Information Processing and Management 54 (2018) 116–128

Contents lists available at ScienceDirect

Information Processing and Management

journal homepage: www.elsevier.com/locate/infoproman

Knowledge based collection selection for distributed information retrieval

Han Baoli, Chen Ling*, Tian Xiaoxue

College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China

ARTICLE INFO

Keywords:
Collection selection
Distributed information retrieval
Knowledge base
Query expansion

ABSTRACT

Recent years have seen a great deal of work on collection selection. Most collection selection methods use *central sample index* (CSI) that consists of some documents sampled from each collection as collection description. The limitations of these methods are the usage of 'flat' meaning representations that ignore structure and relationships among words in CSI, and the calculation of query-collection similarity metric that ignore semantic distance between query words and indexed words. In this paper, we propose a knowledge based collection selection method (KBCS) to improve collection representation and query-collection similarity metric. KBCS models a collection as a weighted entity set and applies a novel query-collection similarity metric to select highly scored collections. Specifically, in the part of collection representation, context- and structure-based measures are employed to weight the semantic distance between two entities extracted from the sampled documents of a collection. In addition, the novel query-collection similarity metric takes the entity weight, collection size, and other factors into account. To enrich concepts contained in a query, DBpedia based query expansion is integrated. Finally, extensive experiments were conducted on a large webpage dataset, and DBpedia was chosen as the graph knowledge base. Experimental results demonstrate the effectiveness of KBCS.

1. Introduction

Distributed Information Retrieval (DIR), also known as Federated Search (FS) or Federated IR (FIR), concerns with aggregating multiple searchable sources of information under a single interface (Crestani & Markov, 2013). DIR consists of four main phases: collection (server/resource) description, collection selection, results merging, and results presentation. Given a query and a set of collection descriptions, collection selection ranks available collections based on their computed scores, then determines which collections to search (Callan, 2002). In a specific search circumstance, users are often interested in top-ranked search results. However, not all collections contain information that users need. If search engine only retrieve a small number of collections and get a similar effect to retrieve all collections, it would significantly enhance the efficiency of retrieval system. Collection selection plays an important role in reducing computational overhead and improving retrieval efficiency.

Recent years have seen a great deal of work on collection selection, which can be divided according to the mechanism to describe a collection: dictionary-based methods (Aly, Hiemstra, & Demeester, 2013, Callan, Lu, & Croft, 1995, Gravano & Garcia-Molina, 1995, Xu & Croft, 1999, Yuwono & Lee, 1997) and sampling-based methods (Baillie, Carman, & Crestani, 2011, Kulkarni, Tigelaar, Hiemstra, & Callan, 2012, Mendoza, Marín, Gil-Costa, & Ferrarotti, 2016, Paltoglou, Salampasis, & Satratzemi, 2011, Shokouhi, 2007, Shokouhi, Zobel, Tahaghoghi, & Scholer, 2007, Si & Callan, 2003, Thomas & Shokouhi, 2009, Wauer, Schuster, & Schill, 2011).

* Corresponding author.
E-mail addresses: litutor@zju.edu.cn (B. Han), lingchen@cs.zju.edu.cn (L. Chen), xxtian@zju.edu.cn (X. Tian).

http://dx.doi.org/10.1016/j.ipm.2017.10.002
Received 1 April 2017; Received in revised form 28 August 2017; Accepted 11 October 2017

Dictionary-based methods use the word statistics of all documents as collection description, and then exploit a scoring function to reflect the similarity between a collection and a query. However, it is unfeasible to acquire the word statistics of all collections in an uncollaborative distributed information retrieval environment. Another problem is that the scoring function based on word statistics loses a large amount of semantic information in calculating collection score, e.g., synonym, polysemy, and the order of words. These methods also have a low effectiveness in the environment of skewed collection sizes.

To overcome the limitation of usage in uncollaborative distributed information retrieval environment, sampling-based methods use some documents that are sampled from a collection as collection description. With the development of text representation technology, some sampling-based methods (Callan & Connell, 2001) begin to exploit ESA (Gabrilovich & Markovitch, 2007) and LDA (Blei, Ng, & Jordan, 2003) to represent collection description. Compared with dictionary-based methods, sampling-based methods can find entities or topics that reflect the semantic information of a collection and have higher accuracy of collection representation. However, they totally ignore the structure and relationships among entities or topics.

The growing popularity of graph knowledge bases, e.g., DBpedia (Bizer et al., 2009), Yago (Hoffart, Suchanek, Berberich, & Weikum, 2013), and Freebase (Bollacker, Evans, Paritosh, Sturge, & Taylor, 2008), has inspired the emergency of new text representation methods. To represent documents with more fine-grained semantic information, knowledge based text representation methods (Ni et al., 2016, Schuhmacher & Ponzetto, 2014) exploit the relational knowledge and network structure encoded in graph knowledge bases. These methods have achieved better performance than ESA and LDA.

Query expansion is also important for collection selection, which can help reduce the vocabulary mismatch problem by adding related terms to a query and find potential related documents and collections (Carpineto & Romano, 2012). If a collection is modeled as a weighted entity set, without query expansion, it would be hard to compute its score for a query containing few entities. There is a vast amount of research work on knowledge based query expansion (Arguello, Elsas, Callan, & Carbonell, 2008, Dang & Croft, 2010, Eiron & McCurley, 2003, Guisado-Gámez, Dominguez-Sal, & Larriba-Pey, 2014), which can obtain expansion terms from external knowledge sources, e.g., Wikipedia and DBpedia. Arguello et al. (2008), Dang and Croft (2010) and Eiron and McCurley (2003) only focus on adding entities directly connected to original query entities in a graph knowledge base. Due to the limitation in connected entities, these entities that are actually related to original query entities but not connected in a knowledge base are excluded in expansion terms. Guisado-Gámez et al. (2014) not only find the semantic relevant terms but also the connected entities in a knowledge base, and thus has a better performance.

As mentioned above, collection representation, query-collection similarity metric, and query expansion are three key points in collection selection research. In this paper, we propose *knowledge based collection selection* (KBCS), which covers the three key points. KBCS explicitly models a collection as a weighted entity set. Entities are firstly extracted from the sampled documents of a collection and the semantic distance of each pair of entities is weighted by employing semantic relation measures, including context- and structure-based measures based on knowledge bases. Then, to identify the important entities, entities are assigned with different weights by calculating semantic distances between an entity and all the other entities in a collection. In addition, we integrate a query expansion method (Arguello et al., 2008) to find more related entities. Specifically, to distinguish entities in a query and avoid query drift, the way to weight query entities is adjusted in our method. Finally, we present a novel query-collection similarity metric based on the knowledge representation that takes sampling factor (the proportion of sampled documents in a collection), collection entity frequency, collection entity weight, and query entity weight into account.

