden before providing this query-SLP as input for TermPicker (cf. Section 5.4.1) are the only relevant candidates. This leads to two major weaknesses. First, many recommendation candidates are identi-

fied as irrelevant, although they are appropriate considering the rdfs:domain and rdfs:range, the owl:equivalentClass, or other information. For example, let us assume the property foaf:made is selected and extracted from the query-SLP $slp_q = (\{\text{mo:MusicArtist}\}, \{\text{foaf:made}\}, \varnothing)$. This results in the query-SLP $slp'_q = (\{\text{mo:MusicArtist}\}, \varnothing, \varnothing)$ which is provided as input for TermPicker. The only relevant recommendation candidate for properties is the extracted and hidden foaf:made property, as it was used in the original SLP. Properties, such as mo:produced or mo:remixed are considered as irrelevant in the evaluation, although they are appropriate recommendations as outgoing properties from mo:MusicArtist. Thus, an employed recommendation approach may identify many *commonly used* vocabulary terms (with an SLP-feature value greater than zero) as irrelevant, which then can result in an insufficiently trained ranking model. Second, if SLPs from the training set contain terms that are used incorrectly, such as using the property foaf:made as data type property, and this term is randomly extracted and hidden, then it is very likely that the SLP-feature value would be zero; although the hidden term is considered as relevant though. Thus, the recommendation approach (in this case only L2R) learns to decrease the usefulness of the SLP-feature.

These limitations can be addressed in a two-fold way: First, one can use a gold-standard data set containing all relevant candidates for a given query. However, such a data set does not exist, so that it has to be elaborated. This is quite cumbersome and might not reflect real-life data. Second, one can address the limitations by requiring human judgment whether a recommendation candidate is relevant. This can be done via crowd sourcing approaches, in which some person, i.e., an anonymous user, receives the query and the list of recommendations. The person then decides which terms in the list are relevant and which terms are not relevant. However, there are not many people with this kind of domain-specific knowledge. A lot of domain experts from various domains would be needed for judging whether a candidate is relevant or not. Such an effort is however not feasible when performing ca. 50 or more evaluations. Due to that reason, human judgment is not used in an offline evaluation.

5.7 Conclusion and Summary of the Chapter

As presented in this Chapter, TermPicker constitutes a novel approach for recommending vocabulary terms based on the information how other data providers on the LOD cloud have modeled their data. Key to this approach is the notion of schema-level patterns (SLPs), which form an effective way to represent which sets of properties other data providers on the LOD cloud have used to connect which sets of RDF classes. Given a query-SLP, TermPicker uses this information and recommends further terms that other data providers have used together with the terms in the query-SLP. The set of features (among them the SLP-feature) presented is used by a Learning To Rank as well as by an Association Rule mining approach to compute a ranking model. It was demonstrated that using the SLP-feature increases the prediction accuracy by about 35% compared to the baselines of recommending vocabulary terms from popular vocabularies and recommending terms from the same vocabulary. The Mean Average Precision (MAP) is approximately MAP ≈ 0.75 and the Mean Reciprocal

Rank at the first 5 positions (MRR@5) is about MRR ≈ 0.8 when using the SLP-feature to calculate term recommendations. Finally, based on the evaluation design that automatically assesses the relevance of a recommendation candidate, point-wise Learning To Rank (L2R) algorithms provide better results than pair-wise or list-wise L2R algorithms. The reason for this is that point-wise algorithms generally tend to achieve higher results when the recommendation candidates have a binary relevance level [60].

Results obtained in this Chapter illustrate the prediction accuracy based on an offline evaluation without users. To finally evaluate whether the L2R-based or the AR-based approach aids a Linked Data engineer in reusing RDF vocabulary terms, one needs to perform an online evaluation with actual users interacting with TermPicker in a real-life scenario.

6 Online Evaluation of TermPicker's Recommendations

As noted in Section 2.4, recommender systems should to be evaluated via an offline evaluation as well as via an online evaluation. High Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR) values indicate TermPicker's potential, but ultimately, the recommendation system has to benefit real users. Therefore, it is necessary to investigate whether these real end-users accept TermPickers's recommendations. That applies all the more since both recommendation approaches achieve similar results in the offline evaluation. However, it is challenging to determine which type of online evaluation to use. One could evaluate both approaches via an A/B-Testing (cf. Section 2.4.2.1), but to this end, one would need a large amount of already existing users. Unfortunately, TermPicker does not have this amount of active users available. Another option is to conduct a user study which provides an environment to examine TermPicker in a practical setting without the need to acquire a lot of users. At the same time, it makes it possible to investigate how users interact with the recommender, and the results of a study provide a very good estimation how satisfied real users might be when integrating the investigated recommender in a full-functioning system [77].

This chapter examines TermPicker's recommendation quality via a user study observing how LOD engineers interact the with term recommendations. The user study compares the recommendations based on the best performing L2R algorithm from the offline evaluation, i.e., the *Random Forests* algorithm, with recommendations based on AR mining. Additionally, both approaches are compared to a baseline of no recommendations.

To evaluate TermPicker via a user study, both the L2R-based and the AR-based recommendation approaches are integrated into a graphical data modeling tool (cf. Section 6.1). Data engineers that deal with Linked Open Data on a regular basis have been invited to participate in the study. The participants were asked to model three data sets as LOD with the requirement to reuse existing vocabulary terms (cf. Section 6.2). The user study follows a within-subjects design, such that each participants is asked to complete all three modeling tasks. The data for each of the modeling tasks is taken from three different domains (from the Music, Museum, and Product Offer domain), in order to eliminate any learning effect. The participants receive a partial LOD model for all tasks, which they are asked to complete. For one task, they receive recommendations from the L2R, for another task they receive recommendations from AR mining, and for the third task they receive no recommendations. The no-recommendation condition is the baseline for comparison. Assessing the recommendation quality is measured based on the participants' *effort* needed to complete a modeling task as well as on the *quality* of the resulting LOD model. The effort is comprised of the task-completion time and the number of selected recommendations, which

is referred to as the *recommendation acceptance score*. The quality of the resulting LOD representation is assessed by comparing the participants' LOD representation against representations generated by five different LOD experts, i.e., to see whether the participants choose the same vocabulary terms as the experts for representing the data as LOD. Additionally, the participants are asked to rate the two recommendation approaches and to report their level of satisfaction regarding the recommendations on a 5-point Likert-scale.

For recruiting participants, data engineers from the Information Sciences Institute (ISI), USA, and from GESIS, Germany, were invited to the lab via e-mail or by directly approaching them in person (cf. Section 6.3). Care was taken to ensure that one group of participants were novices to Linked Open Data and the utilized data modeling tool and another group of participants were rather experienced with LOD and the data modeling tool. This allows the experimenter for evaluating whether TermPicker is able to help unexperienced LOD engineers to model high-quality LOD representations.

The results indicate that the recommendations based on AR mining are more satisfying to the participants than the L2R-based recommendations. The task completion times for AR (4:13 minutes) proved to be significantly faster than the completion times for L2R (5:41 minutes) and the baseline (5:26 minutes). The participants accepted term recommendations from AR most of the time (4.85 out of 7 times), but not from L2R (2.05 out of 7 times). AR-based recommendations also lead to high quality results (75% of the selected terms were also chosen by the experts), whereas the L2R-based recommendations do not perform well (only 58% of the selected terms were also chosen by the experts). Even the participants lacking experience in LOD and with the data modeling tool achieved high-quality results using AR-based recommendations. The user satisfaction survey resembles similar results and indicates a preference for the AR-based recommendations (cf. the results in Section 6.4 and the discussion in Section 6.5).

6.1 Apparatus

To perform a user study evaluating the quality of TermPicker's recommendations, the recommender was integrated into the graphical data modeling tool Karma [55].[79] Karma is an interactive tool that enables data engineers to model data sources as Linked Data. It provides a visual interface to help data engineers to incrementally select RDF vocabulary terms and to build a model for their data. Figure 6.1 depicts a screenshot of Karma with a partially specified model. Karma's user interface is divided into two parts, which are separated by the dotted line in Figure 6.1 (note: only for easier understanding; the dotted line is not part of Karma). The bottom part shows a table with the data being modeled. It contains the column header, the instances that are contained in the tables cells, and the diversity of these instances. For example, three green diversity bars indicate that the column contains three different resources. The top part (above the dotted line) shows a graphical representation of the LOD model. The dark ovals represent classes, and the labeled edges represent properties. The number at the end of the dark oval label is useful for distinguishing different columns

[79] `http://usc-isi-i2.github.io/karma/`, last accessed February 14th, 2017

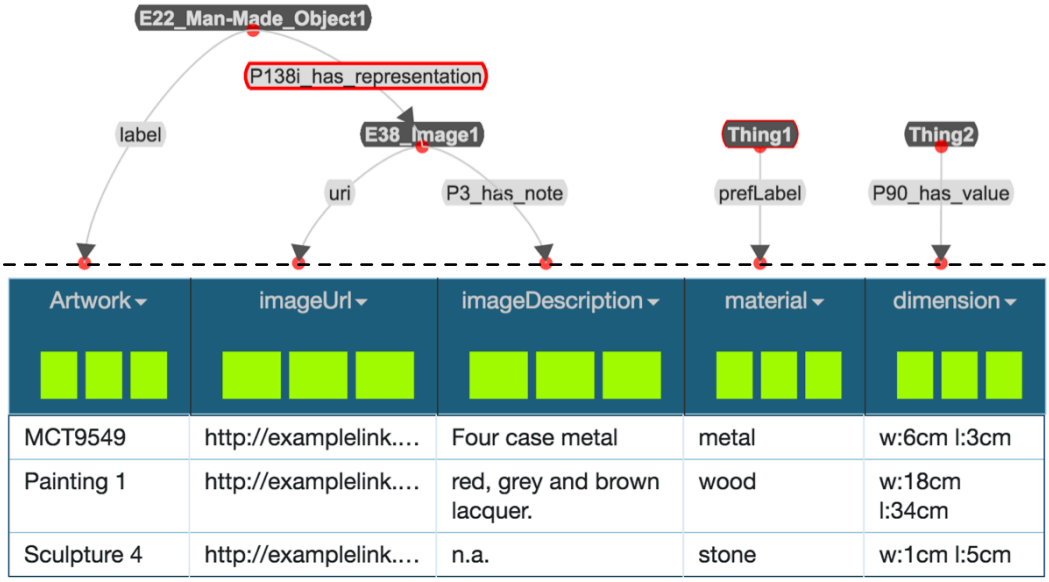


Figure 6.1: **Karma User Interface.** The bottom part (below the dotted line) shows the table data being modeled and the top part shows the data's graph representation including RDF classes (the dark ovals) and properties (the labeled edges). The edges connect classes either to table columns (Semantic Types) or to other classes (object properties)

that are modeled with the same class. For example, there can be a column containing images of artworks and another column containing images of the artist. In the LOD model, resources in these two columns can be both of type ecrm:E38_Image, but in Karma's graphical representation, they are differentiated with E38_Image1 and E38_Image2 for an easier overview. The edges connecting classes to the columns in the data source are called *Semantic Types*, as they specify the semantics of the data in a column. In general, a semantic type can either be a pair consisting of a datatype property and the domain of the data property (in the meaning of rdfs:domain), or a singleton class . For example, in Figure 6.1, the semantic type for the third column labeled "imageDescription" specifies that its contents are notes of resources of type ecrm:E38_Image. It is represented using the pair (ecrm:P3_has_Note, ecrm:E38_Image) as semantic type. The semantic type *uri* is a singleton class specifying that the column contents are the resource URIs for the resources of type ecrm:E38_Image. If such an URI singleton class does not exist for an RDF class, its resources are represented via blank nodes, e.g., the contents of the *Artwork*, *material*, and *dimension* columns are represented via blank nodes. The edges connecting the dark ovals represent object properties specifying the relationships between the instances of the corresponding classes. For example, the edge labeled P138i_has_representation between E22_Man-Made_Object and E38_Image specifies that resources of type ecrm:E22_Man-Made_Object are connected to resources of type ecrm:E38_Image via the property ecrm:P138i_has_representation.

To build LOD models in Karma, users may click on the edge labels or on the classes to edit a model. When users click on an edge label, Karma presents *commands* to let users change the property depicted in the label, the source class of the edge, and the destination class of the edge (this last option is only available for object properties). Similarly, when users click on a class, Karma shows commands to change the class depicted in the oval and to add incoming or outgoing edges.[80]

6.2 Experiment Setup

Based on the discussion in Section 2.4.3 (within-subjects design vs. between-subjects design), a within-subjects user study was chosen. It eliminates the assumption that the split of users is responsible for the results, while it also provides the possibility to ask comparative questions about the different recommendation approaches. In addition, it does not require a high number of participants compared to between-subjects experiments.

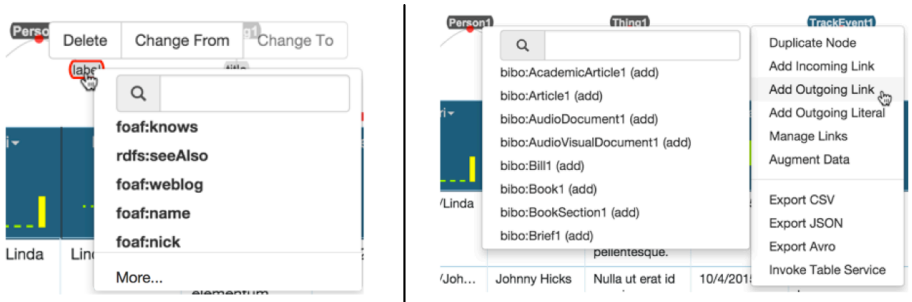
In the following, the usage of Karma for the evaluation is illustrated in Section 6.2.1. Section 6.2.2 explains the evaluation procedure, and Section 6.2.3 comprises the tasks that the participants are asked to complete. The metrics that are used to evaluate the overall quality of the recommended RDF vocabulary terms are described in Section 6.2.4.

6.2.1 Using Karma for Evaluation

For the evaluation, Karma was configured in a way so that the experimenter can select the recommendation approach for a participant and for a modeling task. In such a setup, Karma can be configured to offer no recommendations, recommendations using L2R, or the AR-based term recommendations. When users perform editing operations to change or add properties or classes, Karma uses the pre-configured recommendation approach (including no recommendations) to populate its menus that users see for selecting vocabulary terms. Figure 6.2 illustrates the menus and the underlying operations employed in the user study. In the menu shown in Figure 6.2 (a), a Karma user can delete a labeled edge, change its outgoing and incoming RDF class, and select an RDF property representing the meaning of the labeled edge. The property selection menu has two sections, one showing the recommended properties (marked in bold) and one showing all properties, if the user clicks on "More". In addition, Karma offers a search box which filters terms containing the query string. The menu shown in Figure 6.2 (b) also contains two parts, whereby the left part of the menu is comparable to the property selection menu. It incorporates the recommended RDF classes for the selected dark oval, all available classes (once the user clicks on "More"), and a search for string-based filtering. The right part of the menu can be used to establish links between the dark ovals (marked in the circle).[81] To this end, the Karma user clicks on "Add Outgoing Link" which presents the user a list of recommended properties as well as all available properties in the same manner as in the menu depicted in Figure 6.2 (a). To

[80] More detailed instructions how to install and how to use Karma can be found in the Karma Wiki: `https://github.com/usc-isi-i2/Web-Karma/wiki`, last accessed January 15th, 2017

[81] The other menu items are not relevant for this work and are therefore not explained here



(a) Menu for selecting recommendations

(b) Menu for adding outgoing links

Figure 6.2: **Operation Menus in Karma.** The figure depicts the operation menus in Karma that were used in the user study. In (a), the participants are able to select a property by clicking on an edge label, and in (b), they can select a class for a node, or add an outgoing link to connect classes

make it as easy as possible for the participants, the modeling tasks contained only outgoing links. For the user study, selecting a vocabulary term is defined as one operation in Karma, i.e., if the participants are asked to choose six vocabulary terms, they need to perform six operations.

6.2.2 Evaluation Procedure

The user study takes place in a controlled lab environment with 20 participants (their demographics are mentioned in Section 6.3). All participants were invited to the lab to conduct the modeling tasks. Each participant first performed a training modeling task to become familiar with the Karma user interface and the menus for selecting vocabulary terms. Once they complete the training (about 10 minutes), the participants work on modeling three different data sets (6 minutes each) (cf. Section 6.2.3). A Latin-square design was applied to arrange the tasks and the recommendation approaches, i.e., Karma was configured in a way so that the experimenter would be able to select one of the three recommendation conditions for each data set (no recommendation, L2R-based and AR-based recommendations). To assess the quality of the recommendations, the experimenter measured task completion time (via stop watch), the number of times participants chose a term from the recommendations vs. searching for the desired term manually, as well as the quality of the resulting LOD representation. Upon completion, the participants filled in a satisfaction questionnaire (Section 6.2.4).[82]

6.2.3 Modeling Tasks Used in the Study

The modeling tasks comprise data from three disparate domains. The domains were chosen to eliminate any chance of overlap in the classes and properties used to model them. This

[82] Accompanying material and the modeling results can be found at `https://github.com/WanjaSchaible/termpicker_karmaeval_material`, last accessed September 15th, 2016

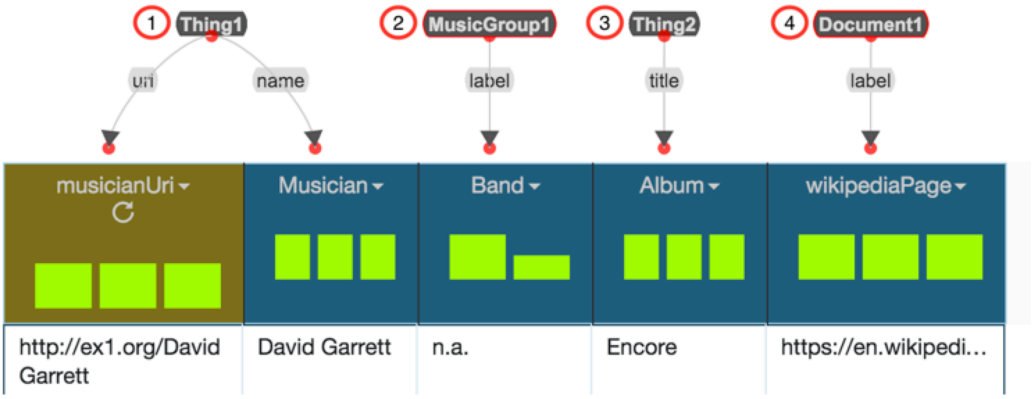


Figure 6.3: **Modeling Task with Data from the Music Domain.** The figure depicts the data from the Music Domain for a modeling task. The participants were asked to replace the owl:Thing classes and the rdfs:label properties, as well as to establish connections from class (1) to the classes (2) to (4).

minimized learning effects across the modeling tasks. One data set is from the music domain, another data set is from the domain of museums, and the third data set contains data about offers and products. The music data set includes information about musicians, their recordings, and their wikipedia page. The data set on museum objects includes information about artworks, materials, dimensions, and images of artworks. The product offers data set comprises information about the seller, location, and offered items.

One concern with the user study is that modeling three full data sets takes a long time. To mitigate this problem, the participants were given partially-defined models. These models already contained a few columns annotated with corresponding Semantic types, i.e., with RDF classes and datatype properties specifying the contents of the table cells. The participants were given a scenario, in which they were asked to complete these partially-defined models. In the following, the data and the three modeling tasks are explained in more detail.

6.2.3.1 Modeling Data from the Music Domain

Figure 6.3 illustrates the modeling assignment for the music data set. The data set contains information about the musician's name, their band's name, albums, wikipedia page, as well as a column containing the URI for each musician. The elements above the table represent the partially-defined model of the data. The instructions to the participants explained that the model is not finished yet and asked them to complete the model. First, several dark ovals (representing the class of resources in a column) were modeled with owl:Thing, which is the most imprecise class for a resource, as every resource is considered to be a thing. The same applies for modeling datatype properties with rdfs:label, as every literal value can be a label. Both modeling decisions are correct, but are likely to lead to a huge loss of semantics in the data, e.g., using rdfs:label instead of foaf:name to define that a datatype property links to a literal value containing a name. Second, no relationship between the classes is established

yet. The participants were given the assignment to finish the LOD representation. The following text snippet was provided:

> You see a data set containing data on musicians. A musician can be part of a band, record an album, and can have a Wikipedia page. Your assignment is to replace all nodes (classes and properties) that are of type **Thing** or marked **label** with "better fitting" vocabulary terms, if possible. In addition, you should connect the nodes to specify their relation to each other.

The instructions for this task are as follows (the digits within brackets represent the semantic type nodes labeled with the same digits in Figure 6.3):

(a) Find a better RDF class for (1) specifying that the entities in the column "Musician" are musicians

(b) Find a better property for (2) specifying the name of a music band

(c) Find a better RDF class for (3) specifying that the entities in the column "Album"are the musician's records

(d) Find a better property for (4) specifying the URL to a Wikipedia page

(e) Connect (1) and (2) to specify that a musician is part of a band

(f) Connect (1) and (3) to specify that a musician recorded a album

(g) Connect (1) and (4) to specify that a musician has a Wikipedia page

Each modeling assignment (a) to (g) can be performed by one Karma operation. Thus, in this modeling task, the participants have to perform seven Karma operations.

6.2.3.2 Modeling Data from the Museum Domain

Figure 6.4 illustrates the modeling assignment for the museum data set. The data set contains information about Artworks. This information comprises an image URI of the Artwork, the description of that image, the material the artwork is made of, and the dimensions of the artwork. Again, the participants' assignment was to finish the model, as it contained representations via owl:Thing and rdfs:label but no relationships. In detail, the assignment is:

> You see a data set containing data on museum items. A museum item is an artwork that is of some specific size, is made out of some material, and has a corresponding image. Your assignment is to replace all nodes that are of type **Thing** or **label** with "better fitting" vocabulary terms, if possible. In addition, you should connect the classes to specify their relation to each other.

The instructions for this task, including which type of relationship should be modeled between the classes, are as follows (the digits within brackets represent the nodes labeled with the same digits in Figure 6.4):

Figure 6.4: **Modeling Task with Data from the Museum Domain.** The figure depicts the modeling task from the Museum Domain. The participants were asked to replace the owl:Thing classes and the rdfs:label properties, as well as to establish connections from class (1) to the classes (2) to (4).

(a) Find a better property for (1) specifying that an artwork item has a specific name

(b) Find a better property for (2) specifying that an image has a specific description

(c) Find a better RDF type for (3) specifying that the entities in the column "material" describe some material

(d) Find a better RDF type for (4) specifying that the entities in the column "dimension" describe a dimension

(e) Connect (1) and (2) to specify that an image represents an artwork

(f) Connect (1) and (3) to specify that the artwork is made out of some material

(g) Connect (1) and (4) to specify that an artwork has some specific length and width

To complete this modeling task, the participants have to perform seven Karma operations.

6.2.3.3 Modeling Data from the Offers and Products Domain

Figure 6.5 illustrates the modeling assignment for the offers and products data set. The data set contains information about a product that is offered at a specific place for a specific price. The entire information comprises the URL of the offer, its title, its price, the location where it can be purchased, as well as the manufacturer of the product that is offered. The participants' assignment is the same as in the previous tasks. The assignment is stated as follows:

> You see a data set containing data on items that are offered for sale. An offer has a specific title, a price, and a place where the offer is taking place. The item that is offered is a product that has a manufacturer. Your assignment is to replace all nodes that are of type **Thing** or **label** with "better fitting" vocabulary terms,

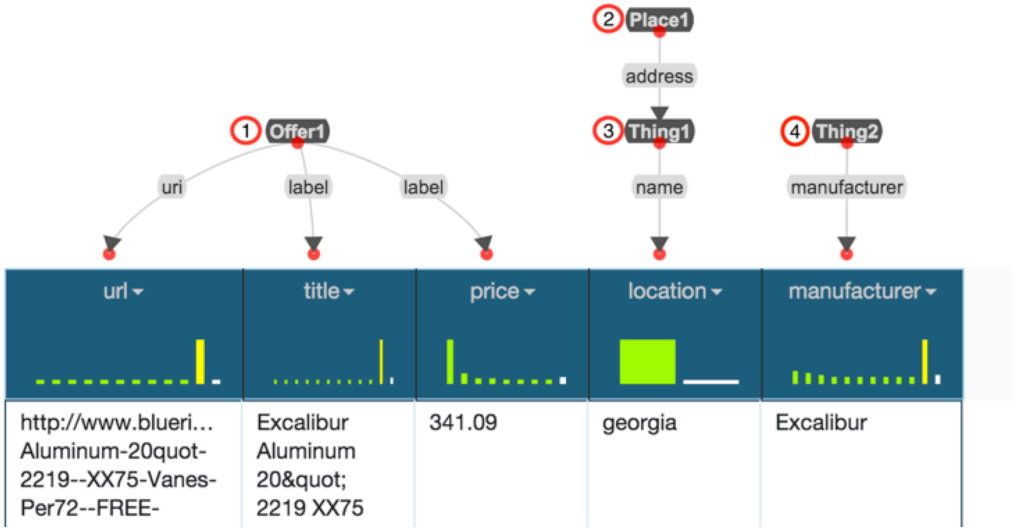


Figure 6.5: **Modeling Task with Data from the Offer and Products Domain.** The figure depicts the modeling task from the Offer and Products domain. The participants were asked to replace the owl:Thing classes and the rdfs:label properties, as well as to establish connections from class (1) to the classes (2) and (4).

if possible. In addition, you should connect the nodes to specify their relation to each other.

The instructions for this task, including which type of relationship should be modeled between the classes, are as follows (the digits within brackets represent the nodes labeled with the same digits in Figure 6.5):

(a) Find a better property for (1) specifying the title of an offer

(b) Find a better property for (1) specifying the price of an offer

(c) Find a better RDF type for (3) specifying that the entities in the column "location" describe the address of a place

(d) Find a better RDF type for (4) specifying that the entities in the column "manufacturer" are manufacturer of a product

(e) Connect (1) and (2) to specify that an offer is available at a location

(f) Connect (1) and (4) to specify that it is an offer of a product

For this modeling task, the participants have to perform only six Karma operations, as the structure of the model is little more complicated.

6.2.4 Evaluation Measurements

To evaluate the recommendation approaches, the following aspects were measured:

- Task completion time, i.e., the time a participant needs to complete a modeling task.
- Recommendation acceptance score, i.e., the number of times participants chose a term from the list of recommendations vs. manual search for a term.
- Quality of the resulting LOD representation assessed by comparing the representation to a LOD representation modeled by a LOD expert.
- Participants' satisfaction with the recommendations.

Task Completion Time. One way to estimate the quality of the recommendations is to measure the amount of time a participant needs for completing a modeling task. Using a stopwatch, the experimenter begins timing once the assignment has been fully read and the participant has begun the modeling process. The assumption is that the less time participants need to finish a task, the more appropriate the recommendations. The reason for this assumption is that if a recommendation approach provides outstanding suggestions, the participants do not need long to search for a fitting vocabulary term. To avoid exhausting the participants, each modeling task was limited to six minutes. Should a participant need more than six minutes, it is assumed that the recommendation approach did not provide satisfactory suggestions to aid the participant in deciding which vocabulary term to use.

Recommendation Acceptance Score. The recommendation acceptance score specifies the number of times the participants choose a recommended vocabulary term from the menu of suggestions, as opposed to searching for terms in the menu containing the full list of vocabulary terms. For example, if a participant selects four recommended terms out of six Karma operations, the recommendation acceptance score would be $4/6 = 0.67$. As each modeling task comprises six or seven operations, choosing a recommended vocabulary term seven times in one assignment is the maximum score, whereas choosing a recommendation zero times is the minimum score. The expectation is that participants would search the full list of terms when the recommended terms are inadequate. This way, the recommendation acceptance score would be very low and indicate that the recommendation approach did not provide satisfying suggestions. On the contrary, a high recommendation acceptance score indicates that the quality of recommendation is very high, as participants do not need to search for appropriate terms in the list of all available classes and properties. Thus, a high score indicates the adequacy of recommendations.

Quality of the Resulting Linked Open Data. To measure the quality of the resulting LOD representations, a panel of five ontology engineering experts was asked to construct a gold-standard model for each data set. Each of the modeling tasks was sent to each expert, together with a proposed model that was created by the principal investigator of this study. The experts provided feedback on the proposed model and suggested additional vocabulary

terms to describe a data entity or a relation between data entities. Some also suggested replacing several of the vocabulary terms that have been proposed with better fitting ones. All terms that were considered adequate to represent a type of a resource or a connection between resources are included in the gold standard. Thus, the gold standard consists of a set of vocabulary terms for each modeling operation. For example, instruction (b) from the modeling task on data from the offers and products domain asks to specify a better fitting property to represent that the data comprises the price of an offer. The property schema:price was part of the suggested model. All five experts confirmed its adequacy, but some also suggested to use gr:hasCurrencyValue from the Good Relations[83] vocabulary. Therefore, both properties constitute the gold standard. A participant's selection of a vocabulary term for a modeling operation is "correct", if the chosen term is in the set of vocabulary terms defined in the gold standard for that operation. The quality score for each data set is the fraction of correct modeling operations. For example, if there are seven Karma operations needed to complete a modeling task, it means that the participants need to select seven vocabulary terms. If the participant selects six out of these seven terms, which were also selected or approved by the five experts, the result would be $6/7 = 0.86$. The higher the ratio, the higher the quality of the resulting data representation.

Participants' Satisfaction with the Recommendations. Another aspect defining the quality of recommendations is assessed by asking the participants direct questions about their satisfaction with the provided recommendations. After all modeling tasks, the participants are told which recommendation approach was used for which modeling task. Subsequently, the participants are asked to specify their satisfaction for each recommendation approach individually on a 5-point-Likert Scale (from "very dissatisfied" to "very satisfied"). In addition, they are asked to compare the two investigated approaches (i.e. AR vs. L2R) via a rating and a ranking. The rating is performed via questioning the participants about their impression how the AR-based recommendations performed against the L2R-based recommendations. Choices available ranged from "much worse" to "much better" on a 5-point-Likert scale. In addition, both recommendation approaches and the no-recommendations approach were ranked by the participants from "best" to "worst".

