LOVBench: Ontology Ranking Benchmark

Niklas Kolbe
University of Luxembourg
niklas.kolbe@uni.lu

Pierre-Yves Vandenbussche
Elsevier
p.vandenbussche@elsevier.com

Sylvain Kubler
Université de Lorraine
s.kubler@univ-lorraine.fr

Yves Le Traon
University of Luxembourg
yves.letraon@uni.lu

ABSTRACT

Ontology search and ranking are key building blocks to establish and reuse shared conceptualizations of domain knowledge on the Web. However, the effectiveness of proposed ontology ranking models is difficult to compare since these are often evaluated on diverse datasets that are limited by their static nature and scale. In this paper, we first introduce the *LOVBench* dataset as a benchmark for ontology term ranking. With inferred relevance judgments for more than 7000 queries, *LOVBench* is large enough to perform a comparison study using learning to rank (LTR) with complex ontology ranking models. Instead of relying on relevance judgments from a few experts, we consider implicit feedback from many actual users collected from the Linked Open Vocabularies (LOV) platform. Our approach further enables continuous updates of the benchmark, capturing the evolution of ontologies' relevance in an ever-changing data community. Second, we compare the performance of several feature configurations from the literature using *LOVBench* in LTR settings and discuss the results in the context of the observed real-world user behavior. Our experimental results show that feature configurations which are (i) well-suited to the user behavior, (ii) cover all features types, and (iii) consider decomposition of features can significantly improve the ranking performance.

KEYWORDS

ontology search, ground truth mining, ontology reuse, semantic interoperability, learning to rank

ACM Reference Format:
Niklas Kolbe, Pierre-Yves Vandenbussche, Sylvain Kubler, and Yves Le Traon. 2020. LOVBench: Ontology Ranking Benchmark. In *Proceedings of The Web Conference 2020 (WWW '20), April 20–24, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3366423.3380245

1 INTRODUCTION

Ontology search provides users with a ranked list of either ontologies[1] or their terms for a given query. Efficient ontology search helps users to find and reuse existing knowledge on the Web [7, 14, 18, 20, 33], which benefits communities by establishing consensus on domain conceptualizations (e.g., as in the biomedical domain [39]) as well as the breakup of vertical data silos (e.g., as in Open Data [5] and the Internet of Things (IoT) [22]). This, ultimately, eases the discovery and reuse of Web data, fostering interoperability among computing systems.

However, the evaluation of ontology ranking models that were proposed in this context is relatively unexplored. The heterogeneity of evaluation strategies and underlying ontology collections makes it difficult to understand, interpret and compare the performance of proposed ranking models. Existing datasets for ontology ranking evaluation, such as the state-of-the-art benchmark CBRBench [7], are often built based on explicit judgments from human experts. However, the laborious approach to building such benchmarks led to relatively small datasets (in case of CBRBench, only comprising ten queries with 819 relevance judgments). Given that ontology collections typically contain hundreds of ontologies and thousands of terms, benchmarks that only comprise ten queries are not sufficient for the evaluation of ranking models in real-world settings.

To overcome this issue and address the lack of ontology ranking comparison studies, we propose the ontology ranking benchmark *LOVBench*. It comprises both a dataset including a ground truth for ranking evaluations and an empirical comparison of state-of-the-art ranking models. The LOVBench dataset contains more than 180,000 inferred relevance judgments for more than 7,000 queries. LOVBench's ground truth relies on implicit, real-world user feedback in the form of queries and clicks that were collected from the Linked Open Vocabularies (LOV) platform[2] [41], which comes with the following advantages. First, the relevance judgments for ontology terms to a query in LOVBench stem from feedback of many actual users, which is more representative than judgments from a few experts; second, collecting user feedback through search logs does not require manual effort, meaning the benchmark can be continuously updated to capture the evolution of ontologies and their relevance in the Semantic Web. The empirical evaluation is performed based on learning to rank (LTR) [24]. Compared to combining ranking features manually, LTR allows to learn the optimal combination of the considered features using supervised machine learning techniques and, thus, allows for a fair comparison of complex ranking models. We measure the models' ranking quality using the Normalized Discounted Cumulative Gain (NDCG) [19] as evaluation metric and we compare ranking configurations as proposed in DWRank [8], AKTiveRank [2] and CBRBench [7]. Our experiments confirm that all considered models outperform a straight-forward Lucene search [28]. We further show that the

[1] In the Semantic Web, ontologies are also referred to as *linked vocabularies.*

[2] https://lov.linkeddata.es/dataset/lov/

WWW '20, April 20–24, 2020, Taipei, Taiwan

Table 1: Comparison of existing benchmarks and the proposed LOVBench dataset.

Reference	Year	Query	Ranking	Relevance Source	Labels	Queries	Judgments	Features	Dataset	Impl.
AKTiveRank [2]	2006	Keyword	Ontology	Four experts	List	7	13	4	No	No
CARRank [43]	2008	Keyword	Class	Four experts	List	80	~ 400	1	No	No
CBRBench [7]	2014	Keyword	Class	Ten experts	Point (graded)	10	819	8	Yes	No
TermPicker [38]	2016	Triple-Pattern	Triple-Pattern	Mining (LOD)	Point (binary)	5,650	3,010,620	5	Yes	No
Quality Ranking [21]	2019	Keyword	Ontology	Mining (scholarly)	Point (graded)	25	1,028	11	Yes	No
LOVBench	2020	Keyword	Term	Mining (click logs)	Point (graded)	7,395	184,224	33	Yes	Yes

consideration of features that are well-suited to the user behavior as well as the consideration of much larger feature sets (given a large enough dataset like LOVBench), can significantly improve the ranking performance. The insights gained in this study help to understand the effectiveness of ontology ranking models and can serve as guidelines for the design of more efficient ontology ranking tools in terms of ranking quality.

In summary, the main contributions of this paper are as follows:

(1) We provide insights of real-world user behavior in ontology search of which some dissent dominant assumptions in the literature;
(2) we introduce and publish the LOVBench dataset, a large-scale ontology ranking benchmark based on actual and timely user feedback that allows for continuous updates and enables comparison studies using learning to rank;
(3) we provide an experimental evaluation of ranking effectiveness of three state-of-the-art ontology ranking models (composed of 13 distinct ranking features) and propose feature variations and configurations that outperform them (using in total 33 distinct ranking features);
(4) we conclude with recommendations for the design of efficient ontology search tools.

The remainder of this paper is structured as follows. In Section 2, we review previous related work. We present our approach to building an ontology ranking benchmark based on user clicks in Section 3. We evaluate the usefulness of user feedback collected through LOV in Section 4 and describe the details of the LOVBench dataset in Section 5. We report the subsequent empirical evaluation in Section 6 and discuss our findings in Section 7. Finally, we conclude the paper in Section 8.

2 RELATED WORK

In this section, we present related research on the evaluation of ontology ranking models and the application of LTR with ontology and term ranking features.

The importance of search within the Semantic Web has been identified from early on [4]. Despite the huge amount of works on ontology ranking through content and link analysis, only few unified approaches to evaluate these models have been proposed. Early works on ranking in the Semantic Web, such as Swoogle [15], lacked any evaluation of the ranking quality, which was soon identified as a general problem [35]. In the following years, authors presented diverse evaluations along with proposed ranking models, either based on small user studies or qualitative discussions on ranking outputs: Ding et al. [16] made a qualitative comparison between two ranking models and highlighted the need for relevance judgments in order to perform a formal evaluation; similarly, Lei et al. agreed that their evaluation is biased based on their qualitative assessment [23]. A first evaluation based on how well the ranking matches user judgments was presented in AKTiveRank [2] using the Pearson Correlation Coefficient. The evaluation includes a comparison with Swoogle's ranking, which resulted in a negative correlation with the ground truth (inverse ranking) and proved the lack of evaluations to be problematic. A similar evaluation approach has been followed later in CARRank [43]. Other evaluation approaches that were followed did not directly assess the ranking performance, but rather the tool's overall user experience based on a questionnaire [32], which requires extensive efforts and does not allow for immediate comparisons with newly proposed rankings. Noy et al. were the first to use more cost-efficient user feedback in the form of clicks to evaluate ontology rankings [30]. The evaluation is based on the Click Through Rate (CTR) on the first ranking position for several modifications of the ranking model over similar time intervals. However, the authors do not mine any relevance labels for the query-ontology pairs from the search logs. More recently published ontology ranking models still consider the evaluation as a challenging task. The complex NCBO2.0 recommender [26] of the BioPortal is evaluated with a qualitative discussion based on absolute feature scores in comparison with a previous version of the NCBO recommender. The authors state that a user-based evaluation would help to understand the ranking's real-world effectiveness, but refrain from it since it would be a laborious task.

Despite the successful emergence of many ontology search tools over time in various domains, no attempts to unify ontology ranking evaluation known to us have been proposed until CBRBench [7] was released, which is the closest work to ours. To the best of our knowledge, CBRBench is the first published dataset with relevance labels for query-term pairs obtained from experts. It has further been used as input for LTR, such as in DWRank [8]. However, CBRBench is limited as follows. First, it only contains a small number of queries and judgments (10 and 819, respectively), meaning that its application in LTR settings is limited by its size. Second, it only compares a few ranking models individually (eight in total), and not a (potentially learnt) combination of these or in configurations as they have been proposed in the literature. Moreover, it mostly compares ontology instead of term ranking features that assign the same score for all classes of the same ontology [7]. Third, CBRBench is restricted to class ranking, meaning that it does not provide any judgments about properties, which limits its potential application

to newly proposed ranking models. Fourth, the dataset might become outdated as soon as new ontologies arise and relevance of terms changes, and because of its laborious way of obtaining the relevance labels, it cannot be easily updated. Apart from CBRBench, other datasets have been proposed for LTR in the context of ontology ranking. In TermPicker [38], a dataset is automatically derived from the Linked Open Data (LOD) cloud, however, relevance judgments only indicate whether a certain triple pattern appears in the LOD cloud or not (binary relevance) and it does not contain any judgments for keyword-based queries. Kolbe et al. [21] propose a dataset derived from scholarly data, which, however, only considers ontology and no term ranking, is also limited by its size (1,028) and focuses on IoT ontologies.

With LOVBench, we aim to overcome these limitations. An overview of existing benchmarks and LOVBench is presented in Table 1. In contrast to the related work, LOVBench satisfies all the following: it relies on keyword-based queries, ranks classes as well as properties of ontologies from various domains, graded relevance labels are mined cost-efficiently from actual, real-world user feedback (views and clicks), and it provides sufficient judgments for complex models in LTR settings (7,395 queries and 184,224 relevance judgments). Moreover, our comparison of state-of-the-art ranking models considers best combinations of originally proposed along with newly designed features (in total 33), and we make the dataset as well as the feature implementation publicly available.

3 LOVBENCH APPROACH

In this section, we present our approach to benchmarking ontology ranking models. As illustrated in Figure 1, our approach comprises four main steps: (1) analyzing the collected LOV search logs (containing users' queries and clicks), (2) inferring and evaluating relevance labels, (3) creating the LOVBench dataset, and (4) performing the empirical evaluation of ontology ranking models. In the following, we present an overview of each step and respective outcomes including key features for the design of LOVBench.

Analyzing LOV search logs. In the first step (cf. Section 4.1), we analyze the LOV search logs to confirm whether the logs fulfill basic requirements to generalize user behavior for ontology search. Moreover, we extract fundamental insights concluded from the observed real-world user actions in the context of literature assumptions on ontology ranking design, which guide our later feature design and discussions on experimental results.

Inferring relevance labels. In the second step (cf. Section 4.2), we learn a user click model [10] based on the LOV search logs in order to infer relevance labels for query-term pairs. Considering user's reactions (i.e., clicks) on presented terms for a query to infer relevance labels has several advantages. Compared to obtaining explicit judgments from human experts, which may contain bias and become outdated, collecting implicit user feedback through logs is cost-efficient and further captures actual, potentially time-varying user preferences [42]. This helps to overcome the problem that the amount of training data in LTR settings is often low [21, 40]. However, clicks cannot be used as direct relevance judgments due to user's inherent biases (e.g., position bias), click incompleteness (missing feedback for relevant terms), and noise (a click does not necessarily imply relevance of a term) [1]. The purpose of user click

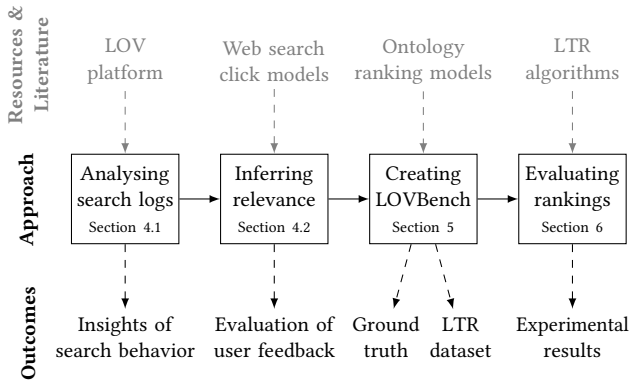


Figure 1: Overview of the LOVBench approach.

models is to take these considerations into account and allow for the inference of actual relevance based on observed user clicks. In order to evaluate whether the LOV search logs allow us to infer meaningful relevance labels, we pursue the following steps:

(1) We learn and compare several user click models with well-established assumptions of user behavior in Web search,
(2) subsequently infer relevance judgments for query-term pairs contained in the logs, and
(3) evaluate these relevance predictions by comparing them with judgments from human assessors of CBRBench [7].

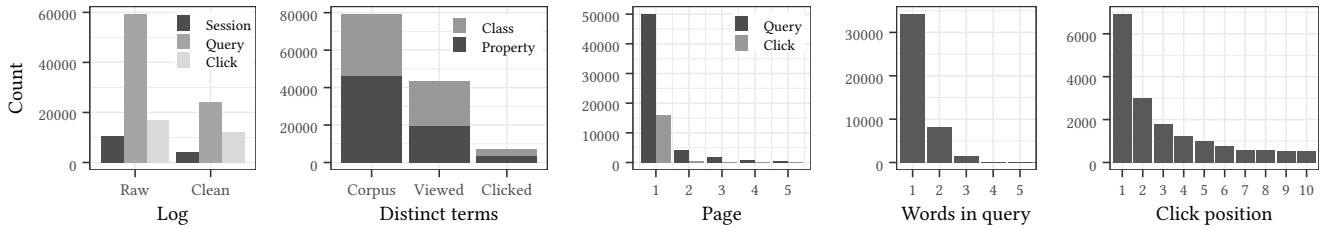
These steps allow us to build a benchmark with relevance labels that are best aligned with both, observed user clicks as well as expert judgments.

Creating LOVBench. In general, an ontology benchmark is composed of two main components: first, a set of sampled query-term pairs with relevance judgments (the ground truth) and second, the extracted scores of selected ranking features for these query-term pairs (used as input for LTR experiments). In the third step of our LOVBench approach (cf. Section 5), we first sample terms for the ground truth based on common practices (similar to the one followed by LETOR [31]). Second, our strategy for the selection of ranking features for LOVBench is guided by related literature surveys [14, 22, 45], and respectively discovered features are selected based on applicability constraints. In summary, we build the LOVBench dataset as follows.

(1) For each query contained in the search log, we sample relevant and non-relevant terms following common practices,
(2) we infer the relevance labels based on the best-performing user click model of the previous step,
(3) we select ranking features from the literature and propose variations, and
(4) we extract corresponding feature scores for each query-term pair of the sample.

As a result, we derive a ground truth file for ontology evaluations and a dataset that can be used for LTR experiments.

Evaluating ranking models. The last step of our benchmark (cf. Section 6) is concerned with the comparison of the ranking models' performance based on the LOVBench dataset. We follow a standard LTR experimental methodology [24] and use standard



(a) Log size (b) Log term coverage (c) Actions per SERP (d) Query length (e) Clicks on first SERP

Figure 2: LOV search log analysis.

evaluation metrics for information retrieval (NDCG). In addition to feature configurations as proposed in the literature, we propose novel variations that are motivated by the observed user behavior from the LOV search logs.

4 LOV USER FEEDBACK EVALUATION

In this section, we provide insights of real-world user behavior for ontology search. We analyze the search logs in Section 4.1 and present the inference of relevance labels in Section 4.2.

4.1 Search Log Analysis

The ontology collection of LOV, at the time of writing, consists of 680 ontologies associated with 43 different domains. User interactions with the LOV interface are logged upon queries and clicks, without storing any user information. Figure 2 provides details of the query and click log files collected from LOV's term search of a 7-months period starting from 01/2019.

The raw log files contain 10,579 user sessions with 59,398 queries and 17,125 clicks (see Figure 2(a)). In Figure 2(b), we illustrate the amount of terms that are defined in the LOV corpus that have also been viewed and clicked by users. The coverage shows that the LOV search logs are sufficient to generalize search behavior with regard to the corpus size. The pre-processing for the clean logs include removing sessions without clicks, queries containing personal information, as well as queries beyond the first Search Engine Result Page (SERP). The latter is motivated by our observation that only very few users made use of the pagination feature (i.e., only few actions were logged beyond the first page), as shown in Figure 2(c). We further made the following observations regarding the user behavior of which some are in contrast with assumptions made in the literature.

Importance of property ranking. The literature often focuses on class ranking, and some ranking and evaluation approaches do not consider properties [2, 7, 8]. As illustrated in Figure 2(b), we observe equal amount of views and clicks of classes and properties in LOV, which should be considered when designing ranking features.

Multiple words in keyword query. Some approaches assume that user queries contain multiple words, e.g., when considering the number of words that match the term [8] and relying on the distances in the ontology graph of matched terms for each word in the query [2]. However, as shown in Figure 2(d), we observe that the majority of queries in LOV are single words, which means the potential usefulness of these ranking models is limited to a small fraction of queries.

Position for evaluation. The frequency of click positions of the first SERP is illustrated in Figure 2(e), which shows a strong bias towards the first position in the result list. Ranking evaluations should thus not only consider the complete SERP (first ten positions), which is a common choice in the literature [7, 8], but further based on metrics that only consider the more important top positions of the SERP.

These considerations guide our approach for the creation of LOVBench, as well as the design and discussion on the LTR experiments in later sections.

4.2 Relevance Inference

In this section, we learn several user click models from the cleaned LOV search logs in order to infer meaningful relevance labels from the implicit user feedback. We then evaluate their accuracy through a comparison with CBRBench, which contains judgments from human experts. We first describe the setup and subsequently present the evaluation results.

4.2.1 Learning User Click Models. In order to model user click behavior from the LOV search logs, we learn several models that were proposed in the literature to later select the best performing one for the inference of relevance judgments in LOVBench. We choose to experiment with the two best-performing modeling approaches presented in [10], i.e., the User Browsing Model (UBM) [17] and the Dynamic Bayesian Network Model (DBN) [9]. As a baseline, we consider a simple Document-based Click-through Rate Model (DCTR) [13]. We randomly select 75% of the logged LOV search sessions for training and hold the remaining 25% out for testing to learn each model[3].

After learning the user click model we can infer relevance for the query-term pairs contained in the search logs. In general, inferring relevance *Rel* for a query Q and a document (i.e., a term t) from a learnt click model is based on user satisfaction probability [10].

$$Rel_{Q,t} = P(S_t = 1|C_t = 1) * P(C_t = 1|E_t = 1) \quad (1)$$

where S_t, C_t, and E_t are binary random variables corresponding to the user's satisfaction probability, click probability, and examination probability for a term t, respectively. The considered user click

[3]Implementation taken from https://github.com/markovi/PyClick

Table 2: Evaluation of learnt LOV click models with regard to click and relevance prediction. Best performance for each metric is highlighted in bold.

Model	Click			Relevance
	Log-likelihood	Perplexity	Gain	r_{CBRBench}
DCTR	−0.274	1.299	-	0.448
DBN	−0.271	1.227	0.240	0.282
UBM	**−0.173**	**1.187**	**0.375**	**0.613**

models differ as follows. UBM – unlike DBN – does not consider actual user satisfaction in the model, meaning this formula simplifies to the attractiveness probability (i.e., perceived relevance). Another fundamental difference of UBM's and DBN's assumptions concern the examination probability, which in UBM depends on previous clicks and their ranks, while in DBN it depends only on the term position [10]. In DCTR, relevance corresponds to the click-through rate, i.e., how often a term was clicked compared to how often it was shown.

4.2.2 Click Prediction Evaluation. As a first step, we evaluate the learnt models with common evaluation measures on the held-out test set with regard to click prediction. One evaluation measure for this purpose is the standard log-likelihood [10].

$$LL(M) = \sum_{s \in \mathcal{S}} \sum_{r=1}^{n} \log P_M(C_r = c_r^s | C_{<r} = c_{<r}^s) \tag{2}$$

where P_M corresponds to model's M click probability measure at rank r for a session $s \in S$ in the test set, and c_r being the actual click information. We further consider perplexity based on full probability and perplexity gain [10].

$$\begin{aligned} \text{Perplexity}(M) &= \frac{1}{n} \sum_{r=1}^{n} 2^{-\frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} (c_r \log_2 + (1-c_r) \log_2 (1-p_r))} \\ \text{Gain}(M_A, M_B) &= \frac{\text{Perplexity}(M_B) - \text{Perplexity}(M_A)}{\text{Perplexity}(M_B) - 1} \end{aligned} \tag{3}$$

where p_r corresponds to the model's predicted click probability ($P_M(C_r = 1|q, t)$).

The results on the model's ability to predict the clicks are summarized in Table 2. Both, DBN and UBM outperform the baseline DCTR model. However, UBM clearly shows a better performance for log-likelihood as well as perplexity compared to DBN. We thus find that the learnt model using UBM is able to predict clicks well and should be preferred over the others. However, we continue the evaluation based on a comparison with human judgments.

4.2.3 Relevance Prediction Evaluation. In addition to the previous evaluation on the held-out test set, we leverage existing term relevance judgments from the literature for evaluation purposes. By evaluating the models' performance with regard to their relevance predictions we ensure that inferred relevance labels are accurate and we gain more confidence in the ground truth. In particular, we rely on the expert judgments from CBRBench [7]. CBRBench is composed of ten queries with 34 to 137 term judgments per query. We are able to compare six queries from CBRBench that have overlapping judgments with the relevance predictions of the learnt click models based on the results shown in the LOV logs (i.e., *location*, *address*, *organization*, *event*, *music*, and *person*). We use the Pearson Correlation Coefficient r to evaluate the similarity of the total order of terms based on the predicted satisfaction probability with the total order of terms based on the CBRBench judgments.

The results of the relevance prediction evaluation are also summarized in Table 2, showing the correlation r for each model with respect to the CBRBench judgments. On the one hand, this evaluation step confirms the previous results that the UBM should be preferred over DCTR and DBN. Furthermore, UBM demonstrates a a strong correlation with CBRBench's relevance judgments (0.613), meaning that the relevance predictions of the UBM model learnt from the LOV logs are close to the expert judgments from CBRBench. We explain the better performance of UBM compared to DCTR and DBN due to its consideration of previous clicks and their position in the same session, based on the experimental findings and intuition presented in [10], where UBM also performs best. Thus, we consider the relevance labels inferred from the UBM model in the LOVBench dataset.

5 LOVBENCH DATASET

In this section, we introduce the considered ranking features and the sampled ground truth for the proposed ontology ranking benchmark dataset. We present the considered features in LOVBench with a joint conceptualization in Section 5.1, introduce our term sampling strategy in Section 5.2, and finally summarize the LOVBench dataset in Section 5.3.

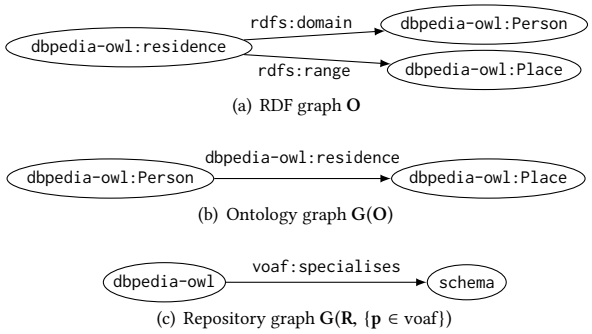
5.1 Feature Description

We first introduce the notation used to describe the features in a joint framework. Subsequently, we provide an overview and more details about the considered ranking features.