To summarize, our contributions are as follows:

1. Propose KBCS, which exploits the relational knowledge and network structure encoded in knowledge bases to solve the ‘flat’ meaning representation problem, and integrates a query expansion method to solve the vocabulary mismatch problem.
2. Present a query-collection similarity metric, which aggregates both context- and structure-based semantic distances between query entities and collection entities. To improve its precision, the sampling factor, collection entity frequency, collection entity weight, and query entity weight are incorporated into the calculation formula.
3. Conduct extensive experiments to verify the effectiveness of KBCS. On one hand, evaluation on KBCS and baselines has demonstrated that KBCS can get a high precision while chooses few collections. On the other hand, evaluation on query expansion has demonstrated that enriching entities found in query terms can significantly improve KBCS's precision.

The rest of this paper is structured as follows: the next section describes related work on collection selection. Section 3 firstly gives some definitions about collection selection and a brief introduction to KBCS's architecture, and then elaborates on KBCS's algorithm. Section 4 describes experimental details and discusses experimental results. We conclude our work in Section 6.

2. Related work

There has been considerable research on collection selection. Collection selection algorithms can be divided into three classes: dictionary-based methods, sampling-based methods, and classification-based methods which combine the above methods and a number of other query- and corpus-based features in a machine learning framework. In this paper, we focus on dictionary- and sampling-based methods, which are most related prior research work.

2.1. Dictionary-based methods

Dictionary-based collection selection methods use the word statistics of all documents in a collection as collection description.

Then these methods calculate the similarity between a query and a collection based on word statistics. Finally, collections are ranked and highly scored collections are selected.

CORI (Callan et al., 1995) uses the TF-IDF (Chowdhury, 2010) scoring function of INDRI (Strohman, Metzler, Turtle, & Croft, 2005) to calculate the similarity between a collection and a query, then selects highly scored collections. Xu and Croft (1999) build a topic language model to represent the distribution of words in a collection. It organizes collections around some topics and selects right topics for a query. The similarity between topics and a query is computed by Kullback-Leibler divergence, which is also viewed as collection score. GGloss (Gravano & Garcia-Molina, 1995) uses a new type of vector space model to represent collection, where the *i*th component reflects the weight of a word statistic feature. The cosine distance of vectors is chosen as collection score. Due to the phenomenon that the same word in different collections has different document frequencies, CVV (Yuwono & Lee, 1997) combines the document frequency and the variance of cue validity to calculate the similarity between a collection and a query. Taily (Aly et al., 2013) models the score distribution of documents in each collection as a Gamma distribution and selects collections containing more than a certain number of documents that are beyond the cut-off score.

This type of methods requires that each collection can provide its own word histograms to describe the statistics of words. However, it is difficult to do in practice, especially in an uncollaborative environment. In addition, word histograms do not consider any semantic information, e.g., synonym, polysemy, and the order of words, which directly lead to a bad collection representation and similarity measure.

2.2. Sampling-based methods

Sampling-based collection selection methods use central sample index (CSI) that consists of some documents sampled from each collection as collection description. These methods generally employ sampling algorithm, e.g., query-based sampling (Callan & Connell, 2001), to get the CSI, and calculate the similarity between a query and a collection based on the CSI. Finally, collections are ranked and highly scored collections are selected.

Some collection selection methods (Kulkarni et al., 2012, Paltoglou et al., 2011, Shokouhi, 2007, Shokouhi et al., 2007, Si & Callan, 2003, Thomas & Shokouhi, 2009) utilize keyword-based similarity measures to calculate the scores of sampled documents, and then estimate collection score. The main phases of these methods are the following: firstly, evaluate the relevance between a query and a document. Then, group sampled documents by the collection they belong to. The scores of strongly related documents (documents with high score) in the same group are accumulated as collection score. Finally, sort collections by score in descending order and select top-ranked collections. The main difference between these methods is the calculation of documents' score. ReDDE (Si & Callan, 2003) treats all documents equally and gives them same score. FIRSTM-PR (Shokouhi et al., 2007) calculates documents' score with BM25 similarity function and reduces the CSI size by exploiting query log information to eliminate terms. CRCS (Shokouhi, 2007) groups documents by their ranking, and documents in a group have same score. CiSS (Paltoglou et al., 2011) and SUSHI (Thomas & Shokouhi, 2009) calculate documents' score with an interpolant function and a fitted curve function, respectively. SHiRE (Kulkarni et al., 2012) uses a tree structure to estimate documents' score.

The above-mentioned six methods have taken many factors into account, e.g., the size of a collection. However, the calculation of document score depends only on the TF-IDF keyword estimation and the fitted function, without considering semantic information. The TF-IDF keyword estimation is a rough similarity measure and the fitted function only simulates the distribution of document scores, so they cannot improve the precision of similarity measure essentially.

Other collection selection methods (Baillie et al., 2011, Mendoza et al., 2016, Wauer et al., 2011) utilize semantic-based similarity measures to calculate collection score. Wauer et al. (2011) use ESA (Gabrilovich & Markovitch, 2007) vector to represent collections and queries, and then get collection score from cosine similarity, but it has dimension disaster problem. Baillie et al. (2011) model a collection in a low dimensional topic space by using LDA (Blei et al., 2003) topic model, under which collections are ranked based on the topical relevance between a collection and a query. However, LDA topic model does not consider the mutual position of the words in a document, which would result in mismatching topics, and it is also computationally time-consuming on large data sets. Mendoza et al. (2016) builds the vector of each collection on query term space by employing a logistic regression algorithm, and then predicts collection score for a new query. Only from the perspective of query term, it represents the semantic information of a collection partially. Moreover, these methods totally ignore the structure and relationships among entities, topics, and terms.

In addition, some collection selection methods (Francès, Bai, Cambazoglu, & Baeza-Yates, 2014, Kim, Callan, Culpepper, & Moffat, 2016, Kulkarni & Callan, 2015) employ collection creation, document replication, and system workload balancing to improve performance. However, these methods only optimize external factors rather than internal factors, e.g., collection representation and query-collection similarity metric.

