Relationship-based Top-K Concept Retrieval for Ontology Search

Anila Sahar Butt[1,2], Armin Haller[2], and Lexing Xie[2]

[1] CSIRO CCI, `firstname.lastname@csiro.au`
[2] Australian National University, `firstname.lastname@anu.edu.au`

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. We applied this algorithm on the task of searching and ranking concepts in ontologies and compare it with state-of-the-art ontology ranking models and traditional information retrieval algorithms such as PageRank and tf-idf. Our evaluation shows that DWRank significantly outperforms the best ranking models on a benchmark ontology collection for the majority of the sample queries.

1 Introduction

The growth in Linked Data coupled with the widespread use of ontologies in vertical domains such as in the bioinformatics domain highlights an increasing need to discover existing ontologies and the concepts and relations within. Prefix.cc[3], a service to register prefixes, for example, counts about 1250 ontologies (April 2014). Currently, the potential to "reuse" ontologies is hampered by the fact that it is hard to find the right ontology for a given use case. There has been previous work, for example [7, 1, 14, 13], to tackle the problem of finding and selecting ontologies. However, only with search engines that employ sophisticated techniques to rank ontologies based on a keyword query, will it be possible to use ontologies to their full potential. Recently, search engines for ontologies have emerged [18], but the ranking algorithms they use are based only on document-ranking algorithms. 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 ranking algorithm, DWRank, 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 which 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. In a second, online query processing phase a candidate set for a *top-k* concept is selected from the offline ranked list of ontologies and then filtered based on two strategies, the diverse results semantics and the intended type semantics. The resulting list of *top-k* ranked concepts is then evaluated against a ground truth derived through a human evaluation published previously [2]. Our evaluation shows that DWRank 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.

[3] `http://prefix.cc`

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 with and without the additional filters in Section 5. We position our work in relation to state-of-the-art in Section 6 before we conclude in Section 7.

2 Relationship-based top-k Concept Retrieval

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 referent concept, 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.

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 $v' \in V(O)$.

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*[4].

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

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

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. The second phase is an online query processing phase where a query is evaluated and the *top-k* concepts are returned to the user.

Offline Ranking and Index construction: The framework first constructs a *ConHubIdx* on all concepts and *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*.

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

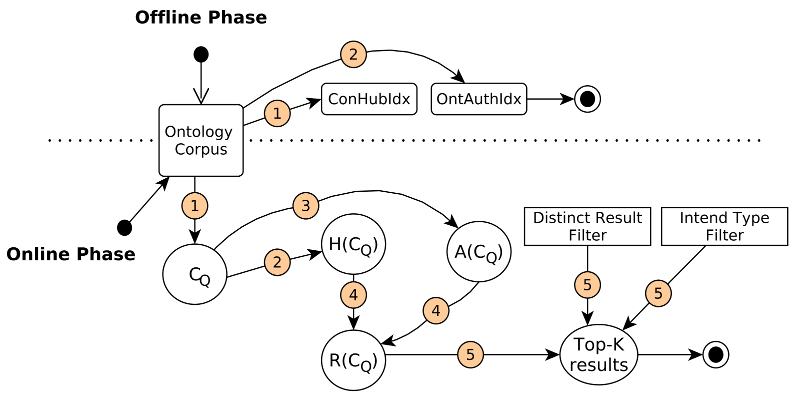


Fig. 1. Relationship-based top-k concept retrieval framework

Online Query Evaluation: Upon receiving a query Q, the framework extracts the *candidate result set* $C_Q = \{(v_1, O_1), ..., (v_i, O_j)\}$ including all matches that are semantically similar to Q, by querying the ontology repository. The *hub score* and *authority score* for all $(v, O) \in C_Q$ is extracted from the corresponding indices as $H(C_Q)$ and $A(C_Q)$ lists. A ranked list $R(C_Q)$ of candidate result set is computed from $H(C_Q)$ and $A(C_Q)$. $R(C_Q)$ is further filtered to satisfy two result set properties, i.e. the *Diverse Result Semantics* and the *Intended Type Semantics*, as introduced in Sec. 4.3.

3 Offline Ranking and Index Construction

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

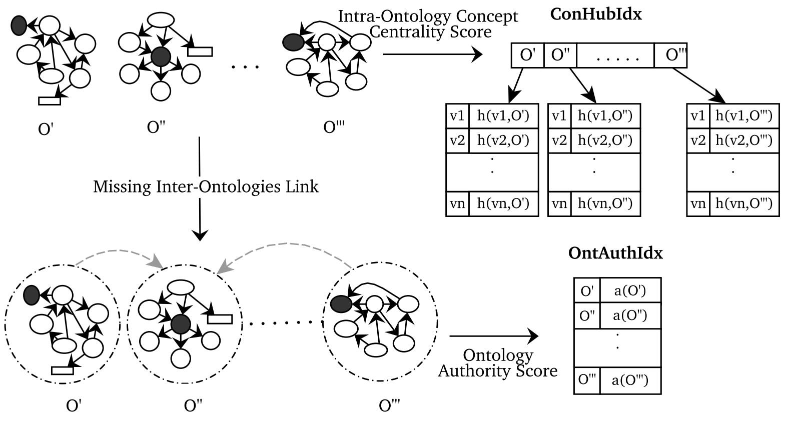


Fig. 2. Offline Ranking Phase

3.1 DWRank: A ***D***ual ***W***alk based ***Rank***ing 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 performed 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 then only on the whole ontology corpus considering an ontology as a node and inter-ontology relationships as links. Secondly, we differentiate the type of relationship (i.e. inter-ontology and intra-ontology) and the direction of the walk varies on the basis of the type of the relationship. Our Model *DualWalkRank* is named after its characteristic of a dual directional walk to compute the ranks of concepts.

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 relationships* 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 [15] where a node accepts scores from its referent nodes i.e. back link concepts. We adopt a Reverse-PageRank [9] 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 featured 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 dumping 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 $\acute{V(O)}$ 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. $\forall\ v_j \in V'$, L(v'_j) = L (v_i,v'_j). After incorporating *weak links* and *weak nodes* notions, 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 [19], but varies from it as we consider *weak nodes* and we are not considering relationships weights. 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 the ontology and $\sigma_h(O)$ is the standard deviation of hub scores of the concepts in the ontology.

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* starting from an authoritative ontology to the 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. The 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 formalise the *authority function* which computes the authoritativeness of O at the *kth* iteration.

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

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 *authority function* as follows:

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

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`.

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*. The function is described 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:

$$R_{(v,O)} = F_V(v, \mathtt{Q}) * [w_1 h(v,O) + w_2 a(O)]$$
$$F_V(v, \mathtt{Q}) = \sum_{q \in Q} f_{ss}(q, \phi(q_v)) \tag{6}$$

In Eq. 6, w_1 and w_2 are the weights for the *hub function* and the *authority function*. $F_V(v, \mathtt{Q})$ aggregates the contribution of all 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 $\mathtt{Q}$.

3.2 Index Construction: An execution of DWRank

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

ConHubIdx. 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 $\mathtt{O}$, (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.

OntAuthIdx. An index where each entry in the index maps an ontology to its *normalised authority score* $a_n(O)$ as shown in Fig.2 (buttom left). To construct the *OntAuthIdx* on the corpus $\mathtt{O}$, (1) the *authority function* is executed to get an *auth score* of all the ontologies in $\mathtt{O}$, (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.

Inter-Ontology Relationships Extraction. As we 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.
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. filled node) that appears in three different ontologies O', O'' and O'''. In the absence of explicit relationships, some implicit relationships exit 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, we use the

namespace of the *reused resource*. 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 other *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.

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 $o_i$
for $i \in [1, n]$ do
    $ns_{o_i} \leftarrow o_i$.topNS();
    $M_{ns}$.put($o_i$, $ns_{o_i}$);
for $r \in o_{i\in[1,n]} \wedge M_{ro}.contains(r) = false$ do
    while $\exists\ o_{j\in[1,n]}\ : r \in o_j \wedge o_i \neq o_j$ do
        $oList_r$.add($o_j$);
    if $oList_r.size() > 0$ then
        $oList_r$.add($o_i$);
        $M_{ro}$.put($oList_r$);
while $\exists\ r_{k\in[1,M_{ro}.size()]}$ do
    $ns_{r_k} \leftarrow r_k$.getNS();
    for $s \in [\ 1,\ oList_{r_k}.size()]$ do
        $ns_{o_s} \leftarrow M_{ns}$.get($o_s$);
        if $ns_{o_s} = ns_{r_k}$ then
            $o_k \leftarrow o_s$
            break;
    if $M_{oo}.contains(o_k)$ then
        $oList_{r_k}$.addAllDistinct($M_{oo}$.get($o_k$))
    $M_{oo}$.put($o_k$, $oList_{r_k}$)
return $M_{oo}$
```

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 namespace of most of the resources in the ontology. 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 the 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* 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 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.

Table 1. Top five reused ontologies based on 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 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

4 Online Query Processing

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 $\in$ `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}.

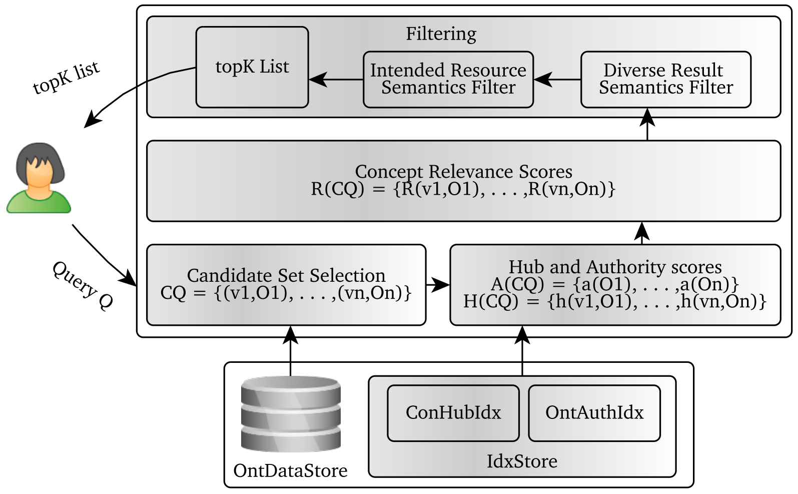
$SenSim(q, \phi(q_v))$ is a word similarity measure of a query keyword and a partially matched word in $L(v)$.

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.

Candidate Result Set Selection. A user 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) `rdfs:label` (b) `rdfs:comment` (c) `rdfs:description` property; or $\exists\, q \in Q \mid \phi(q_v)$ is part of *L(v)*.



topK list
Filtering
topK List
Intended Resource Semantics Filter
Diverse Result Semantics Filter
Concept Relevance Scores
R(CQ) = {R(v1,O1), . . . ,R(vn,On)}
Query Q
Candidate Set Selection
CQ = {(v1,O1), . . . ,(vn,On)}
Hub and Authority scores
A(CQ) = {a(O1), . . . ,a(On)}
H(CQ) = {h(v1,O1), . . . ,h(vn,On)}
ConHubIdx
OntAuthIdx
OntDataStore
IdxStore

Fig. 3. Online Query Processing

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 a 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 *L(v)* and *L(O)* where $v \in$ *V(O)*. Since for the *reused resources* there are multiple *hosting ontologies*, therefore *'v'* may have multiple entries in a candidate set if it is a *reused resource.*

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*. The two scores are combined according to the formulae of Eq. 6, that provides the final *concept relevance* of each v to the Query Q.

4.3 Filtering top-k results

In this section, we discuss the filtering strategies of our framework to enhance the semantic similarity of the results to the keyword query. We introduce two properties for the *top-k* results:

Diverse Results Semantic. Considering the semantics of a query allows us to remove repetitive results from the top-k results to increase the diversity in the result set. As mentioned earlier, if a candidate concept v is reused/extended in *'n' hosted ontologies* i.e. $\{O_1, O_2, ..., O_n\}$ then it may appear multiple times in a *candidate result set* (i.e. $C_Q = \{(v, O_1), (v, O_2),...,(v, O_n)\}$). In this case we remove the duplicates from the candidate result set.

Intended type Semantic. The semantic differentiates the *intended type* from the *context resource* of a concept. The label of a concept v may have multiple keywords as a description of the concept e.g., the label of a concept[5] in the GND ontology has

[5] `http://d-nb.info/standards/elementset/gnd#NameOfThePerson`

the keywords “Name of the Person”. Here “Name” is the intended type, whereas “Person” is the context resource. According to the *intended type semantic property* a concept should appear in the *top-k* if and only if its *intended type* matches to at-least one of the query keywords q $\in$ Q.

```
Algorithm 2: TOP-K FILTER
   Input: Concept Relevance Map $R(C_Q) = \{[(v_1,O_1),r_1], .. , [(v_n,O_n),r_n]\}$
   Output: top-k results $L(C_Q) = \{[(v_1,O_1),r_1], .. , [(v_k,O_k),r_k]\}$
1  $R_s(C_Q)$ /* A map to store intermediate results */
2  for $i \in [1,n]$ do
3      $e \leftarrow$ R($C_Q$).get(i);
4      if $R(C_Q).contains(e') \bigcap v(e)= v(e') \cap O(e) \neq O(e')$ then
5          $R_s(C_Q)$.put($[(v, O_h), r_h]$);
6          for $e''$ where $v(e'') = v$ and $O(e'') \neq O_h$ do
7              $R_s(C_Q)$.put($[(v, O''), (r'' - r_h)]$);
8          R($C_Q$).removeAll($e$ where concept is $v$);
9      else
10         $R_s(C_Q)$.put($e$);
11 $R_s(C_Q) \leftarrow$ sortByValue($R_s(C_Q)$);
12 while $(L(C_Q).size() \leq k) \bigcap (i \in [1,n])$ do
13     $e \leftarrow$ R($C_Q$).get(i);
14     if $\phi$ $(q_v(e))$ is a multi-keyword match then
15         if $I_t(\phi\ (q_v(e))) = q$ then
16             $L(C_Q)$.put(e);
17     else
18         $L(C_Q)$.put(e);
19 return $L(C_Q)$
```

Algorithm 2 explains the *top-k results filtering* process. It takes as input a *Concept Relevance Map* $R(C_Q)$ and returns the *top-k* results. First, the *diverse results semantics* are preserved (line 2-10) for $R(C_Q)$, and then the check for *intended type semantics* is applied (line 11-18) until the *top-k* results are retrieved.

A map $R_s(C_Q)$ is initialised to store the intermediate results that preserve the *diverse results semantics*. All candidate concepts in $R(C_Q)$ that appear only once in $R(C_Q)$ preserve the *diverse results semantics*, therefore they become part of $R_s(C_Q)$ (line 10). For all *reused concepts*, first the *home ontology* $O_h(v)$ of the concept v is identified. The entry e= $[(v,O),r] \in R(C_Q)$ for which ontology of the concept v is its home ontology (i.e. O=$O_h(v)$) becomes part of the $R_s(C_Q)$ (line 5). For all other entries e'' for v a new entry is created by subtracting the relevance score of e i.e. r_h from the r'' and add it to the $R_s(C_Q)$ (line 6-7). The process decreases the relevance score of duplicate entries by a factor of r_h. Then all such e'' from $R(C_Q)$ are removed since they have already been dealt with through candidate concepts of v.

The next step is to check the *intended type semantic*. For brevity, a detailed discussion of the intended type checking is exempted from Algorithm 2. The *ontology structure* and the *Information Retrieval method* are used to identify the *intended type*. For a concept v, its sub-classes, super-classes and inter-ontology relationships are extracted as the context of v. WS4J[6] API is used to calculate the similarity of different words in the concept v with its context. The word that has a higher similarity score in regards

[6] `https://code.google.com/p/ws4j/`

Table 3. DWRank Effectiveness

	Person	Name	Event	Title	Loc.	Addr.	Music	Org.	Author	Time
`AP@10`	0.9	0.7	1	0.7	0.7	0.8	0.7	0.9	0.8	0.8
`MAP@10`	0.98	0.82	1	0.88	0.86	0.94	0.80	0.85	0.78	0.74
`DCG@10`	37.58	19.11	35.12	12.45	24.88	23.53	14.82	33.70	18.24	22.53
`NDCG@10`	0.55	0.44	0.51	0.26	0.60	0.63	0.46	0.53	0.57	0.54

to the context is considered as the intended type of the concept. However, to reduce the cost of ensuring the *intended type semantic* for *top-k* results, the filter is applied until we retrieved the *top-k* results in the final results L. For this, first the $R_s(C_Q)$ is sorted in a decreasing order based on its relevance score r, so the more relevant results for query Q are at the top of the $R_s(C_Q)$ (line 11). Then the intended type of the candidate concept is checked only until 'k' concepts are selected from $R_s(C_Q)$ or there are no more results in $R(C_Q)$ (line 12). If the concept v has a single exact or partial matched word $\phi\ q_v(e)$ then by default it preserves the semantics and becomes part of $L(C_Q)$ (line 18), otherwise we check its *intended type*. If its intended type is equal to the query keyword $q \in Q$, the concept is included in $L(C_Q)$ otherwise, it is ignored.

5 Experimental Evaluation

In the following we present an experimental evaluation of our *relationship based top-k concept retrieval framework* on a benchmark suite [2]. We conducted two sets of experiments to evaluate: (1) the effectiveness of the *DWRank ranking model* presented in Sec. 3 and (2) the effectiveness of the additional filtering phase presented in Sec. 4.

5.1 Experimental Settings

To evaluate our approach we use a benchmark suite [2], that includes a collection of ontologies, a set of benchmark queries and a ground truth established by human experts. This collection is composed of 1011 ontologies and ten keyword queries. The benchmark evaluates eight state-of-the-art ranking algorithms on the task of ranking ontologies. We use the performance of these ranking models as the baseline to evaluate our approach. For a fair analysis, we implemented two versions of our approach: (1) the DWRank model (2) the DWRank model + Filters. The reasoning is that the filters can be applied to any of the evaluated ranking models, thus, in the second step we only evaluate the effectiveness of the *filters*. The effectiveness of the framework is measured in terms of its *Average Precision* (`AP`), *Mean Average Precision* (`MAP`), *Discounted Cumulative Gain* (`DCG`) and *Normalised Discounted Cumulative Gain* (`NDCG`).

5.2 Experimental Results

We next present our findings.

Effectiveness of DWRank. In the first set of experiments, we evaluated the effectiveness of DWRank in comparison with the eight baseline ranking models. We ran the ten sample queries on the ontology collection and retrieved the *top-k* results according to the proposed ranking model. We recorded the `AP@10`, the `MAP@10`, the `DCG@10` and the `NDCG@10`. The effectiveness measure results are shown in Table 3. Next, we compared our results with the baseline for the same dataset with the sample queries. 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 *unit of measure* and the y-axis is the *ranking model*. Each box on a graph presents the range of effectiveness measure for 10 sample queries according to the gold standard.

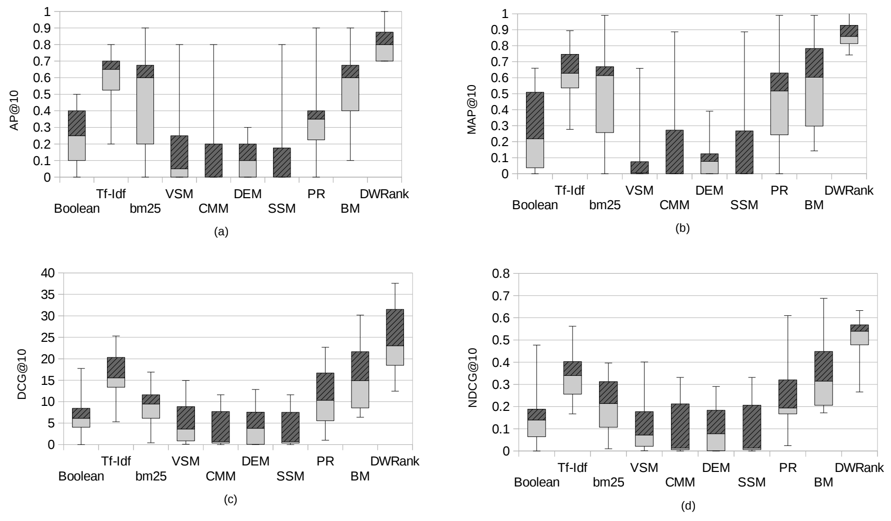


Fig. 4. Effectiveness of Ranking Model

Fig. 4 shows the maximum, minimum and average performance of DWRank in comparison to the best 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 the *location* and *music* query, the `AP@10` and `MAP@10` for DWRank is lower than the other best performing ranking model. However, the maximum average `MAP@10` for DWRank on ten queries is 0.84 that is greater than the average of Tf-Idf, the best baseline ranking models, (i.e., 0.63). The box plot also shows that `AP@10` and `MAP@10` of DWRank ranges from 0.7 ~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)* model, however, the average performance of DWRank is much higher than the average performance of BM.

Fig. 5 compares the `MAP@10` (resp. `NDCG@10`) for DWRank on all ten queries with the maximum `MAP@10` (resp. `NDCG@10`) achieved with any of the baseline ranking model on the sample queries. The result shows that DWRank outperforms the best other ranking model for `MAP@10` (resp. `NDCG@10`) for all but two queries. The experiment confirms our claim about the stable performance of the DWRank algorithm.

Effectiveness of DWRank+Filter. For the evaluation of the filter performance, we ran the ten sample queries of the benchmark collection with the DWRank model extended with the two filters proposed earlier, i.e. *diverse result semantics* and *intended type semantics*. Fig. 6 shows the effectiveness of DWRank compared to DWRank+filters. The average `AP@10` increased from 0.8 to 0.9, i.e. a 12 % increase in the effectiveness of results.

From the evaluation it is obvious that the *filter* improves the overall performance of our framework. A detailed analysis on the precision and recall of the *filter* is out of scope of this paper. However some *True positive* (`TP`), *False positive*(`FP`), *True negative* (`TN`) and *False negative* (`FN`) examples regarding our current implementation of the

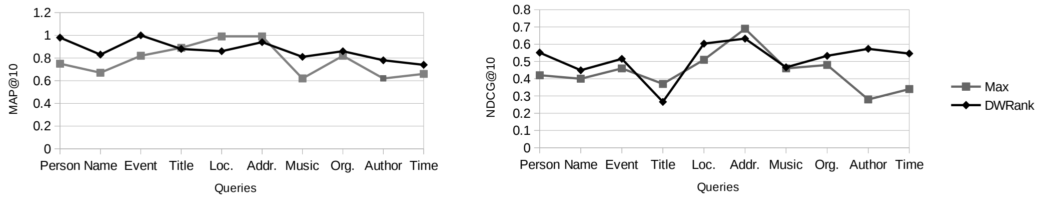


Fig. 5. *MAP@10* and *nDCG@10* for DWRank in comparison with the best value for any ranking model on sample queries

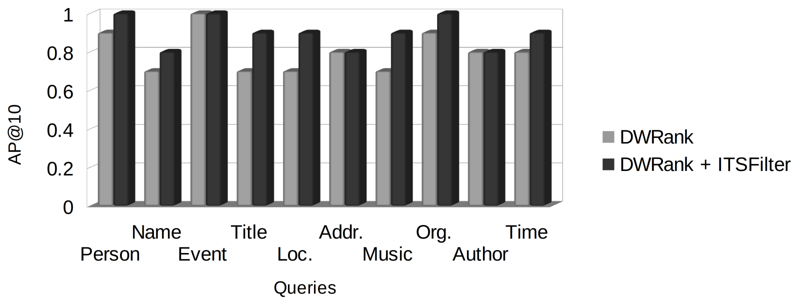


Fig. 6. Filter Effectiveness

intended type semantic filter are shown in Table 4. We analyse the top-10 results of DWRank without *intended type semantic filter* and then with the filter. For each query if there are `TN`, `FN`, `FP` examples we selected them or otherwise a random `TP` example.

Table 4. Intended Type Semantic Filter Performance in Relationship-based top-k Concept Retrieval Framework

Query term	Label of concept	Human Judgement	Intended Filter Judgement
person	personal communication model	×	×
name	gene name	×	×
event	academic event	✓	✓
title	spectrum title	×	×
location	hematopoiesis location trait	×	×
address	E45_address	✓	×
music	sound and music computing	×	×
organization	3D structural organization datrum	×	×
author	author list	✓	×
time	time series observation	×	✓

6 Related Work

The Linked Open Vocabularies (LOV) search engine[7], 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 [18].

[7] `http://lov.okfn.org`

There are also some ontology libraries available that facilitate the locating and retrieving of potentially relevant ontology resources [13]. Some of these libraries are domain-specific such as the Open Biological and Biomedical Ontologies library[8] or the BioPortal [14], whereas others are more general such as OntoSearch [16] or the TONES Ontology Repository[9]. However, as discussed by Noy & d'Aquin [13] only few libraries support a keyword search, only one (Cupboard [4]) supports a ranking of ontologies based on a user 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 [6] (which was initially developed to rank ontologies only), Sindice.com [17], Watson [5], or Yars2 [10] do allow a search of ontology resources through a user query. The ranking in these search engines follows traditional link-based ranking methods [12], in particular adapted versions of the PageRank algorithm [15], where links from one source of information 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 [11]. AKTiveRank [1], ranks ontologies based on how well they cover specified search terms. Falcon [3] is a popularity-based scheme to rank concepts and ontologies. Other strategies, mainly based on methods proposed in the information retrieval community, are employed in Semantic Search [8], 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 [7, 1]. Another related approach is presented in [19] identifies the most important concepts and relationships from a given ontology, however the approach does not support ranking concept that belongs 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 ranking and index construction phase and an online query and evaluation phase. In the offline ranking 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. The online ranking phase filters the top-k ranked list of concepts by removing redundant results and by determining the intended type of the query term and removing concept types that are not closely related to the intended query type. We evaluated our DWRank algorithm without the online query processing filters against state-of-the-art ranking models on a benchmark ontology collection and also evaluated the added performance of the proposed filters. The evaluation shows that DWRank outperforms the best performing ranking algorithm for most queries while exhibiting a much stabler performance (i.e. MAP@10 of 0.84) than the average of the best performing ranking models of the benchmark (i.e. MAP@10 of 0.63). The filters proposed in the online ranking phase further increased the average AP@10 by 12%. Although our algorithm shows significantly improved performance compared to the state-of-the-art in ontology ranking models, we believe further improvements are possible through learning the weights in computing the authority and the hub score using linear classification model. Also, in the online query processing phase we could pre-compute indices for the diverse result semantics and intended type semantics to increase the performance of the online query.

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

References

1. H. Alani, C. Brewster, and N. Shadbolt. Ranking Ontologies with AKTiveRank. In *Proceedings of the International Semantic Web Conference (ISWC)*, pages 5–9. Springer-Verlag, 2006.
2. A. S. Butt, A. Haller, and L. Xie. Ontology search: An empirical evaluation. In *Proceedings of the International Semantic Web Conference*, Riva del Gara, Italy, 2014. [TO APPEAR].
3. 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.
4. M. d'Aquin and H. Lewen. Cupboard — A Place to Expose Your Ontologies to Applications and the Community. In *Proceedings of the 6th European Semantic Web Conference*, pages 913–918, Berlin, Heidelberg, 2009. Springer-Verlag.
5. M. d'Aquin and E. Motta. Watson, More Than a Semantic Web Search Engine. *Semantic Web*, 2(1):55–63, 2011.
6. 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 13th ACM International Conference on Information and Knowledge Management*, pages 652–659, New York, NY, USA, 2004. ACM.
7. L. Ding, R. Pan, T. Finin, A. Joshi, Y. Peng, and P. Kolari. Finding and ranking knowledge on the semantic web. In *Proceedings of the International Semantic Web Conference*, volume 3729 of *Lecture Notes in Computer Science*, pages 156–170, 2005.
8. M. Fernandez, V. Lopez, M. Sabou, V. Uren, D. Vallet, E. Motta, and P. Castells. Semantic Search Meets the Web. In *Proceedings of the 2008 IEEE International Conference on Semantic Computing*, pages 253–260, Washington, DC, USA, 2008. IEEE Computer Society.
9. D. Fogaras. Where to start browsing the web? In *Innovative Internet Community Systems*, pages 65–79. Springer, 2003.
10. 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 6th International Semantic Web Conference*, ISWC'07/ASWC'07, pages 211–224, Berlin, Heidelberg, 2007. Springer-Verlag.
11. 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.
12. 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.
13. N. F. Noy and M. d'Aquin. Where to Publish and Find Ontologies? A Survey of Ontology Libraries. *Web Semantics: Science, Services and Agents on the World Wide Web*, 11(0), 2012.
14. 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, and M. A. Musen. BioPortal: ontologies and integrated data resources at the click of a mouse. *Nucleic Acids Research*, 2009.
15. L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the Web. In *Proceedings of the 7th International World Wide Web Conference*, pages 161–172, Brisbane, Australia, 1998.
16. 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*. CEUR-WS.org, 2007.
17. G. Tummarello, R. Delbru, and E. Oren. Sindice.Com: Weaving the Open Linked Data. In *Proceedings of the 6th International The Semantic Web Conference*, pages 552–565, Berlin, Heidelberg, 2007. Springer-Verlag.
18. P.-Y. Vandenbussche and B. Vatant. Linked Open Vocabularies. *ERCIM news*, 96:21–22, 2014.
19. 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. Springer, 2008.