5.1.1 Notation. Let O be an ontology in a repository R. An ontology O is defined by a set of RDF triples (`subject, predicate, object`) which is referred to as RDF graph. A term t in an ontology can be of type class c[4] or property p[5]. A user query is denoted by Q, with ith word denoted as q_i. The set of ontology terms that matches the query is denoted by $\sigma_{\mathcal{T}}(Q, O)$, where $\mathcal{T}$ corresponds to the type of matched terms, i.e., $\mathcal{T} \in \{t, c, p\}$ (corresponding to all terms, only classes, or only properties, respectively). In addition to the RDF graph O (illustrated in Figure 3(a)), two more graph structures can be derived and used for ranking, namely the ontology graph (Figure 3(b)) and the repository graph (Figure 3(c)). These graph structures allow the application of conventional graph scoring algorithms.

Ontology graph. The ontology graph of O represents classes and properties of the ontology. Unlike the RDF graph, properties are modeled as edges instead of nodes. The ontology graph is denoted by $G(O) = (C, \mathcal{E}_{c_i,c_j})$, where C is a set of nodes representing all classes ($c \in O$) and $\mathcal{E}$ is a set of directed edges. These are derived based on the ontologies' properties ($p \in O$) and the semantics of `rdfs:domain` and `rdfs:range`, i.e., $\mathcal{E}_{c_i,c_j} = \{(c_i, c_j) \in O : (p,$ `rdfs:domain`,

[4]Considered class types: `rdfs:Class`, `owl:Class`.
[5]Considered property types: `rdf:Property`, `rdfs:Property`, `owl:ObjectProperty`, `owl:DatatypeProperty`, `owl:AnnotationProperty`, `owl:OntologyProperty`.



(a) RDF graph **O**

(b) Ontology graph **G(O)**

(c) Repository graph **G(R, {p ∈ voaf})**

Figure 3: Illustration of graph definitions used for ranking.

c_i) & (p, rdfs:range, c_j)}. A more detailed description of the mapping from O to $G(O)$ can be found in [43].

Repository graph. The repository graph expresses the relationships among ontologies in the repository. It is denoted by $G(R, \mathcal{P}) = (\mathcal{O}, \mathcal{E}_{O_i,O_j})$, where $\mathcal{O}$ is a set of nodes corresponding to the ontologies contained in the repository ($O \in R$) and $\mathcal{E}_{O_i,O_j}$ is a set of edges representing all considered properties $p \in \mathcal{P}$ that exist between ontologies, i.e. $\mathcal{E}_{O_i,O_j} = \{(O_i, O_j) \in R : (O_i,\ p \in \mathcal{P},\ O_j)\}$.

5.1.2 Feature Selection. Our strategy for the selection of ranking features for LOVBench is guided by related literature surveys [14, 22, 45], and respectively discovered features are selected based on the following applicability constraints.

- The feature must rely on a keyword-based query format and rank ontologies or terms. We choose to consider both, term and ontology ranking features in LOVBench because the consideration of the overall quality of an ontology in which a term is defined is often an important factor for reuse [37], and thus, ontology features are often applied for term ranking, such as in CBRBench [7].
- The feature must not depend on metadata information that cannot be derived from the ontology collection itself. Such features are often considered in ontology ranking by formulating scores involving user feedback such as ratings and clicks [26], ontology and term usage in LOD [38], etc. While these features can be very useful for ranking, applying these features to different ontology collections can be problematic because the respective metadata is often not available or difficult to obtain for other ontology collections [21]. Since ranking models evaluated by LOVBench should be applicable to any ontology collection, we choose to exclude them.
- The feature must form a significant distinction to already considered features.
- Priority is given to features for which the complete ranking configuration as proposed in the literature can be replicated.

Other considerations for a fair comparison of ranking models include the constraints that define whether a term matches a query or not. We choose to harmonize the query match for all features independent from the respective feature's original approach. This gives a more accurate query match for all features by considering the meta vocabularies used in the LOV collection. Lastly, some ranking features depend on hyperparameters that need to be set by experts. In LOVBench, we stick to the hyperparameters as suggested in the original source of the feature. However, our large ground truth for the application of LTR further allows us to decompose some of these features (e.g., in case of weighted sums) and implicitly learn the parameters from the data instead.

5.1.3 Feature Description. An overview of LOVBench's ranking features, based on the previously introduced notation, is presented in Table 3. The subscript after the feature name indicates whether the feature assigns scores to terms (t), ontologies as a whole (O), or only to the query (Q). We would like to note that Q-features cannot be used stand-alone for ranking and that O-features assign the same score for all terms from the same ontology. The feature category indicates the feature's parameters for scoring. We further organize the features in four groups (*query match*, *repository graph analysis*, *ontology graph analysis*, and *RDF graph analysis*), for which we provide more details in the following. As motivated previously, it is possible to use LOVBench for comparisons of rankings using metadata-based features (as an additional group of features), however, it is considered out of scope in this paper.

Query match (Feature 1-7). Query match features assess how well the words in the query match the words that describe a term in the ontology. The boolean match (Feature 1), e.g., simply states whether a term is contained in the query match or not and the text relevancy (Feature 4) measures how many words in the query match a term. The original matching features followed in LOV (Feature 2 and 3) are based on a standard BM25 matching score [34]. Feature 2 further assigns weights depending on which property of a term matches the query and Feature 3 is based on properties that describe the ontology [41]. We propose two more features in addition to those used in the state-of-the-art. First, we propose a variation of the class match (Feature 5) for properties (Feature 6). These features are based on different weights for exact and partial matches of query words. Albeit not directly related to matching, we further consider a simple feature that counts the number of words in the query (Feature 7). The following details were considered when extracting query match features for LOVBench.

- A term matches the query (i.e., $t \in \sigma_t$) if at least one word $q_i \in Q$ matches the domain of at least one of the following properties of the term: rdfs:label, dce:title, dcterms:title, skos:prefLabel, rdfs:comment, rdfs:description, dce:description, dcterms:description, skos:altLabel, or the local name of the term's URI. These are the same properties that were considered in the original LOV term property boost [41].
- The exact and partial matches (φ^{exact} and φ^{partial}) of the class and property match (Feature 5 and 6) only consider matches in rdfs:label. The hyperparameters are set to $\alpha = 0.6$ and $\beta = 0.4$ [2].

We would like to note that features that use terms or ontologies contained in the query match as input for scoring are not classified as query match features, but assigned to one of the following groups to which the scoring algorithm relates.

Repository graph analysis (Feature 8-10). These features determine the importance of ontologies in the corpus based on the repository

graph $G(R, \mathcal{P})$. PageRank has been the most commonly adapted ranking algorithm for this purpose, especially in the domain of ontology search engines [22]. The repository graph is usually constructed based on `owl:imports` statements (Feature 8), however, it has been pointed out that explicit `owl:imports` statements are often missing, and considering implicit imports derived from term URIs appearing in the ontology (Feature 9) showed better performance [8]. We further propose to build a repository graph using ontologies' properties based on the vocabulary of a friend[6] (`voaf`), which in LOV are automatically derived from the ontology collection [41], and to consider this graph as input for PageRank (Feature 10).

- PageRank *pr* for repository graphs $G(R, \mathcal{P})$ is computed as follows [7].

$$pr(O, G(R, \mathcal{P})) = \frac{1-d}{|R|} + \sum_{O_j \in (O_i,\ p \in \mathcal{P},\ O_j)} \frac{pr(O_j, G(R, \mathcal{P}))}{|(O_j,\ p \in \mathcal{P},\ O_i)|} \tag{4}$$

  where d corresponds to the damping factor (set to $d = 0.85$). Since PageRank computes very small scores, we multiple these with 10^5.

Ontology graph analysis (Feature 11-16). These features assign scores to terms and ontologies based on the ontology graph $G(O)$. PageRank has also been adapted for this purpose. In particular, [8] proposes a Reversed PageRank approach to identify hubs in an ontology (Feature 11-13). However, this approach is limited to classes since PageRank only assigns scores to nodes and not to edges (i.e., these scores ignore properties). Other considered graph scoring algorithms include *betweenness* (Feature 15), for which we propose a variation for scoring on terms (Feature 14), as well as the semantic similarity measure (Feature 16). The following details are considered for creating LOVBench.

- The betweenness measure scores classes based on the number of shortest paths passing through it [2].

$$betweenness(c, G(O)) = \sum_{c_i \neq c_j \neq c \in O} \frac{\lambda(c_i, c_j(c), G(O))}{\lambda(c_i, c_j, G(O))} \tag{5}$$

  where $\lambda(c_i, c_j, G(O))$ corresponds to the total number of shortest paths from c_i to c_j, and $\lambda(c_i, c_j(c), G(O))$ defines the number of shortest paths passing through c.
- Semantic similarity is a measure based on the shortest path between two classes to capture how close these concepts are laid out in the ontology [2].

$$ssm(c_i, c_j, G(O)) = \begin{cases} \frac{1}{|\min(\lambda(c_i, c_j, G(O)))|}, & \text{if } i \neq j \\ 1, & \text{if } i = j \end{cases} \tag{6}$$

RDF graph analysis (Feature 17-33). Features based on the RDF graph O extract statistics about terms in an ontology which, e.g., can give an indication about the importance of terms in the repository and to which detail a term is defined. TF-IDF-based measures (Feature 17-25) differ in ontology ranking compared to conventional text retrieval since scoring is adapted to compute the importance of term URIs in the repository [7]. This implies that TF-IDF-like term features (Feature 17-19) are query-independent. TF-IDF ontology features (Feature 20-22), however, are computed based on all terms in an ontology that match the query [7], and thus remain query-dependent. Since the intuition of TF-IDF does not apply to ontology ranking (IDF assigns a low score when a term appears in many ontologies, even though it is a desirable trait) we decompose the measures into separate features (Feature 17-18 and 20-21). In addition to TF-IDF-based features, simple statistics of the RDF graph are also considered for the ranking of terms and ontologies (Feature 26-33). We propose variations of features from the literature which are adapted for ranking of terms instead of ontologies (Feature 26-30) as well as for the consideration of properties instead

[6] https://lov.linkeddata.es/vocommons/voaf/v2.3/

Table 3: LOVBench ranking features.

ID	Feature	Cat.	Description	Ref.
Query match				
1	Boolean match$_t$	Q, t	1, if $t \in \sigma_t$; 0, otherwise	[7]
2	Match-boost$_t$	Q, t, R	$bm25(Q, t, R) + boost(\sigma_t)$	[41]
3	Match-descr.$_O$	Q, O, R	$bm25(Q, O, R)$	[41]
4	Text relevancy$_t$	Q, t	$\lvert\{q_i \in Q : \sigma_t \neq \emptyset\}\rvert$	[8]
5	Class match$_O$	Q, O	$\sum_{q_i \in Q} \alpha * \lvert\varphi_c^{\text{exact}}\rvert + \beta * \lvert\varphi_c^{\text{partial}}\rvert$	[2]
6	Prop. match$_O$	Q, O	$\sum_{q_i \in Q} \alpha * \lvert\varphi_p^{\text{exact}}\rvert + \beta * \lvert\varphi_p^{\text{partial}}\rvert$	
7	Query length$_Q$	Q	$\lvert\{q_i \in Q\}\rvert$	
Repository graph analysis				
8	PR-imports$_O$	O, R	$pr(O, G(R, \{\texttt{owl:imports}\}))$	[7]
9	PR-implicit$_O$	O, R	$pr(O, G(R, \{implicit\ imports\}))$	[7]
10	PR-voaf$_O$	O, R	$pr(O, G(R, \{p \in \texttt{voaf}\}))$	
Ontology graph analysis				
11	Hub$_t$	c, O	$pr(c, G(O_{\text{reversed}}))$	[8]
12	Max hub$_O$	O	$\max(\{\text{Hub}_t : c \in O\})$	[8]
13	Min hub$_O$	O	$\min(\{\text{Hub}_t : c \in O\})$	[8]
14	Betweenness$_t$	t	$betweenness(t, G(O))$	[2]*
15	Betweenness$_O$	Q, O	$\frac{1}{\lvert\sigma_c\rvert} \sum_{c \in \sigma_c} \text{Betweenness}_t$	[2]
16	Semantic sim.$_O$	Q, O	$\frac{1}{\lvert(c_i, c_j)\rvert} \sum_{c_i, c_j \in \sigma_c} ssm(c_i, c_j, G(O))$	[2]
RDF graph analysis				
17	TF$_t$	t, O	$tf(t, O)$	[7]*,**
18	IDF$_t$	t, R	$idf(t, R)$	[7]*,**
19	TF-IDF$_t$	t, O	$\text{TF}_t * \text{IDF}_t$	[7]*
20	TF$_O$	Q, O	$\sum_{t \in \sigma_t} \text{TF}_t$	[7]**
21	IDF$_O$	Q, R	$\sum_{t \in \sigma_t} \text{IDF}_t$	[7]**
22	TF-IDF$_O$	Q, O, R	$\sum_{t \in \sigma_t} \text{TF-IDF}_t$	[7]
23	BM25$_t$	t, O, R	$bm25(t, O, R)$	[7]*
24	BM25$_O$	Q, O, R	$\sum_{t \in \sigma_t} \text{BM25}_t$	[7]
25	VSM$_O$	Q, O, R	$vsm(Q, O, R)$	[7]
26	Subclasses$_t$	c	$\lvert subclasses(c)\rvert$	[2]*,**
27	Superclasses$_t$	c	$\lvert superclasses(c)\rvert$	[2]*,**
28	Relations$_t$	c	$\lvert relations(c)\rvert$	[2]*,**
29	Siblings$_t$	c	$\lvert siblings(c)\rvert$	[2]*,**
30	Density$_t$	c	$w_1 * \text{Subclasses}_t + w_2 * \text{Superclasses}_t + w_3 * \text{Relations}_t + w_4 * \text{Siblings}_t$	[2]*
31	Density$_O$	Q, O	$\frac{1}{\lvert\sigma_c\rvert} \sum_{c \in \sigma_c} \text{Density}_t$	[2]
32	Subproperties$_t$	p	$\lvert subproperties(p)\rvert$	
33	Superprop.$_t$	p	$\lvert superproperties(p)\rvert$	

* Adapted for terms, originally only proposed for ontology ranking.
** Originally not considered as individual feature.

of classes (Feature 32-33). The following details were considered for LOVBench.

- Term frequency $tf(t, O)$ and inverse document frequency $idf(t, R)$ are computed as follows [7].

$$tf(t, O) = 0.5 + \frac{0.5 * f(t, O)}{\max(\{f(t_i, O) : t_i \in O\})}$$
$$idf(t, R) = \log\left(\frac{|R|}{|\{O : t \in O, O \in R\}|}\right) \quad (7)$$

  where $f(t, O)$ is the frequency of a term t in ontology O.
- The adapted notion of BM25 for ranking of terms t is defined as follows [7].

$$bm25(t, O, R) = \text{IDF}_t(t, R) * \frac{\text{TF}_t(t, O) * k + 1}{\text{TF}_t(t, O) + k * \left(1 - b + b * \frac{|O|}{avgos(R)}\right)} \quad (8)$$

  where k and b are hyperparameters ($k = 2.0$, $b = 0.75$), the ontology size $|O|$ is computed by its number of triples, and $avgos(R)$ is the average ontology size in the repository.
- The Vector Space Model (VSM) [36] feature (Feature 25), adapted to ontology ranking, measures similarity of query and ontology using *tf* and *idf* to compute the weights of query words q_i in the query and ontology [7].

$$vsm(Q, O, R) = \frac{\sum_{q_i \in Q}\left(\sum_{t \in \sigma_t(q_i, O)} (\text{TF-IDF}_t(q_i, t)) * tf\text{-}idf_Q(q_i, Q, R)\right)}{\sqrt{\sum_{t_i \in O} (\text{TF-IDF}_t(t_i, O))^2} * \sqrt{\sum_{q_i \in Q} (tf\text{-}idf_Q(q_i, Q, R))^2}} \quad (9)$$

  where $tf\text{-}idf_Q$ of a query word q_i is defined as follows.

$$tf\text{-}idf_Q(q_i, Q, R) = \frac{f(q_i, Q)}{\max(\{f(q_j, Q) : q_j \in Q\})} * \log\left(\frac{|R|}{|\{O : t \in O, t \in \sigma_t(Q, O)\}|}\right) \quad (10)$$

- The density (Feature 30) depends on a weighted sum, for which the original weights were used: $w_1 = 1.0$, $w_2 = 0.25$, $w_3 = 0.5$, and $w_4 = 0.5$ [2].

In summary, we presented the selected state-of-the-art ranking features and proposed variations and features that are motivated by our observations made from the LOV search logs. In the next step we describe our strategy to select samples for the ground truth.

5.2 Term Sampling

When building a ground truth, it is usually impractical to judge and extract features for all terms in the repository [24]. In the context of LTR, the chosen strategy of sampling some documents for each query for learning will impact the observed effectiveness of the learnt ranking model, referred to as *sample selection bias* [25, 29]. Given the discussions of document sampling's impact [25], our goal is to ensure that the LOVBench dataset contains a similar distribution of relevance labels and average sample size compared to well-known state-of-the-art datasets for conventional ad-hoc Web retrieval tasks. We follow a similar strategy to the one presented in LETOR [31], by (i) considering all query-term pairs without judgment as non-relevant and (ii) selecting samples from an existing ranking, i.e., the term search results of LOV. For each query of the LOV search logs, the relevance of the first ten terms is inferred from the best-performing click model using UBM (see Section 4.2), and an additional amount of non-relevant terms is randomly sampled from the remaining query match.

Table 4: Breakdown of relevance labels in LOVBench, well-known ad-hoc Web retrieval datasets, and CBRBench.

Label	LOVBench	TREC Web	ClueWeb12	CBRBench
Total	73,950 + 110,274	28,906	5,392	819
Very high (4)	0.34%	0.14%	0.07%	4.64%
High (3)	1.63%	1.42%	1.15%	10.01%
Low (2)	7.26%	8.77%	5.47%	19.90%
Very Low (1)	24.10%	23.64%	20.83%	25.76%
Non (0)	66.59%	63.31%	68.64%	39.68%
Junk (−2)	0.06%	2.73%	3.84%	0.00%

As a popular convention, we map the relevance inferred from the click model, which is measured in terms of satisfaction probability ($0 \leq Rel_{Q,t} \leq 1$), to relevance labels $l_{Q,t}$ on a scale from 0-4, and further consider a junk grade (−2). We denote the set of labelled data for query Q and term t by $\mathcal{L} = \{(Q, t, l_{Q,t})\}$, the standard input for supervised LTR settings, and refer to $\mathcal{L}$ with all relevance predictions as *ground truth*. As shown in Table 4, this approach results in a dataset with a similar distribution of relevance judgments compared to established LTR datasets, such as TREC Web 2013 & 2014 [11, 12] and a selection of ClueWeb12[7] entries as presented in [40]. On the other hand, it reveals a different distribution for CBRBench, which is a potential explanation for the differences in our inferred relevance judgments and those made by experts from CBRBench. The authors of CBRBench only consider a term as relevant with a score of 2 or higher [7], while we follow the common practice to consider a score of 1 or higher as relevant. Finally, LOVBench is composed of 73,950 judgments inferred through the user click model and 110,274 randomly sampled non-relevant judgments, with an average sample size of ~ 26 judgments per query.

5.3 Final LOVBench Dataset

In summary, the LOVBench dataset contains: (i) extracted ranking features (Section 5.1) and (ii) inferred relevance judgments (Section 5.2) for 184,224 query-term pairs. We provide a single csv-file containing the query-term pairs and their relevance judgments which can be used as ground truth for evaluation purposes. Moreover, the full dataset (including extracted features) is provided in a format for LTR experiments. It is randomly split by query into five equal-sized partitions, which in turn are used to derive five folds with three partitions for training and the remaining two for validation and testing.

6 EMPIRICAL EVALUATION

In this section, we apply LOVBench to evaluate several ranking configurations using LTR. We first describe our experiment setup and then present our experimental results.

6.1 Experimental Setup

The experiments are based on the standard framework for LTR evaluation [24]. In the following, we describe the considered feature

[7] https://lemurproject.org/clueweb12/

Table 5: Overview of feature configurations. The feature IDs correspond to those presented in Table 3.

Category	Name	Feature configuration	Count
Baseline	Baseline	Lucene search (`rdfs:label`)	1
Literature	DWRank [8]	4, 9, 11-13	5
	AKTiveRank [2]	5, 15-16, 31	4
	CBRBench [7]	1, 5, 9, 15-16, 22, 24-25, 31	9
Variations	LOV-based	2-3	2
	LOVBench full	1-33	33
	LOVBench light	2, 10, 14, 17-19, 26-29, 32-33	12

configurations, baselines and evaluation metrics. We rely on the previously introduced LOVBench dataset for the evaluation.

Configurations. We experiment with feature configurations that are associated with three different groups: baseline, literature, and variations. An overview of all configurations subject to the experiments is given in Table 5. The purpose of our baseline (a standard Lucene search on terms' `rdfs:label` properties) is to provide the performance of a straight-forward search approach as a reference point. We then compare several ranking models as they have been proposed in the literature with the baseline using the feature sets from AKTiveRank [2], DWRank [8], and CBRBench [7]. Lastly, we propose variations that are designed to meet the requirements observed in the LOV search logs. We compare three models to the baseline and literature: first, a simple model only using current LOV features as a reference point (LOV-based), second, a model using all features of LOVBench (LOVBench full), and third, a model with features that are chosen based on low computational complexity, high informativeness and meeting the LOV requirements (LOVBench light). In total, we evaluate ranking performance of seven different configurations with up to 33 different features.

Learning to rank. Each model configuration is trained with well-known LTR techniques. We experiment with both, a pairwise approach (RankNet [6]) as well as a listwise approach (AdaRank [44])[8]. Evaluation and comparison to the baseline is based on five-fold cross validation.

Metrics. We rely on standard evaluation metrics for information retrieval. We measure the retrieval performance in terms of multi-valued relevance based on the Normalized Discounted Cumulative Gain (NDCG) [19]. Motivated by the discussion in Section 4.1, we report the NDCG for the first 3, 5, and 10 ranked terms. We further report the Mean Average Precision (MAP) [3] as a consideration of ranking models' effectiveness, which, however, only considers binary relevance.

6.2 Experimental Results

In this section, we present the experimental results and compare each configuration to the baseline.

6.2.1 Originals. In Table 6, we compare the performance of several feature configurations using two LTR algorithms with the baseline. The first row shows the baseline performance based on a Lucene search. The first group of configurations (DWRank, AKTiveRank, CBRBench) for both LTR algorithms shows the performance of original models from the literature.

[8]Implementation taken from https://sourceforge.net/p/lemur/wiki/RankLib

Table 6: Results for ranking models with considered feature configurations using LOVBench. Best results for each algorithm and metric are highlighted in bold. All configurations significantly improve the performance in all metrics compared to the baseline (p-value ≤ 0.05).

LTR	Configuration	NDCG@3	NDCG@5	NDCG@10	MAP
-	Baseline	0.261	0.299	0.358	0.433
RankNet	DWRank	0.497	0.508	0.536	0.572
	AKTiveRank	0.459	0.469	0.499	0.563
	CBRBench	0.480	0.484	0.512	0.564
	LOV-based	0.646	0.688	0.724	0.717
	LOVBench full	**0.841**	**0.868**	**0.883**	**0.845**
	LOVBench light	0.762	0.781	0.803	0.806
AdaRank	DWRank	0.372	0.382	0.404	0.481
	AKTiveRank	0.454	0.454	0.478	0.548
	CBRBench	0.378	0.379	0.411	0.492
	LOV-based	0.638	0.679	0.716	0.710
	LOVBench full	**0.881**	**0.902**	**0.911**	**0.918**
	LOVBench light	0.780	0.825	0.860	0.871

When comparing the results of the original configuration with the baseline, we observe that all configurations manage to significantly improve the baseline performance in all metrics. Compared with each other, all models show similar performance, albeit DWRank performs slightly better when using RankNet and AKTiveRank performs slightly better when using AdaRank. Another interesting observation is that the additional features considered in CBRBench, which also includes the features proposed in AKTiveRank, barely improve (RankNet) or even harm (AdaRank) the ranking performance. This is surprising, given that the comparison in CBRBench [7] based on single features shows some of these features to perform well. This indicates that ranking models should always be evaluated with all features and weights taken into account, since individually poorly performing features can still perform well when combined using LTR.

6.2.2 Variations. The second group for each algorithm (LOV-based, LOVBench full, LOVBench light) in Table 6 shows the results of the adapted models.

First, we can observe that all three configurations that use the original LOV term matching feature (Feature 2) significantly improve the performance of the baseline as well as the models from the literature. However, we need to keep in mind that the sampled ground truth is biased on the existing ranking of LOV, which includes Feature 2 [41]. Moreover, when comparing the LOVBench full model with the simple LOV-based configuration, we observe that the additional features in LOVBench significantly improve the performance. Given that real-time computational requirements can be a major concern, we are also interested in increasing the ranking performance with a reduced number of features compared to the full configuration. As the results of the LOVBench light configuration show, the smaller feature set is also able to significantly

improve ranking performance compared to the LOV-based model, albeit not as much as the full model. All these observations hold true for both learning algorithms, RankNet and AdaRank.

7 DISCUSSION

In this section, we discuss the practical implications of the results and state the limitations of our approach.

7.1 Practical Recommendations

One of our key findings is that ranking features should be designed with regard to actual user behavior of the targeted domain. The performance results of the considered literature models can be explained with the mismatch of the assumptions for these ranking models and the user behavior we observe in LOV (cf. Section 4.1). First, the original AKTiveRank configuration does not include any term features, meaning that if the ranking contains multiple terms from the same ontology, these receive the same score. Second, in DWRank, three out of five features (Feature 11-13) relate to the hub score, which only scores classes, meaning that this configuration lacks the ability to rank properties, which we have found in LOV to be equally important as ranking of classes. Furthermore, the text relevancy feature that is also part of this configuration (Feature 4) assumes multiple words in the query, which in LOV is often not the case. Third, CBRBench is, similar to AKTiveRank, mostly composed of ontology features, and further of many complex features with predefined hyperparameters, not allowing the LTR algorithms to learn these from the data. As shown by our experiments, adapting the ranking features to these requirements significantly improved the performance. Our results also demonstrate that increasing the number of ranking features, given a large enough dataset like LOVBench, allows to increase ranking performance. However, in practice it is often necessary to form a reasonable trade-off between effectiveness and computation cost. This especially concerns query-dependent features that cannot be pre-computed and have to be extracted at run-time.

As previously mentioned, the experimental results need to be interpreted in the context of the biases imposed by the current LOV ranking from which the relevance labels for the ground truth are derived. The LOV search relies on a Lucene-based match with property boost and a popularity score measured in term of usages in LOD datasets [41]. Thus, it is reasonable that relevance features and qualitative importance features such as PageRank perform better on our dataset than graph-structural criteria such as density.

In conclusion, the following considerations should be taken into account when designing ontology ranking models, which were also followed for the LOVBench light configuration:

(1) mixing and prioritizing of ranking granularities depending on the platform (ontologies/terms and classes/properties),
(2) diverse coverage of all feature categories (query match, repository graph, ontology graph and RDF graph analysis),
(3) preferring decomposed features when possible,
(4) being aware of the user behavior of targeted platform,
(5) the number of query match features should be minimized.

Lastly, LOVBench can be used not only to evaluate, but also to train ranking models and integrate these for re-ranking of ontology terms in other applications with different ontology collections. In this case, we would like to highlight that the query match constraint should be adjusted to the meta-vocabularies considered in the respective ontology collection and all features need to be extractable for the LOV ontology collection.

7.2 Limitations

The applicability of LOVBench for the evaluation of ontology rankings is limited as follows. First, only ranking models that use a keyword-based query format to rank ontology terms can be evaluated. Albeit this is the most popular approach in existing search interfaces [22], this means that LOVBench cannot be considered as a general benchmark in the broader context of ontology reuse. E.g., in some tools other search interfaces and rank granularities are used (such as the ranking of the best combination of ontologies [27]). However, the current lack of search logs from respective platforms hinder the application of our approach to other search interfaces. We believe that the adoption of our approach to other ontology search platforms could ultimately result in a landscape of benchmarks for all variations. Second, applying LOVBench for the evaluation of ranking models that include features that rely on metadata information such as user ratings can be difficult, as this metadata needs to be collected for the LOV collection. However, using metadata-based features in the ranking model in general can be problematic, depending on the way the metadata is collected. Instead, sufficient and well-designed ontology and term features directly derived from the ontology collection might be able to substitute such features [21].

8 CONCLUSION

In this paper, we address the problem of ontology ranking evaluations and comparisons in LTR settings. We analyze logged user interactions of a real-world ontology search platform (LOV) and use this implicit user feedback infer relevance judgments for query-term pairs. Our evaluation shows that inferred relevance judgments are close to those made by human experts. We create the LOVBench dataset that comprises 184,224 relevance judgments for 7,395 queries and considers 33 different ontology ranking features. We then evaluate and compare three state-of-the-art ranking models from the literature based on this ground truth. We explain the results in the context of the observed user behavior in LOV and propose variations and configurations that outperform the baseline. The published dataset can be applied for future ontology ranking evaluations and LTR experiments.

In future work, the LOVBench dataset can be updated in regular time intervals to capture the evolution of ontologies and their relevance in the Semantic Web. The continuous nature of LOVBench further allows the benchmark to grow in terms of queries and judgments over time. Moreover, future work includes online evaluation of learnt models in LOV to evaluate their effectiveness with actual user feedback, e.g., based on users' average click position.

RESOURCES

The LOVBench dataset, the code for the extraction of selected features, the code to run the experiments presented in this paper, and the cleaned search logs collected from LOV are available online[9].

[9]LOVBench resources: https://github.com/nut-hatch/LOVBench

REFERENCES

[1] Aman Agarwal, Xuanhui Wang, Cheng Li, Michael Bendersky, and Marc Najork. 2019. Addressing Trust Bias for Unbiased Learning-to-Rank. In *The World Wide Web Conference*. ACM, 4–14. https://doi.org/10.1145/3308558.3313697

[2] Harith Alani, Christopher Brewster, and Nigel Shadbolt. 2006. Ranking Ontologies with AKTiveRank. In *International Semantic Web Conference*. Springer, 1–15. https://doi.org/10.1007/11926078_1

[3] Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. 1999. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc.

[4] Tim Berners-Lee, James Hendler, Ora Lassila, et al. 2001. The Semantic Web. *Scientific American* 284, 5 (2001), 28–37.

[5] Christian Bizer, Tom Heath, and Tim Berners-Lee. 2011. Linked Data: The Story So Far. In *Semantic Services, Interoperability and Web Applications: Emerging Concepts*. IGI Global, 205–227. https://doi.org/10.4018/978-1-60960-593-3.ch008

[6] Christopher Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Gregory N Hullender. 2005. Learning to Rank using Gradient Descent. In *Proceedings of the 22nd International Conference on Machine Learning*. ACM, 89–96. https://doi.org/10.1145/1102351.1102363

[7] Anila Sahar Butt, Armin Haller, and Lexing Xie. 2014. Ontology Search: An Empirical Evaluation. In *International Semantic Web Conference*. Springer, 130–147. https://doi.org/10.1007/978-3-319-11915-1_9

[8] Anila Sahar Butt, Armin Haller, and Lexing Xie. 2016. DWRank: Learning Concept Ranking for Ontology Search. *Semantic Web* 7, 4 (2016), 447–461. https://doi.org/10.3233/SW-150185

[9] Olivier Chapelle and Ya Zhang. 2009. A Dynamic Bayesian Network Click Model for Web Search Ranking. In *Proceedings of the 18th International Conference on World Wide Web*. ACM, 1–10. https://doi.org/10.1145/1526709.1526711

[10] Aleksandr Chuklin, Ilya Markov, and Maarten de Rijke. 2015. Click Models for Web Search. *Synthesis Lectures on Information Concepts, Retrieval, and Services* 7, 3 (2015), 1–115. https://doi.org/10.2200/S00654ED1V01Y201507ICR043

[11] Kevyn Collins-Thompson, Paul Bennett, Fernando Diaz, Charlie Clarke, and Ellen M Voorhees. 2013. TREC 2013 Web Track Overview. *TREC* (2013).

[12] Kevyn Collins-Thompson, Craig Macdonald, Paul Bennett, Fernando Diaz, and Ellen M Voorhees. 2015. TREC 2014 Web Track Overview. *TREC* (2015).

[13] Nick Craswell, Onno Zoeter, Michael Taylor, and Bill Ramsey. 2008. An Experimental Comparison of Click Position-Bias Models. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*. ACM, 87–94. https://doi.org/10.1145/1341531.1341545

[14] Mathieu d'Aquin and Natalya F Noy. 2012. Where to publish and find ontologies? A survey of ontology libraries. *Journal of Web Semantics* 11 (2012), 96–111. https://doi.org/10.1016/j.websem.2011.08.005

[15] Li Ding, Tim Finin, Anupam Joshi, Rong Pan, R Scott Cost, Yun Peng, Pavan Reddivari, Vishal Doshi, and Joel Sachs. 2004. Swoogle: A Search and Metadata Engine for the Semantic Web. In *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*. ACM, 652–659. https://doi.org/10.1145/1031171.1031289

[16] Li Ding, Rong Pan, Tim Finin, Anupam Joshi, Yun Peng, and Pranam Kolari. 2005. Finding and Ranking Knowledge on the Semantic Web. In *International Semantic Web Conference*. Springer, 156–170. https://doi.org/10.1007/11574620_14

[17] Georges E Dupret and Benjamin Piwowarski. 2008. A User Browsing Model to Predict Search Engine Click Data from Past Observations. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 331–338. https://doi.org/10.1145/1390334.1390392

[18] Jens Hartmann, Raúl Palma, and Asunción Gómez-Pérez. 2009. Ontology Repositories. In *Handbook on Ontologies*. Springer, 551–571. https://doi.org/10.1007/978-3-540-92673-3_25

[19] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated Gain-Based Evaluation of IR Techniques. *ACM Transactions on Information Systems (TOIS)* 20, 4 (2002), 422–446. https://doi.org/10.1145/582415.582418

[20] Megan Katsumi and Michael Grüninger. 2017. Choosing ontologies for reuse. *Applied Ontology* 12, 3-4 (2017), 195–221. https://doi.org/10.3233/AO-160171

[21] Niklas Kolbe, Sylvain Kubler, and Yves Le Traon. 2019. Popularity-driven Ontology Ranking using Qualitative Features. In *International Semantic Web Conference*. Springer. https://doi.org/10.1007/978-3-030-30793-6_19

[22] Niklas Kolbe, Sylvain Kubler, Jérémy Robert, Yves Le Traon, and Arkady Zaslavsky. 2019. Linked Vocabulary Recommendation Tools for Internet of Things: A Survey. *ACM Computing Surveys (CSUR)* 51, 6 (2019), 127. https://doi.org/10.1145/3284316

[23] Yuangui Lei, Victoria Uren, and Enrico Motta. 2006. Semsearch: A Search Engine for the Semantic Web. In *International Conference on Knowledge Engineering and Knowledge Management*. Springer, 238–245. https://doi.org/10.1007/11891451_22

[24] Tie-Yan Liu. 2009. Learning to Rank for Information Retrieval. *Foundations and Trends in Information Retrieval* 3, 3 (2009), 225–331. https://doi.org/10.1007/978-3-642-14267-3

[25] Craig Macdonald, Rodrygo LT Santos, and Iadh Ounis. 2013. The whens and hows of learning to rank for web search. *Information Retrieval* 16, 5 (2013), 584–628. https://doi.org/10.1007/s10791-012-9209-9

[26] Marcos Martínez-Romero, Clement Jonquet, Martin J O'Connor, John Graybeal, Alejandro Pazos, and Mark A Musen. 2017. NCBO Ontology Recommender 2.0: an enhanced approach for biomedical ontology recommendation. *Journal of Biomedical Semantics* 8, 1 (2017), 21. https://doi.org/10.1186/s13326-017-0128-y

[27] Marcos Martínez-Romero, José M Vázquez-Naya, Javier Pereira, and Alejandro Pazos. 2014. BiOSS: A system for biomedical ontology selection. *Computer Methods and Programs in Biomedicine* 114, 1 (2014), 125–140. https://doi.org/10.1016/j.cmpb.2014.01.020

[28] Michael McCandless, Erik Hatcher, and Otis Gospodnetic. 2010. *Lucene in action: covers Apache Lucene 3.0*. Manning Publications Co.

[29] Tom Minka and Stephen Robertson. 2008. Selection bias in the LETOR datasets. In *SIGIR Workshop on Learning to Rank for Information Retrieval*. Citeseer, 48–51. https://www.microsoft.com/en-us/research/publication/selection-bias-letor-datasets/

[30] Natalya F Noy, Paul R Alexander, Rave Harpaz, Patricia L Whetzel, Raymond W Fergerson, and Mark A Musen. 2013. Getting Lucky in Ontology Search: A Data-Driven Evaluation Framework for Ontology Ranking. In *International Semantic Web Conference*. Springer, 444–459. https://doi.org/10.1007/978-3-642-41335-3_28

[31] Tao Qin, Tie-Yan Liu, Jun Xu, and Hang Li. 2010. LETOR: A benchmark collection for research on learning to rank for information retrieval. *Information Retrieval* 13, 4 (2010), 346–374. https://doi.org/10.1007/s10791-009-9123-y

[32] Yuzhong Qu and Gong Cheng. 2011. Falcons Concept Search: A Practical Search Engine for Web Ontologies. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 41, 4 (2011), 810–816. https://doi.org/10.1109/TSMCA.2011.2132705

[33] Antonio J Roa-Valverde and Miguel-Angel Sicilia. 2014. A survey of approaches for ranking on the web of data. *Information Retrieval* 17, 4 (2014), 295–325. https://doi.org/10.1007/s10791-014-9240-0

[34] Stephen E Robertson. 1997. Overview of the okapi projects. *Journal of Documentation* 53, 1 (1997), 3–7. https://doi.org/10.1108/EUM0000000007186

[35] Cristiano Rocha, Daniel Schwabe, and Marcus Poggi Aragao. 2004. A Hybrid Approach for Searching in the Semantic Web. In *Proceedings of the 13th International Conference on World Wide Web*. ACM, 374–383. https://doi.org/10.1145/988672.988723

[36] Gerard Salton, Anita Wong, and Chung-Shu Yang. 1975. A Vector Space Model for Automatic Indexing. *Commun. ACM* 18, 11 (1975), 613–620. https://doi.org/10.1145/361219.361220

[37] Johann Schaible, Thomas Gottron, and Ansgar Scherp. 2014. Survey on Common Strategies of Vocabulary Reuse in Linked Open Data Modeling. In *European Semantic Web Conference*. Springer, 457–472. https://doi.org/10.1007/978-3-319-07443-6_31

[38] Johann Schaible, Thomas Gottron, and Ansgar Scherp. 2016. TermPicker: Enabling the Reuse of Vocabulary Terms by Exploiting Data from the Linked Open Data Cloud. In *International Semantic Web Conference*. Springer, 101–117. https://doi.org/10.1007/978-3-319-34129-3_7

[39] Barry Smith, Michael Ashburner, Cornelius Rosse, Jonathan Bard, William Bug, Werner Ceusters, Louis J Goldberg, Karen Eilbeck, Amelia Ireland, Christopher J Mungall, et al. 2007. The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature Biotechnology* 25, 11 (2007), 1251. https://doi.org/10.1038/nbt1346

[40] Bram van den Akker, Ilya Markov, and Maarten de Rijke. 2019. ViTOR: Learning to Rank Webpages Based on Visual Features. In *The World Wide Web Conference*. ACM, 3279–3285. https://doi.org/10.1145/3308558.3313419

[41] Pierre-Yves Vandenbussche, Ghislain A Atemezing, María Poveda-Villalón, and Bernard Vatant. 2017. Linked Open Vocabularies (LOV): a gateway to reusable semantic vocabularies on the Web. *Semantic Web* 8, 3 (2017), 437–452. https://doi.org/10.3233/SW-160213

[42] Chao Wang, Yiqun Liu, and Shaoping Ma. 2016. Building a click model: From idea to practice. *CAAI Transactions on Intelligence Technology* 1, 4 (2016), 313–322. https://doi.org/10.1016/j.trit.2016.12.003

[43] Gang Wu, Juanzi Li, Ling Feng, and Kehong Wang. 2008. Identifying Potentially Important Concepts and Relations in an Ontology. In *International Semantic Web Conference*. Springer, 33–49. https://doi.org/10.1007/978-3-540-88564-1_3

[44] Jun Xu and Hang Li. 2007. Adarank: A Boosting Algorithm for Information Retrieval. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 391–398. https://doi.org/10.1145/1277741.1277809

[45] Amrapali Zaveri, Anisa Rula, Andrea Maurino, Ricardo Pietrobon, Jens Lehmann, and Sören Auer. 2016. Quality assessment for Linked Data: A Survey. *Semantic Web* 7, 1 (2016), 63–93. https://doi.org/10.3233/SW-150175