Applied Ontology 11 (2016) 301–324
IOS Press

RecoOn: Ontology Recommendation for Structureless Queries

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

Abstract. Ontology search is becoming increasingly important as the number of available ontologies on the Web steadily increases. Ontology recommendation is done by analyzing various properties of ontologies, such as syntax, structure and usage, in order to find and recommend high quality matches for a user defined query. Only a few ontology libraries and search engines facilitate this task for a user who is looking for an ontology that models all or some of the concepts she is looking for. In this paper, we introduce $RecoOn$, a framework that helps users in finding the best matching ontologies to a multi-keyword query. Our approach recommends a ranked list of relevant ontologies using metrics that include the matching cost of a user query to an ontology, an ontology's informativeness and its popularity. Based on these metrics two versions of $RecoOn$ are implemented: $RecoOn_{ln}$, where the metrics are combined in a linear model to find the relevance score of an ontology to a query, and $RecoOn_{opt}$ that formalizes ontology recommendation as an optimization problem to recommend ontologies to the user that are as informative and popular as possible while incurring the least matching costs. We compare both versions of $RecoOn$ with the state-of-the-art approach in ontology ranking by conducting a user study over the CBRBench ontology collection. Our experimental results show that both versions of the proposed approach are promising: they identify high-quality matches for keyword queries over real-life ontologies, and outperform the state-of-the-art ranking method significantly in terms of effectiveness, while $RecoOn_{opt}$ is more effective than $RecoOn_{ln}$. We further test the scalability of our proposed approach; and results show $RecoOn_{opt}$ is more efficient than $RecoOn_{ln}$.

Keywords: Ontology Search, Ontology Recommendation, Ontology Ranking

1. Introduction

Ontologies are a shared conceptualization of knowledge in a specific domain of discourse. However, only if an ontology is reused and thus its conceptualization validated by others it becomes truly a shared conceptualization. The process of reusing existing ontologies is also cost-effective and produces high-quality conceptualization, because referring to an established ontological term in another domain of discourse builds an interlinked model of conceptualizations with strong formal semantics. It also facilitates data interoperability on both the syntactic and the semantic level. The growth of available ontologies in vertical domains such as bioinformatics, e-commerce and the internet-of-things highlights an increasing need of ontology search, which is the process of finding ontologies for users' defined queries from an ontology collection. In order to find the relevant ontologies various terms within those ontology, such as classes and properties, are searched and matched to the queries. However, it is often difficult to find the right ontology for a given use case. A user may not know the exact classes or properties and their positions in an ontology (ontological structure) she wants, but requires that the ontology contains a set of resources as its constituents. To mitigate the problem, a schema-less and structure-less keyword-based query is commonly used for ontology search. The problem here is that it is still hard to choose between ontologies that match to such a keyword query. Further, if there is no exact match for a given query string, a subset of the query may be used in order to find ontologies of interest. However, considering ontology matches for all subsets of the query terms results in a large number of matches. Consequently, it is often too time consuming for a user to explore the matched ontologies to find the most suitable one.

*Corresponding author: Anila Sahar Butt, anila.butt@anu.edu.au

Some previous work (Alani, Brewster, & Shadbolt, 2006; Noy et al., 2009; Noy & d'Aquin, 2012) has tackled the problem of finding and recommending ontologies. More recently, a dedicated ontology search engine has emerged (Vandenbussche & Vatant, 2014). Some of the search engines (e.g. (Ding et al., 2004)) adopt document-ranking algorithms to introduce ranking to their search results; most consider the popularity of terms in the ontology corpus. For this they often use the PageRank algorithm as the ranking factor, which although effective in some cases, as (Butt, Haller, & Xie, 2014a) showed, hinders the visibility of newly emerging, but well defined ontologies. Moreover, most of the ontology search systems retrieve ontological terms (concepts and relations) and only a few provide ontology search based on a keyword query. Only a few ontology libraries and search engines facilitate the task of ontology retrieval for a user who is looking for an ontology that models all or some of the concepts she is looking for. The National Center for Biomedical Ontology (NCBO) proposed a biomedical ontology recommender web service (Jonquet, Musen, & Shah, 2010) that is one of the most prominent approaches to find an ontology based on the text description. It is also a domain dependent ontology library and does not deal with all types of ontologies. A general solution is required for ontology search based on text descriptions or at least a multi-term query string.

`RecoOn`[1] , an Ontology Recommendation approach, is an effort towards a dedicated ontology search engine that recommends relevant ontologies in response to a multi-term query string. Given a keyword query Q and a partial match approach, one might find many matches of Q in an ontology corpus. Thus, a user friendly ontology search engine must address the following two questions: (1) how to determine which match is better, and (2) how to identify the top k matches? We propose an ontology recommendation approach that first finds the matched (relevant) ontology set to a query string; and then identifies the up to k most relevant ones. To identify the k most relevant ontologies for a query string, three measures are computed for each ontology: *matching cost* - the syntax and structural difference of the ontology from the query, *informativeness* - the information an ontology contains about the concepts that match the query string, and *popularity* - the popularity of the ontology in the ontology corpus. We then find the relevance of an ontology to the query by formulating and solving ontology recommendation as a linear model, referred to as $RecoOn_{ln}$, and as an optimization problem referred to as $RecoOn_{opt}$. The aim is to find the ontologies that are as informative and popular as possible while incurring the least matching costs. The approach is evaluated on the CBRBench dataset (Butt et al., 2014a) against $AKTiveRank$ by conducting a user study. The results of our user study show that $RecoOn_{opt}$ and $RecoOn_{ln}$ outperforms the baseline state-of-the-art algorithm $AKTiveRank$; and $RecoOn_{opt}$ is efficient as well as effective as compared to $RecoOn_{ln}$ on CBRBench ontology collection and sample queries designed in this work.

The remainder of the paper is structured as follows. Sec. 2 discusses related work. Sec. 3 gives some preliminaries and outlines the $RecoOn$ workflow. Sec. 4 describes the ontology recommendation approach. Sec. 5 reports on the evaluation results and Sec. 6 concludes the paper.

2. Related Work

The *Linked open Vocabularies (LOV)*[2] system contains domain independent ontologies retrieved from the datasets in the Linked Data Cloud; and facilitates ontology search for a given query term. For term search, it uses a ranking algorithm based on the term popularity in Linked Open Data (LOD) and in the LOV ecosystem (Vandenbussche & Vatant, 2014). However, the vocabulary search results are unordered. *WebCORE* (Cantador, Fernández, & Castells, 2007) recommends the most appropriate ontologies for a given domain. The tool allows a user to refine and enlarge query terms using WordNet, and then automatically recommends ontologies based on a query terms' frequency in the matched ontology and the knowledge base. WebCORE modifies the vector space model to compute the similarity between the query vector and ontology vector. Moreover, it considers the manual user evaluations in order to incorporate a

[1] A demo based on CBRBench ontology collection is available at `www.activeraul.org/RecoOn/`, and the code is available at `https://github.com/anilabutt/RecoOn`

[2] http://lov.okfn.org

human, collaborative assessment of ontologies. However, (Butt et al., 2014a) shows that the vector space model is not an optimal ontology ranking model, and we are not considering manual ontology evaluation in this work. *Swoogle* (Ding et al., 2004) is a crawler-based indexing and retrieval system for the Semantic Web. It supports keyword-based search over RDF documents and orders results (i.e, ontologies, properties, or classes) according to their popularity. It adapts PageRank (Page, Brin, Motwani, & Winograd, 1998) as the method to compute popularity of ontologies or its terms. *WATSON* (d'Aquin & Motta, 2011) collects and indexes available semantic content on the Web (both ontologies and entities). It computes some quality measures (e.g., structural measures, topic relevance,etc.) for ontologies and their terms and made them available to the user or application to design their own ranking scheme according to the application requirement. *OntoKhoj* (Patel, Supekar, Lee, & Park, 2003), a crawler-based semantic Web portal, indexes the crawled ontologies in a local repository. A *PageRank* (Page et al., 1998) based ranking model is adapted to rank the large number of ontologies within each category. Sindice (Tummarello, Delbru, & Oren, 2007) is a registry and lookup service for Semantic Web data. In the case of a literal search, resources are ranked according to their text relevancy with the search terms. For a resource search, search results with common host-names to the search resource are ranked higher than the other results. These ranking techniques were already deemed as error-prone in conventional Web search engines.

Several other ranking techniques are proposed in the literature and tested on an ontology collection, but are not available online. One such technique, *AKTiveRank* (Alani et al., 2006) is used in our evaluation as the baseline. The approach uses keyword/s to find the relevant set of ontologies from a semantic Web search engine (i.e. Swoogle (Ding et al., 2004)) and then applies four ranking models (Freeman, 1977; Rada, Mili, Bicknell, & Blettner, 1989; Spanoudakis & Constantopoulos, 1993) that were originally proposed for *graph retrieval* or *document retrieval*, to the task of ranking ontologies. The final rank of an ontology is calculated using fixed weights for each of the four measures. For each query, the weights of all four measures are required to be computed individually to get the maximum performance. However, that is practically impossible to do without applying some other techniques. (Alani, Noy, Shah, Shadbolt, & Musen, 2007) presents a context-based ontology search approach which finds the most relevant terms, in a given domain, using Wikipedia and outputs results based on the coverage of these terms in an ontology. The approach addresses the problem of ontology search, but not ontology ranking, and thus considers all returned ontologies equally relevant for a query. OntoQA (Tartir & Arpinar, 2007) evaluates ontologies related to a certain set of terms and rank them according to a set of metrics. The evaluation of an ontology is made on two dimensions: Schema and Instances. The final relevance score of a candidate ontology is the weighted average of the schema and instance metrics; weights are set based on empirical testing. However, most ontologies in the wild do not have instances which would result in a lower relevance score for those ontologies. (Sabou, Lopez, & Motta, 2006) presents an approach that considers a query text, retrieves triples out of the query text, expands query terms by considering their synonyms and hyponyms, identifies the matching ontologies and ranks them for the given query text. The ranking model relies on the generality deviation of a matching ontology from the query triples. The ranking quality heavily depends upon the query expansion and triple extraction processes. *CARRank* (Wu, Li, Feng, & Wang, 2008) proposed to rank ontology terms (i.e. class and properties), but not ontologies themselves. The approach finds the important classes and properties within an ontology.

