TermPicker: Enabling the Reuse of Vocabulary Terms by Exploiting Data from the Linked Open Data Cloud

Johann Schaible[1(✉)], Thomas Gottron[2], and Ansgar Scherp[3]

[1] GESIS – Leibniz Institute for the Social Sciences, Cologne, Germany
johann.schaible@gesis.org
[2] Innovation Lab, SCHUFA Holding AG, Wiesbaden, Germany
thomas.gottron@schufa.de
[3] ZBW – Leibniz Information Center for Economics, Kiel University, Kiel, Germany
asc@informatik.uni-kiel.de

Abstract. Deciding which RDF vocabulary terms to use when modeling data as Linked Open Data (LOD) is far from trivial. In this paper, we propose *TermPicker* as a novel approach enabling vocabulary reuse by recommending vocabulary terms based on various features of a term. These features include the term's *popularity*, whether it is from an already used vocabulary, and the so-called *schema-level pattern* (SLP) feature that exploits which terms other data providers on the LOD cloud use to describe their data. We apply Learning To Rank to establish a ranking model for vocabulary terms based on the utilized features. The results show that using the SLP-feature improves the recommendation quality by 29–36 % considering the Mean Average Precision and the Mean Reciprocal Rank at the first five positions compared to recommendations based on solely the term's popularity and whether it is from an already used vocabulary.

1 Introduction

When modeling Linked Open Data (LOD), engineers employ Resource Description Framework (RDF) vocabularies—a collection of (unique) vocabulary terms, i.e., RDF types (also referred to as "classes") and properties—to represent their data. It is considered best practice to reuse terms from existing vocabularies before defining proprietary ones, since this reduces heterogeneity in the data representation [1]. However, finding vocabulary terms for reuse that are commonly used for representing similar data is far from trivial. Prominent vocabulary search services such as LOV [2] or LODstats [3] can be used to find specific terms based on string search. Vocabulary term recommendation tools, like CORE [4], suggest entire vocabularies based on a string search. They also provide statistics on vocabulary terms such as their total number of occurrence and which datasets use them. However, none of the services provide information on how data providers on the LOD cloud actually combine the RDF types and properties from the different vocabularies to model their data (cf. Sect. 2).

© Springer International Publishing Switzerland 2016
H. Sack et al. (Eds.): ESWC 2016, LNCS 9678, pp. 101–117, 2016.
DOI: 10.1007/978-3-319-34129-3_7

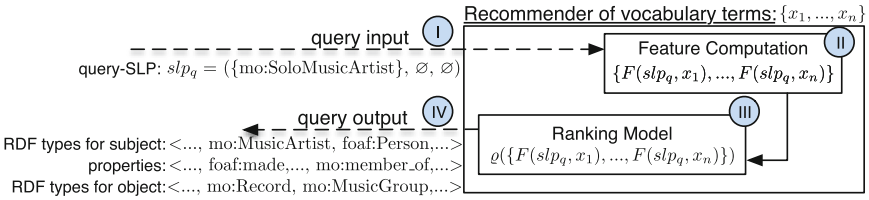

Fig. 1. *Example Run of TermPicker.* The engineer started modeling the data and chose to reuse mo:SoloMusicArtist from the Music Ontology. TermPicker uses the corresponding query-SLP (slp_q) (step (I)) and calculates feature values for each recommendation candidate x_i from the set of all terms published on the LOD cloud ($\{x_1, ..., x_n\}$) (step (II)). The ranking model ϱ uses the values $F(slp_q, x_i)$ (step (III)) to provide the engineer ranked lists of vocabulary terms (step (IV)).

In this paper, we introduce *TermPicker*[1] - a novel vocabulary term recommendation approach enabling the reuse of vocabulary terms by exploiting the following information: given that the engineer already generated a part of her LOD model, TermPicker suggests further RDF types and properties (from other vocabularies) that other LOD providers use together with the part the engineer has already modeled. We capture such information by using so-called *schema-level patterns* (SLPs), which are tuples describing the connection between two sets of RDF types via a set of properties (cf. Sect. 3).

Figure 1 shows TermPicker's workflow via an example: The input is an SLP describing a part of the data model the engineer has already designed, e.g. the query-SLP $slp_q = (\{\text{mo:SoloMusicArtist}\}, \varnothing, \varnothing)$. TermPicker allows both: adding further RDF types or properties to the query-SLP, and replacing already used terms in slp_q with better fitting ones (cf. Sect. 4). This applies for RDF types for resources in the subject or object position of an RDF triple, as well as properties between the resources. TermPicker's ranking model uses SLPs and other features, such as the popularity of a recommendation candidate or if the candidate is from an already used vocabulary, for presenting three ranked lists. These lists contain RDF type and property recommendations as output. In this way, the engineer can use TermPicker to find vocabulary terms for reuse that other data providers used together with the terms in slp_q.

To train the ranking model, we use a *Learning To Rank* approach. Learning to Rank approaches constitute 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 items in the training data. In our case the items to be ranked are RDF vocabulary terms. To evaluate the benefit of using SLPs, we conduct a 10-fold leave-one-out evaluation for different situations where engineers need to select a vocabulary term for reuse (cf. Sect. 5). The recommendation quality is assessed using the Mean Average Precision (MAP) and the Mean Reciprocal Rank at the first five positions

[1] http://termpicker.lodrec.org, last access 3/4/16.