6.3 Participants

Overall, $n = 20$ participants (5 female) took part in the user study, all working with LOD in academia and two also in industry. The participants' professions ranges from master students (2) to research associates (14) to post doctoral researchers (3) and professors (1) with an average age range between $30 - 35$ years ($SD = 7$ years). On average, the participants have worked for 3.05 years with LOD. However, they rated their own expertise consuming LOD as moderate ($M = 2.8, SD = 1.1$) and publishing LOD as little ($M = 2.1, SD = 1.6$) on a 5-point-Likert scale from 1 (*none at all experienced*) to 5 (*expert*). The same applies to the participants' knowledge about the domains of the data from the three modeling tasks,

[83] http://www.heppnetz.de/ontologies/goodrelations/v1, last accessed February 14th, 2017

Table 6.1: **Average Task Completion Time in Minutes.** The table shows the average time the participants needed to complete the modeling tasks followed by the standard deviation in brackets.

Approach	Music domain	Museum domain	Product Offer domain	Average
L2R	5:31 (0:41)	5:48 (0:24)	5:42 (0:25)	5:41 (0:30)
AR	4:50 (1:10)	4:37 (1:02)	3:16 (1:02)	4:13 (1:03)
No Rec	5:22 (0:59)	5:28 (0:37)	5:34 (0:42)	5:28 (0:47)

that is with $M = 2.1$ ($SD = 1.1$) rather low across the three domains. Only three participants consider themselves highly experienced or expert in LOD (4 or 5 on the Likert-scale). The majority ($57,3\%$) consider themselves to have moderate knowledge. Regarding Karma, 13 participants had never used it, while seven individuals declared themselves to be experts or to have high knowledge about it. Overall, the participants had a moderate knowledge of Linked Open Data, but were quite inexperienced regarding LOD publishing and the domain of the modeling tasks' data.

The participants were recruited from the Information Integration group at the USC[84] Information Sciences Institute (ISI), USA, and from GESIS - Leibniz Institute for the Social Sciences, Germany. ISI participants were familiar with Karma but not with the recommendation systems being evaluated in this work, while participants from GESIS had never used Karma before. All participants were not familiar with the recommendation approaches, but a majority did have moderate experience in working with Linked Open Data. Participants were acquired in person or via personal email inviting them to take part of the study within a laboratory or a private office space, in which solely the experimenter was present. An informed consent form was signed by each participant prior to the experiment; no compensation was offered to anyone involved.

6.4 Evaluation Results

The results of the user study[85] are illustrated and discussed based on the measurements considering the efficiency, quality, and level of satisfaction (cf. Section 6.2.4).

6.4.1 Task Completion Time

To obtain the average time participants needed to complete a modeling task, recorded times from participants who used the same recommendation approach were summed and then divided by the number of participants. For example, if seven out of the 20 participants modeled the music data using AR mining, the times these seven participants needed to complete modeling the music data are summed up and then divided by seven, the number of participants. Table 6.1 illustrates these calculated average times for each of the three data models and the combined average time.

[84] University of Southern California

[85] Research data is available at `http://dx.doi.org/10.7802/1206`, last accessed February 1st, 2017

Table 6.2: **Recommendation Acceptance Score.** The table shows the score representing the average number of times the participants selected a recommended vocabulary term out of the number of operations per task (standard deviation in brackets).

Approach	Music domain	Museum domain	Product Offer domain	Average
L2R	0.28 (0.89)	0.34 (0.91)	0.28 (0.81)	0.31 (0.88)
AR	0.62 (0.51)	0.69 (0.69)	0.88 (0.95)	0.73 (0.95)

The results show that the AR-based recommendations led to the fastest completion times (4:13 minutes). This was more than a minute faster than the baseline of having no recommendations (5:28 minutes) and about 90 seconds faster than using recommendations based on L2R (5:41 minutes). When given no recommendations, ten participants needed more than the allowed six minutes. Nine participants were unable to complete a modeling task withing six minutes using L2R-based recommendation, and only three participants did not finish in time when having AR-based recommendations. Applying a Friedman test, the differences observed between the task completion times are significant ($\mathcal{X}^2(3) = 15.083$, $p = .001$). However, the Wilcoxon signed-rank test (with a Bonferroni correction applied) shows that the difference in recommendations based on L2R and having no recommendations at all is not significant ($Z = -1.256$, **n.s.**, $p = .209$), whereas having recommendations based on AR mining is significantly preferred (compared to L2R: $Z = -3.220$, $p = .001$; compared to the baseline of no recommendations: $Z = -3.018, p = .003$).

Noteworthy, even surprising, was that users took less time to complete tasks with the baseline system than with L2R-based recommendations. The survey reveals that participants were dissatisfied with the recommendations provided by L2R (details on that manner in cf. Section 6.4.4). The results suggest that participants spent time evaluating these recommendations and proceeded to search for terms as they would have done in the baseline system, thus requiring more time overall. The L2R-based recommendations use multiple features, including the popularity of a term and whether or not it is from an already used vocabulary. Thus, the recommendations sometimes include terms that are popular or are taken from a commonly used vocabulary, but that do not fit the desired semantics. For example, instead of suggesting the datatype property foaf:name to specify the name of a person, it suggests the property foaf:knows. This occurred regardless of the domain of the data.

6.4.2 Recommendation Acceptance Score

The average number of selected recommendations, displayed in Table 6.2, supports the observations from Table 6.1. In total, participants did not use the L2R-based recommendations very often (on average 2.05 times out of 6.67 indicating a recommendation acceptance score of $2.05/6.67 = 0.31$), which evidently leads to longer completion times, as the participants more often searched manually for the needed terms. The difference between the two recommendation approaches is most noticeable when participants modeled data from the product offer domain. Whereas the participants accepted a L2R-based recommended terms only 1.7 out of 6 times (recommendation acceptance rate of 0.28), they used terms recommended

Table 6.3: **Quality of the Resulting RDF Representation.** The table shows the average number of vocabulary terms that were also chosen by the LOD experts. The standard deviation is shown in brackets.

Approach	Music domain	Museum domain	Product Offers domain	Average
L2R	0.57 (0.64)	0.55 (1.05)	0.65 (0.83)	0.58 (.14)
AR	0.70 (0.88)	0.68 (1.03)	0.87 (0.64)	0.74 (.26)
No Rec	0.55 (0.83)	0.48 (0.91)	0.54 (0.70)	0.54 (.24)

by the AR-based approach 5.3 out of 6 times (recommendation acceptance rate of 0.88). Such a difference is not that large for modeling tasks from the other two domains. For the music domain it was 2 vs. 4.3 out of 7 times (recommendation acceptance score of 0.28 vs. 0.62) and for the museum domain, it was 2.4 vs. 4.9 out of 7 times (recommendation acceptance score of 0.34 vs. 0.69). Worth noting here is that participants selected more terms coming from the AR-based recommendations than from the L2R-based recommendations. In total, compared to the number of selected recommendations based on Association Rule mining (4.85 selected recommendations out of 6.67 possibilities with a recommendation acceptance score of 0.73), the recommendation acceptance score for the L2R-based recommender is lower (on average 2.05 out of 6.67 possibilities selected with a recommendation acceptance score of 0.31). This difference is significant, as the Wicoxon-signed rank test illustrates ($Z = -3.848, p < .001$).

6.4.3 Quality of the Resulting LOD Model

Table 6.3 shows the average number of *correctly* selected vocabulary terms for each modeling task and in total. As mentioned, participants selected a correct (or appropriate) term for an operation, if this term was also used by at least one of the five LOD experts. One can observe that the recommendations based on Association Rule mining (on average 0.74) helped the participants more in selecting correct vocabulary terms than the recommendations based on L2R (0.58) or not using any recommendations at all (0.54). This especially applies to modeling the data from the Product Offers domain which uses mainly the schema.org vocabulary.

When searching for an appropriate vocabulary term, participants tried to find a term that matched the description in the assignment. For example, a task from the Product Offers domain states "Find a better data type property specifying that the table column contains the price of an item". The participants were instantly looking for a term that contained the string "price" in its specification, such as hasPrice. Even if the recommendations did not contain such a term, the participants easily found the property schema:price using Karma's string search and incorporated it into the model. However, not every assignment was that easy. Another example from the task with the Product Offers data states "Find a better RDF class specifying that the table column contains the address of a place". Only in the conditions with AR-based recommendations, participants were able to find and choose the correct

vocabulary term which is schema:PostalAddress. L2R-based recommendations provided that term as a recommendation, but only at the 16th position of the results list.

In total, recommendations based on L2R did not provide as good results as AR-based recommendations. The quality of the resulting data is only practically as high as modeling data without any recommendations. Once again, participants were not able to find the desired terms in the set of recommendations, such that they used Karma's string search or they browsed through all vocabulary terms. Since participants also searched for the appropriate terms manually when they received no recommendations, the quality of the LOD representations of the tasks modeled with L2R-based recommendations approximates the quality of the LOD representations of the tasks modeled without any recommendations.

A Friedman test was applied to show that the differences between the quality of LOD representations are significant ($\mathcal{X}^2(3) = 13.125, p = .001$). However, a post hoc analysis using the Wilcoxon signed-rank test with Bonferroni correction applied shows that only the AR-based recommendations helped participants to achieve modeling LOD of significantly higher quality ($Z = -2.952, p = .003$ compared to the L2R-based approach, and $Z = -2.980, p = .003$ compared to using no recommendations). There was no significant difference between using L2R-based recommendation and having no recommendation at all ($Z = -1.303$, **n.s.** , $p = .193$).

6.4.4 Participants' Satisfaction with the Recommendations

The participants were asked various questions to assess their level of satisfaction regarding the recommendations based on AR mining and L2R. All who completed the tasks were first requested to indicate their general level of satisfaction on a 5-point Likert scale from "1 - Very Dissatisfied" to "5 - Very Satisfied", with "3 - Unsure" indicating a neutral position. As to whether they were satisfied with the recommendations based on L2R, the majority of the respondents stated that they were rather unsure ($M = 2.7, SD = 0.97$). The recommendations based on AR were considered rather satisfying ($M = 4.35, SD = 0.67$). A pairwise Wilcoxon signed-rank test shows that these this difference is significant ($Z = -3.602, p < .001$).[86]

In addition, the participants were asked to directly compare the two recommendation approaches. They were first asked to compare the AR-based recommendation approach to L2R using a 5-point Likert scale from "1 - Much worse than L2R" to "5 - Much better than L2R". A total of 19 participants stated that the AR recommendations were either somewhat better or much better than the L2R recommendations (10 participants stated "Much better than L2R", only one selected "3 - About the same"). These results are supported by the second question to rank the two recommendation approaches (and the no recommendations baseline) from "best" to "worst" regarding the helpfulness of the recommendations. Recommendations based on AR mining were unanimously considered to be most helpful (all participants ranked AR mining at the first position). A Friedman showed that the difference in the ranking positions is significant ($\mathcal{X}^2(3) = 32.500, p < .001$). A pair-wise analysis using the Wilcoxon signed-rank test with Bonferroni correction applied shows that the

[86] A prior Friedman test was not necessary, as only two variables (AR vs. L2R) were compared

recommendations based on AR mining are significantly favored compared to recommendation based on L2R and compared to no recommendations ($Z = -4.134, p < .001$ in both cases). However, the differences between getting recommendation based on L2R and getting no recommendations at all are not significant ($Z = -2.236,$ **n.s.** $, p = .025$).

6.5 Discussion

The results illustrate that the participants favor RDF vocabulary term suggestions produced by the AR-based recommendation approach. It helped them to complete the modeling tasks with less effort and the resulting LOD representations were of higher quality compared to using the L2R-based approach or no recommendations. Furthermore, the participants explicitly rated and ranked the AR-based approach better than the L2R-based approach. By monitoring the participants' modeling steps and how they selected vocabulary terms for reuse the experimenter observed that participants felt more confident when using the AR-based recommendations. The correct vocabulary terms in the list of L2R-based recommendations were rather *overlooked*, meaning that the list of recommendation contained a correct vocabulary term, but the participants were not able to identify it, as the list contained many, obviously incorrect, terms as well. As a result, they mentioned feeling unsure about whether the recommended terms were correct or not.

No matter how well-conducted a user study might be, it can still contain various threats to the validity of its results, e.g., there is always a possibility that carryover effects cannot be entirely eliminated. Therefore, a user study can only estimate how users interact with the investigated recommendation approaches in a full-functioning productive system.

A discussion of the results and the threat to the validity of the results of the user study are presented in more detail in the following.

6.5.1 Discussion of the Results

The results show a clear picture: vocabulary term recommendations based on Association Rule mining aid the participants more in reusing vocabulary terms than recommendations based on the machine learning approach Learning To Rank. Both approaches use the SLPs calculated from data sets on the LOD cloud in order to identify how other data providers model their data. However, the L2R-based approach uses additional features describing the popularity of a recommendation candidate and whether it is from an already used vocabulary. Based on all these features, the list of recommendations can also contain very popular terms from a vocabulary that is already used, but that are not used by others in a similar model. Nevertheless, recommendations based on these features can be irrelevant, such that the list of recommendations calculated by L2R contains many incorrect vocabulary terms around the correct ones. It appeared that participants felt unsure which recommendation was correct and which was not. If the engineer is not sure whether the recommended terms are appropriate for reuse, it can be assumed that she is more likely to overlook the relevant recommendations. In about 70% of all assignments across the three modeling tasks, the L2R-based approach recommended a correct vocabulary term at the third or fifth position

(sometimes also the tenth position or 17th position) of the list of all recommendations. The participants however, either skipped it, or were not sure whether to select it, as the list of recommendations contained various terms at a lower rank that seemed rather inappropriate. This uncertainty stopped most participants from selecting the correct vocabulary term from the recommendation list. On the contrary, the AR-based approach does not rely on the popularity of a vocabulary term, or whether it is from an already used vocabulary. It only exploits the information regarding which terms are used by other data sets on the LOD cloud in conjunction with the terms in the modeling task. Evidently, the lists of recommendations generated by the AR-based approach supported the participants in selecting the correct terms, even when participants were not aware of the existence of the correct term, they felt confident enough to select a term from the list. For example, the assignment to model an outgoing object property from mo:MusicArtist to mo:MusicGroup from the modeling task on music data, most participants were not aware of the existence of the property mo:member_of. Having no recommendations, they searched for "part_of" or "is_member", but with recommendations based on AR, they saw this property very quickly and selected it. One can therefore argue that this is the main reason why participants were not satisfied with L2R-based recommendations, and why the task completion time, the recommendation acceptance score, as well as the quality of the resulting LOD representation were significantly worse compared to using AR-based recommendations.

In summary, most participants mentioned that the L2R-based recommendations were generally useful, but they were unsure whether the recommended terms were appropriate for reuse. Recommendations based on AR mining alleviated this situation and supported the participants in selecting the correct terms. Therefore, Association Rules seem to provide more appropriate vocabulary term recommendations, and can be seen as the preferred option to choose when reusing vocabulary terms for LOD modeling.

6.5.2 Threat To Validity

Although care was taken, the within-subjects design chosen for the user study may still result in carryover effects, where the first modeling task potentially influences the other following tasks. For one, participants who were not familiar with Karma might have needed a longer time to complete the first modeling task, as they needed to get accustomed to Karma. Presumably then, they were more confident for the second and third modeling task, such that they complete the second and third task in less time. To address this issue, the participants started by modeling a data set from the publications domain for practicing with Karma and its menus which contain the recommended vocabulary terms. The user study started only once the participant felt ready to begin with the actual modeling tasks. Second, the participants might become tired during the user study. This can decrease their performance towards the final modeling task. To mitigate this problem, the time for each modeling task in the user study was limited to six minutes. This way, each participant was able to complete all three modeling tasks within 18 minutes, reducing the likelihood that their performance would suffer during the last modeling task.

A within-subjects designed user study also comprises the need to develop several tasks for participants. In this work, it was necessary to develop three different modeling tasks

containing data from disparate domains, while they must simultaneously be approximately equally difficult to model. In such a design, one can argue that the developed modeling tasks did not have the same complexity, and that one task was easier to model than the others. In addition, even when letting the participants practice with some data before beginning with the study, there is the possibility that it is easier to complete the third task, since the participant has by then become even more acclimated to the system. This problem is addressed by changing which recommendation approach is used with which modeling task in which order based on a Latin square design. With three modeling tasks and two recommendation approaches plus the no-recommendation approach, this system allows for various equivalent classes of a 3×3 array filled with the order in which the tasks were performed, each occurring exactly once in each row and exactly once in each column. Based on such a design, biases towards a specific recommendation approach or a specific modeling task is eliminated, and one can conclude that the AR-based recommendation approach performed best for all three tasks.

Another aspect to be considered is the number of recommended terms provided by L2R and AR mining. Association Rule mining is a filtering approach showing only terms which have higher support and confidence values than the minimum thresholds. On the contrary, Learning To Rank is a ranking approach that does not cut off the list of recommendations. This way, it is possible that the size of the lists of recommendations differ between both recommendation approaches. To address this problem, one can apply Hick's law [43], which essentially normalizes the time taken to complete a task by the number of recommended items. This enables to compare two lists of recommendations with different amounts of items. However, applying Hick's law [43] is only feasible, if the needed time is measured for every single operation. In the user study, only the overall time for the entire task (including six to seven operation) was measured, such that Hick's law is not applicable. Another option to mitigate this problem is to equal the number of recommended terms for both approaches. However, it is of additional interest whether the participants are rather interested in only reusing terms that other data providers have reused for similar data (this would favor AR-based recommendations, as they filter out every other terms), or whether they want to explore which terms is it generally possible to reuse (this would favor L2R-based recommendations, as they include terms that are not used by others). Filling up the list of recommendations provided by the AR-based approach results in a list that also contains terms which have not been used by other data providers. On the contrary, filtering terms from the list of recommendations provided by the L2R-based approach results in a list which cannot be explored. Thus, the lists of recommendations based on both approaches have not been altered.

6.6 Conclusion and Summary of the Chapter

This chapter presented an online evaluation examining TermPicker's recommendation quality via a user study. The study compared vocabulary term recommendations using the Association Rule mining vs. the machine learning approach Learning To Rank. LOD practitioners were invited to the lab to participate in the user study. Some of the respondents

were quite experienced with LOD and the data modeling tool Karma that was used for the study. However, most participants had little to moderate experience in publishing Linked Open Data. All participants were asked to finish three LOD models for data sets from the music domain, museum domain, and product offer domain. The results showed that recommendations based on AR mining were clearly favored by a majority of the participants. With the AR-based approach, they were able to complete the modeling tasks with less effort, the quality of the resulting RDF representation was higher than using the L2R-based recommendations or no recommendations, and the user satisfaction survey showed a clear preference towards the AR-based recommendations.

Although a user study can show how real users interact with the evaluated recommendation approaches in a practical setting, it is solely an estimation of the usage of the approaches in a productive system. For example, the utilized modeling tasks needed to be manageable for the participants in order to perform a within designed study with three modeling tasks. However, in a productive system, a user might encounter a very complex modeling task, e.g., modeling museum data that has over 100 columns in its tabular representation. Evaluating a modeling task of this size is not feasible with the help of a user study. To address this challenge, it is rather advised to perform additional long-term evaluations after a recommendation approach is implemented in the productive system such that users interact with it on a regular basis.

7 Conclusion

This thesis presented the design and the evaluation of *TermPicker*: a novel recommendation approach suggesting Resource Description Framework (RDF) vocabulary terms that can be reused when modeling Linked Open Data (LOD). TermPicker uses existing data from the LOD cloud to suggest vocabulary terms, i.e., RDF classes and properties. It is desgined to enable and to facilitate the reuse of vocabulary terms. So-called *schema-level patterns* (SLPs), which represent how vocabulary terms are combined in a LOD model, are the mechanism for accomplishing this. They capture a part of the LOD model that the engineer is currently focused on and also determine how other data providers on the LOD cloud combine terms to model their data as LOD. The former is defined as a *query-SLP* which is used as input for TermPicker, and the latter is used to calculate the so-called *SLP-feature* specifying how many other data sets use a recommendation candidate in conjunction with the terms in the query-SLP.

TermPicker was evaluated via an offline evaluation and an online evaluation. Both evaluations investigated the performance of two recommendation methods, i.e., the machine learning approach *Learning To Rank* (L2R) and the data mining approach *Association Rule* (AR) mining. The offline evaluation simulates which vocabulary terms potential users would select, and based on such a simulation, the prediction accuracy of recommendation algorithms is measured. The online evaluation employs a user study examining how actual TermPicker users interact with the recommender in a practical setting, ascertaining which recommendation method they prefer. LOD engineers whose task it is to model some data as Linked Open Data were engaged as TermPicker test users. Results obtained from the offline evaluation illustrate that the SLP-feature significantly improves the prediction accuracy by 35% compared to using solely the current strategies for reusing vocabulary terms. These current strategies comprise the popularity of a vocabulary term and whether it is from an already used vocabulary. The measures used to calculate the prediction accuracy are the Mean Average Precision (MAP) and the Mean Reciprocal Rank at the first five positions (MRR@5), and the average values for the best performing L2R algorithm were MAP $= 0.75$ and MRR@5 $= 0.80$. This illustrates that a relevant candidate is very likely to appear within the first five results of the list of all recommendations. However, the difference between the prediction accuracy of L2R-based recommendations and the prediction accuracy of AR-based recommendations was not significant, as both performed equally well in the offline evaluation. Such a result served as additional motivation for conducting an online evaluation in the form of a user study. Participants, who are LOD practitioners, were invited to the lab for modeling three data sets from disparate domains as LOD and to subsequently provide feedback on the utilized recommendation methods. The results of the user study indicated that a clear majority of the participants favored recommendations based on AR mining. The AR-based recommendations allowed them to complete the mod-

eling tasks with less effort, and the resulting LOD representation of the three data sets had a higher quality than the LOD representations modeled with the help of L2R-based recommendations. In total, it was observed that Association Rule mining provides high quality RDF vocabulary term recommendations, such that it aids LOD practitioners to reuse RDF classes and properties and thereby enables them to model high-quality Linked Open Data. This especially applies to participants rating their knowledge in LOD publishing as "moderately experienced", who completed the modeling tasks with the same results as participant with more experience in publishing LOD.

The experiments described in this work make it possible to answer the main research question and its sub-questions (RQ1) to (RQ3) (cf. Section 1.2). The findings described in Chapter 4 illustrate that the most typical vocabulary reuse strategies currently used by LOD engineers are based on the popularity of a vocabulary term and whether it is from a vocabulary that is already used. The features that are induced from these findings constitute the main contribution to (RQ1), as they can be used for calculating RDF vocabulary term recommendations. Chapter 5 illustrates TermPicker's prediction accuracy when using schema-level patterns to recommend RDF vocabulary terms. Both the Learning To Rank and the Association Rule mining technique outperform recommendations based solely on the features induced from the most typical reuse strategies which constitutes the main contribution to (RQ2). Chapter 6 describes further insights gained by evaluating both recommendation approaches via a user study examining how real users interact with the recommender. The results show a clear preference towards vocabulary term recommendations based on Association Rule mining, which contributes to answer (RQ3). The contributions to (RQ1), (RQ2), and (RQ3) thereby show that TermPicker is able to aid LOD engineers in reusing RDF vocabulary terms to a large extent compared to the current typical vocabulary reuse strategies. Including the SLP-feature into the set of utilized features increases the quality of TermPicker's RDF vocabulary term recommendations by 35%.

In the following, Section 7.1 enlists the lessons learned and Section 7.2 describes the outlook and future work.

7.1 Lessons Learned

The main challenges that were faced in this work were (i) designing a survey to establish a baseline to which TermPicker can be compared, (ii) designing an offline evaluation to measure the recommendation quality without users, and (iii) conducting a user study to derive meaningful results. Lessons learned with respect to (i) and (ii) are discussed in the following in more detail, and a description as well as a solution of/for the latter challenge is found in Section 6.5.

To address challenge (i), having a baseline against which to compare a recommender system is an essential aspect when evaluating the system's benefit for the community or for a particular group of users [50, 77]. If an existing recommender, which can be used as a baseline, does not exist, it is widely accepted to conduct a survey for inferring some type of a baseline [77]. The goal of the survey presented in Chapter 4 was to infer from the knowledge and experiences of LOD engineers a model to decide which vocabulary

terms to reuse and which not to reuse. It was not possible to simply ask questions such as "Would you rather reuse popular vocabulary terms?", as the participants would have answered according to the best practices stated in [42]. It was rather necessary to present the participants several LOD models that represent the same data entities with different vocabulary reuse strategies. The main challenge here was designing these LOD models. First, the represented data had to be broadly understandable, as participants do not want to spend a lot of time trying to understand the data. Second, it must be possible to represent the data via different vocabulary reuse strategies, i.e., same data entities and their relations are represented via different vocabulary terms. To address this challenge, it was necessary to make three iterations of pre-surveys, in which a small fraction of LOD experts commented on the models. This way, the pre-surveys and those participating helped a lot in designing the LOD models and the survey itself.

Considering (ii), when designing an offline evaluation, one has to find a data set comprising representative information that can simulate user interaction with the recommender [77]. Unfortunately, there is no data set containing pre-defined queries and relevance information on recommending RDF vocabulary terms. Therefore, it was necessary to select a LOD collection — a data collection that consists of various data sets from the LOD cloud — for an evaluation in which a part of the information is first hidden and then used to assess the prediction accuracy. The main challenge here was to calculate the SLPs from the selected LOD collections that were used for training and testing the ranking model. Splitting a LOD collection by RDF triples into training set, test set, and a knowledge base representing the LOD cloud, was not possible. This would result in incomplete SLPs that typically do not contain the class information for resources in the object position of RDF triples. For example, given the three RDF triples in n-quad syntax (the fourth element is the context URI specifying the pay-level domain where the RDF triple can be found)

<http://ex1.org/001> rdf:type foaf:Person <http://ex1.org>. (7.1)

<http://ex1.org/001> foaf:knows <http://ex1.org/002> <http://ex1.org>. (7.2)

<http://ex1.org/002> rdf:type foaf:Person <http://ex1.org>. (7.3)

If splitting these RDF triples between the triple in (7.2) and the triple in (7.3) and subsequently calculate the schema-level patterns, it results in the following two SLPs:

$$slp_1 = (\{\text{foaf:Person}\}, \{\text{foaf:knows}\}, \varnothing) \tag{7.4}$$

$$slp_2 = (\{\text{foaf:Person}\}, \varnothing, \varnothing) \tag{7.5}$$

Many SLPs would contain only one or two terms, and such SLPs are less usable to calculate an SLP-feature greater than zero. Therefore, it was necessary to split the data by pay-level domain (PLD). For example, as the triples in equations (7.1) to (7.3) are from the same PLD, they would be in the same split, and computing SLPs would result in the SLP:

$$slp = (\{\text{foaf:Person}\}, \{\text{foaf:knows}\}, \{\text{foaf:Person}\}) \tag{7.6}$$

However, splitting the data by PLDs randomly yielded very frustrating results, i.e., the SLP-feature also did not improve the prediction accuracy. Only after further analysis did it

become clear that most PLDs contain only a few SLPs with only two or three vocabulary terms, since the data size is rather small for many LOD data sets and the diversity of the utilized terms is small as well. To address this challenge, it seemed most appropriate to establish two measures representing the PLDs. These metrics specify that a PLD must contain a high number of distinct vocabulary terms (solves the challenge to calculate many SLPs from a PLD) and that a PLD must have a high ratio between reused vocabulary terms and all utilized terms (solves the challenge that many SLPs contain only a few terms). This way, it was possible to find the PLDs that were most appropriate as training and test sets for the evaluation.

7.2 Outlook

The work and the results presented in this thesis serve as motivation for further investigations in recommending RDF vocabulary terms based on schema information. First, the data collections utilized in the offline evaluations can be exchanged with the recently developed *LODLaundromat* [7] collection which contains a set of RDF triples from a larger set of data sets on the LOD cloud compared to the BTC 2014 data. They collect the data from the LOD cloud, clean it, and provide it back to the community guaranteed to conform to a specified set of best practices. Such data has not been used in the evaluations of this work, as the LODLaundromat was not developed at the time of the experiments. However, it would be of broad interest whether the LODLaundromat collection can be used to compute a ranking model that outperforms the ranking model computed from the BTC 2014 data set. Second, due to its size, what remains to be investigated is whether the LODLaundromat collection is able to provide a larger set of appropriate recommendation candidates. In the evaluation using the BTC 2014 data set, it was observed that 37% more relevant vocabulary terms had an SLP-feature greater than zero compared to the evaluation using the DyLDO data set. It is thereby likely that an evaluation using the LODLaundromat collection could produce even more relevant vocabulary terms with an SLP-feature greater than zero.

It is also possible to investigate further features representing recommendation candidates and their impact on the recommendation quality. For example, participants of the survey and the user study mentioned that a recommender system of RDF vocabulary terms should not only provide term suggestions, but also various meta-information on the recommendation candidates. Such meta-information could include the domain and range information of properties, the documentation of the vocabulary terms, i.e., what they mean, and whether the terms are connected to other classes or properties from other vocabularies. Additionally, in [51], the authors provide a first specification of a *5-star LOD vocabulary use*, which can be used for annotating a recommendation candidate. One could also use the PageRank information of a given pay-level domain that uses a recommended vocabulary term. This way, recommendations can be more specific to a given PLD. It is very likely that using such information to calculate the corresponding feature values can provide more favorable recommendations and thereby aid LOD engineers even more in making a decision which vocabulary terms to reuse.

In addition, the utilized schema-level patterns (SLPs) can be used for applications other than solely for recommending vocabulary terms. The SLP representation of the own data set can be used to analyze whether other data sets use a similar schema. If there are LOD providers using a similar schema, i.e., similar SLPs, to represent their data, it might be possible that these LOD providers share common resources. Once such data sets have been identified, engineers can try to establish relationships between the resources. This will help them to achieve the 5th star in the 5-star LOD paradigm.

For the future, it would be of great use to investigate whether the proposed recommendation approach is, in fact, regularly used in the LOD community. This includes the usage of the system on its website[87] as well as via API calls from modeling tools like Karma, in which TermPicker is integrated. This will be part of the investigation of TermPicker's long term effect.

[87] `http://lod-rec.org`, last accessed February 14th, 2017

Bibliography

[1] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. *ACM SIGMOD Record*, 22(2):207–216. ACM, 1993.

[2] X. Amatriain, A. Jaimes, N. Oliver, and J. M. Pujol. *Recommender Systems Handbook*, chapter Data Mining Methods for Recommender Systems, pages 39–71. Springer, 2011.

[3] S. Auer, J. Demter, M. Martin, and J. Lehmann. LODStats – An extensible framework for high-performance dataset analytics. In *The international conference on Knowledge Engineering and Knowledge Management (EKAW)*, pages 353–362. Springer, 2012.

[4] F. Baader. *The description logic handbook: Theory, implementation and applications*. Cambridge University Press, 2 edition, 2003. ISBN: 9780521150118.

[5] R. Baeza-Yates, B. Ribeiro-Neto, et al. *Modern information retrieval*, volume 463. ACM, 1999. ISBN: 020139829X.

[6] M. Bates. Models of natural language understanding. *The National Academy of Sciences*, 92(22):9977–9982. PMC, 1995.

[7] W. Beek, L. Rietveld, H. R. Bazoobandi, J. Wielemaker, and S. Schlobach. LOD laundromat: a uniform way of publishing other peoples dirty data. In *The International Semantic Web Conference (ISWC)*, pages 213–228. Springer, 2014.

[8] J. Beel. *Towards effective research-paper recommender systems and user modeling based on mind maps*. PhD thesis, Otto von Guericke University Magdeburg, 2015. `http://nbn-resolving.de/urn:nbn:de:gbv:ma9:1-6121`, last accessed October 20th, 2016.

[9] C. Bizer and R. Cyganiak. D2RQ - Lessons Learned. In *W3C Workshop on RDF Access to Relational Databases*. Cambrigde, 2007.

[10] C. Bizer, R. Cyganiak, and T. Heath. How to Publish Linked Data on the Web, July 2008. `http://www4.wiwiss.fu-berlin.de/bizer/pub/LinkedDataTutorial/`, last accessed October 20th, 2016.

[11] C. Bizer, T. Heath, and T. Berners-Lee. Linked Data - the story so far. *Semantic Services, Interoperability and Web Applications: Emerging Concepts*, pages 205–227, 2009.

[12] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. DBpedia - A crystallization point for the Web of Data. *Web Semantics: science, services and agents on the World Wide Web*, 7(3):154–165. Elsevier, 2009.

[13] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[14] D. Bridge, M. H. Göker, L. McGinty, and B. Smyth. Case-based recommender systems. *The Knowledge Engineering Review*, 20(3):315–320. ACM, 2005.

[15] S. Brin and L. Page. Reprint of: The anatomy of a large-scale hypertextual web search engine. *Computer networks*, 56(18):3825–3833. Elsevier, 2012.

[16] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to Rank Using Gradient Descent. In *The International Conference on Machine Learning (ICML)*, pages 89–96. ACM, 2005.

[17] R. Burke. Knowledge-based recommender systems. In *Encyclopedia of Library And Information Systems*, pages 180–200. Marcel Dekker, 2000.

[18] R. Busa-Fekete, G. Szarvas, T. Elteto, B. Kégl, et al. An apple-to-apple comparison of learning-to-rank algorithms in terms of normalized discounted cumulative gain. In *The European Conference on Artificial Intelligence (ECAI)*, volume 242. IOS Press, 2012.

[19] S. Campinas, T. E. Perry, D. Ceccarelli, R. Delbru, and G. Tummarello. Introducing RDF graph summary with application to assisted SPARQL formulation. In *The Workshop on Database and Expert Systems Applications (DEXA)*, pages 261–266. IEEE, 2012.

[20] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning To Rank: from pairwise approach to listwise approach. In *The International Conference on Machine Learning (ICML)*, pages 129–136. ACM, 2007.

[21] G. Charness, U. Gneezy, and M. A. Kuhn. Experimental methods: Between-subject and within-subject design. *Journal of Economic Behavior and Organization*, 81(1):1–8. Elsevier, 2012.

[22] G. Cheng, W. Ge, and Y. Qu. Falcons: searching and browsing entities on the semantic web. In *The international conference on the World Wide Web (WWW)*, pages 1101–1102. ACM, 2008.

[23] G. Cheng, S. Gong, and Y. Qu. An empirical study of vocabulary relatedness and its application to recommender systems. In *The International Semantic Web Conference (ISWC)*, pages 98–113. Springer, 2011.

[24] G. Cheng and Y. Qu. Searching linked objects with falcons: Approach, implementation and evaluation. *The international Journal on Semantic Web and Information Systems (IJSWIS)*, 5(3):49–70. ACM, 2009.

[25] H. Cramér. *Mathematical Methods of Statistics (PMS-9)*, volume 9. Princeton University Press, 2016. ISBN: 0691005478.

[26] N. Craswell. *Encyclopedia of Database Systems*, chapter Mean Reciprocal Rank, pages 1703–1703. Springer, 2009. ISBN: 9780387399409.

[27] W. W. Daniel. *Applied Nonparametric Statistics*. The Duxbury advanced series in statistics and decision sciences. PWS-Kent Publ., 1990. ISBN: 9780534919764.

[28] M. d'Aquin, C. Baldassarre, L. Gridinoc, M. Sabou, S. Angeletou, and E. Motta. Watson: Supporting next generation semantic web applications. In *The IADIS International WWW/Internet Conference*, pages 363–371. IADIS Press, 2007.

[29] C. Desrosiers and G. Karypis. *Recommender Systems Handbook*, chapter A Comprehensive Survey of Neighborhood-based Recommendation Methods, pages 107–144. Springer, 2011.

[30] M. Dudáš, V. Svátek, and J. Mynarz. Dataset Summary Visualization with LODSight. In *The Semantic Web: ESWC 2015 Satellite Events*, pages 36–40. Springer, 2015.

[31] T. Eiter, G. Ianni, R. Schindlauer, and H. Tompits. Effective integration of declarative rules with external evaluations for semantic-web reasoning. In *The European Semantic Web Conference (ESWC)*, pages 273–287. Springer, 2006.

[32] J. Euzenat, P. Shvaiko, et al. *Ontology matching*, volume 18. Springer, 2007.

[33] A. Felfernig, G. Friedrich, D. Jannach, and M. Zanker. *Recommender Systems Handbook*, chapter Developing Constraint-based Recommenders, pages 187–215. Springer, 2011.

[34] M. Fernandez, I. Cantador, and P. Castells. CORE: a tool for collaborative ontology reuse and evaluation. In *The International Workshop on Evaluation of Ontologies for the Web (EON)*. ACM, 2006.

[35] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *The Journal of machine learning research*, 4:933–969. JMLR.org, 2003.

[36] J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232. JSTOR, 2001.

[37] M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701. JSTOR, 1937.

[38] T. Gottron, M. Knauf, A. Scherp, and J. Schaible. Ellis: Interactive exploration of linked data on the level of induced schema patterns. In *The Workshop on Summarizing and Presenting Entities and Ontologies (SumPre)*. CEUR-WS.org, 2016.

[39] A. G. Greenwald. Within-subjects designs: To use or not to use? *Psychological Bulletin*, 83(2):314. American Psychological Association, 1976.

[40] L. Hang. A short introduction to Learning To Rank. *Transactions on Information and Systems*, 94(10):1854–1862. The Institute of Electronics, Information and Communication Engineers (IEICE), 2011.

[41] A. Harth. Billion Triples Challenge data set. Downloaded from `http://km.aifb.kit.edu/projects/btc-2012/`, 2012. last accessed October 20th, 2016.

[42] T. Heath and C. Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Synthesis Lectures on the Semantic Web. Morgan & Claypool Publishers, 2011. ISBN: 978-1-60845-431-0.

[43] W. E. Hick. On the rate of gain of information. *Quarterly Journal of Experimental Psychology*, 4(1):11–26. Taylor and Francis Online, 1952.

[44] P. Hitzler, M. Krötzsch, Y. Sure, and S. Rudolph. *Semantic Web: Grundlagen*. Springer, 2007. ISBN: 9783540339946.

[45] B. Hjørland. The foundation of the concept of relevance. *Journal of the American Society for Information Science and Technology*, 61(2):217–237. Wiley, 2010.

[46] A. Hogan. *Exploiting RDFS and OWL for Integrating Heterogeneous, Large-Scale, Linked Data Corpora*. PhD thesis, National University of Ireland, Galway, 2011. `https://users.dcc.uchile.cl/~ahogan/docs/thesis/thesis-one-sided.pdf`, last accessed November 20th, 2016.

[47] A. Hogan, A. Harth, A. Passant, S. Decker, and A. Polleres. Weaving the Pedantic Web. In *The international Linked Data on the Web Workshop (LDOW)*. CEUR-WS, April 2010.

[48] A. Hogan, J. Umbrich, A. Harth, R. Cyganiak, A. Polleres, and S. Decker. An empirical survey of Linked Data conformance. *Web Semantics: Science, Services and Agents on the World Wide Web*, 14:14–44. Elsevier, 2012.

[49] E. Hyvönen. Publishing and using cultural heritage linked data on the semantic web. *Synthesis Lectures on the Semantic Web: Theory and Technology*, 2(1):1–159. Morgan & Claypool, 2012.

[50] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich. *Recommender systems: an introduction*. Cambridge University Press, 2010. ISBN: 0521493366.

[51] K. Janowicz, P. Hitzler, B. Adams, D. Kolas, I. Vardeman, et al. Five stars of Linked Data vocabulary use. *Semantic Web Interoperability, Usability, Applicability*, 5(3):173–176. IOS Press, 2014.

[52] T. Käfer, A. Abdelrahman, J. Umbrich, P. O'Byrne, and A. Hogan. Observing Linked Data Dynamics. In *The Extended Semantic Web Conference (ESWC)*, pages 213–227. Springer, 2013.

[53] T. Käfer and A. Harth. Billion Triples Challenge data set. Downloaded from `http://km.aifb.kit.edu/projects/btc-2014/`, 2014. last accessed October 20th, 2016.

[54] G. Klyne, J. J. Carroll, and B. McBride. Resource Description Framework (RDF): Concepts and abstract syntax. 2006. `https://www.w3.org/TR/rdf11-concepts/`, last accessed October 20th, 2016.

[55] C. Knoblock, P. Szekely, J. Ambite, A. Goel, S. Gupta, and et al. Semi-automatically Mapping Structured Sources into the Semantic Web. In *The European Semantic Web Conference (ESWC)*, pages 375–390. Springer, 2012.

[56] Y. Koren and R. Bell. *Recommender Systems Handbook*, chapter Advances in Collaborative Filtering, pages 145–186. Springer, 2011.

[57] R. Krishnamurthy, A. Mittal, C. A. Knoblock, and P. Szekely. Assigning Semantic Labels to Data Sources. In *The Extended Semantic Web Conference (ESWC)*, pages 403–417, 2015.

[58] W. H. Kruskal and W. A. Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association*, 47(260):583–621, 1952.

[59] J. Lazar, J. H. Feng, and H. Hochheiser. *Research Methods in Human-Computer Interaction*. Wiley Publishing, 2010.

[60] T. Liu. *Learning to Rank for Information Retrieval*. Springer, 2014. ISBN: 978-3-642-14266-6.

[61] G. Lodi, A. Maccioni, M. Scannapieco, M. Scanu, and L. Tosco. Publishing Official Classifications in Linked Open Data. In *The international Workshop on Semantic Statistics (SemStats)*. Springer, 2014.

[62] P. Lops, M. de Gemmis, and G. Semeraro. *Recommender Systems Handbook*, chapter Content-based Recommender Systems: State of the Art and Trends, pages 73–105. Springer, 2011.

[63] C. D. Manning, P. Raghavan, H. Schütze, et al. *Introduction to information retrieval*, volume 1. Cambridge University Press, 2008. ISBN: 0521865719.

[64] P. E. McKnight and J. Najab. Mann-whitney u test. *Corsini Encyclopedia of Psychology*, 2010.

[65] D. Metzler and W. B. Croft. Linear feature-based models for information retrieval. *Information Retrieval*, 10(3):257–274, 2007.

[66] G. Navarro. A Guided Tour to Approximate String Matching. *ACM Computing Surveys*, 33(1):31–88. ACM, 2001.

[67] A.-C. Ngonga Ngomo and S. Auer. LIMES - A Time-Efficient Approach for Large-Scale Link Discovery on the Web of Data. In *The International Joint Conference on Artificial Intelligence (IJCAI)*. ACM, 2011.

[68] N. F. Noy and D. L. McGuinness. *Ontology development 101: A guide to creating your first ontology*. Stanford knowledge systems laboratory technical report KSL-01-05 and Stanford medical informatics technical report SMI-2001-0880, 2001.

[69] V. Presutti, L. M. Aroyo, A. Gangemi, A. Adamou, B. Schopman, and G. Schreiber. A knowledge pattern-based method for linked data analysis. In *The International Conference on Knowledge Capture (K-CAP)*, pages 173–174. ACM, 2011.

[70] F. Ricci, L. Rokach, and B. Shapira. *Recommender Systems Handbook*, chapter Introduction to Recommender Systems Handbook, pages 1–35. Springer, 2011.

[71] M. M. Romero, J. M. Vázquez-Naya, C. R. Munteanu, J. Pereira, and A. Pazos. An Approach for the Automatic Recommendation of Ontologies Using Collaborative Knowledge. In *International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES)*, pages 74–81. Springer, 2010.

[72] C. Sammut and G. I. Webb. *Encyclopedia of machine learning*. Springer, 2011. ISBN: 9780387345581.

[73] J. Schaible, T. Gottron, S. Scheglmann, and A. Scherp. LOVER: support for modeling data using linked open vocabularies. In *The International Workshop On Linked Web Data Management (LWDM)*, pages 89–92. ACM, 2013.

[74] J. Schaible, T. Gottron, and A. Scherp. Survey on Common Strategies of Vocabulary Reuse in Linked Open Data Modeling. In *The Extended Semantic Web Conference (ESWC)*, pages 457–472. Springer, 2014.

[75] F. Scharffe, G. Atemezing, R. Troncy, F. Gandon, and et al. Enabling Linked-Data publication with the Datalift platform. In *The AAAI Conference on Artificial Intelligence*. AAAI, 2012.

[76] M. Schmachtenberg, C. Bizer, and H. Paulheim. Adoption of the linked data best practices in different topical domains. In *The International Semantic Web Conference (ISWC)*, pages 245–260. Springer, 2014.

[77] G. Shani and A. Gunawardana. *Recommender Systems Handbook*, chapter Evaluating Recommendation Systems, pages 257–297. Springer, 2011.

[78] S. Stadtmüller, A. Harth, and M. Grobelnik. Accessing information about linked data vocabularies with vocab.cc. In *Semantic Web and Web Science*, pages 391 – 396. Springer, 2013.

[79] M. Taheriyan, C. Knoblock, P. Szekely, and J. L. Ambite. Learning the Semantics of Structured Data Sources. *Journal of Web Semantics Special Issue on Knowledge Graphs*. Elsevier, 2016.

[80] T. Tran and G. Ladwig. Structure index for RDF data. In *International Workshop on Semantic Data Management (SemData)*, volume 2. CEUR-WS, 2010.

[81] S. Van Hooland and R. Verborgh. *Linked Data for Libraries, Archives and Museums: How to clean, link and publish your metadata*. Facet, Londres, 2014. ISBN: 978-1-85604-964-1.

[82] P.-Y. Vandenbussche, G. A. Atemezing, M. Poveda-Villalón, and B. Vatant. Linked Open Vocabularies (LOV): a gateway to reusable semantic vocabularies on the Web. *Semantic Web – Interoperability, Usability, Applicability. IOS Press*, (Preprint):1–16, 2015.

[83] J. Volz, C. Bizer, M. Gaedke, and G. Kobilarov. Silk-A Link Discovery Framework for the Web of Data. In *The International Workshop on Linked Data on the Web (LDOW)*. ACM, 2009.

[84] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83. JSTOR, 1945.

[85] T. D. Wilson. On user studies and information needs. *Journal of Documentation*, 37(1):3–15. MCB UP Ltd., 1981.

[86] T. H. Wonnacott and R. J. Wonnacott. *Introductory Statistics*. Wiley, 1972. ISBN: 9780471615187.

[87] Q. Wu, C. J. Burges, K. M. Svore, and J. Gao. Adapting boosting for information retrieval measures. *Information Retrieval*, 13(3):254–270. Springer, 2010.

[88] J. Xu and H. Li. Adarank: a boosting algorithm for information retrieval. In *The Special Interest Group on Information Retrieval (SIGIR)*, pages 391–398. ACM, 2007.

[89] C. Zhang and S. Zhang. *Association Rule Mining: Models and Algorithms*. Springer, 2002. ISBN: 9783540460275.

[90] Z. Zhang, A. L. Gentile, E. Blomqvist, I. Augenstein, and F. Ciravegna. Statistical knowledge patterns: Identifying synonymous relations in large linked datasets. In *The International Semantic Web Conference (ISWC)*, pages 703–719. Springer, 2013.

Acknowledgments

First of all, I would like to thank Prof. Dr. Ansgar Scherp for supervising this thesis. He offered me the opportunity to work with him and his group as well as accompanied my research with help and encouragement. In this context, I would also like to thank Prof. Dr. York Sure-Vetter for agreeing to be the second reviewer of my thesis.

I am very grateful to GESIS - Leibniz-Institute for the Social Sciences for ensuring funding as well as to my colleagues for always encouraging me and my work.

Naturally, I thank all respondents and Linked Open Data experts for participating in the online survey as well as the user study.

A special appreciation goes to my family, which always supported me in writing the thesis and in my everyday life.

Erklärung

Hiermit versichere ich,

I. dass diese Arbeit – abgesehen von der Beratung durch den Betreuer Prof. Dr. Ansgar Scherp – nach Inhalt und Form meine eigene ist,

II. dass Vorversionen einiger Teile dieser Arbeit bereits veröffentlicht wurden, nämlich

   a) Johann Schaible, Thomas Gottron, Stefan Scheglmann, and Ansgar Scherp: *LOVER: Support for Modeling Data Using Linked Open Vocabularies.* In the Proceedings of the 3rd International Workshop on Linked Web Data Management (LWDM), ACM, 2013

   b) Johann Schaible, Thomas Gottron, and Ansgar Scherp: *Survey on Common Strategies of Vocabulary Reuse in Linked Open Data Modeling.* In the Proceedings of the 11th Extended Semantic Web Conference (ESWC), Springer, 2014

   c) Johann Schaible, Thomas Gottron, and Ansgar Scherp: *TermPicker: Enabling the Reuse of Vocabulary Terms by Exploiting Data from the Linked Open Data Cloud.* In the Proceedings of the 13th Extended Semantic Web Conference (ESWC), Springer, 2016

   d) Johann Schaible, Pedro Szekely, and Ansgar Scherp: *Comparing Vocabulary Term Recommendations using Association Rules and Learning To Rank: A User Study.* In the Proceedings of the 13th Extended Semantic Web Conference (ESWC), Springer, 2016

III. dass kein Teil dieser Arbeit bereits einer anderen Stelle im Rahmen eines Prüfungsverfahrens vorgelegen hat,

IV. und dass diese Arbeit unter Einhaltung der Regeln guter wissenschaftlicher Praxis der Deutschen Forschungsgemeinschaft entstanden ist.

Johann Schaible
17. Februar 2017