There are some ontology recommendation approaches that are specifically designed for biomedical ontologies. For instance, BiOSS (Martínez-Romero, Vázquez-Naya, Pereira, & Pazos, 2014), a system for the selection of biomedical ontologies, outputs single or combined matching ontologies for a user specified keywords. BiOSS proposes three evaluation metrics to rank matching ontologies: 'domain coverage', 'semantic richness', and 'popularity'. The output matching ontologies are ordered according to the aggregated scores combined from these metrics. Similarly, The National Center for Biomedical Ontology (NCBO) proposed a biomedical ontology recommender web service (Jonquet et al., 2010) to suggest the most appropriate ontologies required to annotate a given biomedical text data. The recommender service recognizes relevant concepts from an ontology repository to annotate the given text data, and then expands the first set of annotations using the UMLS meta-thesaurus and the NCBO ontologies. The relevance of an ontology is computed based on the context and matching terms in that ontology; that mainly depends

on the accuracy of the NCBO annotator. The methods proposed by such techniques use some common schema or meta-data to evaluate the ontologies. For example, the former uses the UMLS meta-thesaurus, PubMed and BioPortal to compute semantic richness and popularity, and the latter uses UMLS meta-thesaurus and NCBO ontology. However, our approach works without the use of a common schema or meta-data.

In RecoOn, three evaluation metrics are considered i.e., `Informativeness`, `Popularity` and `Coverage` (referred to as 'matching cost'); and these three metrics are then combined using an optimization problem to compute a final relevance score for each ontology to a user query. Each of these three metrics have only been considered individually as a ranking metric previously.

Informativeness is considered by entity ranking approaches (Cheng, Tran, & Qu, 2011; Meymandpour & Davis, 2013). They adopt Shannon entropy as an informative measure; according to which the least frequent property or concept with fewer instances is more informative. RecoOn considers connectivity of a concept as the measure of informativeness as discussed in Sec. 4.2.2. The relation frequency is taken into consideration for this purpose. AKTiveRank (Alani et al., 2006) has considered relation frequency as a density measure where four different types of relationships of a concept are identified and each type is assigned a fixed weight. The final density measure for an ontology is an average density of all its matching concepts. However, in contrast to RecoOn the informativeness is not query-dependent.

Popularity is mostly calculated using PageRank (Patel et al., 2003; Ding et al., 2004), with the difference to RecoOn being mostly around what type of relationships are used for the calculation of the PageRank score. OntoKhoj (Patel et al., 2003) assigns fixed weights to three different types of links (owl:imports, rdfs:subClassOf and rdfs:domain/rdfs:range) from one ontology to another ontology and considers the edge (i.e. link) weights in the PageRank execution. Swoogle (Ding et al., 2004) identifies five different types of link categories and assigns different weights to each category to model their probability of being explored. In RecoOn we consider all types of links from one ontology to another as a positive vote for the referred ontology.

Coverage is mostly measured in terms of presence of query keywords in an ontology. Some approaches prefer exact matches (Vandenbussche & Vatant, 2014; d'Aquin & Motta, 2011) and others partial matches (Butt, Haller, & Xie, 2016; Sabou et al., 2006; Cantador et al., 2007). Such techniques either enhance query terms using a thesaurus (e.g. Wordnet) by considering synonyms and antonyms (Sabou et al., 2006; Cantador et al., 2007) or assign a fixed weight to each partial match e.g., (Alani et al., 2006) considers a 0.5 weight for each partially matched concept. RecoOn, considers Jaccard distance as a label matching cost. It also considers a structural matching cost to prefer the ontologies that contain matched concepts in close vicinity.

Also, we consider a combination of the aforementioned three metrics for ontology evaluation in a novel way. Most of the existing approaches use either one of the three measures or assign fixed weights to combine more than one evaluation metrics. (Butt et al., 2014a) shows that none of the commonly used evaluation metrics performs adequately. Moreover, for optimal performance of an algorithm, the metrics' weights need to be reset for each user query (Alani et al., 2006), which is not a practical solution.

3. RecoOn: Ontology Recommendation

In the following we first define the terms used throughout the paper, and then give a brief overview of the ontology recommendation workflow.

3.1. Preliminaries

Fig. 1 and Fig. 3 introduce a motivating example ontology and query that are used throughout this paper.

An ontology here refers to a labelled directed graph based formalisation $o = (C, R, L)$ of a domain knowledge. *C(o)* is a finite set of nodes where $c \in C(o)$ denotes a domain concept in 'o' e.g., `'Publication'` or `'Conference'`. *R(o)* is the set of edges where $r(c_i, c_j) \in R(o)$ denotes a re-

lationship between c_i and c_j e.g., `authorOf(Publication,Person)`. L is a labelling function which assigns a label $L(c)$ (resp. $L(r)$ or $L(o)$) to node c (resp. an edge $r \in R(o)$, or the ontology *'o'*). In practice, the labelling function L may specify (1) the node labels to relate the node to the referred concept, e.g. `'Person'`, `'Publication'` and `'Author'`; and (2) the edge labels as explicit relationships between/of concepts e.g., `'publicationOf'`, and `'title'` or implicit relationships e.g., `'subClassOf'` and `'superClassOf'`, and (3) the ontology label to relate the ontology to the domain or some identity.

Function	Output
l_{Person}	Person@en
$l_{authorOf}$	authorOf@en
$domain(authorOf)$	Person
$range(authorOf)$	Publication
$super(Author)$	Person
$sub(Publication)$	ConferencePaper, JournalPaper

Table 1: Functions with outputs

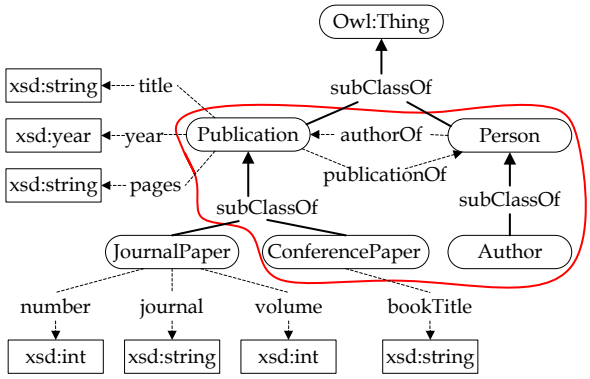


Fig. 1.: An Example Ontology

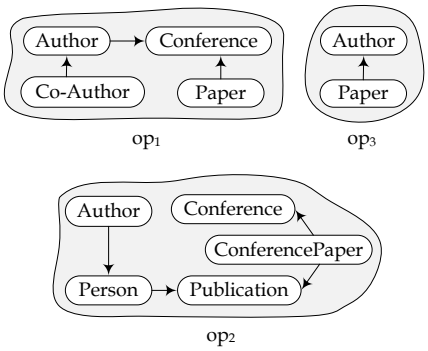


Fig. 2.: Match Patterns for 'Q'

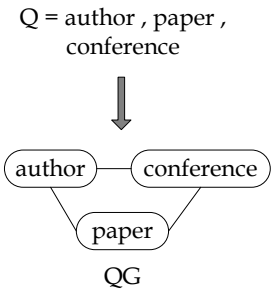


Fig. 3.: Query 'Q' and Query Graph 'QG'

Based on the description above we define the following functions:

- $l_c : C \rightarrow L(C)$ returns the label of concept 'c'
- $l_r : R \rightarrow L(R)$ returns the label of relation 'r'
- $domain(r) : R \rightarrow C$ returns the source concept/s of relation 'r'
- $range(r)$: $R \rightarrow C$ returns the target concept/s of relation 'r'
- $super(c)$: $C \rightarrow C$ returns the immediate super concept/s of concept 'c'
- $sub(c)$: $C \rightarrow C$ returns the immediate sub concept/s of concept 'c'

Table 1 presents the results of the execution of these functions on the example ontology shown in Fig. 1. Moreover, we define some terms on graph 'g', applicable to all graph based formalizations (e.g., ontology and query graph) used throughout this paper, while Table 2 summarizes the notations used for these terms.

Definition 1 : Concept Subsumption *($\subset_C$) . A concept set $C_1(g)$ is subset of another set $C_2(g)$ if every node (i.e., concept) in $C_1(g)$ is in $C_2(g)$. $C_1(g)$ may have exactly the same nodes (i.e., concepts) as $C_1(g)$.*

$$C_1(g) \subset_C C_2(g) \text{ iff } \forall\, c,\ c \in C_1(g) \rightarrow c \in C_2(g).$$

Table 2
Notations used throughout the paper

Variable	Description
o	An ontology in Ontology Collection 'O'
op_i	Ontology pattern in $o_i : C(op_i) \subset_C C(o_i)$ and $R(op_i) \subset_R R(o_i)$
Q	Query String
q_i	Single query term in Q (i.e. $q_i \in$ Q)
QG	Query Graph w.r.t. Q
c_q	A query concept in Query Graph: $c_q \in C(QG)$
Q_{mat}	Match of QG in o
$\phi(c_q)$	$c \in C(Q_{mat}) : c \approx c_q$
$C_{Q_{mat}}$	Concept Match Set: $C_{Q_{mat}} = \{\phi_i(c_q) : \phi_i(c_q) \in C(Q_{mat})$ and $c_q \in C(QG)\}$

In Fig. 1, if $C_1(o)$ is the set of all concepts in the ontology i.e. $C_1(o)$ = {$Author$, $Person$, $Publication$, $JournalPaper$, $ConferencePaper$} and $C_2(o)$ is the set of some concepts, i.e. $C_2(o)$ = {$Author$, $Publication$ } then $C_2(o) \subset_C C_1(o)$.

Definition 2 : Relation Subsumption *($\subset_R$) . A relation set $R_1(g)$ is subset of another set $R_2(g)$ if every edge (i.e., relation) in $R_1(g)$ is in $R_2(g)$. $R_1(g)$ may have exactly the same edges (i.e., relations) as $R_1(g)$.*

$$R_1(g) \subset_R R_2(g) \text{ iff } \forall\ r,\ r \in R_1(g) \rightarrow r \in R_2(g).$$

Considering Fig. 1, if $R_1(o)$ is the set of some relations or edges in the ontology e.g., $R_1(o)$ = {$authorOf$, $publicationOf$, $title$, $journal$, $bookTitle$} and $R_2(o)$ is the set of object relations only i.e. $R_2(o)$ = {$authorOf$, $publicationOf$} then $R_2(o) \subset_R R_1(o)$.

Definition 3 : Ontology Pattern *(op). An ontology pattern op in an ontology 'o' is a directed labelled graph, comprising of nodes and edges (i.e. concepts and relations), where*

$$C(op) \subset_C C(o) \text{ and } R(op) \subset_R R(o) \text{ and } \forall\ \mathrm{r}(c_i, c_j),\ \mathrm{r}(c_i, c_j) \in R(op) \rightarrow \mathrm{r}(c_i, c_j) \in R(o).$$

The example ontology in Fig. 1 comprises of a set of concepts $C(o)$ = {$Author$, $ConferencePaper$, $JournalPaper$, $Person$, $Publication$ } and a set of relations $R(o)$ = { $authorOf$, $bookTitle$, $journal$, $number$, $page$, $publicationOf$, $subClassOf$, $title$, $volume$, $year$ }. The part of ontology marked as red in Fig. 1 (referred to as o_{red}) is composed of a set of concepts $C(o_{red})$ = {$Author$, $ConferencePaper$, $Person$, $Publication$} and a set of relations $R(o_{red})$ = { $authorOf$, $publicationOf$, $subClassOf$ }. Since $C(o_{red}) \subset_C C(o)$, $R(o_{red}) \subset_R R(o)C$ and for all relations $r \in R(op)$ (i.e. $authorOf(Person, Publication)$, $publicationOf(Publication, Person)$, $subClassOf(Author, Person)$, $subClassOf(ConferencePaper, Publication)$) $r \in R(o)$, according to the definition 3, o_{red} is an *op* in o.

Definition 4 : Query Graph *(QG) . Given a query string Q = {$q_1, q_2, ..., q_n$} containing 'n' query terms, a query graph 'QG' w.r.t. Q is defined as an unlabelled undirected graph where each query term is mapped to a node, and each node has one or more unlabelled and undirected edges that connect this node to every other node in the graph. C(QG) is a set of nodes or concepts where $c_q \in C(QG)$ denotes a concept (query term) in 'QG'. R(QG) is the set of edges where (c_i, c_j) ∈ R(QG) denotes a relationship between c_i and c_j.*

An example three-term query string Q = 'author conference paper', and a possible query graph 'QG' for 'Q' is shown in Fig. 3, where each query term $q \in Q$ is mapped to a query graph concept c_q and nodes are connected to each other.

Definition 5 : Query Match (Q_{mat}) . *A query match Q^j_{mat} in an ontology 'o_j' is a set of ontology patterns 'op' in 'o' such that for a $C_i(QG)$ that is a subset of $C(QG)$, $C_i(QG)$ is also a subset of C(op).*

$$Q_{mat} = \{\ op : C_i(QG) \subset_C C(op) \text{ and } C_i(QG) \subset_C C(QG)\ \}$$

According to the definition, a query match is a partial match for a query graph. Consider an example query Q = 'author paper conference', a query graph QG for this query consists of three nodes and therefore $C(QG)$= $\{Author, Paper, Conference\}$ as shown in Fig. 3. Fig. 2 shows example ontology patterns op_1, op_2 and op_3 in ontologies o_1, o_2 and o_3, respectively. Since, a subset of $C(QG)$ is a subset of each of $C(op_1)$, $C(op_2)$ and $C(op_3)$ therefore op_1, op_2 and op_3 are query matches in o_1, o_2 and o_3 for QG.

Due to the adaptation of a partial match approach in RecoOn, a query match Q_{mat} may introduce some additional concepts or relations, or drop a few existing concepts and relations compared to the concepts and relations contained in the query graph QG itself as shown in matches 'op_1', 'op_2' and 'op_3' of Fig. 2. The newly introduced concepts in Q_{mat} may or may not match a query node. For instance, the ontology pattern op_2 of Fig. 2, introduces two new concepts 'Person' and 'Publication' that do not exist in the query graph. Similarly 'op_3' lacks the query concept 'conference'.

Definition 6 : Concept Match Set ($C_{Q_{mat}}$) . *If a concept $c \in C(Q_{mat})$ i.e. concept set of query match in 'o', is a match for at least one of the query nodes $c_q \in C(QG)$ (i.e. $c \in C(Q_{mat}) \approx c_q \in C(QG)$), then the concept c is called a* ***match concept*** *$\phi(c_q)$ of c_q. The set of all matched nodes of a query graph match is the* ***concept match set*** *$C_{Q_{mat}}$ of the query graph match i.e.,*

$$C_{Q_{mat}} = \{c : c \in C(Q_{mat}) \wedge c = \phi_i(c_q) \wedge c_q \in C(QG)\}$$

For instance, for a query 'q' and query graph 'QG', as shown in Fig. 3, a query match Q_{mat} is 'op_2' (i.e. ontology pattern in 'o_2') as shown in Fig. 2, the match for a query concept 'author', 'paper' and 'conference' is $\phi(author)$ = 'Author', $\phi(paper)$ = 'ConferencePaper', and $\phi(conference)$ = 'ConferencePaper' respectively. Therefore, the concept match set $C_{Q_{mat}}$ for Q_{mat} is {'Author', 'ConferencePaper'}.

3.2. RecoOn Workflow

RecoOn is implemented as a Java web application that uses Virtuoso as an ontology repository. Fig. 4 shows the overall execution flow of RecoOn. Starting from the input, there are four components that participate in the ontology recommendation task. Here, a step-by-step explanation of how different RecoOn components participate in the recommendation task is given.

Query preprocessing. This component takes a query string as an input and extracts keywords from the string by stemming and removing stop words. A query graph is generated from the extracted keywords, where each keyword is matched to a node and each node is connected to each other node in a query graph. If a keyword appears several times in the query string, only one corresponding node is created for that keyword in the query graph. A query graph considers each keyword as a single word, however we look for them as compound words in their matches in the ontologies (cf. Sec. 4.2.1). The query graph is then used to find the appropriate ontology matches for the query.

Ontology Retrieval. This component considers a query graph and finds candidate ontologies for the query graph as discussed in Sec. 4.1. RecoOn dynamically maps a query graph to a SPARQL query and retrieves the query matches. RecoOn is implemented and tested for the English language only, therefore the SPARQL query is defined to lookup English labels of concepts only (i.e. labels that are defined with an $@en$ tag). The output of this component is the ontology match set, that is passed on to the next component for the ontology evaluation.

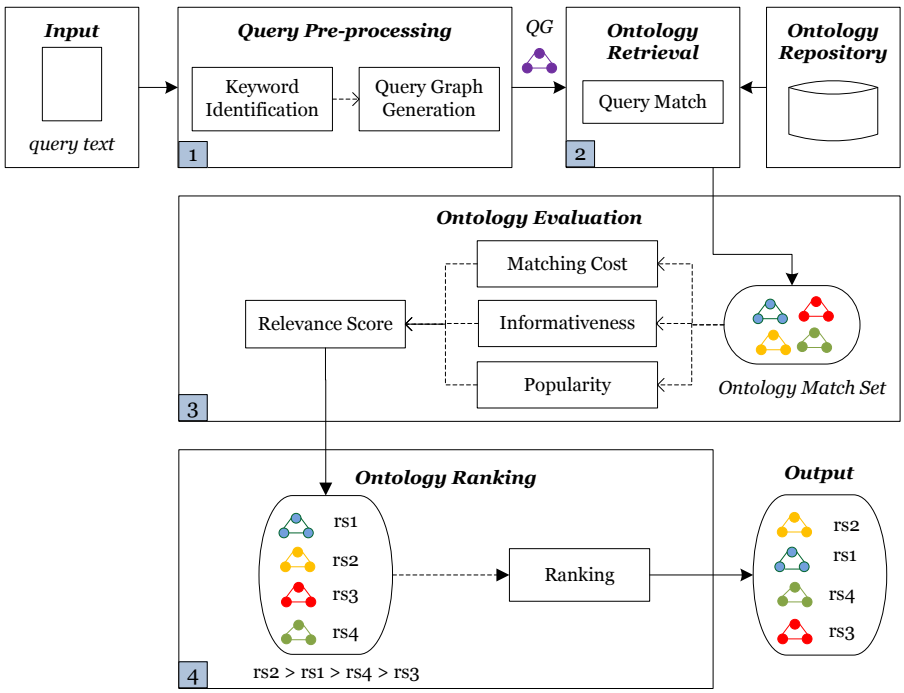


Fig. 4. RecoOn Workflow

Ontology Evaluation. This component preprocesses query matches before evaluating the ontology match set. The labels from each query match are considered and for each label the language tag (i.e. , '$@en$') is removed. A labelSplit() function is used to split the label based on capital letters to retrieve all words from the label, and each word obtained from the label is then stemmed. For instance, '$ConferencePapers@en$' results, after removing the language tag, splitting the label and stemming in '$Conference\ Paper$'. After this preprocessing the matching cost (cf. Sec. 4.2.1), informativeness (cf. Sec. 4.2.2), and popularity (cf. Sec. 4.2.3) for each ontology in the ontology match set are computed. Finally the relevance score for each ontology to the user query is determined as shown in Sec. 4.2.4.

Ontology Ranking. This component orders the matching ontologies in order of their relevance score to the user query and outputs a ranked list of matched ontologies and their concepts.

4. Ontology Recommendation

Based on the functions and definitions above, we now explain our ontology recommendation model. Given a query string Q and an ontology collection O, the purpose is to find O_{MAT} - a set of matching ontologies to the query string Q in O, and recommend up to k ontologies to the user to help her finding the right ontology. To achieve that, the ontologies that are relevant to the query are first retrieved and then k ontologies are selected based on their matching cost, informativeness and popularity to the query string.

4.1. Ontology Retrieval

Given an ontology collection O and a query string Q, to characterise the match of an ontology 'o' to Q, we define the candidacy of 'o' w.r.t. Q as

$$cand(o, Q) = \begin{cases} true & \text{if } \exists\ Q_{mat} \text{ in 'o'} \\ false & \text{otherwise} \end{cases} \quad (1)$$

which is either 'true' (a candidate ontology) or 'false' (not a candidate ontology). As mentioned in Eq. 1, an ontology 'o' is a candidate ontology for a query Q, if it contains at least one match Q_{mat}.

Example: For instance, Fig. 3 shows a query graph for a three keyword query string i.e., 'paper author conference'. The example ontology shown in Fig. 1 contains a query match Q_{mat} (ontology pattern marked as red in the figure) for Q, therefore the example ontology is a candidate ontology for Q.

A set of all match ontologies in an ontology corpus O for a query Q is referred to as the ***ontology match set*** *(O_{MAT}). i.e.,*

$$O_{MAT} = \{o_i : cand(o_i, Q) \equiv true\} \quad (2)$$

4.2. Recommending k Ontologies

Among all the ontologies in the ontology match sets O_{MAT} for a query Q, we aim to recommend up to k ontologies that are informative and popular while incurring the least matching costs. In the following, firstly we define the matching cost of an ontology to the query, the informativeness and the popularity of the ontology. Then we integrate these measures to get the final score for the ontologies.

4.2.1. Matching Cost

We consider the matching cost for each ontology $o_i \in O_{MAT}$ to find the k best matching ontologies out of the ontology match set. The matching cost for o_i is the difference between the content and structure of the concepts in the query match Q^i_{mat} and the corresponding query graph QG. To quantify the matching cost of an ontology o_i, the matching cost for a query match Q^i_{mat} is computed. The matching cost considers both, the structure and the content of a query match, referred to as the *structure matching cost* and the *label matching cost*, respectively.

Label matching cost. The label matching cost for a query match Q_{mat} is the average difference between the label of the matched concept c $\in C_{Q_{mat}}$ and its corresponding query concept $c_q \in C(QG)$, where c is a match of c_q i.e. $c = \phi(c_q)$, as shown in Eq. 3 .

$$cost_{lb}(Q_{mat}) = \frac{1}{|C_{Q_{mat}}|} \sum_{\forall c_q \in C(QG)} \sum_{\forall c \in C_{Q_{mat}}} dist_{lb}(c, c_q) : c = \phi(c_q)$$

$$dist_{lb}(c, c_q) = \frac{|l_c \cup l_{c_q}| - |l_c \cap l_{c_q}|}{|l_c \cup l_{c_q}|} \quad (3)$$

where, $dist_{lb}(c, q)$ is the difference in the labels' contents of c (i.e. l_c) and the query term c_q (i.e. l_{c_q}), and $|C_{Q_{mat}}|$ is the size of the concept match set. $dist_{lb}(c, c_q)$ is measured according to information retrieval principles and is computed by using the *Jaccard distance* metric. *Jaccard distance* is a commonly used measure of distance between two sets; we compute the *Jaccard distance* of the set of words. Here, l_c and l_{c_q} represent a set of words in the label of c and c_q respectively. $l_c \cup l_{c_q}$ is the set of all distinct words in labels of c and c_q, and $l_c \cap l_{c_q}$ are the common words in l_c and l_{c_q}.

Example: For a match 'op_2' in Fig. 2, the $dist_{lb}(ConferencePaper, paper)$ is computed as:

$$dist_{lb}(ConferencePaper, paper) = \frac{|\{conference, paper\} \cup \{paper\}| - |\{conference, paper\} \cap \{paper\}|}{|\{conference, paper\} \cup \{paper\}|}$$

$$= \frac{2-1}{2} = 0.5$$

Note that $dist_{lb}(c, c_q)$ is maximum (i.e. 1) if there is no match for a keyword in this query match. The label matching cost is low for the query matches that contain all or more common words with the query concept labels. For example, in Fig. 2, 'op_2' is favoured over match 'op_3' as a query match for QG, since 'op_2' contains more common words with concepts of QG than 'op_3'.

Structural matching cost. A structure matching cost measures the difference in the connectivity structure of the matched concepts of a query match Q_{mat}. The purpose of this metric is to prefer the ontologies that contains Q_{mat} where the concepts of concern are in close vicinity. The intuition behind this is, that the more the concepts are connected to each other, the more closely they are defined in the domain of discourse. An ideal match Q_{mat} is the one that is at least as connected as the least connected query graph QG, i.e. all the concepts in the match Q_{mat} should have a direct connection to at least one other query match in Q_{mat}. We compute the structural cost as:

$$cost_{st}(Q_{mat}) = \frac{1}{n}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} dist_{st}(c_i, c_j) : c_i = \phi(c_{q_i}) \& c_j = \phi(c_{q_j})$$

$$dist_{st}(c_i, c_j) = \begin{cases} 0 & \text{if } c_i = c_j \\ |R_{SP}(c_i \rightarrow c_j)| & \text{if } c_i \neq c_j \\ +\infty & \text{if } \nexists c_i \vee \nexists c_j \end{cases}$$

$$n = |C(QG)| \tag{4}$$

In Eq. 4, $dist_{st}(c_i, c_j)$ is the minimum structural distance of any of the match for c_{q_i} and c_{q_j} i.e., concept c_i to c_j in Q_{mat}. The structural distance $dist_{st}(c_i, c_j)$ is 0 if c_i = c_j (i.e., two query terms matches to the same concept in an ontology are considered as a compound word in that ontology) or the shortest distance, in terms of number of edges, of concept c_i to c_j in Q_{mat}. The length of the shortest path between c_i and c_j is positive infinity (+∞) if c_i and c_j are disconnected (i.e., either one or both $\phi(c_{q_i})$ or $\phi(c_{q_j})$ does not exist in Q_{mat}. The structure cost of a query match $cost_{st}(Q_{mat})$ is the average distance among all the concepts of the concept match set $C_{Q_{mat}}$ of Q_{mat}.

`Example:` Let us consider two query concepts 'author' as c_{q_i} and 'paper' as c_{q_j} of QG, shown in Fig. 3. The structural distances of the matches ($\phi(c_{q_i})$ and $\phi(c_{q_j})$) in op_1, op_2, and op_3, shown in Fig. 2, is as follows:

- In op_1, $\phi(author)$ is 'Author' and $\phi(paper)$ is 'Paper', and $dist_{st}(Author, Paper)$ is 2.
- In op_2, $\phi(author)$ is 'Author' and $\phi(paper)$ is 'ConferencePaper', and $dist_{st}(Author, ConferencePaper)$ is 3.
- In op_3, $\phi(author)$ is 'Author' and $\phi(paper)$ is 'Paper', and $dist_{st}(Author, Paper)$ is 1.

To combine the structural cost and the label costs of the query fold match, we take the harmonic mean of $cost_{lb}(Q_{mat})$ and $cost_{st}(Q_{mat})$ of Q_{mat}, as shown in Eq. 5.

$$cost(Q_{mat}) = \frac{2.cost_{lb}(Q_{mat}).cost_{st}(Q_{mat})}{cost_{lb}(Q_{mat}) + cost_{st}(Q_{mat})} \tag{5}$$

The cost of an ontology o for Q is the cost of Q_{mat} in o as shown in Eq. 6.

$$cost(o, Q) = cost(Q_{mat}) : Q_{mat} \in o \tag{6}$$

4.2.2. Informativeness

Informativeness is defined as a measure of knowledge an ontology provides in relation to a user query. The informativeness measure is characterized by a feature that an ontology 'o' is more informative for a query Q, if a query match Q_{mat} exists in 'o', and the concepts in $C_{Q_{mat}}$ have more relations with other concepts of that ontology or they have datatype relations defined for them. The intuition behind this is, the

more relations that are defined for a concept, the more important the concept is in the ontology, and the more information it includes. Therefore, an ontology in which the concepts defined in the ontology have no or very few relations with other concepts in the ontology are considered less informative for a query.

Our algorithm prefers to recommend more informative ontologies, i.e. ontologies that have stronger connections between its concepts/datavalues. More precisely, the informativeness of an ontology 'o' is a measure in terms of the informativeness of the query match Q_{mat} it contains. To quantify the informativeness of a query match Q_{mat}, we measure the informativeness of each concept in $C_{Q_{mat}}$.

The informativeness of a concept of $C_{Q_{mat}}$ is quantified by measuring the connectivity of the concept in its ontology. The informativeness of each concept c $\in C_{Q_{mat}}$ is the informativeness of the event that 'c' is indeed observed as a concept involving relations in an ontology.

$$inf(c, C_{Q_{mat}}) = 1 + log\frac{rf(c,o)}{max\{rf(c_j,o_j) : c_j \in C_{Q^j_{mat}} \bigcap c \wedge c_j = \phi(c_q) : c_q \in QG\}}$$

$$rf(c,o) = |\{r \in R(o) : domain(r) \vee range(r) = c \; or \; super(c)\}| \tag{7}$$

$Inf(c, C_{Q_{mat}})$ of a concept is a query dependent metric as shown in Eq. 7. The informativeness of c of a query match $C_{Q_{mat}}$ is equal to the log of $rf(c,o)$, the *relation frequency* of the concept in the ontology it belongs to, divided by the maximum $rf(c_j,o_j)$ of the concept c_j in its home ontology o_j, where c_j belongs to a query match Q_{mat} for the query Q in o_j, and c and c_j are concept matches for the same query node $c_q \in QG$ in Q_{mat} in and Q^j_{mat}, respectively.

Example: Let us suppose that op_1, op_2, and op_3 (resp. o_1, o_2, and o_3) are the only query matches (resp. matched ontologies) for the example QG in an ontology corpus. For $Q_{mat} = op_3$, C_{op_3} = {Author, Paper}, and

$$inf(Author, C_{op_3}) = 1 + log\frac{rf(Author, o_3)}{rf(Author, o_1)} = 1 + log\frac{1}{2} = 0.7$$

$$inf(Paper, C_{op_3}) = 1 + log\frac{rf(Paper, o_3)}{rf(ConferencePaper, o_2)} = 1 + log\frac{1}{2} = 0.7$$

Based on the informativeness of the concepts of a query match, we compute the informativeness of the query match Q_{mat} as shown in Eq. 8.

$$inf(Q_{mat}, o) = \frac{1}{|C(QG)|} \sum_{\forall c: c \in C_{Q_{mat}}} inf(c, C_{Q_{mat}}) \tag{8}$$

Example: For the example query match op_3,

$$inf(op_3, o) = \frac{1}{|\{Author, Paper, Conference\}|}(inf(Author, C_{op_3}) + inf(Paper, C_{op_3})$$
$$= \frac{1}{3}(0.7 + 0.7) = 0.47$$

The informativeness of an ontology is then measured as the informativeness of a query match it contains for a query Q as shown in Fig. 6.

$$Inf(o, Q) = inf(Q_{mat}, o) \tag{9}$$

4.2.3. Popularity

The *popularity* of an ontology is measured based on the level of reuse of the ontology in an ontology corpus or based on the size of RDF data populated according to the ontology. In this paper, the *popularity* of an ontology is measured in terms of its reuse within the ontology corpus. Therefore, we define *pop(o,O)*, to measure the *popularity* of an ontology o in ontology corpus O. Our *popularity* function is characterised by the following two features: (i) *reuse:* an ontology is more popular, if there are more ontologies using the ontology. (ii) *neighbourhood:* an ontology is more popular, if other popular ontologies use the ontology. Based on these two features, a *reuse* of ontology o by ontology o_i is considered as a "positive vote" for the popularity of ontology o from o_i. The PageRank algorithm is adopted as the *popularity function*, whereby each ontology is considered a node. Eq. 10, formalises the *popularity function* which computes the popularity of *o* at the *kth* iteration.

$$pop_k(o, O) = \frac{1-\alpha}{|O|} + \alpha \sum_{o_i \in BO(o)} \frac{pop_{k-1}(o_i, O)}{|FO(o_i)|} \tag{10}$$

In Eq. 10, |O| is the total number of ontologies in the ontology corpus, $BO(o)$ is a set of ontologies reusing the ontology o and $FO(o)$ is a set of ontologies, ontology 'o' is reusing. We further normalized the popularity of an ontology within the matched ontology set for query i.e., O_{MAT}.

$$pop(o, Q) = \frac{pop(o, O)}{max\{pop(o_j, O) : o_j \in O_{MAT}\}} \tag{11}$$

In Eq. 11, $pop(o, Q)$ returns a value from [0-1]. It is the relative popularity of an ontology 'o' in O_{MAT} that is achieved by dividing the popularity of o with the maximum popularity for any ontology among the matched ontologies for query Q.

4.2.4. Relevance Score

Finally, we define the relevance score of an ontology 'o' to the query Q, as a function of the matching cost, the informativeness and the popularity of 'o' for Q.

Linear Model: We describe a linear model containing fixed weights as a quantitative metric to measure the overall relevance between the query Q and the ontology 'o', and choose the up to k ontologies that have high relevance to the query.

$$rel(o, Q) = \alpha\ [inf(o, Q)] + \beta\ [pop(o, Q)] + \gamma\ [\frac{1}{cost(o, Q)}] \tag{12}$$

According to Eq. 12, ontologies with high informativeness and popularity, and low matching costs are preferred among all matching ontologies O_{MAT}. Here, α, β and γ are the variable sets to combine the three features of a linear model.

Optimization Problem: In O_{MAT}, the set of ontologies matched to Q in O, we aim to find up to k ontologies that are as informative and popular as possible while having the least matching costs (i.e. the best matches to the query). It can be formulated as an optimization problem, in particular, as a 2-dimensional 0-1 knapsack problem, where each $o_i \in O_{MAT}$ corresponds to an 'item' to be selected whose 'value' v_i is the informativeness and popularity, and 'weight' w_i is 1, when the 'capacity' of the 'knapsack' w.r.t the number of items is k and w.r.t. the matching cost is γ .

$$maximize \sum_{\forall i: o_i \in O_{MAT}} x_i * [\alpha(inf(o, Q)) + \beta(pop(o, Q))]$$

$$subject\ to$$

$$\sum_{\forall i: o_i \in O_{MAT}} x_i \leq k$$

$$minimize \sum_{\forall i: o_i \in O_{MAT}} x_i * cost(o_i, Q)$$

$$x_i \in [0,1], 1 < i < |O_{MAT}| \tag{13}$$

In Eq. 13, the optimization algorithm maximizes the informativeness and popularity of the ontology in the result set, where α and β are the variable sets to combine the two, considering the constraint of the result set size, i.e. k and its matching costs.

We implemented the 2-dimensional knapsack problem as a less optimum greedy algorithm solution. We traded speed for effectiveness, since it is important in real time query execution and as shown in the evaluation, it is still very accurate and easily outperforming existing state of the art. An optimal solution using dynamic programming takes a lot of time but results in an optimal solution, whereas a less optimum greedy solution is efficient but the results are not optimal. The greedy algorithm first sorts the ontologies in increasing order of their matching cost and then selects the one with high popularity and informativeness. If two of the ontologies have the same matching cost, it prefers the one that is first evaluated for its popularity and informativeness.

5. Evaluation

In this section we report on a set of experiments and a user study that we performed to demonstrate the effectiveness and efficiency of $RecoOn$.

5.1. Experimental Setup

For our experiments we use our previously established CBRBench[3]. It contains an ontology collection of a representative set of ontologies used on the Web, benchmark queries, and a gold standard established by human experts on the task of ranking ontology concepts for the benchmark queries. All the experiments are performed on a machine with Intel Core i7 3.4 GHz Octa-core CPU and 8GB RAM.

5.1.1. DataSet

For our ontology corpus we use CBRBench ontology collection. This ontology collection is composed of 1011 OWL & RDF(S) ontologies that we use as our ontology corpus. We stored each ontology as a named graph in a Virtuoso database.

5.1.2. Query selection

CBRBench contains ten single-term queries and a gold standard composed of a relevance score for matching concepts to the queries on the task of ontology concept retrieval. CBRBench queries are selected using the query log[4] of the Linked Open Vocabularies (LOV) search engine (Vandenbussche & Vatant, 2014). The most popular search terms in the log covering the period between 06/01/2012 and 16/04/2014 are selected as benchmark queries. These general queries cover a wide range of domains from the list of the most popular query terms in the LOV query log. This helped evaluators who were experts in different domains to correctly evaluate the concepts. All CBRBench queries are single word queries – that is for two reasons. First, only about 11% of all queries posed on the LOV search engine use compound search queries and no compound query was among the 200 most used queries and second, for no compound query in the top 1000 query terms did the benchmark collection contain enough relevant resources to derive at

[3]See `https://zenodo.org/record/11121`

[4]See `http://lov.okfn.org/dataset/lov/stats/searchLog.csv`

Table 3
Query strings derived from benchmark query terms

Q-Id	Multi term queries	Q-Id	Multi term queries
Q1	person & agent	Q20	name & person
Q2	person & organization	Q21	name & title
Q3	person & organization & project	Q22	name & person & agent
Q4	person & student & professor & university		
Q5	organization & location	Q23	title & identifier
Q6	organization & student	Q24	title & organization
Q7	organization & student & course	Q25	title & author & document
Q8	organization & student & course & university		
Q9	event & location	Q26	location & place
Q10	event & conference	Q27	location & place & geographic
Q11	event & conference & paper		
Q12	event & conference & paper & article		
Q13	author & publication	Q28	music & event
Q14	author & newspaper	Q29	music & group
Q15	author & publication & research		
Q16	author & publication & research & issue		
Q17	address & organization	Q30	time & date
Q18	address & organization & place		
Q19	address & organization & place & country		

a meaningful ranking. However, varying length queries composed of multi-terms are required to evaluate the effectiveness of *RecoOn* that are not available in CBRBench. Thus, we first need to establish a set of queries to be used in our experiment as well as in future research. Our queries are derived from our earlier established CBRBench queries and ontology collection in two ways:

Single-term queries. Single-term queries proposed in the CBRBench are used as is to evaluate the performance of the ranking algorithms on ontology ranking. For each single-term benchmark query, the relevance score of the matched concepts to the query terms in the gold standard is considered as the relevance of the corresponding ontologies to the query term. The ten single keyword queries used for the evaluation of *RecoOn* are: `'address'`, `'author'`, `'event'`, `'location'`, `'music'`, `'name'`, `'organization'`, `'person'`, `'time'` and `'title'`.

Multi-term queries. Multi-term queries are created to evaluate the effect of the query size on the performance of the algorithm as follows.

1. First, for each of the ten query terms of CBRBench the top three matching ontologies are considered. This results in a collection of 28 ontologies (some ontologies appear in the top three for more than one query) while for two queries we considered 4 ontologies because there was a tie for the third ranked ontology.
2. Each concept in these ontologies is assigned a single-term label by finding the *intended type of class* using the method proposed by (Butt, Haller, & Xie, 2014b), e.g., the label "An Organization - a base class for instances of organizations" for a class[5] is reduced to 'organization'.
3. Once each concept has a single-term label, all possible combinations of length 2, 3 and 4 terms are generated for each ontology from the labels of the concepts of the ontology. E.g., a string 'person & university & student' is generated by combining the labels of the 'person', 'university' and 'student' classes for the query term `'person'` from one of its matched ontologies i.e., UNIV_BENCH ontology[6].

[5] `http://data.press.net/ontology/stuff/Organization`
[6] `http://swat.cse.lehigh.edu/onto/univ-bench.owl\#`

4. For each combination of concept labels the number of times they occur collectively in the ontology corpus is computed. The most frequently occurring 2, 3 and 4 length concept label combinations are then selected from each ontology. The intuition behind this process is that the more frequently concepts occur together in the ontology corpus, the more related they are (they belong to the same domain of discourse). We could not find a meaningful combination for some multi-term queries because they do not occur together in any other ontology and ended up with 30 additional multi-term queries to evaluate `RecoOn` as shown in Table 3.

5.1.3. Baseline

To evaluate the quality of results produced by ***RecoOn***, two versions of ***RecoOn*** are implemented.

- $RecoOn_{opt}$ - Optimized RecoOn , ***RecoOn*** recommends up to 'k' results based on the relevance score computed through the optimization model.
- $RecoOn_{ln}$ - Linear RecoOn , ***RecoOn*** recommend up to 'k' results based on relevance score computed with linear relevance model. The weights for calculating relevance score (Eq. 12) for our experiments are set to 0.4, 0.3, and 0.3 for the 'matching cost', 'informativeness' and 'popularity' metrics respectively. The relative weighs for these metrics are selected based on how well each metrics performed in our pre-evaluation tests.

To evaluate the effectiveness, we compare the result set of $RecoOn_{opt}$ with the result sets of `AKTiveRank` and $RecoOn_{ln}$ for all queries shown in Table 3 .

To the best of our knowledge, `LOV` is the most recent purpose built ontology library available on the Web. However, `LOV` and most of other ontology libraries do not focus on the ranking of search results (Butt, Haller, & Xie, 2015). Moreover, some domain specific ontology libraries and search engines (Noy et al., 2009; Martínez-Romero et al., 2014) purposed effective ranking models, however, the ranking methods used are specific to that domain and cannot be adapted for a domain independent ontology collection. We consider `AKTiveRank` as our baseline, since the approach is one of the two state-of-the-art generic ontology ranking techniques (the second one is `Swoogle` (Ding et al., 2005)). However, as the evaluation results presented in (Alani et al., 2006) prove, `AKTiveRank` outperforms `Swoogle` on the task of ontology ranking. Another reason for not considering `Swoogle` as a baseline is that it computes the ranks for the matched ontologies on the basis of the instances of that ontology in the Swoogle database, which is not possible in our case where the dataset is merely composed of ontologies. A list of $RecoOn_{opt}$ containing 'k' elements is compared with $RecoOn_{ln}$ to verify that the quality of the results produced through the optimization model is better than a linear model.

5.2. User Study

5.2.1. Approach.

Table 4
Comparison statistics of $RecoOn_{opt}$ with baselines

Baseline	Minimum	Maximum	Average	SDev
AKTiveRank	3	12	8	1.95
$RecoOn_{ln}$	4	13	8	1.92

We implemented $RecoOn_{opt}$ and $RecoOn_{ln}$, and re-implemented $AKTiveRank$ to the best of our abilities. A list of relevant results for all three models is produced for multi-term queries over the CBR-Bench dataset to compare the effectiveness of $RecoOn$ in a user study. We conducted the user study with sixteen human experts from the ANU, Monash University, the University of Queensland, CSIRO, Fraunhofer Institute, Vienna University of Business, KIT, Universidad de Chile, the Australian Bureau of Statistics and the Polytechnical University of Madrid. All of the evaluators have developed ontologies before and some are authors of widely cited ontologies.

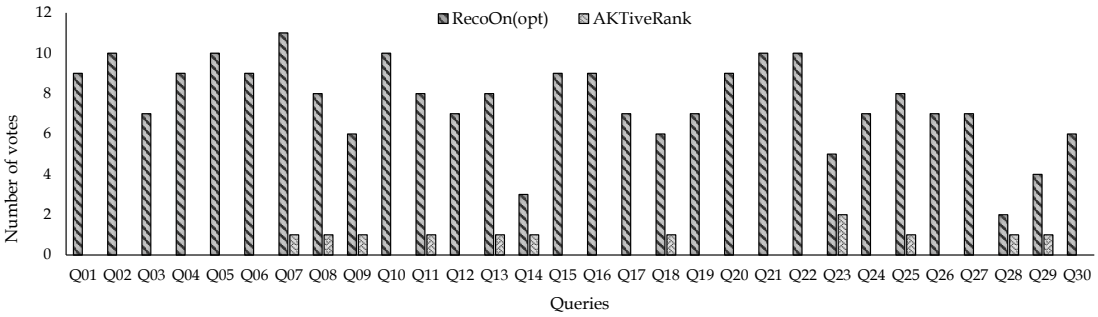


Fig. 5. Effectiveness of $RecoOn_{opt}$ vs. $AKTiveRank$ for multi-term query strings

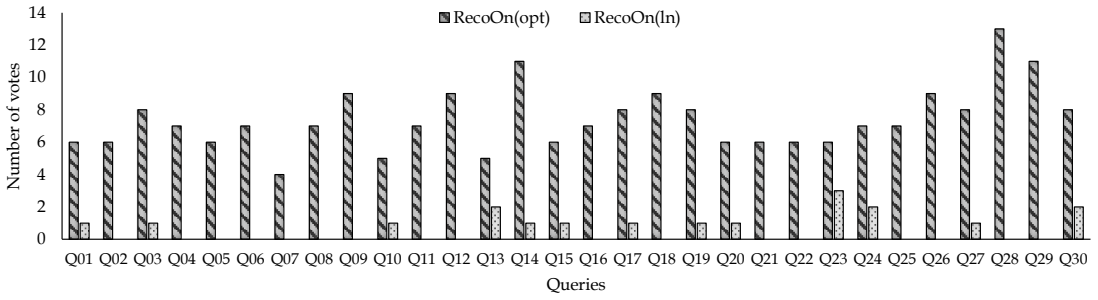


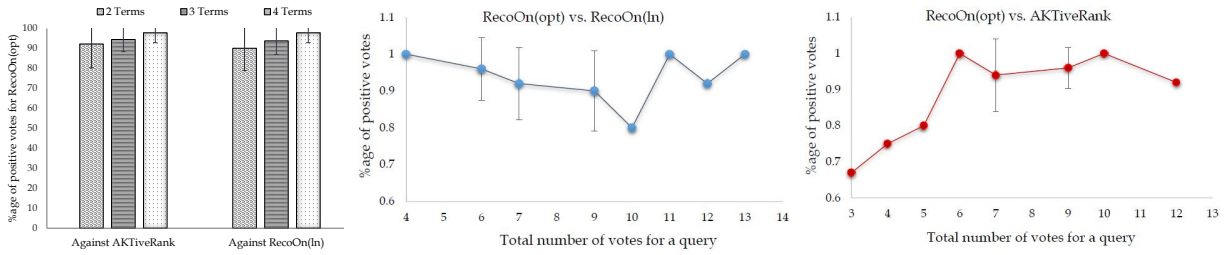
Fig. 6. Effectiveness of $RecoOn_{opt}$ vs. $RecoOn_{ln}$ for multi-term query strings

For the user study, an evaluation tool[7] was developed that allowed the experts to evaluate $RecoOn_{opt}$ for the thirty multi-term query strings in comparison to the $AKTiveRank$ and $RecoOn_{ln}$ produced result sets. Two lists of matched ontologies for a given query string are presented to the evaluators, and they are requested to choose the list, from the two, that is more relevant to the query string. To make our evaluation robust and neutral, the following decisions were taken: (1) The order of the queries along with their relevant results shown to the evaluators was chosen randomly. Every participant was shown queries in a random order to eliminate the effect of the query sequence on the performance of the approaches. (2) For each query two lists 'List A' and 'List B', each consisting of the ten most relevant ontologies along with the matched concepts in the ontology, were shown to the participants. For each query $RecoOn_{opt}$ was compared with one of the other two algorithms; either $AKTiveRank$ or $RecoOn_{ln}$ as baseline. The selection of the baseline was random. (3) The positioning of the ranked lists as 'List A' or 'List B' was random too, i.e. results of $RecoOn_{opt}$ appeared either as 'List A' or 'List B'. Table 4 shows the statistics about the evaluation strategy. The Table shows minimum, maximum, average, and standard deviation of the number of times the results of $RecoOn_{opt}$ for a query was evaluated in comparison to the $AKTiveRank$ or $RecoOn_{ln}$ results. The statistics shows a balance in the evaluation as on average 8 evaluations for each query were received for $RecoOn_{opt}$ in comparison to both baselines.

5.2.2. Results

To derive our results, we considered a positive vote for $RecoOn_{opt}$ (resp. $RecoOn_{ln}$ or $AKTiveRank$) when the list comprised of results of $RecoOn_{opt}$ (resp. $RecoOn_{ln}$ or $AKTiveRank$) was selected by the expert as the "more relevant" result set for a given query. Fig. 5 and Fig. 6 show all 16 votes for all thirty queries; the x-axis shows all 30 queries while the y-axis shows the number of votes (evaluations) in favour of each approach in comparison to the one other approach. $RecoOn_{opt}$ (resp. $AKTiveRank$) bars in Fig. 5 show the number of votes for $OptimizedRecoOn$ (resp. $AKTiveRank$) in comparison to $AKTiveRank$ (resp. $OptimizedRecoOn$). Similarly, the results for $RecoOn_{opt}$ and $RecoOn_{ln}$ in Fig. 6

[7] http://activeraul.org/ontologySearch/



(a) Effect of query length

(b) Effect of number of evaluations (votes)

Fig. 7. $RecoOn_{opt}$ performance

show the number of votes for $OptimizedRecoOn$ in comparison to $AKTiveRank$. Fig. 5 shows that $RecoOn_{opt}$ incurred more positive votes in comparison to the baseline $AKTiveRank$, i.e. 94% of the time experts voted for the result set produced by $RecoOn_{opt}$ to be "more relavant" than `AktiveRank` for the example queries. Moreover, 92% of the time $RecoOn_{opt}$ generated "more relevant" result-sets in comparison to $RecoOn_{ln}$ as shown in Fig 6.

Fig. 7a shows the effect of the query size on the effectiveness of $RecoOn_{opt}$. The percentage of positive votes, in comparison to the baselines, for 2, 3, and 4 terms queries are shown here. The figure confirms that an increase in the number of the query terms increased the effectiveness of $RecoOn_{opt}$ in comparison to the baselines. The average of positive votes increased from 90% to 98% (resp. 92% to 98%) and the standard deviation decreased from 11.9% to 4.9% (resp. 11.2% to 4.9%) in comparison to $AKTiveRank$ (resp. $RecoOn_{ln}$).

In another analysis we examined the effect of the number of evaluations for a query on the performance of $RecoOn_{opt}$. Fig. 7b, shows the statistics, the x-axis shows the number of evaluations (votes) for a query and y-axis shows the performance of $RecoOn_{opt}$ in comparison to both baselines. The results presented here show that an increase in the number of evaluations corresponding to a query result in a stable and improved performance of $RecoOn_{opt}$.

5.3. Experiments

Table 5
MAP and NDCG for Single term queries

Metric	Approach	Queries									
		Person	Name	Event	Title	Loc.	Addr.	Music	Org.	Author	Time
MAP@10	$RecoOn_{opt}$	0.87	0.69	0.74	0.62	0.53	0.72	0.64	0.88	0.65	0.41
	$RecoOn_{ln}$	0.61	0.33	0.45	0.31	0.42	0.55	0.39	0.61	0.41	0.46
	$AKTiveRank$	0.47	0.16	0.25	0.34	0.34	0.48	0.42	0.32	0.54	0.23
NDCG@10	$RecoOn_{opt}$	0.66	0.51	0.49	0.37	0.32	0.54	0.42	0.54	0.43	0.28
	$RecoOn_{ln}$	0.43	0.21	0.29	0.33	0.27	0.38	0.33	0.39	0.25	0.19
	$AKTiveRank$	0.32	0.11	0.14	0.20	0.16	0.23	0.33	0.18	0.29	0.09

Other than the user study, we also compared our approach to the gold standard available as part of the CBRBench benchmark. As mentioned in Sec. 5.1 we consider the rank of a resource in CBRBench, for a given query, as the rank for the ontology this resource belongs to. If more than one matched resources for a query term belong to the same ontology, the highest rank of a matched resource that belongs to the same ontology is assigned as the rank of the ontology for a given query term. We then measure the Mean Average Precision @ 10 (MAP@10) and Normalized Discounted Cumulative Gain @ 10 (NDCG@10) for all ten single term queries based on the gold standard derived from the CBRBench gold standard.

Table 5 shows the MAP@10 and NDCG@10 of $RecoOn_{opt}$, $RecoOn_{ln}$ and $AKTiveRank$ on ten single term queries. The results shows that $RecoOn_{opt}$ performs better than $RecoOn_{ln}$ and $AKTiveRank$.

Moreover, $RecoOn_{ln}$ outperforms $AKTiveRank$ on all ten sample queries. The average MAP@10 is 0.68 and NDCG@10 is 0.46 for $RecoOn_{opt}$ in comparison to the baselines where average MAP@10 is 0.45 and NDCG is 0.31 for $RecoOn_{ln}$ and for $AKTiveRank$, MAP@10 is 0.36 and NDCG is 0.21 .

5.4. Scalability Analysis

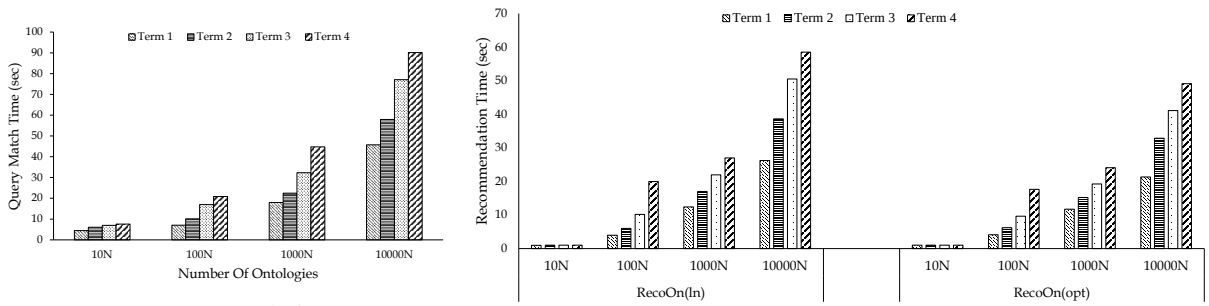
In the final experiment, we demonstrate the scalability of our approach and runtime improvements of $RecoOn_{opt}$ over $RecoOn_{ln}$. $RecoOn_{opt}$ employs a greedy algorithm solution, that tries to minimize the matching cost, to quickly find high-quality matches. We choose k = 10, and use the same queries as in the previous experiments (both single term and multi-term queries). The analysis are conducted on four ontology corpora with differing sizes in terms of the number of ontologies it contains (i.e. 10N, 100N, 1000N and 10000N).

The following experiments are based on varying sized corpora that are randomly sampled from the CBRBench. Since CBRBench contains 1011 ontologies, to generate 10000N corpus we randomly picked an ontology removed one or more of it concepts or relations and added it into 10000N corpus with a different ontology's URI. The process is repeated until ontology count reaches to 10000N. The ontologies in CBRBench varies in terms of the number of triples they contain. Therefore, a random selection of ontologies for all four corpora has high chances of measurement bias. In order to minimize the bias, we randomly generated 10 samples of each corpus size and recorded the query runtime for all 40 queries on each sample. An average of 40 queries on 10 samples is recorded as query execution time for one ontology corpus.

Table 6
Comparison statistics of $RecoOn_{opt}$ with baselines

Ontologies Count	Minimum Triples	Maximum Triples	Average Triples	SDev
10	2357	103548	38782.5	41504.4
100	215934	765842	467776.1	174549.5
1000	3021562	8453289	5798053.9	1604409
10000	35381037	68672132	53217881.8	10667258.4

Table 6 shows the maximum, minimum and average number of triples for the samples of all 10N, 100N, 1000N and 100000N corpora. The statistics show that samples for each size of ontology corpus vary in the number of triples. A measurement on a single sample may not be the true reflection of the performance. Therefore, for all experiments reported in this section, we consider an average of ten samples for an ontology corpus.



(a) Query match time

(b) Query recommendation time

Fig. 8. Query match and recommendation time for different length queries

We recorded the query execution time for all single-term and multi-terms queries. The experiments are divided into those that measure the query match time i.e., the time it takes to find matched ontologies (as discusses in Sec. 4.1), and those that measure the top-k recommendation time i.e. time it takes to

recommend up to 'k' high quality matches for a query. The former are referred to as '*Query match time*' experiments and the latter as '*Recommendation time*' experiments. The reasons for this is twofold: 1) query match time is dependent on the underlying RDF store (Virtuoso repository in our implementation), while for 2) for both $RecoOn_{opt}$ and $RecoOn_{ln}$the query match time is the same, but the time to recommend k ontologies differs, due to the different implementations of the recommendation model.

The query match time here also contains the connection time with a local Virtuoso repository, which is the same for each query on all different sized ontology corpora. Fig. 8a shows the query match time in seconds with varying query size and varying corpus size. The figure shows an increase in the query match time with an increase in the number of ontologies (resp. triples) and query size (i.e., number of terms in a keyword query). The increase in the query match time is logarithmic with the increase in the number of ontologies/triples in $RecoOn$. However, the experiment was performed on a desktop machine that can be easily distributed to perform faster for a reasonable size of ontologies.

Similarly, Fig. 8b show the ontology recommendation time in seconds with varying query size, and varying corpus size, for $RecoOn_{opt}$ and $RecoOn_{ln}$. The results show that $RecoOn_{opt}$ performs better than $RecoOn_{ln}$. The reason for an improved performance is that we implemented a greedy algorithm for the knapsack optimization that measures the matching cost for all the matched ontologies, while recording the popularity and informativeness only until the top-k ontologies are retrieved. Although, $RecoOn_{opt}$ is more efficient in our experiments, the runtime for $RecoOn$ can further be reduced by employing a multi-thread implementation.

5.5. Results Summary

From the statistics presented in Sec. 5.2 and Sec. 5.3, it is evident that $RecoOn_{opt}$ outperforms the state-of-the-art ontology ranking algorithm $AKTiveRank$ on the sample queries for the CBRBench ontology collection; while an optimized solution to recommend relevant ontologies is preferred over a linear model of $RecoOn$ in most cases. On average $RecoOn_{opt}$ received 94% positive votes as compared to $AKTiveRank$ and 92% positive votes as compared to $RecoOn_{ln}$ of relevant ontologies for the multi-term queries in the user study. The results also show that the effectiveness of $RecoOn_{opt}$ increases with an increase in the length of the query string (i.e., number of query terms); and an increase in the number of votes for a query result set increases the average of positive votes for $RecoOn_{opt}$ which means that the lowest positive votes $RecoOn_{opt}$ received for a query (i.e. 67%) could have improved with more evaluations for this query. However, in this user study, for multi-term queries the relevance and the order were evaluated at once by performing a comparative study of two lists. As future work, we aim to test them separately to improve the ranking independently of the relevance of an ontology.

Similarly, for a single term query strings $RecoOn_{opt}$ scored higher for MAP@10 (i.e. 0.68) as compared to $AKTiveRank$ (i.e. 0.36) and higher for NDCG@10 (i.e. 0.46) as compared to the $AKTiveRank$ (i.e. 0.21). These statistic shows that even for the evaluation of the order of a ranking list, $RecoOn_{ln}$ yields better scores as compared to $AKTiveRank$. Further experiments were conducted to evaluate the design decisions made for the proposed recommendation model (i.e. $RecoOn$). The results presented in Sec. 5.4 show that $RecoOn_{opt}$ performs better than $RecoOn_{ln}$ in terms of total query execution time because of the greedy implementation of the knapsack optimization.

6. Conclusion and Future Work

In this paper we have presented $RecoOn$, an ontology recommendation approach to select and rank relevant ontologies against a multi-term structureless query. Our approach first finds a set of matched ontologies for a query string and then identifies the up to k most relevant matches using three measures, the *Matching cost*, the *Informativeness* and the *Popularity* of the matched ontologies. Then we integrate these measures by formulating and solving a linear model (i.e. $RecoOn_{ln}$) and then as an optimization problem (i.e. $RecoOn_{opt}$). The evaluation of our approach against $AKTiveRank$ and a linear version of

our algorithm $RecoOn_{ln}$ in a user study performed on the CBRBench dataset (Butt et al., 2014a) shows that $RecoOn_{opt}$ outperforms the baseline algorithm $AktiveRank$ in 94% of the cases and $RecoOn_{ln}$ in 92% of the cases. Moreover, our experiments also show that $RecoOn_{opt}$ is efficient in terms of query execution time on the CBRBench ontology collection for the sample queries. Although our algorithm shows significantly improved effectiveness compared to the state-of-the-art in ontology ranking models, we believe further improvements are possible. In this work we mainly focus on the ranking model. In future work we want to extend $RecoOn$ in different ways. The query graph generation can be improved to enhance the matching quality. In the current implementation, user queries are mapped to the ontology concepts in the form of query graphs. We want to dynamically map structureless queries to structured graph queries, where a term in a query can be mapped to a concept (node) or to a relationship (edge) of the ontology graph through a query graph. Secondly, we want to introduce a faceted search over the ontology corpus that will help a user explore the result sets effectively and efficiently. We also plan to implement an efficient algorithm to determine the 'intended type' of a concept (Butt et al., 2014b) to improve the matching quality of a resource to the user query.

References

Alani, H., Brewster, C., & Shadbolt, N. (2006). Ranking Ontologies with AKTiveRank. In *Proceedings of iswc* (pp. 1–15).

Alani, H., Noy, N. F., Shah, N., Shadbolt, N., & Musen, M. A. (2007). Searching ontologies based on content: experiments in the biomedical domain. In *Proceedings of the 4th international conference on knowledge capture* (pp. 55–62).

Butt, A. S., Haller, A., & Xie, L. (2014a). Ontology search: An empirical evaluation. In *Proceedings of iswc* (pp. 130–147). Riva del Gara, Italy.

Butt, A. S., Haller, A., & Xie, L. (2014b). Relationship-based top-k concept retrieval for ontology search. In *Proceedings of ekaw* (pp. 485–502).

Butt, A. S., Haller, A., & Xie, L. (2015). A taxonomy of semantic web data retrieval techniques. In *Proceedings of the 8th international conference on knowledge capture* (p. 9).

Butt, A. S., Haller, A., & Xie, L. (2016). *Relationship-based top-k concept retrieval for ontology search.* http://semantic-web-journal.net/system/files/swj883.pdf.

Cantador, I., Fernández, M., & Castells, P. (2007). Improving ontology recommendation and reuse in webcore by collaborative assessments.

Cheng, G., Tran, T., & Qu, Y. (2011). Relin: relatedness and informativeness-based centrality for entity summarization. In *The semantic web–iswc 2011* (pp. 114–129). Springer.

d'Aquin, M., & Motta, E. (2011). Watson, More Than a Semantic Web Search Engine. *Semantic Web*, *2*(1), 55–63.

Ding, L., Finin, T., Joshi, A., Pan, R., Cost, R. S., Peng, Y., ... Sachs, J. (2004). Swoogle: A Search and Metadata Engine for the Semantic Web. In *Proceedings of cikm* (pp. 652–659). New York, USA.

Ding, L., Pan, R., Finin, T., Joshi, A., Peng, Y., & Kolari, P. (2005). Finding and ranking knowledge on the semantic web. In *Proceedings of iswc* (pp. 156–170).

Freeman, L. C. (1977). A set of measures of centrality based on betweenness. *Sociometry*, 35–41.

Jonquet, C., Musen, M. A., & Shah, N. H. (2010). Building a biomedical ontology recommender web service. *Journal of biomedical semantics*, *1*(1), 1.

Martínez-Romero, M., Vázquez-Naya, J. M., Pereira, J., & Pazos, A. (2014). Bioss: A system for biomedical ontology selection. *Computer methods and programs in biomedicine*, *114*(1), 125–140.

Meymandpour, R., & Davis, J. G. (2013). Linked data informativeness. In *Web technologies and applications* (pp. 629–637). Springer.

Noy, N. F., & d'Aquin, M. (2012). Where to Publish and Find Ontologies? A Survey of Ontology Libraries. *Web Semantics: Science, Services and Agents on the World Wide Web*, *11*(0).

Noy, N. F., Shah, N. H., Whetzel, P. L., Dai, B., Dorf, M., Griffith, N., ... Musen, M. A. (2009). BioPortal: ontologies and integrated data resources at the click of a mouse. *Nucleic Acids Research*.

Page, L., Brin, S., Motwani, R., & Winograd, T. (1998). The PageRank citation ranking: Bringing order to the Web. In *Proceedings of the www conference* (pp. 161–172). Brisbane, Australia.

Patel, C., Supekar, K., Lee, Y., & Park, E. (2003). Ontokhoj: a semantic web portal for ontology searching, ranking and classification. In *Proceedings of acm widm* (pp. 58–61).

Rada, R., Mili, H., Bicknell, E., & Blettner, M. (1989). Development and application of a metric on semantic nets. *Systems, Man and Cybernetics, IEEE Transactions on*, *19*(1), 17–30.

Sabou, M., Lopez, V., & Motta, E. (2006). Ontology selection for the real semantic web: How to cover the queen's birthday dinner? In *Managing knowledge in a world of networks* (pp. 96–111). Springer.

Spanoudakis, G., & Constantopoulos, P. (1993). Similarity for analogical software reuse: A conceptual modelling approach. In *Advanced information systems engineering* (pp. 483–503).

Tartir, S., & Arpinar, I. B. (2007). Ontology evaluation and ranking using ontoqa. In *Semantic computing, 2007. icsc 2007. international conference on* (pp. 185–192).

Tummarello, G., Delbru, R., & Oren, E. (2007). Sindice.Com: Weaving the Open Linked Data. In *Proceedings of iswc* (pp. 552–565). Berlin, Heidelberg.

Vandenbussche, P.-Y., & Vatant, B. (2014). Linked Open Vocabularies. *ERCIM news*, *96*, 21–22.

Wu, G., Li, J., Feng, L., & Wang, K. (2008). Identifying potentially important concepts and relations in an ontology. In *Proceedings of iswc* (pp. 33–49).