(MRR@5). Summarizing, the contributions of this paper are (cf. Sects. 6 and 7 for results and discussion): (i) Evaluation of Learning To Rank algorithms when calculating a ranking model for TermPicker's recommendations. (ii) Evaluation of the SLP-feature's impact on the recommendation quality. (iii) Evaluation of RDF type vs. property recommendations as this reflects different real-world LOD modeling scenarios.[2] For the interested reader, formalizations regarding the SLPs and the features for the L2R algorithm can be found in [5].

2 Related Work

To motivate the use of a recommendation service like TermPicker, we first illustrate current approaches that aid Linked Data engineers in reusing RDF vocabulary terms. Subsequently, we motivate the use of SLPs compared to current approaches for inducing schema information from Linked Open Data.

Vocabulary Search Engines. Services like LOV [2], LODstats [3], Watson [6], or Falcon's concept Search [7] aid the engineer in finding vocabulary terms based on a keyword-based approach. The input is a string describing the desired vocabulary term and the output is a set of vocabulary terms that are similar to the search-string. Each service provides various meta-information on the vocabulary terms and their vocabularies, such as the term's number of usages on the LOD cloud or which datasets use them. In addition, most services exploit schema-information encoded in the vocabularies (within the T-Box specification of vocabularies) such as sub-class, sub-property, or equivalence relations between vocabulary terms (in LOV also via SPARQL queries). However, sometimes engineers cannot express the needed vocabulary term via keywords, and not every relationship between vocabulary terms is explicitly defined in a vocabulary (especially across vocabularies). TermPicker does not use a string-based approach for search but a structure-based approach. This enables it to recommend specific vocabulary terms unknown to the engineer based on other usages of the term on the LOD cloud. Furthermore, for exploiting schema-information, TermPicker does not rely on T-Box information but rather induces the schema-information directly from the datasets on the LOD cloud, i.e. from the encoded A-Box specification of datasets.

Vocabulary Recommender Systems. Services that recommend RDF types and properties are generally based on syntactic and semantic similarity measures. Prominent examples are [4,6,8] and the "data2Ontology" module of the Datalift platform [9]. The input is a set of initial keywords describing a vocabulary term. The services determine a ranked list of domain-specific ontologies considerable for reuse based on string similarity and semantic similarity measures, such as synonyms (in [4] also on manual user evaluations of suggested ontologies). Falcons' Ontology Search [10] also identifies which vocabulary terms might

[2] https://www.w3.org/TR/ld-bp/#MODEL, last access 2/7/16.

express similar semantics, but it is rather designed to specify that different vocabularies contain terms describing similar data. Two other approaches [11,12], both embedded in the data integration tool Karma [13], use NLP techniques to recommend an RDF type in conjunction with a data type property for a column of a table, or use a user's previously designed RDF model for recommendations. Whereas these services provide recommendations based on string analysis, or on previously modeled data, they do not exploit any structural information on how other data providers on the LOD cloud use RDF vocabulary terms to describe the connection between objects. By using the schema-information induced directly from the datasets on the LOD cloud, TermPicker is able to alleviate this situation and recommend vocabulary terms with the goal to reduce the heterogeneity in data representation.

Inducing Schema Information. There are various existing concepts to induce schema information from data on the LOD cloud. Prominent examples are the Knowledge Patterns (KPs) [14], Statistical Knowledge Patterns (SKPs) [15], or the RDF graph summary [16]. KPs identify *PathElements* between all RDF types in the data. Statistical Knowledge Patterns (SKPs) find synonymous properties from different RDF types, 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 type, nor do they contain data type properties to some literal values. SKPs only find synonymous vocabulary terms for a given term and not further vocabulary 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. Therefore, we introduce the notion of *schema-level patterns* (SLPs) and utilize them to induce the schema information from datasets on the LOD cloud (cf. Sect. 3). SLPs have a rather flat structure and do not contain any information from the Entity Layer making them useful for fast queries. They also enable recommending data type properties and object properties that do not have an RDF type as rdfs:range encoded in the data. Statistical Knowledge Patterns can rather be used in addition to SLPs to suggest further synonymous terms. Other tools like LODSight [17] provide visualizations of a dataset's schema information, but are rather useful for exploring relations within and across datasets instead of providing recommendations.

3 Schema-Level Patterns (SLPs)

When reusing vocabularies with the goal to preferably reduce heterogeneity in data representation, one must investigate how other Linked Data providers modeled their data. To this end, we introduce the notion of schema-level patterns (SLPs). They illustrate how resources from a dataset on the LOD cloud are connected with each other. For example, the schema-level pattern