3. Methodology

3.1. Definitions

Before introducing the details of our method, some related definitions are given as follows:

Definition 1. (Collection) *A collection c is a set of documents,* $c = \{d_1, d_2, \cdots, d_{|c|}\}$, $|c|$ *is the total number of documents in c. A DIR system contains many collections* $C = \{c_1, c_2, \cdots, c_N\}$, *N is the total number of collections.*

Definition 2. (Collection Selection) *Given a query q, the DIR system would calculate the similarity between q and c, and then get collection*

score R_c:

$$R_c = Rel(q, c) \tag{1}$$

where Rel(q, c) is a query-collection similarity metric. $R = \{R_{c_1}, R_{c_2}, \cdots, R_{c_N}\}$ *is the collection score list, where each collection* c_i *in C would have a score* R_{c_i}*. Finally, the DIR system would rank collections according their scores and select top k collections.*

Definition 3. (Sampled Document Set of Collection) *A sampled document set* s_c *of c is a set of documents that are sampled from c through a certain sampling algorithm,* e.g.*, query-based sampling,* $s_c = \{d_1, d_2, \cdots, d_{|s_c|}\}$*, where* $|s_c|$ *is the total number of sampled documents and* $s_c \subset c$*.*

Definition 4. (Entity Set of Collection) *An entity set* E_c *of c is a set of entities that are extracted from* s_c*,* $E_c = \{e_1, e_2, \cdots, e_{|E_c|}\}$*, where an entity* e_i *references to an entity in DBpedia,* $|E_c|$ *is the total number of entities.*

Definition 5. (Weighted Entity Set of Collection) *A weighted entity set* WE_c *of c is a set of entities that are weighted,* $WE_c = \{w_{e_1}, w_{e_2}, \cdots, w_{e_{|E_c|}}\}$*, where* w_{e_i} *is the weight of* e_i *and* $e_i \in E_c$*.*

3.2. Architecture

Fig. 1 depicts the architecture of KBCS. The sampled documents of a collection and user query are inputs. KBCS builds collection representation and adds related entities to original query through query expansion, and then the similarity between the collection and the query is calculated. KBCS depends on the knowledge base to provide relational knowledge and network structure. No matter what knowledge base is used, e.g., DBpedia, Yago, or Freebase, the steps of our method do not need to be changed. In order to elaborate KBCS, DBpedia is exploited as an example.

KBCS consists of Entity linking, relation weighting, entity weighting, query expansion, and query-collection similarity metric. Entity linking uses DBpedia spotlight (Mendes, Jakob, García-Silva, & Bizer, 2011) to process sampled documents and obtain disambiguated entities through references to entities in DBpedia. Note that the disambiguated entities are divided by their belonging collection, i.e., the number of sets of the disambiguated entities is as same as the number of collections. Relation weighting employs context- and structure-based measures to weight the relatedness of two entities whose distance is within two hops. Entity weighting is to identify most important entities in a collection, which is realized by calculating the semantic distances between an entity and all other entities in a collection. Query expansion integrates a modified DBpedia based query expansion method to add more related entities to the original query. Query-collection similarity metric applies a new similarity metric, which take sampling factor, collection entity frequency, collection entity weight, and query entity weight into account to calculate collection score.

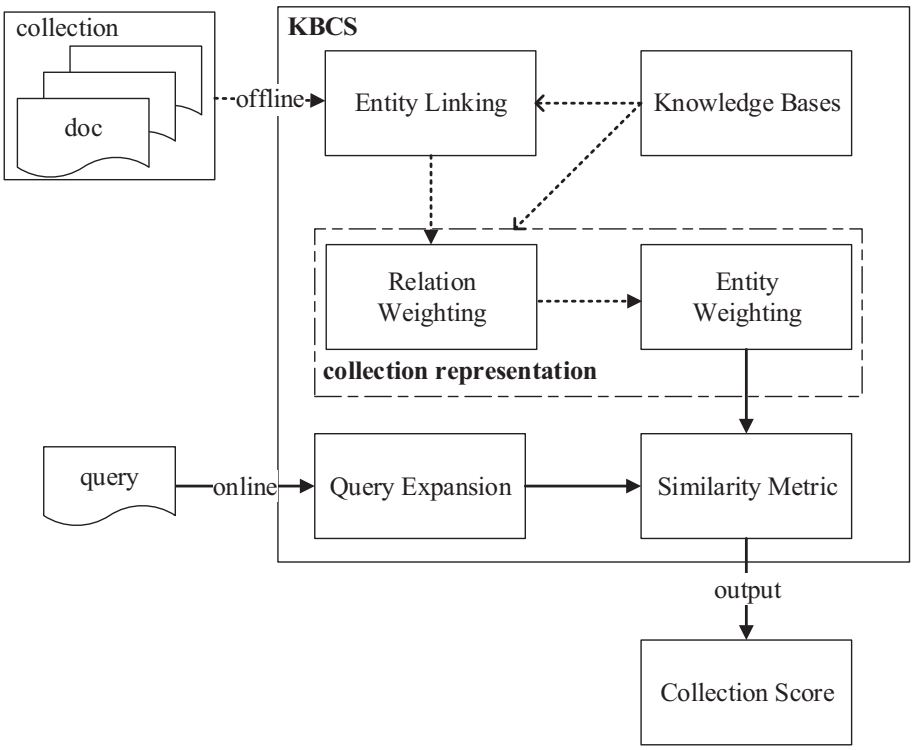

Fig. 1. The architecture of KBCS.

Table 1
Table of annotated DBpedia URIs (Raw Text: First documented in the 13th century, **Berlin** was the **capital** of the Kingdom of **Prussia** (1701–1918), the **German Empire** (1871–1918), the **Weimar Republic** (1919–33) and the **Third Reich** (1933–45). **Berlin** in the 1920s was the third largest **municipality** in the world.).

Original terms	DBpedia URIs
Berlin	http://dbpedia.org/resource/Berlin
capital	http://dbpedia.org/resource/Capital_city
Prussia	http://dbpedia.org/resource/Kingdom_of_Prussia
German Empire	http://dbpedia.org/resource/German_Empire
Weimar Republic	http://dbpedia.org/resource/Weimar_Republic
Third Reich	http://dbpedia.org/resource/Nazi_Germany
municipality	http://dbpedia.org/resource/Municipalities_of_Germany

3.3. Entity linking

DBpedia is chosen as the knowledge base. There is more valuable semantic information encoded within the knowledge base, e.g., relational knowledge and network structure. Wikipedia[1] is a multilingual, free Internet encyclopedia that allows users to edit almost any article. It is currently the largest knowledge repository on the Web, and the English version of it contains more than 5 million articles. DBpedia[2] is a data set that extracts structured content from Wikipedia pages. DBpedia allows users to semantically query relationships and properties associated with Wikipedia resources, and it has a broad scope of entities covering different areas of human knowledge.

In the phase of entity linking, the DBpedia Spotlight (Mendes et al., 2011) can automatically annotate text documents with DBpedia URIs. An entity set E_c of c is derived from sampled documents in s_c through DBpedia Spotlight. For example, the following text would be annotated seven entities. The seven original terms and their annotated DBpedia URIs are shown in Table 1. In order to pursue simple representation of entities and eliminate redundancy, the prefix of 'http://dbpedia.org/resource/' would be removed for each entities. That is to say, '*Berlin*' would be stored as the entity of a collection rather than 'http://dbpedia.org/resource/Berlin'.

3.4. Collection representation

Traditional collection selection methods usually use TF-IDF, ESA, and LDA to represent collections. Although they have considered word frequency and topics, they only build collection representations from the level of morpho-syntactic information and ignore fine-grained semantic information. In order to overcome these problems, we present a collection representation approach that exploits the relational knowledge and network structure encoded in knowledge bases to model a collection as a weighted entity set. That is to say, the semantic information of a collection c is represented by a weighted entity set WE_c.

The main difficulty of our collection representation approach is how to weight entities in a collection. If the weight of an entity is higher, the entity is more important in a collection. To overcome the difficulty, we combine the context- and structure-based measures to calculate the semantic distance between any two entities, and then get their weights.

3.4.1. Relation weighting

Given all collections, CSI is firstly built by sampled documents of each collection. The disambiguated entities are extracted from sampled documents. Relation weighting can evaluate the relatedness of any two entities whose distance is limited in two hops. In other words, the distance of any two entities $\{e_i, e_j\}$ is the shortest path from e_i to e_j in a knowledge base, e.g., DBpedia. There is sufficient evidence from previous works (Hulpus, Hayes, Karnstedt, & Greene, 2013, Navigli & Ponzetto, 2012) to demonstrate that increasing the distance to three hops would tend to produce a time-consuming computation and introduce a lot of noise. The relatedness finally contributes in calculating entity weight.

3.4.1.1. Context-based measure. There are some methods to measure the semantic distance between two words. Cilibrasi and Vitanyi (2007) define a Normalized Google Distance measure, which uses two words to search related web pages through Google search engine and gets the total number of search results. The main idea of this measure is viewing the total number of web pages that contain both words as the semantic distance between two words. Following the idea of normalized Google distance measure, Witten and Milne (2008), as well as Ni et al. (2016), utilize the number of links (links to other Wikipedia articles) shared by two Wikipedia articles to calculate the semantic distance between two Wikipedia titles.

Based on the ideas of above methods, we define the context-based measure, which is based on the number of articles that both entities share on Wikipedia. Intuitively, if two entities are relevant, it is inevitable that these two entities would share some substantive Wikipedia articles, whereas it is impossible that two irrelevant entities con-occur in lots of articles. The number of co-occurrences on all Wikipedia articles is used to capture the semantic distance of an entity pair, which is defined as context-based

[1] https://dumps.wikimedia.org/backup-index.html
[2] http://wiki.dbpedia.org/Downloads2014

semantic distance. Therefore, the more shared articles there are, the more relevant the two entities are. The context-based measure can be defined as follows:

$$ctxt(e_1, e_2) = 1 - \frac{\log(\max(|E_1|,|E_2|)) - \log(E_1 \cap E_2)}{\log|W| - \log(\min(|E_1|,|E_2|))} \tag{2}$$

where e_1 and e_2 are the two entities, E_1 and E_2 are the respective sets of articles to e_1 and e_2, and W is the total number of articles in Wikipedia.

3.4.1.2. Structure-based measure. DBpedia can be viewed as a structural knowledge graph, whose nodes are entities and edges are the predicates that denote relationships between them. For instance, the predicate of 'dbpediaowl: capital' connect the entity of 'dbpedia:Germany' to the entity of 'dbpedia: Berlin' . A naïve solution to calculate the structure semantic similarity of entities would be to use the shortest distance between them in DBpedia. However, it is unreasonable to treat each edge equally. The jointIC (Schuhmacher & Ponzetto, 2014) exploits the number of edges, e.g., 'dbpediaowl: capital', and the number of entities the predicate is pointing to, e.g., 'dbpedia: Berlin', to distinguish the importance of each edge. We apply the jointIC metric to weight different edges. Formally, given an edge, *edge* = (*Subj, Pred, Obj*), where *Subj* and *Obj* are entities, *Pred* is the predicate between *Subj* and *Obj*. The measure can be calculated as follows:

$$IC(\omega_{Pred}) = \log\left(\frac{|T|}{|\omega_{Pred}|}\right) \tag{3}$$

$$IC\ (\omega_{Obj}|\omega_{Pred}) = \log\left(\frac{|\omega_{Pred}|}{|\omega_{(Pred,\ Obj)}|}\right) \tag{4}$$

$$w_{jointIC}\ (edge) = IC\ (\omega_{Pred}) + \ IC\ (\omega_{Obj}|\omega_{Pred}) \tag{5}$$

where $IC(\omega_{Pred})$ is the information content of the predicate, which indicates that a large number of same predicate has a low specificity. $|\omega_{Pred}|$ is the total number of same predicate, and $|T|$ is the count of all triples in the full DBpedia graph. $IC\ (\omega_{Obj}|\omega_{Pred})$ considers the specialty of the entity that the predicate is pointing to. $|\omega_{(Pred,\ Obj)}|$ is the total number of the predicate connected to the entity. The information content of the joint probability distribution is denoted as $w_{jointIC}(edge)$, which is the sum of the above two information content measures.

Given the shortest path between e_1 and e_2, the structure-based measure is defined as the sum of each edges' weights $edge_1$, $edge_2$, …, $edge_k$. The length of the shortest path is limited in two hops (Hulpus et al., 2013, Navigli & Ponzetto, 2012). In other words, k is less than or equal to two. The structure-based measure is defined to be:

$$struct(e_1, e_2) = \sum_{1}^{k} w_{jointIC}\ (edge_i) \tag{6}$$

If there are multiple shortest paths between e_1 and e_2, the one that has the maximum accumulated weight is selected.

3.4.2. Entity weighting

After calculating the semantic distances of all entity pairs by context- and structure-based measures, the two evaluation results of relatedness can be integrated into an overall result. The linear combination of the two measures is used as the overall semantic distance of any two entities globally. Formally,

$$sim_{overall}(e_1, e_2) = \lambda \times ctxt(e_1, e_2) + (1 - \lambda) \times struct(e_1, e_2) \tag{7}$$

where λ is the weight parameter.

According to Borgatti (2005), the importance of entities would be quantified by the sum of relatedness of one entity to all other entities if the relatedness of any two entities is known. Formally, for each entity extracted from sampled documents of a collection, the sum of its overall semantic distances to all other entities is defined as the entity weight. Given the set of entities V, the weight of an entity e is:

$$w(e) = \frac{1}{|V|} \sum_{e_j \in V} sim_{overall}(e, e_j) \tag{8}$$

All entities would be assigned with weights through Eq. (8), which aims to identify entities that can reflect more relevance to the aspects of a collection. In addition, it is a part of the composition of the query-collection similarity metric, which will be described in Section 3.5.

We summarize the process of collection representation in Algorithm 1. Given an entity set extracted from a collection, $E = \{e_1, e_2, \ldots, e_n\}$, and the weight parameter λ, we perform the following steps:

Step 1: lines 1–10. For each pair of entities e_i and e_j in E, we firstly find the shortest path between them in DBpedia using breadth-first search algorithm. If the length is larger than 2, the overall semantic distance between them is assigned with 0. $matrix[i][j]$ is used to store this value. If the length is less than or equal to 2, the context and structure semantic distance are calculated in line 7

and line 8, respectively. Finally, the overall semantic distance is calculated in line 9 and stored in $matrix[i][j]$.
Step 2: lines 11–20. For each entity e_i, the sum of its overall semantic distances to all other entities e_j $(i \neq j)$ is calculated and denoted as w_i. Then, the w_i is divided by the total number of entities in E. Finally, the entity e_i and its weight are added to the weighted entity set WE.

3.5. Query expansion

Query expansion is an important part of collection selection. In our collection selection method, collection and query are represented by entities. It is a serious problem that no entity or fewer entities can be found in a query. To deal with the problem, a DBpedia based query expansion method (Guisado-Gámez et al., 2014) is integrated into our method. Since the method is designed for image retrieval, some adaptation work is needed. The phases of (Guisado-Gámez et al., 2014) are in the following:

Step 1: relevant article selection. Given original query θ and the its description κ, two sets of relevant titles would be found in Wikipedia through Lucene,[3] which are denoted as R_θ and R_κ, respectively.
Step 2: path analysis. According to the structure of total DBpedia graph, this phase builds two graphs for R_θ and R_κ to exclude unrelated entities. For each entity in R_θ, it computes all shortest paths that reach all entities in R_κ. Once all paths are computed, they are ranked in a descending order based on the score of each path. The entities that are in highly scored paths are retained. Otherwise, the entities are removed. Formally, given a path of $P = a_1 \rightarrow a_2 \rightarrow \ldots \rightarrow a_s$, its score is:

$$Score(P) = \frac{1}{s}\left(\sum_{i=1}^{s} f(a_i, \theta) + f(a_i, \kappa)\right) \tag{9}$$

where $f(x, y)$ is a function that counts the number of common terms between x and y, and s is the length of P.
Step 3: community search. It uses a community search algorithm to enrich the previously computed paths with semantically strong neighbors of DBpedia entities.
Step 4: query building. It combines the original query θ, the synonyms Q_R of query terms, and the relevant entities Q_T in the communities already found. The final query Q_E is defined as follows: $Q_E = \langle \theta, Q_R, Q_T \rangle$.

There are two main problems during integration. One problem is that it requires users provide original query and its description as inputs. It is impractical to get the extended description of a query. In our work, only the original query is used to find some relevant articles, denoted as R_θ too, and one graph is built for R_θ, where the nodes are entities in R_θ and the edges are the relations between them in DBpedia. The main difference is that paths are enumerated between any two nodes in the graph. Then the paths are ranked according to the score function defined in Eq. (10), which is a variant of Eq. (9).

$$Score'(P) = \frac{1}{s}\left(\sum_{i=1}^{s} f(a_i, \theta)\right) \tag{10}$$

The other problem is that the Q_E derived in Step 4 is not suitable in our method, as there are many same and redundant terms in Q_E, and the final query is represented by entities rather than query terms in our method. Therefore, entities are found in the corresponding query terms of Q_E, and redundant entities are removed. Then the final query denoted as Q'_E consists of two parts that are original entities C_θ found in θ and the extended entities C_E found in Q_R, Q_T, respectively. In other words, $Q'_E = \langle C_\theta, C_E \rangle$.

However, the expanded query terms are only a supplement to original query terms in query expansion. If the new query depends on expanded query terms too much, it would lead to significant deviation from original query intention. Therefore, it is necessary to set different weights for expanded query terms according their relatedness with original query. To distinguishing the entities in C_θ, C_E and avoid query drift, the entities are weighted as follows:

$$w'(e) = \begin{cases} \gamma \times \frac{1}{|C_\theta|}, & (e \epsilon C_\theta) \\ (1-\gamma) \times \frac{L-i}{\sum_{i \leq L} i} \times \frac{1}{|H_i|}, & (e \epsilon H_i, H_i \subseteq C_E) \end{cases} \tag{11}$$

where γ is the weight parameter. When entities belong to C_θ, their weights are equal. When entities belong to C_E, they are weighted differently according to the height in the hierarchy that is built from the community search in Step 3. The first level is formed by the entities from the computed paths. Like in (Guisado-Gámez et al., 2014), the ith level of a hierarchy of L levels $(1 < i \leq L)$ is formed by the entities that have a link from an entity in the $(i-1)$th level. H_i is the set of entities in the ith level. If extended entities are in H_L, they will have a weight equal to 0. This can remove unimportant extended entities.

3.6. Query-Collection similarity metric

The goal of collection selection is to select a small number of information collections that contain the largest number of relevant

[3] http://lucene.apache.org/

documents for a query. After the entities of a collection are weighted and query terms are expanded, collections need to be ranked by calculating the similarity between query terms and the weighted entities of a collection.

Most existing collection selection methods use the keyword-based similarity measures to calculate the relevance between a collection and a query. The TF-IDF measure is a typical example among them. Given 'iPhone' as a query, a collection would get a high score if its indexed words contain 'iPhone'. However, this type of measures would give a collection a low score if the collection's indexed words do not contain 'iPhone' but contain 'Apple', 'Phone', etc. Actually, such a collection should be assigned with a high score. To address this issue, query-collection similarity metric should consider the semantic distance between query words and indexed words in a collection, which can be calculated by Eq. (7).

The semantic information of a collection is represented as a weighted entity set. However, the weight of an entity is calculated by the combination of context- and structure-based measures without considering entity frequency. In traditional information retrieval models, an entity is more important in a collection if it appears more in the collection but fewer in other collections. Therefore, the entity frequency is considered as the second impact factor for query-collection similarity metric, which is defined to be:

$$f(e,\ c) = \frac{|e|}{\sum_{e_i \in s_c} |e_i|},\ (e \in s_c) \tag{12}$$

where s_c consists of documents sampled from the collection c by query-based sampling (Callan & Connell, 2001), $|e|$ is the time that the entity e appears in s_c.

Collection size is also an important factor in evaluating the relevance between a collection and a query. Assume that there are two collections containing the same number of relevant sampled documents for a query and the same number of sampled documents in CSI. It is clear that the larger collection contains more relevant documents. Therefore, the collection size is considered as the third impact factor for query-collection similarity metric. To normalize the collection size, the size of the largest collection (denoted by $|c_{max}|$) is divided. It is calculated as shown below:

$$\varepsilon_c = \frac{|c|}{|c_{max}| \times\ |s_c|} \tag{13}$$

where $|c|$ is the total number of documents in the collection c. $|s_c|$ is the number of sampled documents in c. ε_c is also called sampling factor.

In addition to the three factors mentioned above, there are still two factors to consider: collection entity weight and query entity weight. Given a query q and a collection c, the query-collection similarity metric is defined formally as:

$$Rel(q,\ c) = \varepsilon_c \times \sum_{i=1}^{m} w_i' \times \frac{\sum_{j=1}^{n} w_j \times f(e_j,\ c) \times sim_{overall}(e_i,\ e_j)}{\sum_{j=1}^{n} w_j} \tag{14}$$

where e_i $(1 \le i \le m)$ is the entity in the expanded query and e_j $(1 \le j \le n)$ is the entity in s_c. In addition, w_i' and w_j are their weights, respectively.

In summary, the process of query-collection similarity metric is shown in Algorithm 2. Given two weighted entity sets of collection and query, the steps are as follows:

Step 1: Lines 1–13. For each entity e_i in WQ and each entity e_j in WE, if the length between them is larger than 2, the overall semantic distance $sim_{overall}(e_i,\ e_j)$ is assigned with 0. If the length is less than or equal to 2, the context and structure semantic distance are calculated in line 11 and line 12, respectively. Finally, the $sim_{overall}(e_i,\ e_j)$ is calculated in line 13.
Step 2: Lines 14–18. The collection score is calculated based on the overall semantic distance, collection entity weight, collection entity frequency in line 14, query entity weight in line 15, and samping factor in line 17.

4. Experiments

In this section, we describe the experimental details and compare the performances of KBCS and other methods. We first describe the dataset used in experiments, and then elaborate on experimental setup. Finally, we present experimental results and discuss findings.

4.1. Dataset

The dataset used in experiments is ClueWeb09 Dataset Category B (CW09-CatB). It was created to support research on information retrieval and related human language technologies, which consists of about 50 million English webpages. The dataset is used by several tracks of the TREC conference. If the dataset is divided randomly or base on chronology, the performance of collection selection methods would be affected and experimental results would be of questionable validity (D'Souza, Thom, & Zobel, 2004, Kulkarni & Callan, 2010). Therefore, we divide this dataset into 100 collections by applying topical clustering algorithm (Kulkarni & Callan, 2010). The summary statistics of the dataset are shown in Table 2.

In order to fully verify the effectiveness of our method, the evaluation queries (WT09: 1–50) and their corresponding relevant documents are chosen and summarized in Table 3.

Table 2
Statistics about the dataset and collections.

DataSet(Collections)	Documents (K)	Avg Doc Len	Collection		
			Min Docs (K)	Max Docs (K)	AVG Docs (K)
CW09-CatB(100)	50,220	918	3	9542	502

Table 3
Statistics about the query set (In the entries of the form WT = Web Track).

QuerySet(Queries)	Avg Qry Len	Avg Rel Docs / Qry
WT09:1–50(50)	2.1	262.3

4.2. Experimental setup

To simulate a distributed information retrieval environment, each collection is converted to a Lucene index that serves as a search engine. Collection selection decides which search engines to retrieve. Then for each query, top n documents are returned from each search engine and merged directly. Finally, the top n documents in merged list are shown to users.

A CSI is created for the dataset by using query-based sampling algorithm (Callan & Connell, 2001). An initial query is extracted randomly from the titles of Wikipedia articles. In each round of sampling, the top five documents returned from each collection are built in CSI. As a result, the sampled size of each collection is 400 documents.

In the environment of uncooperative distributed information retrieval, the statistics of each collection are transparent to retrieval proxy server. It is necessary to estimate the size of each collection (the number of documents) used in KBCS (e.g., Eq. (12)). The capture-history method (Shokouhi, Zobel, Scholer, & Tahaghoghi, 2006), which is a classical and effective estimation algorithm, is utilized to capture the collection size.

Since the vast majority of sampling based methods are superior to dictionary based methods, and our method is also a sampling based method, ReDDE (Si & Callan, 2003) and CRCS (Shokouhi, 2007) are selected as baselines. Since CRCS(e) is significantly better than CRCS(l), CRCS(e) is chose, which is denoted as CRCS in the following for simplicity. Another state-of-the-art method called Rank-S is chose as one baseline. Rank-S is one of the SHiRE (Kulkarni et al., 2012) algorithms, which showed the best performance. In experiments, the optimal parameters of each method are set according to the original paper. In ReDDE, *ratio* is 0.003. In CRCS(e), α and β are 1.2 and 2.8, respectively. In Rank-S, B is 50.

Based on the above mentioned baselines, KBCS (sim) and KBCS are compared with them. Specifically, KBCS (sim) contains query-collection similarity metric without query expansion. KBCS contains not only query-collection similarity metric but also query expansion.

For the comparative analysis of different collection selection methods, the evaluation metric of P@n is employed to measure the average search accuracy of all queries. The greater value of P@n means the higher accuracy of search results. In an information retrieval system, per page usually presents 10 results. If the first page contains few related results for a query, it would affect the satisfaction degree of users. Therefore, we choose P@10 as the evaluation metric.

4.3. Results and discussion

4.3.1. The impact of parameters

The KBCS method contains two parameters, which are λ in Eq. (7) and γ in Eq. (11), respectively. The parameter λ denotes the impact of context-based semantic distance in overall semantic distance between entities. The parameter γ denotes the impact of entities in original query. To get the optimal value of λ, experiments are conducted on KBCS(sim), in which the weight of w_i' in Eq. (14) is set equally according to the number of entities found in query terms. After the optimal value of λ is got, experiments are conducted on KBCS to determine the optimal value of γ.

In the experiment, we increase the value of λ from 0 to 1 with a fixed step of 0.1. The precision of KBCS(sim) is shown in Fig. 2. The experimental results show that when $\lambda = 0.4$, the precision of KBCS(sim) is maximized, which reflects that the structure-based semantic distance is more important in calculating the overall semantic distance. Therefore, the value of λ is set to 0.4.

In order to determine the optimal value of γ, we firstly set the value of λ to 0.4 and then compare the precision of KBCS under different values of γ (from 0 to 1, with a fixed step of 0.1). The experimental results are shown in Fig. 3. When $\gamma \leq 0.6$, the precision of KBCS increases consistently and reaches the maximum ($\gamma = 0.6$). It indicates that the weight of entities in original query has a greater impact on query-collection similarity metric. Therefore, the value of γ is set to 0.6.

4.3.2. Comparison with other methods

There are two sets of experiments to verify the effectiveness of our method. One set compares KBCS with baselines, including ReDDE, CRCS and Rank-S, to demonstrate KBCS's performance. The other set compares KBCS with KBCS(sim) to evaluate the

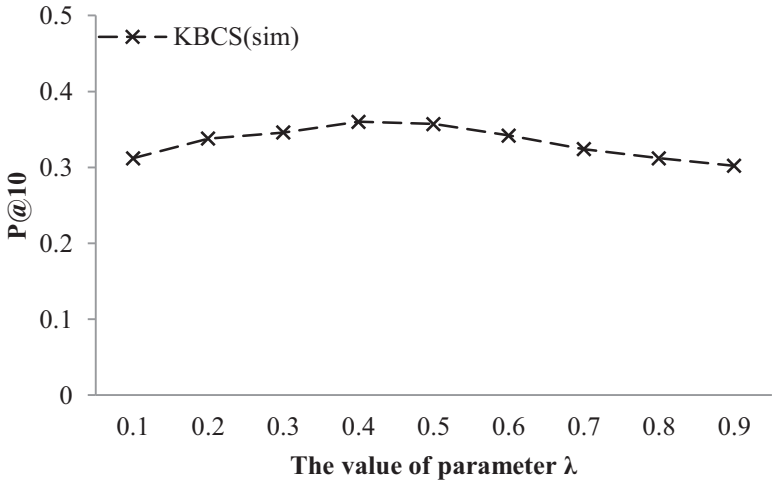

Fig. 2. The precision of KBCS(sim) under different values of λ when it selects 5 collections.

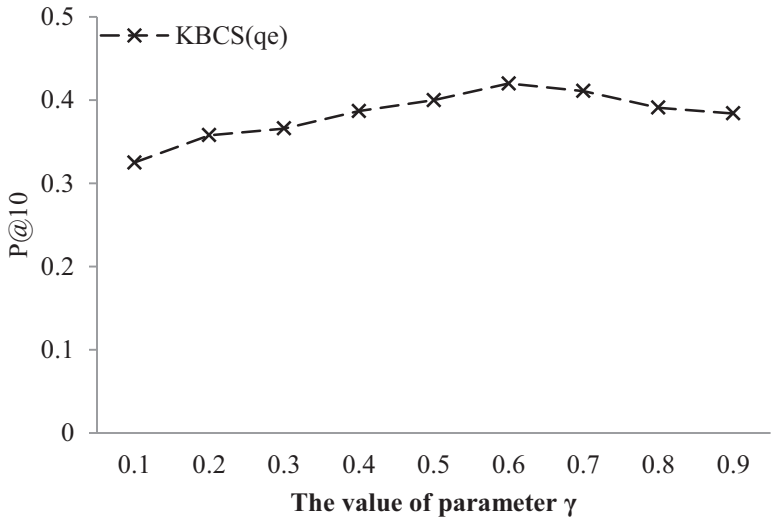

Fig. 3. The precision of KBCS under different values of γ when it selects 5 collections.

contribution of query expansion.

The performances of these methods are shown in Table 4. The mean of precision is recorded for each method under different numbers of selected collections (i.e., *k*). Furthermore, to statistically measure the significance of performance difference, pairwise *t*-

Table 4
Precision (P@10) comparison for ReDDE, CRCS, Rank-S, KBCS(sim) and KBCS under different numbers of selected collections ($1 \le k \le 10$). * indicates whether KBCS is statistically superior to the compared method.

K	Methods				
	ReDDE	CRCS	Rank-S	KBCS(sim)	KBCS
1	0.176*	0.212*	0.263*	0.332*	**0.340**
2	0.236*	0.212*	0.272*	0.344*	**0.368**
3	0.244*	0.208*	0.316*	0.348*	**0.380**
4	0.244*	0.236*	0.316*	0.340*	**0.370**
5	0.244*	0.252*	0.357*	0.360*	**0.420**
6	0.252*	0.252*	0.365*	0.380*	**0.400**
7	0.276*	0.252*	0.353*	0.376*	**0.394**
8	0.300*	0.260*	0.348*	0.400*	**0.422**
9	0.336*	0.248*	0.357*	0.408*	**0.414**
10	0.356*	0.232*	0.361*	0.404*	**0.420**

Algorithm 1
Collection representation.

Input: an entity set extracted from a collection, $E = \{e_1, e_2, ..., e_n\}$, the weight parameter λ
Output: a weighted entity set $WE = \{e_1: w_1, e_2: w_2, ..., e_n: w_n\}$
1. initialize the set of weighted entities $WE = \varnothing$, the weight matrix $matrix[n][n] = [0, 0, ..., 0]$
2. **for all** $e_i \in E$, $e_j \in E$, $i \leq j$ **do**
3. $l \leftarrow$ the length of shorted path between e_i and e_j
4. **if** $l > 2$ **then**
5. $matrix[i][j] \leftarrow 0$
6. **else**
7. $ctxt(e_i, e_j) \leftarrow$ use Eq. (2)
8. $struct(e_i, e_j) \leftarrow$ use Eq. (6)
9. $sim_{overall}(e_i, e_j) \leftarrow$ use Eq. (7)
10. $matrix[i][j] \leftarrow sim_{overall}(e_i, e_j)$
11. **for all** $e_i \in E$ **do**
12. $w_i \leftarrow 0$
13. **for all** $e_j \in E$ **do**
14. **if** $i \leq j$ **then**
15. $w_i \leftarrow w_i + matrix[i][j]$
16. **else**
17. $w_i \leftarrow w_i + matrix[j][i]$
18. $w_i \leftarrow w_i/n$
19. add e_i and w_i to WE
20. **return** WE

Algorithm 2
Query-collection similarity metric.

Input: a weighted entity set of collection WE, and a weighted entity set of query WQ
Output: the collection score
1. initialize the collection score, $score = 0$
2. **for all** $e_i \in WQ$, $i \leq m$ **do**
3. $sum \leftarrow 0$
4. $temp \leftarrow 0$
5. **for all** $e_j \in WE$, $i \leq n$ **do**
6. $sum \leftarrow sum + w_j$
7. $l \leftarrow$ the length of shorted path between e_i and e_j
8. **if** $l > 2$ **then**
9. $sim_{overall}(e_i, e_j) \leftarrow 0$
10. **else**
11. $ctxt(e_i, e_j) \leftarrow$ use Eq. (2)
12. $struct(e_i, e_j) \leftarrow$ use Eq. (6)
13. $sim_{overall}(e_i, e_j) \leftarrow$ use Eq. (7)
14. $temp \leftarrow temp + w_j \times f(c, e_j) \times sim_{overall}(e_i, e_j)$
15. $temp \leftarrow w_i' \times temp/sum$
16. $score \leftarrow score + temp$
17. $score \leftarrow score \times \varepsilon_c$
18. **return** $score$

tests at 95% significance level are conducted between the results of these methods.

4.3.2.1. Compare KBCS with baselines. The experimental results show that KBCS has a clear advantage while baselines have similar performance. KBCS achieves an average improvement of 46% on precision. Although there is no significant difference between ReDDE and CRCS, ReDDE performs better than CRCS in the condition of selecting more than 2 collections. Since the topical clustering algorithm applied to partition collections results in skewed sizes of collections, larger collections usually have a higher number of relevant documents. ReDDE prefers to select larger collections, which in this case would perform better. Unlike ReDDE, CRCS differentiates the scores of different documents, which is not biased, i.e., not giving larger collections higher scores. Therefore, CRCS performs poorer in this case, and it is significantly worse than others when $k \geq 8$. Rank-S shows stronger effectiveness than ReDDE and CRCS. However, it does not achieve stable performance improvement when $k \geq 5$. This might be because some relevant documents are lower-ranked, which would result in less contribution to their own collections scores.

Another finding in the result is that KBCS can achieve a high precision while choosing few collections. This characteristic of KBCS can substantially reduce network resource consumption. Since P@10 does not change greatly when $k \geq 5$, it is reasonable for KBCS to select fewer collections than the other three methods. With the increase of k, the precision of CRCS and Rank-S increases slowly or even decreases. Although the precision of ReDDE increases continuously and stably (due to its preference to larger collections), its performance cannot exceed KBCS.

In summary, the advantages of our method are as follows: first, all entities in sampled documents are weighted to represent the

semantic information of a collection, and the entities related to the query would have a greater contribution to collection score. While, ReDDE and CRCS only estimate collection score by a number of top-ranked documents returned from CSI without considering the impact of other lower-ranked documents. Although Rank-S takes lower-ranked documents into account, it dampens or even ignores their votes to collections scores. Second, our method is more fine-grained on the query-collection similarity metric that calculates the semantic distance between query entities and collection entities, while the other three methods calculate only from the perspective of relevant documents.

4.3.2.2. Compare KBCS with KBCS(sim). To evaluate the effectiveness of query expansion on collection selection, KBCS is compared with KBCS(sim). As is shown in Table 4, KBCS performs better than KBCS(sim) in all conditions of k. Compared with KBCS(sim), the precision of KBCS is improved by an average of 6%. It indicates that query expansion is critical to collection selection. This is because query would contain rich entities through query expansion, and collections that contain more entities related with query entities would get a higher score. The results suggest that except collection representation and query-collection similarity metric, query expansion is a promising way to further improve the performance of collection selection.

5. Conclusions and future work

In this paper, we present KBCS that represents collection as a weighted entity set based on context- and structure-based measures, and ranks collections according to a query-collection similarity metric that considers sampling factor, collection entity frequency, collection entity weight, and query entity weight. Context and structure semantic information encoded in DBpedia is exploited, and a DBpedia based query expansion method is also integrated to enrich entities found in query terms. We evaluate KBCS on a large dataset of CW09-CatB that is partitioned into topical collections, and experimental results demonstrate the effectiveness of KBCS.

In the future, we will take other semantic measures (e.g., content-based measure) to weight the entities in CSI accurately, and combine documents score and entities weight to improve the query-collection similarity metric. Furthermore, we will apply context- and structure-based measures to estimate the quality of expanded entities to improve the performance of community search in query expansion, on which the weights assigned to expanded entities can be optimized.

Acknowledgments

This work was funded by China Knowledge Centre for Engineering Sciences and Technology (No. CKCEST-2014-1-5), the National Natural Science Foundation of China (No. 61332017), the Science and Technology Department of Zhejiang Province (No. 2015C33002).

References

Aly, R., Hiemstra, D., & Demeester, T. (2013). Taily: Shard selection using the tail of score distributions. *Proceedings of the 36th international ACM SIGIR conference on research and development in information retrieval* (pp. 673–682). .

Arguello, J., Elsas, J. L., Callan, J., & Carbonell, J. G. (2008). Document representation and query expansion models for blog recommendation. *Proceedings of international AAAI conference on web and social media. 2008*. *Proceedings of international AAAI conference on web and social media* (pp. 1–). .

Baillie, M., Carman, M., & Crestani, F. (2011). A multi-collection latent topic model for federated search. *Information Retrieval, 14*, 390–412.

Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., et al. (2009). DBpedia-A crystallization point for the web of data. *Web Semantics: Science, Services and Agents on the World Wide Web, 7*, 154–165.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet allocation. *Journal of Machine Learning Research, 3*, 993–1022.

Bollacker, K., Evans, C., Paritosh, P., Sturge, T., & Taylor, J. (2008). Freebase: A collaboratively created graph database for structuring human knowledge. *Proceedings of the 2008 ACM SIGMOD international conference on management of data* (pp. 1247–1250). .

Borgatti, S. P. (2005). Centrality and network flow. *Social Networks, 27*, 55–71.

Callan, J. (2002). *Distributed information retrieval. Advances in information retrieval*. Springer127–150.

Callan, J., & Connell, M. (2001). Query-based sampling of text databases. *ACM Transactions on Information Systems, 19*, 97–130.

Callan, J. P., Lu, Z., & Croft, W. B. (1995). Searching distributed collections with inference networks. *Proceedings of the 18th international ACM SIGIR conference on research and development in information retrieval* (pp. 21–28). .

Carpineto, C., & Romano, G. (2012). A survey of automatic query expansion in information retrieval. *ACM Computing Surveys, 44*, 1.

Chowdhury, G. (2010). *Introduction to modern information retrieval.* Facet publishing.

Cilibrasi, R. L., & Vitanyi, P. M. (2007). The Google similarity distance. *IEEE Transactions on Knowledge and Data Engineering, 19*.

Crestani, F., & Markov, I. (2013). Distributed information retrieval and applications. *Proceedings of European conference on information retrieval* (pp. 865–868). .

Dang, V., & Croft, B. W. (2010). Query reformulation using anchor text. *Proceedings of the 3rd ACM international conference on web search and data mining* (pp. 41–50). .

D'Souza, D., Thom, J. A., & Zobel, J. (2004). Collection selection for managed distributed document databases. *Information Processing and Management, 40*, 527–546.

Eiron, N., & McCurley, K. S. (2003). Analysis of anchor text for web search. *Proceedings of the 26th international ACM SIGIR conference on research and development in informaion retrieval* (pp. 459–460). .

Francès, G., Bai, X., Cambazoglu, B. B., & Baeza-Yates, R. (2014). Improving the efficiency of multi-site web search engines. *Proceedings of the 7th ACM international conference on web search and data mining* (pp. 3–12). .

Gabrilovich, E., & Markovitch, S. (2007). