Semantic Web 7 (2016) 447–461
IOS Press

DWRank: Learning Concept Ranking for Ontology Search

Editor(s): Krzysztof Janowicz, STKO Lab, University of California, Santa Barbara, USA; Stefan Schlobach, Vrije Universiteit Amsterdam,The Netherlands
Solicited review(s): Yingjie Hu, University of California, Santa Barbara, USA; Laurens Rietveld, Vrije Universiteit Amsterdam, The Netherlands; one anonymous reviewer

Anila Sahar Butt [a,b,*], Armin Haller [a], Lexing Xie [a]
[a] *Australian National University*
Canberra, Australia
E-mail: firstname.lastname@anu.edu.au
[b] *CSIRO Digital Productivity*
Canberra, Australia

Abstract. With the recent growth of Linked Data on the Web there is an increased need for knowledge engineers to find ontologies to describe their data. Only limited work exists that addresses the problem of searching and ranking ontologies based on a given query term. In this paper we introduce DWRank, a two-staged bi-directional graph walk ranking algorithm for concepts in ontologies. DWRank characterises two features of a concept in an ontology to determine its rank in a corpus, the centrality of the concept to the ontology within which it is defined (HubScore) and the authoritativeness of the ontology in which it is defined (AuthorityScore). DWRank then uses a Learning to Rank approach to learn the feature weights for the two aforementioned ranking strategies. We compare DWRank with state-of-the-art ontology ranking models and traditional information retrieval algorithms. This evaluation shows that DWRank significantly outperforms the best ranking models on a benchmark ontology collection for the majority of the sample queries defined in the benchmark. In addition, we compare the effectiveness of the HubScore part of our algorithm with the state-of-the-art ranking model to determine a concept centrality and show the improved performance of DWRank in this aspect. Finally, we evaluate the effectiveness of the design decisions made for the AuthorityScore method in DWRank to find missing *inter-ontology links* and present a graph-based analysis of the ontology corpus that shows the increased connectivity of the ontology corpus after extraction of the *implicit inter-ontology links*.

Keywords: Ontology Search, Learning to Rank, Ontology Ranking

1. Introduction

The growth in Linked Data coupled with the widespread use of ontologies in vertical domains (e.g. bioinformatics, e-commerce, internet-of-things etc.) highlights an increasing need to discover existing ontologies and the concepts and relations within. The benchmark ontology collection [3] that we use in the evaluation of this paper, for example, includes 1022 ontologies that were retrieved through a Web crawl. However, the potential to "reuse" these and other ontologies is hampered by the fact that it is hard to find the right ontology for a given use case. There are several established ontology libraries in vertical domains such as the Open Biological and Biomedical Ontolo-

*Corresponding author. E-mail: anila.butt@anu.edu.au

gies library[1] or the BioPortal [19], where keyword queries are still the preferred method to find concepts and relations in the registered ontologies. However, since there may exist many ontologies that contain concepts and relations with their label matching the keyword query, the matches need to be usefully ranked. There has been some previous work, for example [11,1,19,9], to tackle the problem of finding and ranking ontologies. More recently, dedicated ontology search engines have emerged [27], but the ranking methods they use are based only on document-ranking algorithms. Moreover, most of the ranking techniques in these ontology libraries and search engines only consider the popularity of terms in the ontology corpus, often using the PageRank algorithm, which although effective in some cases [3] penalizes the rank of newly emerged, but well defined ontologies.

In this paper we propose a new ontology concept retrieval framework that uses a number of techniques to rate and rank each concept in an ontology based on how well it represents a given search term. The ranking in the framework is conducted in two phases. First, our offline learning and index construction phase, computes the centrality of a concept within an ontology based on its connectivity to other concepts within the ontology itself. Then, the authority of a concept is computed. It depends on the number of relationships between ontologies and the weight of these relationships based on the authority of the source ontology. The assumption behind this is that ontologies that reuse and are reused by other ontologies are more authoritative than others. Next, a ranking model is learnt from the features like centrality and authority of the concepts using LamdaMART, a learning to rank algorithm. In a second, online query processing phase a list of *top-k* concepts for a user query is selected using the ranking model learnt in the offline phase. The resulting list of k ranked concepts is then evaluated against a ground truth derived through a human evaluation published previously [3]. Our evaluation shows that DWRank, the ranking model proposed as a part of the framework, significantly outperforms the state-of-the-art ranking models on the task of ranking concepts in ontologies for all ten benchmark queries in the ontology collection.

The remainder of the paper is structured as follows. In Section 2 we describe the overall framework and briefly define some of the terms used throughout the paper. Section 3 describes the offline ranking phase of our framework, in particular the DWRank algorithm. Section 4 then describes the online query processing and filtering phase that is independent of the offline ranking model. We evaluate the DWRank algorithm and different design decisions made in DWRank in Section 5. We position our work in relation to state-of-the-art in Section 6 before we conclude in Section 7.

2. Concept Retrieval Framework

In the following we first define the terms used throughout the paper. We then give a brief overview of the mechanics of the ranking framework.

2.1. Preliminaries

An ontology in this paper refers to a graph based formalisation $O = (V, E, L)$ of a domain knowledge. V is a finite set of nodes where $v \in V$ denotes a domain concept in O, E is the edge set where $(v, v') \in E$ denotes an explicit or implicit relationship between v and v'. L is a labelling function which assigns a label $L(v)$ (resp. $L(e)$ or $L(O)$) to node v (resp. an edge $e \in E$ or the ontology O). In practice the labelling function L may specify (1) the node labels to relate the node to the concept it is referring to, e.g. person, place and role; and (2) the edge labels as explicit relationships between concepts e.g., friendship, work and participation or implicit relationships e.g., sub-concept and super-concept, and (3) the ontology label to relate the ontology to the domain or some identity.

2.1.1. Intra-Ontology Relationships.
An *intra-ontology relationship* $I_a = ((v, v'), O)$ is a directed edge (v, v'), where $(v, v') \in E(O)$ for $v \in V(O)$ and $v' \in V(O)$.

2.1.2. Inter-Ontology Relationships.
An *inter-ontology relationship* $I_e = ((v,v'), O, O')$ is a directed edge (O, O'), where $(v, v') \in$ E(O), $L(v) = L(O)$, $L(v') = L(O')$ and $L\ (v,v') = owl{:}imports$[2].

[1] http://www.obofoundry.org/

[2] http://www.w3.org/2002/07/owl#imports

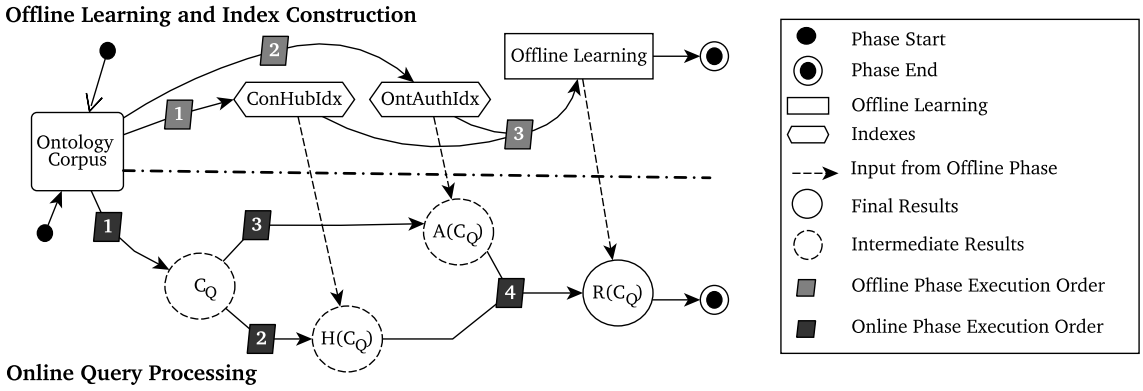

Fig. 1. Relationship-based top-k concept retrieval framework

2.1.3. Forward Link Concepts.
Forward link concepts $C_{FLinks}(v, O)$ is a set of concepts V' in an ontology O, where $V' \subset V(O)$ and $\forall\ v_i \in V'$, $\exists\ (v, v_i) \in E(O)$.

2.1.4. Back Link Concepts.
Back link concepts $C_{BLinks}(v, O)$ is a set of concepts V'' in an ontology O, where $V'' \subset V(O)$ and $\forall\ v_j \in V''$, $\exists\ (v_j, v) \in E(O)$.

2.2. Overview of the framework

The framework is composed of two phases as shown in Fig. 1. The first phase is an offline phase where two indices, i.e. *ConHubIdx* and *OntAuthIdx*, are constructed for the whole ontology corpus. A learning to rank technique is used to learn how to combine the information from these two indices to calculate the final relevance score of a concept to a query. The second phase is an online query processing phase where a query is evaluated and the *top-k* concepts are returned to the user.

2.2.1. Offline Learning and Index construction:
The framework first constructs a *ConHubIdx* on all concepts and an *OntAuthIdx* on all ontologies in the ontology corpus `O`. The *ConHubIdx* maps each concept of an ontology to its corresponding *hub score*. Similarly, the *OntAuthIdx* maps each ontology to its precomputed *authority score*. Finally, the ranking model for the hub and authority scores are learned using the learning to rank algorithm. The *hub score*, *authority score* and learning to rank algorithms are defined in Sec. 3.1

2.2.2. Online Query Processing:
Upon receiving a query `Q`, the framework extracts the *candidate result set* $C_Q = \{(v_1, O_1), \ldots, (v_i, O_j)\}$ including all matches that are semantically similar to `Q` by querying the ontology repository. The *hub score* and *authority score* for all $(v, O) \in C_Q$ are extracted from the corresponding indices as lists $H(C_Q)$ and $A(C_Q)$. A ranked list $R(C_Q)$ of the candidate result set is generated from $H(C_Q)$ and $A(C_Q)$ along with the text relevancy measure by applying the ranking model leant during the offline learning and index construction phase.

3. Offline Learning and Index Construction

In this section the offline learning and index construction phase of the *relationship-based top-k concept retrieval* framework is described. First, we introduce the ranking model in Section 3.1 and then we introduce the index construction based on the ranking model in Section 3.2 (cf. Fig. 2).

3.1. DWRank: A Dual Walk based Ranking Model

Our ranking model characterises two features of a concept to determine its rank in a corpus:

1. A concept is more important, if it is a *central concept* to the ontology within which it is defined.
2. A concept is more important, if it is defined in an *authoritative* ontology.

More precisely, first, the offline ranking module generates for each concept in the corpus a *hub score*, a measure of the *centrality of a concept*, i.e. the extent that the concept is `related` to the domain for which the ontology is formalised. Second, the *authority score* is generated as a measure of the `authoritativeness` of the ontology. A link analysis algorithm, i.e. PageRank, is used that leverages the ontological structure and semantics to compute these scores. However, the difference between our model and a traditional PageRank-like algorithms is two-fold. Firstly, we perform the link analysis independently on each ontology to find a *hub score* and only then on the whole ontology corpus considering an ontology as a node and *inter-ontology relationships* as links. Secondly, we differentiate between the type of relationship (i.e. inter-ontology and intra-ontology) and the direction of the walk varies based on the type of the relationship.

Our Model *DualWalkRank* is named after its characteristic of a dual directional walk to compute the ranks of concepts.

3.1.1. HubScore: The centrality of a concept within an ontology

The *hub score* is a measure of the centrality of a concept within an ontology. We define a *hub function* `h(v,O)` that calculates the *hub score*. The *hub function* is characterised by two features:

- **Connectivity:** A concept is more central to an ontology, if there are more *intra-ontology relationships* starting from the concept.
- **Neighbourhood:** A concept is more central to an ontology, if there is an *intra-ontology relationship* starting from the concept to another central concept.

According to these features, a concept accepts the centrality of another concept based on its forward link concepts (like a `hub`). The *hub function* is therefore a complete reverse of the PageRank algorithm [21] where a node accepts scores from its referent nodes i.e. back link concepts.

We adopt a Reverse-PageRank [13] as the *hub function* to find the centrality of a concept within the ontology. The hub function is an iterative function and at any iteration k, the *hub function* is defined as Eq. 1.

$$h_k(v,O) = \sum_{v_i \in C_{FLinks}(v,O)} \frac{h_{k-1}(v_i,O)}{|C_{BLinks}(v_i,O)|} \tag{1}$$

Within the original PageRank framework there are two types of links in a graph, strong and weak links. The links that actually exist in the graph are *strong links*. *Weak links* are artificially created links by a damping factor α, and they connect all nodes to all other nodes. Since *data-type relationships* of a concept do not connect it to other concepts in an ontology, most PageRank-like algorithms adopted for ontology ranking consider only *object-type relationships* of a concept while ignoring others. We adopt the notion of weak links in our *hub function* to be able to also consider *data-type relationships* along with *object-type relationships* for the ontology ranking. We generate a set of artificial concepts $V'(O)$ in the ontology that act as a sink for every *data-type relationship* and label these concepts with the data-type relationship label, i.e. $\forall\ v_j \in V'$, $\mathrm{L}(v'_j) = \mathrm{L}\ (v_i,v'_j)$. After incorporating *weak links* and *weak nodes*, Eq. 2 reflects the complete feature of our *hub function*.

$$h_k(v,O) = \frac{1-\alpha}{|V|} + \alpha * \sum_{v_i \in C_{SFLinks}(v,O) \cup C_{WFLinks}(v,O)} \frac{h_{k-1}(v_i,O)}{|C_{BLinks}(v_i,O)|} \tag{2}$$

In Eq. 2, $C_{SFLinks}(v,O)$ is a set of *strong forward link concepts* and $C_{WFLinks}(v,O)$ is a set of *weak forward link concepts*. Our *hub function* is similar to [28], but varies from it as we consider *weak nodes* and we are not considering relationships weights. The results presented in [28] also justify our choice of ReversePageRank over other algorithms to measure the centrality. We normalise the hub scores of each concept v within an ontology O through the `z-score` of the concept's hub score after the last iteration of the *hub function* as follows:

$$h_n(v,O) = \frac{h(v,O) - \mu_h(O)}{\sigma_h(O)} \tag{3}$$

In Eq. 3, $h_n(v,O)$ is a normalised hub score of v, $\mu_h(O)$ is an average of hub scores of all concepts in

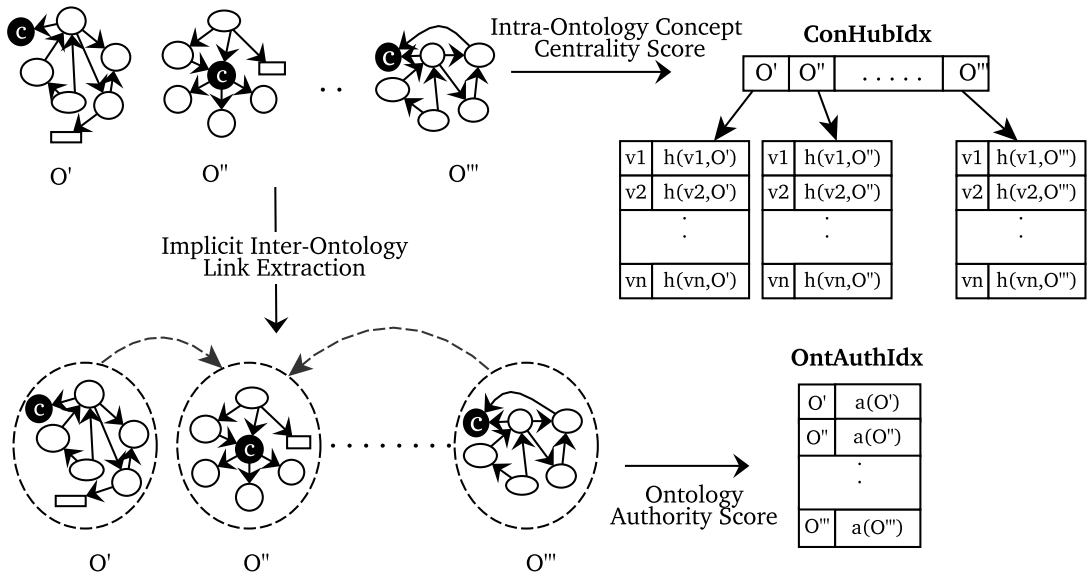

Fig. 2. Offline Index Construction

the ontology and $\sigma_h(O)$ is the standard deviation of hub scores of the concepts in the ontology.

3.1.2. AuthorityScore: The authoritativeness of a concept

The *authority score* is the measure of the authoritativeness of a concept within an ontology. As mentioned earlier, the *authoritativeness of a concept* depends upon the *authoritativeness of the ontology* within which it is defined. Therefore, we define the *authority function* a(O) to measure the *authority score* of an ontology. Our *authority function* is characterised by the following two features:

- **Reuse:** An ontology is more authoritative, if there are more *inter-ontology relationships* ending at the ontology.
- **Neighbourhood:** An ontology is more authoritative, if there is an *inter-ontology relationship* originating at an authoritative ontology and ending at that ontology.

Based on these two features, an *inter-ontology relationship* $I_e((v, v'), O, O')$ is considered as a "positive vote" for the authoritativeness of ontology O' from O. PageRank is adopted as the *authority function*, whereby each ontology is considered a node and *inter-ontology relationships* are considered links among nodes. Eq. 4 formalises the *authority function* which computes the authoritativeness of O at the *kth* iteration.

$$a_k(O) = \frac{1-\alpha}{|O|} + \alpha \sum_{O_i \in O_{BLinks}(O)} \frac{a_{k-1}(O_i)}{|O_{FLinks}(O_i)|} \quad (4)$$

In Eq. 4, $O_{BLinks}(O)$ is a set of *back link ontologies* and $O_{FLinks}(O)$ is a set of *forward link ontologies*. The definition of $O_{FLinks}(O)$ (resp. $O_{BLinks}(O)$) is similar to $C_{FLinks}(v, O)$ (resp. $C_{BLinks}(v, O)$), however, the links are *inter-ontology relationships*.

Similar to the *hub score*, we also compute the `z-score` of each ontology after the last iteration of the *authority function* as follows:

$$a_n(O) = \frac{a(O) - \mu_a(\mathtt{O})}{\sigma_a(\mathtt{O})} \quad (5)$$

In Eq. 5, $a_n(O)$ is the normalised authority score of v, $\mu_a(\mathtt{O})$ is an average of the authority scores of all ontologies in the corpus and $\sigma_a(\mathtt{O})$ is the standard deviation of the authority scores of ontologies in `O`.

3.1.3. DWRank Score

Finally, we define the *DWRank* $R_{(v,O)}$, as a function of the *text relevancy*, the *normalised hub score* and the *normalised authority score* features.

We initially described a linear ranking model with fixed weights in [4] as a quantitative metric for the overall relevance between the query Q and the concept v; and the concept *hub and authority* score as follows:

$$R_{(v,O)} = \gamma F_V(v, \mathtt{Q}) * [\alpha h(v, O) + \beta a(O)]$$
$$F_V(v, \mathtt{Q}) = \sum_{q \in Q} f_{ss}(q, \phi(q_v)) \quad (6)$$

In Eq. 6, α, β and γ are the weights for the *hub function*, the *authority function* and text relevancy of the concept to the query respectively. The *text relevancy* function $F_V(v, \mathtt{Q})$, aggregates the contribution of all fully or partially matched words of a node v, in an ontology O, to the query keywords $q \in \mathtt{Q}$. f_{ss} returns a binary value : it returns 1 if q has a match $\phi(q_v)$ in v, and 0 otherwise. The metric favours the nodes v that are semantically matched to more keywords of the query Q.

Learning the Ranking Model: In our current implementation of the DWRank algorithm a ranking model is learnt from the features i.e. the *text relevancy*, the *hub score* and the *authority score*. This way, it constructs the ranking model 'M' automatically in order to provide the most relevant results by automatically determining the weights of features. To combine different features in a quantitative metric DWRank automatically learns from a set of training instances. Training instances can be regarded as past query experiences, which can teach the system how to rank the results when new queries arrive. Each training instance is composed of the query, one of its relevant/irrelevant answers and a list of features. Intuitively we want to identify the model 'M' from features that can rank relevant answers as high as possible for a given query in the training set T. We want a model such that the log-likelihood of relevant matches is maximized.

This type of model has been extensively studied in the machine learning community. We used `LambdaMART` [29] as a learning to rank tool. The main reason for our choice is its ability to optimise the non-smooth cost functions (i.e. NDCG). The implementation details are provided in Sec. 3.3.

3.2. Index Construction: An execution of DWRank

In this section, we explain the execution model of *DWRank* and the construction of the indices.

3.2.1. ConHubIdx

The ConHubIdx is a bi-level index where each entry in the index maps a concept of an ontology to its *normalised hub score* $h_n(v, O)$ as shown in Fig. 2 (top left). To construct the *ConHubIdx* for all ontologies in O, (1) the *hub function* is executed in an iterative way to get the *hub score* of all the concepts in ontology O, and (2) after the last iteration, we compute the normalised hub scores and (3) insert the concepts along with their normalised hub scores in an ontology to the index.

3.2.2. OntAuthIdx

The OntAuthIdx is an index where each entry in the index maps an ontology to its *normalised authority score* $a_n(O)$ as shown in Fig. 2 (bottom left). To construct the *OntAuthIdx* on the corpus O, (1) the *authority function* is executed to get an *authority score* of all the ontologies in O, (2) after the last iteration, the normalised authority scores are computed, and (3) the ontology along with its normalised authority scores is inserted as an entry to the index.

3.2.3. Inter-Ontology Relationships Extraction

As mentioned earlier, the *authority function* leverages the *inter-ontology relationships* that are directed links among ontologies. If ontology *OntA* reuses the resources in ontology *OntB*, ontology *OntA* declares the reuse of resources through an OWL import property, i.e. `owl:imports`. Since some ontology practitioners fail to explicitly declare the reuse of ontologies, the `owl:imports` relationships in an ontology are often inaccurate representations of the *inter-ontology relationships*. We therefore identify the implicit *inter-ontology relationships* by considering the reused resources in the corpus. Finding the implicit *inter-ontology relationships* involves the following steps:

1. ***Missing Relationships Detection***: To find all missing *inter-ontology relationships* we identify the resources that appear in multiple ontologies. If a resource (referred to as *"reused resource"*) is used in multiple ontologies (referred to as *"hosting ontologies"*) then there must be some *inter-ontology relationships*. If

these relationships are not explicitly defined then there are missing relationships among the ontologies.

Listing 1: SPARQL query

```
SELECT
   ?namespace (count(?s) AS ?count)
FROM <aGraph>
WHERE {
   { {?s rdf:type owl:Class.}
     UNION
     {?s rdf:type rdfs:Class.}
   }
     Bind (REPLACE(str(?s),
     "[^/#]+$", "") AS ?namespace)
} Group By ?namespace
  ORDER BY DESC(?count)
```

2. ***Relationship Direction Identification***: Since *inter-ontology relationships* are directed links between ontologies, another challenge is to find the direction of the missing relationships. A part of the ontology corpus in Fig. 2 (top right), contains a *reused resource* (i.e. node 'c') that appears in three different ontologies O', O'' and O'''. In the absence of explicit relationships, some implicit relationships exist and to create these relationships we need to identify the direction of the relationships, i.e. from O' to O'' and from O''' to O''. To identify the direction, the *namespace* of the *reused resources* are used. If the namespace of the *reused resource* matches to the *namespace* of a *hosting ontology* (e.g. O''), then the ontology is selected as the *"home ontology"* of the *reused resource* and the *inter-ontology relationships* are directed from the *hosting ontologies* (i.e. O', O''') to the *home ontology*, i.e. O''.
3. ***Explicit relationships Creation***: Once the missing relationships and their directions are identified, we create explicit *inter-ontology relationships* using `owl:imports` properties.

The *inter-ontology relationship* extraction process is briefly described in Algorithm 1. Firstly the namespace of each ontology is identified (line 1-3). `TopNS()` returns the namespace that is the

Algorithm 1: FINDREL: Inter-Ontology Relationships Extraction

Input: A finite set $O = \{o_1, \ldots, o_n\}$ of Ontologies
Output: An Index M_{oo} that maps inLinks of all o_i

1 **for** $i \in [1, n]$ **do**
2 $\quad ns_{o_i} \leftarrow o_i$.topNS();
3 $\quad M_{ns}$.put(o_i, ns_{o_i});
4 **for** $r \in o_{i\in[1,n]} \wedge M_{ro}.contains(r) = false$ **do**
5 $\quad$ **while** $\exists\ o_{j\in[1,n]} : r \in o_j \wedge o_i \neq o_j$ **do**
6 $\quad\quad oList_r$.add(o_j);
7 $\quad$ **if** $oList_r.size() > 0$ **then**
8 $\quad\quad oList_r$.add(o_i);
9 $\quad\quad M_{ro}$.put($oList_r$);
10 **while** $\exists\ r_{k\in[1,M_{ro}.size()]}$ **do**
11 $\quad ns_{r_k} \leftarrow r_k$.getNS();
12 $\quad$ **for** $s \in [\ 1, oList_{r_k}.size()]$ **do**
13 $\quad\quad ns_{o_s} \leftarrow M_{ns}$.get($o_s$);
14 $\quad\quad$ **if** $ns_{o_s} = ns_{r_k}$ **then**
15 $\quad\quad\quad o_k \leftarrow o_s$
16 $\quad\quad\quad$ break;
17 $\quad$ **if** $M_{oo}.contains(o_k)$ **then**
18 $\quad\quad oList_{r_k}$.addAllDistinct(M_{oo}.get(o_k))
19 $\quad M_{oo}$.put(o_k, $oList_{r_k}$)
20 **return** M_{oo}

namespace of most of the resources in the ontology. The SPARQL query to find the namespaces in an ontology and the count of resources defined with each namespace is shown in Listing 1. Secondly, all *reused resources* are identified and each resource and a corresponding list of *hosting ontologies* are recorded in M_{ro} as a key value pair (line 4-9). Finally, for each resource in M_{ro} the *home ontology* is identified and the resource URI is replaced with the ontology URI and all missing *inter-ontology relationships* for an ontology are recorded in M_{oo} (line 10-19).

An important point to consider is that although an ontology *OntA* may reuse more than one resource from another ontology *OntB* there will only be one *inter-ontology relationship* from *OntA* to *OntB* according to the semantics of the *owl:imports* property. Therefore, independently of the number of resources that are reused in *OntA*

Table 1
Top five reused ontologies based on their explicit inter-ontology relationships

URI	Count
http://def.seegrid.csiro.au/isotc211/iso19150/-2/2012/basic	36
http://purl.org/dc/elements/1.1/	25
http://www.ifomis.org/bfo/1.1	16
http://www.w3.org/2006/time	16
http://www.ontologydesignpatterns.org/schemas/cpannotationschema.owl	15

Table 2
Top five reused ontologies based on their implicit inter-ontology relationships

URI	Count
http://www.w3.org/2002/07/owl#	881
http://www.w3.org/2000/01/rdf-schema	361
http://www.w3.org/1999/02/22-rdf-syntax-ns	298
http://xmlns.com/foaf/0.1/	228
http://www.w3.org/2004/02/skos/core	140

from *OntB*, we create a single *inter-ontology relationship* from *OntA* to *OntB*.

Table 1 and Table 2 show the top five ontologies in the benchmark ontology collection [3] and the corresponding number of *inter-ontology relationships* that are directed to these ontologies (i.e. *reuse count)* counted through *explicit* and *implicit* relationships, respectively. A detailed analysis on the effectiveness of `FindRel` is presented in Sec. 5.2.3

3.3. Learning Feature Weights

4. Online Query Processing

Learning to rank has been extensively studied in the machine learning community. We use `LambdaMART` [29], a listwise learning to rank algorithm, which means that a list of training example of resources is provided where it is known which of the resource should be ranked higher in the result set. To learn the ranking model by the `LambdaMART` algorithm we use the RankLib[3] Library that provides an implementation of several learning to rank algorithms including `LambdaMART`. The design choices we made to learn the ranking model are as follows:

Feature Set: Other than the core features of DWRank, i.e. *text relevancy*, *hub score* and *authority score*, two extra features *max_hub score* and *min_hub score* are introduced while training the ranking model to normalize the *hub score* across ontologies.

- **`max_hub score:`** For a concept v, `max_hub` score for v is the maximum score of any concept v' in the ontology where $v \in V(O)$ and $v' \in V(O)$.
- **`min_hub score:`** For a concept v, `min_hub` score for v is the minimum score of any concept v' in the ontology where $v \in V(O)$ and $v' \in V(O)$

Training Data: The training dataset is prepared from the ground truth published as part of CBRBench [3]. The benchmark[4] provides manually created relevance judgements for 10 sample queries on the dataset. Each query has a list of relevant resources and the relevance score (0-4) of the resource to the query. For each resource of the ground truth, DWRank features' values are computed.

Metrics: The ranking model is optimized for `Normalized Discounted Cumulative Gain (NDCG)` while training the model. The model

[3] `http://sourceforge.net/p/lemur/wiki/RankLib/`

[4] `https://zenodo.org/record/11121#.VDcYdK3I9yA`

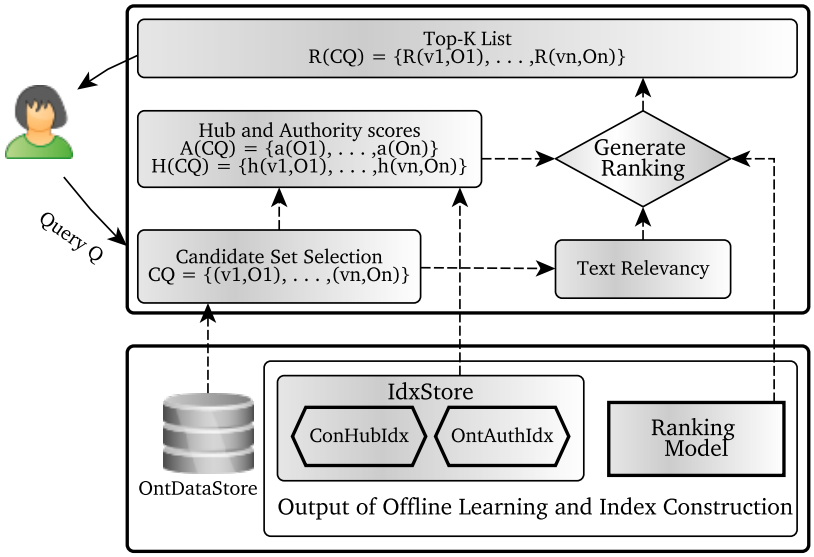

Fig. 3. Online Query Processing

is then tested against `Precision`, `Mean Average Precision (MAP)` and `Discounted Cumulative Gain (DCG)` metrics.

The ranking model is learned with the above mentioned features, the training data and the metrics by `LambdaMART`; and saved on disk.

In this section, we first describe the concept retrieval task and then we outline the online query processing technique that finds the top-k ranked concepts for `Q` in `O` with the highest semantic relevance.

4.1. Concept Retrieval Task

Given a query string $Q = \{q_1, q_2, \ldots, q_k\}$, an Ontology corpus $O = \{O_1, O_2, \ldots, O_n\}$ and a word sense similarity threshold θ, the `concept retrieval task` is to find the $C_Q = \{(v_1, O_1), \ldots, (v_i, O_j)\}$ from `O`, such that there is a *surjective function* f_{sj} from `Q` to C_Q where (a) v has a partial or an exact matched word $\phi(q_v)$ for q ∈ `Q` (b) for a partially matched word, $SenSim(q, \phi(q_v)) \geq \theta$. We refer to C_Q as a candidate set of Q introduced by the mapping f_{sj}.

$SenSim(q, \phi(q_v))$ is a word similarity measure of a query keyword and a partially matched word in $L(v)$.

4.2. Query Evaluation

In the online query evaluation (c.f. Fig. 3), first a candidate set for a *top-k* concept is selected from the ontology data store, i.e. *OntDataStore*, and then the relevance of each concept is calculated based on the formulae defined in Eq. 6.

4.2.1. Candidate Result Set Selection

A keyword query evaluation starts with the selection of a candidate set C_Q for `Q`. A candidate result set C_Q is characterised by two features:

1. To be part of the candidate set a candidate concept v must have at least one exact or partial match $\phi(q_v)$ for any query keyword $q \in Q$ as part of the value of (a) the `rdfs:label`, (b) the `rdfs:comment` (c) the `rdfs:description` properties; or $\exists\ q \in Q \mid \phi(q_v)$ is part of $L(v)$.
2. The word sense similarity of q and $\phi(q_v)$ i.e. $senSim(q, \phi(q_v))$ should be greater than the sense similarity threshold θ.

In our current implementation, we check the word sense similarity using WordNet and set the word sense similarity threshold $\theta = 0.85$. Each entry in a candidate list denotes a candidate concept *'v'* and is a pair *(v,O)* (shown in Fig. 3) of v and O where $v \in V(O)$. Since for the *reused resources*

Table 3
DWRank Effectiveness

	DWRank Fixed Weight Linear Model				DWRank with LTR			
Query Terms	P@10	MAP@10	DCG@10	NDCG@10	P@10	MAP@10	DCG@10	NDCG@10
Person	0.9	0.98	37.58	0.51	1.0	0.8762	55.2391	0.8105
Name	0.7	0.72	19.11	0.41	0.6	0.6761	17.3283	0.4243
Event	1.0	1.0	35.12	0.51	1.0	0.8049	33.5500	0.6868
Title	0.7	0.78	12.45	0.26	1.0	0.9299	17.1655	0.4874
Location	0.7	0.86	24.88	0.60	1.0	0.8831	26.5708	0.5419
Address	0.8	0.89	23.53	0.59	0.9	0.8687	19.4308	0.6109
Music	0.7	0.80	14.82	0.40	0.7	0.7823	14.8165	0.6135
Organization	0.9	0.85	33.70	0.53	1.0	0.8902	56.7080	0.8321
Author	0.8	0.78	18.24	0.48	0.6	0.6725	16.7875	0.8129
Time	0.8	0.74	22.53	0.49	0.5	0.6601	14.5987	0.3907
Average	0.8	0.84	24.196	0.49	0.82	0.8044	27.2195	0.6211

there are multiple *hosting ontologies*, therefore *'v'* may have multiple entries in a candidate set if it is a *reused resource*.

4.2.2. Concept Relevance

For each entry in the candidate list, two scores are retrieved from the stored indices built during the *offline ranking phase*. The entry *(v,O)* is used to retrieve the *hub score* of concept v in ontology O from the *ConHubIdx*, and the *authority score* of ontology O from the *OntAuthIdx*. Along with the hub and authority score, the `max_hub` score and the `min_hub` score for each concept in the candidate list is computed from *ConHubIdx*, while the text relevancy $F_V(v, Q)$ is calculated as described in Eq. 6. Once a complete feature set for all concepts in the candidate list is retrieved, the framework generates the relevance score for the concepts using the ranking model learned in 3.3.

4.2.3. Top-k results

Once the relevance score for all concepts in the candidate list is computed, the framework ranks the list according to the relevance score and returns the top-k results to the user.

5. Experimental Evaluation

In the following we present an experimental evaluation of our *concept retrieval framework* on a benchmark suite, i.e. the CBRBench - Canberra Ontology Ranking Benchmark [3]. We conducted a set of experiments to evaluate different design decisions of DWRank.

5.1. Experimental Settings

To evaluate our approach we use a benchmark suite CBRBench [3], that includes a collection of ontologies, a set of benchmark queries and a ground truth established by human experts. This collection is composed of 1022 ontologies and ten keyword queries: `Person`, `Name`, `Event`, `Title`, `Location`, `Address`, `Music`, `Organization`, `Author` and `Time`. The benchmark evaluates eight state-of-the-art ranking algorithms: *Tf-Idf*[23], *BM25*[22], *Vector Space Model* (`VSM`)[24], *Class Match Measure* (`CMM`) [1], *PageRank* (`PR`)[21], *Density Measure* (`DEM`)[1], *Semantic Similarity Measure* (`SSM`)[1] and *Betweenness Measure* (`BM`)[1] on the task of ranking ontologies. We use the performance of these ranking models as the baseline to evaluate our approach. The effectiveness of the framework is measured in terms of its *Precision* (`P`), *Mean Average Precision* (`MAP`), *Discounted Cumulative Gain* (`DCG`) and *Normalised Discounted Cumulative Gain* (`NDCG`).

5.2. Experimental Results

5.2.1. Effectiveness of DWRank

In the first set of experiments, we evaluated the effectiveness of `DWRank` in comparison with the eight baseline ranking models.

Experiment-1: Offline Learning In this experiment, we study the impact of the offline training on the quality of the ranking. For the evaluation we implemented two versions of DWRank:

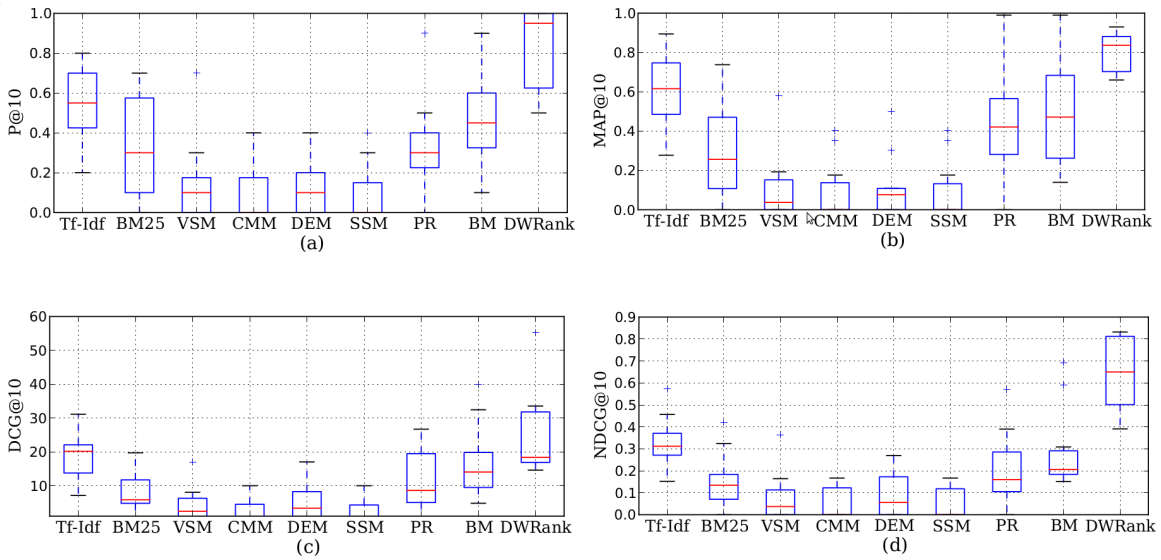

Fig. 4. Effectiveness of Ranking Model

1. `DWRank Fixed Weight Linear Model:` where hub score, authority score and text relevancy are combined in a linear model (i.e. Eq. 6) and the values of weights α , β and γ are set to 0.5, 0.5 and 1 respectively.
2. `DWRank with Learning to Rank Approach:` By using LambdaMART, a LTR algorithm, a ranking model is learnt from the hub score, the authority score and the text relevancy along with two deduced features i.e. the max_hub score and the min_hub score.

For `DWRank fixed weight linear model` we ran the ten sample queries on the ontology collection and retrieved the *top-k* results according to the proposed linear ranking model in Eq. 6. We recorded the `P@10`, the `MAP@10`, the `DCG@10` and the `NDCG@10`. The effectiveness measure results of this implementation of `DWRank` are shown in Table 3. To evaluate `DWRank with learning to rank approach`, the `Leave-one-out Cross Validation (LOOCV)` approach is adopted as follows: for n number of queries we remove the relevance judgement for the training examples of one query and train the ranking model on the training examples of the remaining $n-1$ queries and then we evaluate the performance of the trained model on the n_{th} query. Once the process is repeated for n queries, the mean performance is computed. We applied LOOCV on the queries and the gold standard; and record the `P@10`, the `MAP@10`, the `DCG@10` and the `NDCG@10`. The results are presented in Table 3.

The experimental results show that DWRank with the hub score, the authority score and the text relevancy combined by a model learnt through LTR performs better than the DWRank fixed weight linear model.

Experiment-2: Effectiveness of Top-k Search Next, we compared our results with the available baseline for the sample queries. We compare the performance of the `DWRank fixed weight linear model` (so onward referred as DWRank) with the baseline algorithms. The results are shown in Fig. 4. Each graph here presents an effectiveness measure of a ranking model for all ten queries, where the x-axis is the *ranking model* and the y-axis is the *unit of measure*. Each box on a graph presents the range of effectiveness measure for 10 sample queries according to the gold standard. Fig. 4 shows the maximum, minimum and average performance of DWRank in comparison to the performance of the baseline ranking models for each of the ten queries. The graph shows that DWRank performs better than the best performing ranking algorithm for most queries. For some of the queries, the `P@10` and `MAP@10` for DWRank is lower than the other best performing ranking models. However, the maximum average `MAP@10` for DWRank on ten queries is 0.80 which is greater

than the average of Tf-Idf, the best baseline ranking model, (i.e., 0.55). The box plot also shows that `MAP@10` of DWRank ranges from 0.65 ~1.0 that means the performance of DWRank is more stable on the ontology collection for the sample queries than the baseline ranking models.

Similarly, the `DCG@10` values in Fig. 4(c) and `NDCG@10` values in Fig. 4(d) for the ranking models show that DWRank is more effective than the baseline models. The maximum and minimum measures are closer to the *Betweenness Measure (BM)* or the *Tf-Idf* model, however, the average performance of DWRank is much higher than the average performance of the BM and Tf-Idf models.

Discussion: The results show that `DWRank with learning to rank approach` outperforms `DWRank fixed weight linear model` as well as the baseline ranking models. Moreover, the results presented in [4] show that `DWRank fixed weight linear model` is more effective as compared to the baseline ranking models. This implies that although the learning to rank approach increases the effectiveness of the base model, i.e. DWRank, the characteristics of DWRank (centrality and authoritativeness) are effective enough to produce a good ranking. The results in [3] shows that popularity (i.e. reuse of ontology concepts) or graph based analysis (i.e. coverage or centrality) alone are not an effective model for ontology ranking. However, our approach leverages both, centrality (graph based approach) and reuse (popularity based approach) together to produce a ranking much closer to human expectations than the state-of-the-art ranking algorithms.

5.2.2. Quality of HubScore - Measure of the centrality of a concept

To evaluate the quality of the *hub score* we consider CARRank [28] as a baseline. The reason of comparing the *hub score* quality with the quality of CARRank is two fold: (1) CARRank uses a similar approach (i.e. ReversePage Rank), and (2) the performance results in [28] prove it a better approach than other centrality measures e.g. Betweenness Measure [1] and Density Measure [1]. Since the CARRank algorithm and the gold standard are not available online, we implemented CARRank in Java and adopted a similar evaluation strategy as presented in [28].

To evaluate the two approaches, we tried to collect ontologies and their top-10 concepts. Four representative ontologies, where members of CSIRO were part of the ontology engineering team, were selected as shown in Table 4. We asked the ontology engineers of the four ontologies to list the top 10 central concepts of the ontology they co-designed. We then compare the reference ranking produced by the ontology engineer with the top-10 ranked list generated by HubScore and CARRank. Table 5 presents the comparison on the concepts ranking for the SSN and the ABS ontology. Concepts listed in bold font are the relevant ranking results.

Table 4
Representative Ontologies

Ontology	Concept#
ABS Ontology (abs.owl)	43
SSN Ontology (ssn.owl)	51
Project Ontology (project.owl)	51
Science Ontology (science.owl)	83

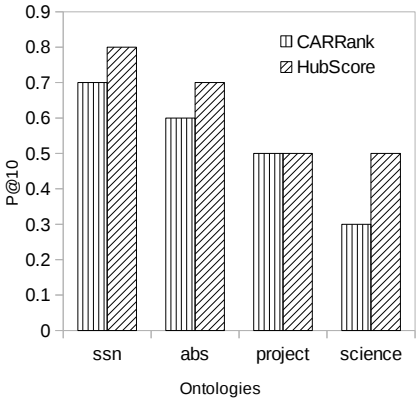

Fig. 5. Comparison of Ranking Concepts

In Table 5 `HubScore` ranks 8 (resp. 7) relevant answers in the top 10 ranking results for the SSN (resp. ABS) ontology in comparison to `CARRank` that ranks 7 (resp. 6) relevant answers for the SSN (resp. the ABS) ontology. Moreover, relatively more relevant results are ranked at the top of the list by HubScore. The quality of both these algorithms is measured in terms of P@10 for the four representative ontologies and presented in Fig. 5.

Though the precision of HubScore on the representative ontologies increases by up to 20% compared to CARRank, the ranked list also seems more meaningful than CARRank. This can be seen in Table 6 that presents the top 5 concepts of the

Table 5
HubScore Quality: Centrality of Concepts

Rank	SSN Ontology			ABS Ontology		
	Reference Answers	HubScore	CARRank	Reference Answers	HubScore	CARRank
1	Sensor	**System**	**System**	Employer	**Employer**	**Person**
2	Observation	**Observation**	**Observation**	Person	**Period**	**EmployeeRole**
3	Property	**FeatureOfInterest**	**Deployement**	EmployeeRole	**Person**	**LegalEntity**
4	SensorOutput	**Deployement**	Platform	EmployerRole	**EmployerRole**	**EmployerRole**
5	SensorInput	Platform	SensingDevice	Period	**EmployeeRole**	Company
6	Stimulus	**SensorOutput**	**Sensor**	AusBusinessNo.	**LegalEntity**	**Employer**
7	FeatureOfInterest	**Sensing**	**SensorOutput**	Address	Company	PrivateCompany
8	Sensing	**Property**	Device	LegalEntity	**TypeOfActivityUnit**	PublicCompany
9	System	**Sensor**	**Sensing**	TypeOfActivityUnit	IncomePeriod	**TypeOfActivityUnit**
10	Deployement	Device	SurvivalRange	EnterpriseGroup	Occupation	AsicRegistration

Table 6
HubScore Quality: Top-5 Concepts of Foaf Ontology

Rank	HubScore	CARRank
1	Person	Person
2	Agent	Agent
3	Group	OnlineGamingAccount
4	Organization	OnlineChatAccount
5	OnlineGamingAccount	OnlineEcom.Account

FOAF ontology ranked by HubScore and CARRank.

5.2.3. *Effect of FindRel: Extraction of Implicit Inter-Ontology links*

The Authority score calculation of DWRank in Sec. 3.1.2 is based on a link-based analysis (i.e. PageRank) that computes the popularity of an ontology in the ontology corpus. However, missing links among ontologies lead to wrong popularity scores [6]. We, therefore, find the missing *inter-ontology links* and present a graph-based analysis of the ontology corpus that shows the increased connectivity of the ontology corpus after extraction of the *implicit inter-ontology links* with `FindRel`.

Table 7 presents different statistical properties of the ontology corpus with and without considering explicit *inter-ontology relationships*. The `Node` notation represents the number of ontologies in the corpus. `Sink Node` is the number of ontologies that are imported (reused) by other ontologies and `Source Node` represents the number of ontologies that import at least one ontology. `Isolated Node` counts the ontologies which neither import nor are imported by any other ontology in the corpus. Similarly, `Edge` is the count of links in the ontology corpus and `Average Degree` is the number of inlinks and outlinks for each node (ontology). `Highest Degree`, `Highest Indegree` and `Highest Outdegree` are the maximum number of inlinks and outlinks, maximum number of inlinks and maximum number of outlinks for a node, respectively.

Table 7
Statistical Properties: Explicit vs. Implicit Inter-Ontology Links

	Explicit Link Graph	Implicit Link Graph
Node	1019	1019
Sink Node	177	348
Sink Node(%)	17.37(%)	34.15(%)
Source Node	204	815
Source Node(%)	20(%)	79.98(%)
Isolated Node	742	135
Isolated Node(%)	72.81(%)	13.25(%)
Edge	431	2311
Average Degree	0.85	4.54
Highest Degree	38	228
Highest Indegree	36	228
Highest Outdegree	26	29

Three language-level vocabularies, namely RDF[5], RDFS[6] and OWL[7], and all the *inter-ontology links*

[5] `http://www.w3.org/1999/02/22-rdf-syntax-ns#`
[6] `http://www.w3.org/2000/01/rdf-schema#`
[7] `http://www.w3.org/2002/07/owl#`

involving them makes our statistics biased towards the improved results through the implicit link extraction. Therefore, they are excluded from the following analysis.

`Nodes` remains the same in the explicit link ontology corpus and the implicit link ontology corpus, as only the missing links among ontologies are extracted in `FindRel`; however, they differ in the number of edges. As shown in Table 6, the `Implicit-Link Graph` contains more edges (links) than the `Explicit-Link Graph`. `Average Degree` and `Isolated Node` values for both the graphs represent that without *implicit inter-ontology links*, the ontology corpus is disconnected and most of the ontologies (i.e. isolated nodes) will end up with the same *authority score* that will minimise the effect of the authority score contribution towards the DWRank model. The `Implicit-Link Graph` statistics are more interesting, because it shows that ontologies often miss an import statement in their meta-description despite reusing some concepts of an external ontology. Therefore, it will be insufficient to only leverage meta-descriptions to perform the link analysis for carrying out tasks such as ranking.

6. Related Work

Ranking of Semantic Web data depends upon a multitude of factors that include globally computed ranks to query-dependent ranks, RDF Schema ranking (T-Box) to RDF data ranking (A-Box) and Schema-aware ranks to Schema-agnostic ranks. Existing ranking techniques can further be classified based upon the type of the user query (i.e. keywords, structured query, faceted search) and semantic search engines' information display mode (i.e. document-centric, relation-centric, entity-centric). There have been several methods proposed to handle the task of ranking Semantic Web data. Most of the existing techniques fulfil the user's information need by prioritising matched resources in accordance with their popularity (or importance), authority, content informativeness and/or relatedness. Although most ranking algorithms are rooted in traditional graph based ranking methods, such as PageRank [21] and HITS [17], they are modified to make them suitable for Semantic Web data.

The Linked Open Vocabularies (LOV) search engine[8], initiated in March 2011, is to the best of our knowledge, the only purpose-built ontology search engine available on the Web. It uses a ranking algorithm based on the term popularity in Linked Open Data (LOD) and in the LOV ecosystem [27]. There are also some ontology libraries available that facilitate the locating and retrieving of potentially relevant ontology resources [9]. Some of these libraries are domain-specific such as the Open Biological and Biomedical Ontologies library[9] or the BioPortal [19], whereas others are more general such as OntoSearch [25] or the TONES Ontology Repository[10]. However, as discussed by Noy & d'Aquin [9] only few libraries support a keyword search, only one (Cupboard [7]) supports a ranking of ontologies based on a keyword query using an information retrieval algorithm (i.e. tf-idf), and none support the ranking of resources within these ontologies.

Semantic Search engines such as Swoogle [10] (which was initially developed to rank ontologies only), Sindice.com [26], Watson [8], or Yars2 [14] do allow a search of ontology resources through a keyword query. Swoogle [10] ranks documents using a variation on PageRank which iteratively calculates the rank for ontologies based on references to ontology vocabulary (classes and properties) defined in other ontologies. It further describes TermRank to sort classes and properties by their popularity in Semantic Web documents, based on which a class-property relationship ranking is proposed. However, the index is not maintained and out-of-date. Sindice [20] is a registry and lookup service for Semantic Web data. In case of a literal search, resources are ranked according to their text relevancy with the search terms, meaning that for resources, search results with common host-names of the search resource are ranked higher than the other results. These ranking techniques were already deemed as error-prone in conventional Web search engines. Watson [8] focuses on retrieving relevant resources with less focus on the ranking and the Yars2 [14] ranking model follows traditional link-based ranking methods [16], in particular, adapted versions of the PageRank algorithm [21], where links from one source of infor-

[8] `http://lov.okfn.org`
[9] `http://www.obofoundry.org/`
[10] `http://owl.cs.manchester.ac.uk/repository/`

mation to another are regarded as a 'positive vote' from the former to the latter. Often, these ranking schemes also take the provenance graph of the data into account [15].

AKTiveRank [1], ranks ontologies based on how well they cover specified search terms. The approach applies graph centrality measures (e.g. betweenness centrality) on the task of ontology ranking. Falcon [5] is a popularity-based scheme to rank concepts and ontologies.

SEAL (SEmantic portAL) [18] is one of the early relation-centric approaches that ranks direct relationships higher than inferred ones. Ranks are query specific and computed only for focussed graph. The SemRank [2] relevance model presents a property-centric modulative approach. It uses a blend of semantic techniques, information theoretic techniques, and heuristics to determine the rank of semantic associations between two resources. This approach is limited to rank different associations that exist between a pair of resources and is probably ineffective to rank the resources and semantic associations that exist among different pairs of resources.

Other strategies, mainly based on methods proposed in the information retrieval community, are employed in Semantic Search [12], but what all these methods have in common is that they are targeted to rank instances, but do not work well for ranking concepts and properties in ontologies [11,1]. Another related approach is presented in [28] that identifies the most important concepts and relationships from a given ontology. However, the approach does not support the ranking of concepts that belong to multiple ontologies.

7. Conclusion and Future Work

In this paper we have presented a relationship-based *top-k* concept retrieval and ranking framework for ontology search. The ranking model is comprised of two phases, an offline learning and index construction phase and an online query and evaluation phase. In the offline phase our DWRank algorithm computes a rank for a concept based on two features, the centrality of the concept in the ontology, and the authority of the ontology that defines the concept; and then combines these scores in a model learnt through LambdaMART. The online query evaluation phase filters the top-k ranked list of concepts by using the ranking model learnt during the offline learning phase. We evaluated our two versions of DWRank; having a fixed weight linear model and the proposed learning to rank model. The learning to rank approach proposed in the offline learning phase increased the performance of DWRank. We then compare our approach against state-of-the-art ranking models on a benchmark ontology collection. The evaluation shows that DWRank outperforms the best performing ranking algorithm for most queries while exhibiting a more stable performance (i.e. MAP@10 of 0.80) than the average of the best performing ranking models of the benchmark (i.e. MAP@10 of 0.55).

While our algorithm shows significantly improved effectiveness compared to the state-of-the-art in ontology ranking models, in future work, we will focus on improving the performance in order to make the framework more efficient. We intend to create efficient indices that can retrieve candidate result sets efficiently to increase the performance of the online queries.

A learning to rank approach needs a query log with a set of good-quality query-answer pairs to have its weights tuned. Since we did not have a real-world query log at hand, in our current implementation the offline ranking model is learnt using the gold standard available as a part of CBR-Bench. However, in future work, we shall focus on improving the offline ranking model incrementally by employing a real query log. Another planned future work is to extend the approach to consider ranking relationships among concepts and ontologies themselves.

Acknowledgements

This paper is an extension of our EKAW paper [4]. We thank the EKAW and SWJ reviewers for their valuable feedback.

References

[1] H. Alani, C. Brewster, and N. Shadbolt. Ranking Ontologies with AKTiveRank. In *The Semantic Web – ISWC 2006, 5th International Semantic Web Conference*, pages 1–15. Springer, 2006.

[2] K. Anyanwu, A. Maduko, and A. Sheth. SemRank: ranking complex relationship search results on the se-

mantic web. In *Proceedings of the 14th international conference on World Wide Web*, pages 117–127. ACM, 2005.

[3] A. S. Butt, A. Haller, and L. Xie. Ontology search: An empirical evaluation. In *The Semantic Web – ISWC 2014*, pages 130–147. Springer, 2014.

[4] A. S. Butt, A. Haller, and L. Xie. Relationship-based top-k concept retrieval for ontology search. In *Knowledge Engineering and Knowledge Management*, pages 485–502. Springer, 2014.

[5] G. Cheng, W. Ge, and Y. Qu. Falcons: searching and browsing entities on the semantic web. In *Proceedings of the 17th international conference on World Wide Web*, pages 1101–1102. ACM, 2008.

[6] G. Cheng and Y. Qu. Relatedness between vocabularies on the web of data: A taxonomy and an empirical study. *Web Semantics: Science, Services and Agents on the World Wide Web*, 20:1–17, 2013.

[7] M. d'Aquin and H. Lewen. Cupboard–a place to expose your ontologies to applications and the community. In *The Semantic Web: Research and Applications*, pages 913–918. Springer, 2009.

[8] M. d'Aquin and E. Motta. Watson, more than a semantic web search engine. *Semantic Web*, 2(1):55–63, 2011.

[9] M. d'Aquin and N. F. Noy. Where to publish and find ontologies? a survey of ontology libraries. *Web Semantics: Science, Services and Agents on the World Wide Web*, 11:96–111, 2012.

[10] L. Ding, T. Finin, A. Joshi, R. Pan, R. S. Cost, Y. Peng, P. Reddivari, V. Doshi, and J. Sachs. Swoogle: a search and metadata engine for the semantic web. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 652–659. ACM, 2004.

[11] L. Ding, R. Pan, T. Finin, A. Joshi, Y. Peng, and P. Kolari. Finding and ranking knowledge on the semantic web. In *The Semantic Web–ISWC 2005*, pages 156–170. Springer, 2005.

[12] M. Fernandez, V. Lopez, M. Sabou, V. Uren, D. Vallet, E. Motta, and P. Castells. Semantic search meets the web. In *Semantic Computing, 2008 IEEE International Conference on*, pages 253–260. IEEE, 2008.

[13] D. Fogaras. Where to start browsing the web? In *Innovative Internet Community Systems*, pages 65–79. 2003.

[14] A. Harth, J. Umbrich, A. Hogan, and S. Decker. Yars2: a federated repository for querying graph structured data from the web. In *Proceedings of the The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference*, pages 211–224. Springer-Verlag, 2007.

[15] A. Hogan, A. Harth, and S. Decker. Reconrank: A scalable ranking method for semantic web data with context. In *Proceedings of the 2nd Workshop on Scalable Semantic Web Knowledge Base Systems*, 2006.

[16] A. Hogan, A. Harth, J. Umbrich, S. Kinsella, A. Polleres, and S. Decker. Searching and browsing Linked Data with SWSE: The Semantic Web Search Engine. *Web Semantics: Science, Services and Agents on the World Wide Web*, 9(4):365–401, 2011.

[17] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1999.

[18] A. Maedche, S. Staab, N. Stojanovic, R. Studer, and Y. Sure. Semantic portal-the seal approach. *Spinning the Semantic Web*, pages 317–359, 2003.

[19] N. F. Noy, N. H. Shah, P. L. Whetzel, B. Dai, M. Dorf, N. Griffith, C. Jonquet, D. L. Rubin, M.-A. Storey, C. G. Chute, et al. Bioportal: ontologies and integrated data resources at the click of a mouse. *Nucleic Acids Research*, 37(suppl 2):W170–W173, 2009.

[20] E. Oren, R. Delbru, M. Catasta, R. Cyganiak, and G. Tummarello. Sindice.com: A document-oriented lookup index for open linked data. *International Journal of Metadata, Semantics and Ontologies*, 3(1):37–52, 2008.

[21] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: bringing order to the web. 1999.

[22] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, M. Gatford, et al. Okapi at trec-3. *NIST SPECIAL PUBLICATION SP*, pages 109–109, 1995.

[23] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

[24] G. Salton, A. Wong, and C.-S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[25] E. Thomas, J. Z. Pan, and D. Sleeman. Ontosearch2: Searching ontologies semantically. In *Proceedings of the OWLED 2007 Workshop on OWL: Experiences and Directions*, volume 258 of *CEUR Workshop Proceedings*, 2007.

[26] G. Tummarello, R. Delbru, and E. Oren. Sindice. com: weaving the open linked data. In *Proceedings of the 6th international The semantic web and 2nd Asian conference on Asian semantic web conference*, pages 552–565. Springer-Verlag, 2007.

[27] P.-Y. Vandenbussche and B. Vatant. Linked Open Vocabularies. *ERCIM news*, 96:21–22, 2014.

[28] G. Wu, J. Li, L. Feng, and K. Wang. Identifying potentially important concepts and relations in an ontology. In *The Semantic Web-ISWC 2008*, pages 33–49. 2008.

[29] Q. Wu, C. J. Burges, K. M. Svore, and J. Gao. Ranking, boosting, and model adaptation. Technical report, Technical report, Microsoft Research, 2008.