$$(\{\text{foaf:Person}, \text{dbo:ChessPlayer}\}, \{\text{foaf:knows}\}, \{\text{foaf:Person}, \text{dbo:Coach}\}) \quad (1)$$

```
@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 1.1. 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

illustrates that resources of types foaf:Person and dbo:ChessPlayer are connected to resources of types foaf:Person and dbo:Coach via the property foaf:knows and is calculated from the fictive RDF triples in Listing 1.1. We compute such SLPs based on two hash maps. The first map $M1$ contains a resource as key and the set of the resource's RDF types as value. For example, after iterating over the triples in Listing 1.1, $M1$ includes the entries [http://ex1.org/sports_001, {foaf:Person, dbo:ChessPlayer}] and [http://ex1.org/sports_002, {foaf:Person, dbo:Coach}]. The second map $M2$ contains a pair of the subject and object resource as key and the set of all properties between these resources as value. After iterating over the triples in Listing 1.1, $M2$ would contain the entry [(http://ex1.org/sports_001, http://ex1.org/sports_002), {foaf:knows}]. Subsequently, iterating over $M2$, we construct the SLPs using the RDF type information of every resource in $M1$. For the remainder of the paper, we refer to the set of SLPs that are calculated this way from all datasets on the LOD cloud as $\mathbb{SLP}$.

Formally, an SLP is a tuple $slp = (sts, ps, ots)$ (we use the generic term "tuple" to avoid a misunderstanding with the common Semantic Web RDF triple), where sts is the set of RDF types describing resources in subject position of a triple, ots the set of RDF types describing resources in object position of a triple, and ps the set of properties interlinking the resources of types in sts and ots. To operate with SLPs, we define the two operators $\oplus$ and $\ominus$. The operator $\oplus$ can be used for extending an SLP with a further vocabulary term by adding it either to the set sts, ps, or ots. The operator $\oplus_{sts}$ adds an RDF type to the set sts, operator $\oplus_{ots}$ adds a RDF type to ots and the operator $\oplus_{ps}$ adds a property to the set of properties ps. The operation to remove or extract terms from an SLP via the $\ominus$ is defined accordingly.

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

$$slp_i \leq slp_j, \quad \text{iff} \quad (sts_i \subseteq sts_j) \wedge (ps_i \subseteq ps_j) \wedge (ots_i \subseteq ots_j) \tag{2}$$

4 Recommending Vocabulary Terms Using SLPs

TermPicker's recommendations are based on utilizing SLPs. Its input is the query-SLP slp_q, which defines a part of the already modeled dataset. Via the $\oplus$ operator it is first extended with a recommendation candidate x from the set of terms from all data sets on the LOD cloud. Subsequently, TermPicker calculates various feature values for each candidate x in conjunction with the query-SLP slp_q. These feature values are then used by a ranking model (calculated via Learning To Rank) to provide an ordered list of vocabulary terms. Optimally, all relevant candidates are ranked at the top of the list.

Features for Recommendation Candidates. Table 1 presents the features we have used to calculate the recommendations. They represent the popularity of each recommendation candidate, whether it is from an already used vocabulary, and the number of SLPs in $\mathbb{SLP}$ that contain all terms of the query-SLP together with the candidate.

Features f_1 to f_4 were derived from [18], which presents that the most common strategies and influencing factors to choose a vocabulary term for reuse is its *popularity* (features $f_1 - f_3$) and whether or not it is from a vocabulary that is already used (feature f_4). Reusing popular vocabulary terms is supposed to enable an easier consumption of the data, as many Linked Data tools provide support for popular vocabularies [1].[3] Whereas the total number of occurrences of a recommendation candidate x show its overall usage, the amount of data sources using x and its vocabulary specify whether the usage of x is spread across many datasets on the LOD cloud or concentrates on only a few ones. Reusing terms from the same vocabulary (feature f_4) is considered as important strategy as well, because of the following reason: When searching for vocabularies covering the domain of interest and subsequently using the vocabulary's RDF

Table 1. Overview of the utilized features. The features are computed for every recommendation candidate x from the set of terms from all data sets on the LOD cloud

Feature	Definition of the feature
f_1	Number of datasets on the LOD cloud using the recommendation candidate x
f_2	Number of datasets on the LOD cloud using the vocabulary V_x 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

[3] http://www.w3.org/TR/ld-bp/#VOCABULARIES, last access 12/12/15.

types and properties for particular needs, it seems quite likely that one specific domain vocabulary contains many RDF types or properties that can be reused for describing data from that specific domain. For example, the geo[4] vocabulary contains many terms on specifying geographical information.

In addition, we use the introduced SLPs to generate the so-called SLP-feature (feature f_5) which calculates how many SLPs $slp_i \in \mathbb{SLP}$ exist with $slp_q \oplus x \leq slp_i$. Using recommendations based on this feature are likely to result in reducing heterogeneity in the data representation by relying on ontological agreement. The more SLPs in $\mathbb{SLP}$ use the recommendation candidate x together with all terms in the query-SLP, the more it seems to be appropriate to reuse it. Using the SLP-feature is basically analogous to a collaborative filtering approach, as it suggest to use terms that have been used similarly by other data providers on the LOD cloud.

To provide the engineer with relevant recommendation candidates, we must utilize the features $f_1 - f_5$ such that the recommendations are ranked from most relevant to least relevant. To this end, we must assign a weight to each feature, but doing so manually is time-consuming and error-prone. Thus, we make use of the machine learning approach Learning to Rank that calculates the weights for the features automatically.

Learning To Rank. (L2R) refers to a class of supervised machine learning algorithms for inducing a ranking model computed from a set of given features [19]. In detail, a ranking model ϱ is derived from some training data by observing correlations between the feature values and the relevance of a recommendation candidate. Ideally, the derived model lists all relevant vocabulary terms high and above less relevant vocabulary terms.

In our case, the training data is a set of query-SLPs with existing relevance information on each recommendation candidate. It contains SLPs such as $slp_q = (\{\text{mo:SoloMusicArtist}\}, \varnothing, \varnothing)$ with the relevance information that, e.g., for recommending properties solely the terms foaf:made and mo:member_of are considered relevant. Using this information, the L2R algorithm iterates 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 at the top of the result list. This way, the learned ranking model can be used in new and previously unknown situations with new and unknown query-SLPs. For example, a query-SLP that was not part of the training set using terms from the Creative Commons[5] ontology and from an ontology for managing presentations at W3C[6] $slp_q = (\{\text{cc:Work}\}, \{\text{w3:presenter}\}, \varnothing)$ can get recommendations, such as the RDF types foaf:Person and/or dc:Agent to reuse for resources in object position.

L2R algorithms are categorized into three different groups according to their method for learning a ranking model [19]: (A) *point-wise* L2R algorithms,

[4] https://www.w3.org/2003/01/geo/wgs84_pos, last access 2/7/16.
[5] http://creativecommons.org/ns, last access 09/06/15.
[6] http://www.w3.org/2004/08/Presentations.owl, last access 09/06/15.

(B) *pair-wise* L2R algorithms, and (C) *list-wise* L2R algorithms. A point-wise approach ranks vocabulary terms directly by allocating a ranking score to each recommendation candidate individually. Pair-wise methods rank vocabulary terms solely in a given pair of two recommendation candidates. This way, a term is considered a better recommendation compared to the terms in a lower ranking position. List-wise approaches rank recommendation candidates by optimizing the quality measure of the result list, such as the Mean Average Precision (MAP).

5 Evaluation

We evaluate the proposed approach by using a 10-fold leave-one-out evaluation. Each fold comprises a *training set* to induce the ranking model, a *test set* to evaluate 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.[7]

We evaluate TermPicker's recommendation quality by investigating the following aspects: (i) Which Learning To Rank algorithm provides the best recommendation quality? (ii) To which extent does the SLP-feature (using features $f_1 - f_5$) enhance the recommendation quality compared to the baseline of reusing only popular vocabulary terms [18] (using features $f_1 - f_3$) and to the baseline of reusing popular vocabularies from the same vocabulary [20] (using features $f_1 - f_4$)? (iii) Is the recommendation quality better for recommending RDF types for resources in subject position of a triple, for recommending RDF types describing resources in object position, or for recommending properties?

To evaluate different L2R algorithms, we chose to use the RankLib[8] library, as it contains various L2R algorithms and provides an entire framework to train and evaluate an algorithm's ranking model. The recommendation quality is measured using the Mean Average Precision (MAP) and the Mean Reciprocal Rank at the first five positions (MRR@5). In the following, Sect. 5.1 describes the evaluation design in detail and we formalize the quality measures MAP and MRR@5 in Sect. 5.2.

5.1 Evaluation Design

TermPicker's recommendations are evaluated by simulating a search for a vocabulary term that can be reused. Thus, the training set and test set, which are used to induce and evaluate the ranking model, are disjoint sets of distinct SLPs. Before providing TermPicker with an SLP as input, i.e., the query-SLP, one or more vocabulary terms from that SLP are randomly selected. The selected terms are extracted from the query-SLP using the $\ominus$ operator and represent the set of *relevant* recommendation candidates, since they are the ones that have been initially used. All other recommendation candidates that are not contained in

[7] Evaluation data and results available at: https://github.com/WanjaSchaible/l2r_eval_material, last access 3/7/16.

[8] http://sourceforge.net/p/lemur/wiki/RankLib/, last access 12/12/15.

the set of the selected terms are considered as irrelevant recommendations. This way, for each query-SLP, the L2R algorithm is provided a set of recommendation candidates, five features categorizing each candidate, and the relevance information for each recommendation candidate. For example, given an SLP slp_j from the training or test set with

$$slp_j = (\{\text{mo:SoloMusicArtist}\}, \{\text{foaf:made}\}, \{\text{mo:Record}\})$$

The property foaf:made is randomly selected and extracted via the $\ominus_{ps}$ operator.

$$slp_q = slp_j \ominus_{ps} \text{foaf:made} = (\{\text{mo:SoloMusicArtist}\}, \varnothing, \{\text{mo:Record}\})$$

The query-SLP slp_q is now provided as input for TermPicker, and the property foaf:made is the single relevant recommendation candidate. This makes it possible to induce and evaluate a ranking model by interpreting a ranked list of recommendations

$$< \text{foaf:name}, \text{mo:remixed}, \textbf{foaf:made}, \ldots >$$

in the following way: the first two recommendations are irrelevant, and the first relevant recommendation is at the third rank of the result list.

We validate TermPicker's recommendation quality by performing one evaluation on the DyLDO [21][9] dataset and a second evaluation on the Billion Triple Challenge (BTC) 2014 dataset [22][10] (crawl no. 1).

DyLDO comprises a considerably large amount of data from the LOD cloud with about 10.8 million triples from 382 different pay-level domains[11] (PLDs). In total there are about 2.3 million distinct vocabulary terms from about 600 vocabularies. In this context, we regard a vocabulary simply by its URI namespace, which is either a *hash namespace* or a *slash namespace* as specified by the W3C.[12] The BTC 2014 dataset contains about 1.4 billion triples, of which we use the first 34 millions, to reduce the memory requirements for the SLP computation. These 34 million triples are provided by 3,493 pay-level domains. Within these triples there are about 5.5 million distinct RDF types and properties from about 1,530 different vocabularies. The datasets 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 dataset is split by PLDs such that the data from ten PLDs is used as training and test set and the data from the remaining PLDs is used to simulate the datasets published on the LOD cloud. The split could not be

[9] http://swse.deri.org/dyldo/, last access 12/12/15.

[10] http://km.aifb.kit.edu/projects/btc-2014/, last access 12/12/15.

[11] 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, we use the Google guava library: https://code.google.com/p/guava-libraries/, last access 2/3/16.

[12] http://www.w3.org/2001/sw/BestPractices/VM/http-examples/2006-01-18/#naming, last access 9/25/15.

Table 2. *PLDs that were selected for training and testing in our evaluation.* 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

performed randomly, as we needed to make sure that the training and test data contained enough SLPs to train and evaluate the ranking model. Thus, to generate representative results, we selected the ten pay-level domains for training and testing based on two criteria.

($C1$) A high number of distinct vocabulary terms within a PLD
($C2$) A high ratio between the reused vocabulary terms and all used 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. This ensures that the relevant recommendation candidates (the selected 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 inducing a ranking model impossible. In our evaluation, we define a reused vocabulary term as follows: if a vocabulary term does not contain the PLD in its namespace, then it is considered as a reused vocabulary term. In summary, using ($C1$) and ($C2$) enables us to calculate SLPs that most likely contain many reused terms, which is important to generate valuable recommendations.

Table 2 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) dataset 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. Of course, SLPs that are used to train the ranking

model are different from the SLPs that are used to evaluate the model. 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.

5.2 Evaluation Metrics

As an engineer who searches for possible RDF types and properties for reuse is likely to browse only through the top-k vocabulary terms (where k is generally a small number such as 5 or 10), it is important to evaluate the ranking model by measures that use ordered sets of vocabulary terms. We use the Mean Average Precision (MAP) and the Mean Reciprocal Rank to the k-th position (MRR@k). Both measures present the quality of the ranking model well as they compute values using such ordered sets of vocabulary terms (in contrast to basic measures such as precision and recall).

On the one hand, MAP provides a measure of quality across recall levels. It illustrates the quality of the entire result list in which the ranking position of the relevant vocabulary term is considered. The higher the MAP value, the more relevant vocabulary terms are ranked to the top positions of the result list. On the other hand, MRR@k investigates the result list only to the rank position of the first relevant vocabulary term or to the k-th position. 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.

We define the set of query-SLPs as $Q = \{slp_{q_1}, ..., slp_{q_n}\}$. If the set of relevant vocabulary terms for a query $slp_{q_j} \in Q$ is $\{rt_1, \ldots, rt_{m_j}\}$ 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 vocabulary term rt_h, then the **Mean Average Precision** and the **Mean Reciprocal Rank** of Q defined as

$$\text{MAP}(Q) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} \frac{1}{m_j} \sum_{h=1}^{m_j} \text{Precision}(R_{jh}) \qquad \text{MRR}(Q) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} \frac{1}{|R_{jh}|} \tag{3}$$

In the following, we use $k = 5$. For MRR@5 it means that one relevant terms must be in the first five results ($|R_{jh}| \leq 5$). Furthermore, MRR@5 uses solely the ranked retrieval results R_{jh} with the minimum amount of vocabulary terms, such that iterating over h is not necessary.

6 Results

The results of our evaluation are presented in Fig. 2 as recommendation quality via box-plots based on MAP. The recommendation quality based on the MRR@5 measure is very much identical compared to the MAP values. The figure depicts the measurements of the recommendation quality considering the aspects (i), (ii), and (iii) mentioned in Sect. 5, where the most competitive L2R algorithms in the RankLib library are: *Coordinate Ascent* [23], *LambdaMART* [24], and *Random Forests* [25]. Both reusing solely popular vocabulary terms (using features $f_1 - f_3$

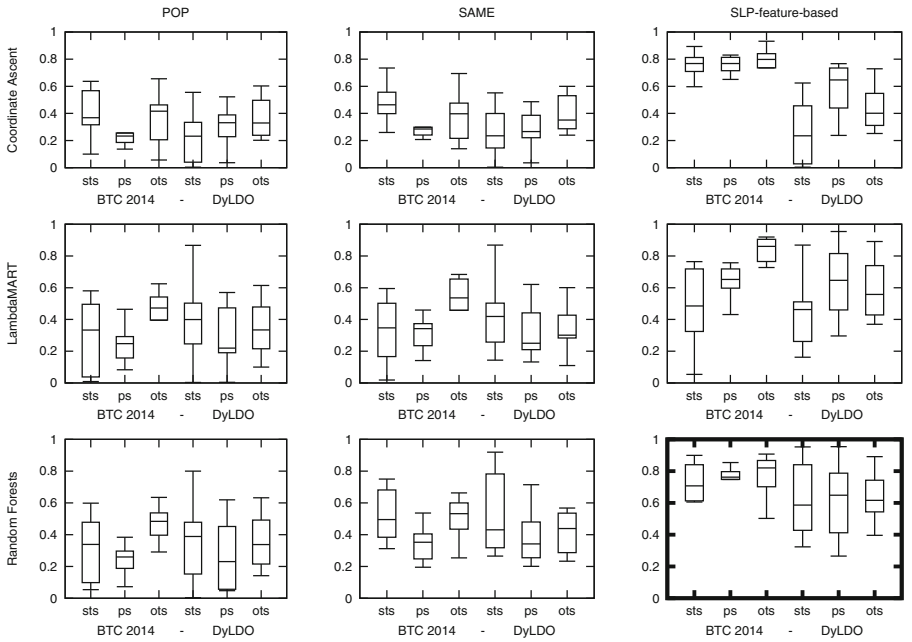

Fig. 2. Evaluation results based on MAP. The plots show the recommendation quality for RDF types for resources in subject position *sts*, properties *ps*, and RDF types for resources in object position *ots* (BTC on the left, DyLDO on the right) based on the different set of features (POP: $f_1 - f_3$; SAME: $f_1 - f_4$; SLP-feature-based: $f_1 - f_5$). The plot in bold font depicts the overall best results, which is the Random Forests algorithm using the SLP-feature.

and marked as POP) and reusing vocabulary terms from the same vocabulary (using features $f_1 - f_4$ and marked as SAME) resemble the baseline. Our SLP-feature-based approach uses features $f_1 - f_5$. Each boxplot displays the different recommendations of an RDF type for resources in subject position (abbreviated as *sts*), a RDF type for resources in object position (abbreviated as *ots*), and a property (abbreviated as *ps*) for both the BTC 2014 and the DyLDO dataset, and comprises the measured values from each fold. The plot in bold font presents the best performing configuration.

In addition, Table 3 shows the average MAP and MRR@5 values (including the standard deviation) for the evaluations using the best performing algorithm *Random Forests* based on the BTC 2014 and the DyLDO datasets, respectively. They underpin the increase of the recommendation quality when adding the SLP-feature to the set of features, which is used by the ranking model. For the BTC 2014 dataset, using the SLP-feature provides on average a higher MAP and MRR@5 value by 29 % compared to the SAME baseline ($f_1 - f_4$), and by 36 % compared to the POP baseline ($f_1 - f_3$). For the DyLDO data, these differences are not as distinctive, but they are still 13 % compared to the baseline

Table 3. *MAP and MRR@5 values for BTC 2014 and DyLDO using the Random Forests algorithm.* Each row depicts the average MAP and MRR@5 values and their standard deviation for the Random Forests algorithm and the utilized set of features.

Data set	Features	sts		ps		ots		overall	
		MAP	MRR@5	MAP	MRR@5	MAP	MRR@5	MAP	MRR@5
BTC	POP	.32 (.20)	.40 (.21)	.26 (.12)	.28 (.12)	.45 (.17)	.48 (.15)	.34 (.16)	.39 (.16)
	SAME	.52 (.16)	.56 (.15)	.37 (.14)	.39 (.14)	.49 (.16)	.50 (.17)	.46 (.15)	.48 (.15)
	SLP	.72 (.11)	.80 (.10)	.75 (.10)	.77 (.10)	.78 (.12)	.83 (.08)	.75 (.11)	.80 (.09)
DyLDO	POP	.44 (.29)	.55 (.31)	.35 (.28)	.36 (.28)	.43 (.25)	.49 (.26)	.41 (.27)	.47 (.28)
	SAME	.59 (.27)	.65 (.24)	.46 (.24)	.46 (.24)	.49 (.21)	.52 (.21)	.51 (.24)	.54 (.23)
	SLP	.65 (.26)	.70 (.24)	.63 (.25)	.63 (.24)	.64 (.17)	.68 (.15)	.64 (.23)	.67 (.21)

SAME and 23 % compared to the baseline POP. In summary, the overall best recommendation quality is provided by the point-wise L2R algorithm *Random Forests* using all features, including the SLP-feature.

(i) Differences between L2R algorithms. Comparing the three most competitive L2R algorithms and making use of all features, one can observe that with MAP ≈ 0.8 based on the BTC 2014 dataset, the algorithms *Coordinate Ascent* and *Random Forests* outperform the *LamdaMART* algorithm (MAP ≈ 0.6). Using the DyLDO data set, the differences between the L2R algorithms are not as noticeable, but are still significant, as a Friedman test shows ($\mathcal{X}^2 = 14,000, p = .001$). 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 ($Z = -4.237, p < .001$).

The reason why *Random Forests* outperforms the other algorithms can be explained by the fact that we use only binary relevance, i.e., a recommendation candidate is either relevant or irrelevant, in our evaluation. Point-wise approaches perform better in such scenarios, especially if there are solely one or a few relevant recommendation candidates for most queries [26].

(ii) Impact of the SLP-feature. Most noticeable is the influence of the SLP-feature when using the BTC 2014 dataset as evaluation data. The median recommendation quality increases by about 30 % compared to the baseline of reusing solely popular vocabulary terms (POP) and by 20 % compared to the SAME baseline. Such differences are not as noticeable when performing the evaluation on the DyLDO dataset. However, they are still around 15–20 % compared to the baselines POP and SAME. In total, using the SLP-feature increases the average MAP value up to MAP ≈ 0.75. Applying a Friedman test ($\mathcal{X}^2 = 51,667, p < .001$) explains that the differences between using the SLP-feature and the baselines are significant. A post-hoc Wilcoxon signed-rank test shows that using the SLP-feature provides significantly better recommendations ($Z = -4.782, p < .001$ compared to the SAME baseline and $Z = -5.832, p < .001$ compare to the POP baseline).

Such a result shows to which extent the SLP-feature is relevant for providing valuable vocabulary term recommendations. It makes it very likely to recommend the engineer vocabulary terms that might lead to reducing heterogeneity in data representation.

(iii) Differences between recommendation types. Before modeling data as LOD, it is common to describe how the data objects are related.[13] One could first define a set of classes, which depict the objects, and then define relationships connecting these classes, or vice versa. Objects are often described using more than one RDF type[14], whereas there is usually one property to describe a relationship. For that reason, one can argue that TermPicker could provide better property recommendations, as there are many possible RDF types, of which only a few are considered relevant in our evaluation design. However, this was not an influencing factor in our experiment. Comparing the recommendations between RDF types for resources in subject position, RDF types in object position, or properties, only slight differences (between 5–10 %) in the recommendation quality can be perceived when utilizing all features. A Friedman test shows significant differences ($\mathcal{X}^2 = 14,000$, **n.s.**, $p = .449$).

7 Discussion

Our experiments show that using the SLP-feature significantly improves the recommendation quality of vocabulary terms. However, this improvement was not as noticeable when performing the evaluation with the DyLDO dataset. Further investigations showed that the training sets based on the BTC 2014 dataset and the DyLDO dataset had differences regarding their query-SLPs. In detail, the training sets based on the BTC 2014 dataset contained 37 % more relevant recommendation candidates with an SLP-feature value greater than zero ($f_5 > 0$) than the DyLDO data. This means the ranking model induced from the DyLDO data did not encounter the SLP-feature to be improving the recommendation quality. Using the BTC 2014 dataset, the number of SLPs simulating datasets on the LOD cloud is twice as high compared to the number of such SLPs using the DyLDO data. In general, the larger the data for calculating the feature values, the more representative are the generated results [26]. Therefore, we can argue that the results based on using the BTC 2014 dataset are more representative than the results of the evaluation based on the DyLDO data.

A potential threat to the validity of our experiments is the utilized evaluation design. It considers solely the recommendation candidates as relevant that have been selected and extracted from a query-SLP before providing this query-SLP as input for TermPicker (cf. Sect. 5.1). This leads to two major weaknesses. First, many recommendation candidates are identified as irrelevant, although they are appropriate considering the rdfs:domain and rdfs:range,

[13] https://www.w3.org/TR/ld-bp/#MODEL, last access 2/7/16.

[14] See the rdf:type information on Pelé: http://dbpedia.org/page/Pele, last access 12/12/15.

the owl:equivalentClass, or other information. For example, let us assume we selected and extracted the property foaf:made from an SLP, resulting in the query-SLP ($\{$mo:MusicArtist$\}, \varnothing, \varnothing$). The only relevant recommendation candidate for properties is this foaf:made property, as it was used in the original SLP. Properties, such as mo:produced or mo:remixed are considered as irrelevant in our evaluation, although it would make sense to reuse them as outgoing properties from mo:MusicArtist. Thus, an L2R algorithm may identify many *commonly used* vocabulary terms (with an SLP-feature value greater than zero) as irrelevant, which then can result in a ill-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 selected and extracted, then it is very likely that the SLP-feature value would be zero. This selected term is considered relevant though. Thus, the L2R algorithm learns to decrease the usefulness of the SLP-feature. However, addressing these limitation requires human judgment whether a recommendation candidate is relevant. We do not utilize human judgment, as the automatic evaluation enables us to use many queries and many recommendation candidates to establish a generalized ranking model using a lot of data. Otherwise, we would need a lot of domain experts from various domains judging whether a candidate is relevant or not, which is not feasible.

8 Conclusion

This paper presented TermPicker, a novel approach for recommending vocabulary terms for reuse based on the information how other data providers on the LOD cloud modeled their data. We introduced the notion of schema-level patterns (SLPs) that represent such information and presented a set of features (among them the SLP-feature) that is used by Learning To Rank algorithms to induce a ranking model. We demonstrated that using the SLP-feature increases the recommendation quality by about 35 % compared to the baselines of recommending vocabulary terms from popular vocabularies and recommending terms from the same vocabulary. In total numbers, the Mean Average Precision (MAP) is approximately 0.75 for our recommendation tasks used in the evaluation. Finally, based on the evaluation design that assesses the relevance of a recommendation candidate automatically, it seems that point-wise Learning To Rank (L2R) algorithms provide better results than pair-wise or list-wise L2R algorithms. As future work, we will perform a user study evaluating the recommendation quality by assessing the users' satisfaction with the recommendations.

References

1. Heath, T., Bizer, C.: Synthesis lectures on the semantic web. In: Linked Data: Evolving the Web into a Global Data Space. Morgan & Claypool Publishers, San Rafael (2011)

2. Vandenbussche, P.Y., Atemezing, G.A., Poveda-Villalón, M., Vatant, B.: Linked Open Vocabularies (LOV): a gateway to reusable semantic vocabularies on the Web. Semantic Web J. (Preprint) 1–16 (2015)
3. Auer, S., Demter, J., Martin, M., Lehmann, J.: LODStats – an extensible framework for high-performance dataset analytics. In: ten Teije, A., Völker, J., Handschuh, S., Stuckenschmidt, H., d'Acquin, M., Nikolov, A., Aussenac-Gilles, N., Hernandez, N. (eds.) EKAW 2012. LNCS, vol. 7603, pp. 353–362. Springer, Heidelberg (2012)
4. Fernandez, M., Cantador, I., Castells, P.: Core: a tool for collaborative ontology reuse and evaluation. In: 4th International Workshop on Evaluation of Ontologies for the Web (2006)
5. Schaible, J., Gottron, T., Scherp, A.: TermPicker: enabling the reuse of vocabulary terms by exploiting data from the linked open data cloud - an extended technical report. ArXiv e-prints, December 2015. http://arxiv.org/abs/1512.05685
6. d'Aquin, M., Baldassarre, C., Gridinoc, L., Sabou, M., Angeletou, S., Motta, E.: Watson: supporting next generation semantic web applications. In: Proceedings of the IADIS International Conference WWW/Internet 2007, pp. 363–371 (2007)
7. Cheng, G., Ge, W., Qu, Y.: Falcons: searching and browsing entities on the semantic web. In: Proceedings of the 17th International Conference on World Wide Web. ACM (2008)
8. García-Santa, N., Atemezing, G.A., Villazón-Terrazas, B.: The ProtégéLOV plugin: ontology access and reuse for everyone. In: Gandon, F., Guéret, C., Villata, S., Breslin, J., Faron-Zucker, C., Zimmermann, A. (eds.) ESWC 2015. LNCS, vol. 9341, pp. 41–45. Springer, Switzerland (2015)
9. Scharffe, F., Atemezing, G., Troncy, R., Gandon, F., et al.: Enabling linked-data publication with the datalift platform. In: AAAI 2012, 26th Conference on Artificial Intelligence (2012)
10. Cheng, G., Gong, S., Qu, Y.: An empirical study of vocabulary relatedness and its application to recommender systems. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 98–113. Springer, Heidelberg (2011)
11. Ramnandan, S.K., Mittal, A., Knoblock, C.A., Szekely, P.: Assigning semantic labels to data sources. In: Gandon, F., Sabou, M., Sack, H., dAmato, C., Cudre-Mauroux, P., Zimmermann, A. (eds.) ESWC 2015. LNCS, vol. 9088, pp. 403–417. Springer, Heidelberg (2015)
12. Taheriyan, M., Knoblock, C.A., Szekely, P., Ambite, J.L.: Learning the semantics of structured data sources. Web Semant. Sci. Serv. Agents World Wide Web (2016). ISSN: 1570-8268. doi:10.1016/j.websem.2015.12.003
13. Knoblock, C.A., et al.: Semi-automatically mapping structured sources into the semantic Web. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 375–390. Springer, Heidelberg (2012)
14. Presutti, V., Aroyo, L.M., Gangemi, A., Adamou, A., Schopman, B., Schreiber, G.: A knowledge pattern-based method for linked data analysis. In: Proceedings of the Sixth International Conference on Knowledge Capture, pp. 173–174. ACM (2011)
15. Zhang, Z., Gentile, A.L., Blomqvist, E., Augenstein, I., Ciravegna, F.: Statistical knowledge patterns: identifying synonymous relations in large linked datasets. In: Alani, H., et al. (eds.) ISWC 2013, Part I. LNCS, vol. 8218, pp. 703–719. Springer, Heidelberg (2013)

16. Campinas, S., Perry, T.E., Ceccarelli, D., Delbru, R., Tummarello, G.: Introducing RDF graph summary with application to assisted SPARQL formulation. In: 23rd International Workshop on Database and Expert Systems Applications (DEXA), pp. 261–266. IEEE (2012)
17. Dudáš, M., Svátek, V., Mynarz, J.: Dataset summary visualization with LODSight. In: Gandon, F., Guéret, C., Villata, S., Breslin, J., Faron-Zucker, C., Zimmermann, A. (eds.) ESWC 2015. LNCS, vol. 9341, pp. 36–40. Springer, Heidelberg (2015). doi:10.1007/978-3-319-25639-9_7
18. Schaible, J., Gottron, T., Scherp, A.: Survey on common strategies of vocabulary reuse in linked open data modeling. In: Presutti, V., d'Amato, C., Gandon, F., d'Aquin, M., Staab, S., Tordai, A. (eds.) ESWC 2014. LNCS, vol. 8465, pp. 457–472. Springer, Heidelberg (2014)
19. Liu, T.Y.: Learning to rank for information retrieval. Found. Trends Inf. Retrieval **3**(3), 225–331 (2009)
20. Lodi, G., Maccioni, A., Scannapieco, M., Scanu, M., Tosco, L.: Publishing official classifications in linked open data. In: Proceedings of the 2nd International Workshop on Semantic Statistics (SemStats2014) in conjunction with the 13th International Semantic Web Conference (ISWC). Springer, Riva del Garda, Italy (2014)
21. Käfer, T., Abdelrahman, A., Umbrich, J., O'Byrne, P., Hogan, A.: Observing linked data dynamics. In: Cimiano, P., Corcho, O., Presutti, V., Hollink, L., Rudolph, S. (eds.) ESWC 2013. LNCS, vol. 7882, pp. 213–227. Springer, Heidelberg (2013)
22. Käfer, T., Harth, A.: Billion Triples Challenge data set (2014). http://km.aifb.kit.edu/projects/btc-2014/
23. Metzler, D., Croft, W.B.: Linear feature-based models for information retrieval. Inf. Retrieval **10**(3), 257–274 (2007)
24. Wu, Q., Burges, C.J., Svore, K.M., Gao, J.: Adapting boosting for information retrieval measures. Inf. Retrieval **13**(3), 254–270 (2010)
25. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
26. Busa-Fekete, R., Szarvas, G., Elteto, T., Kégl, B., et al.: An apple-to-apple comparison of learning-to-rank algorithms in terms of normalized discounted cumulative gain. In: 20th European Conference on Artificial Intelligence, ECAI 2012, vol. 242 (2012)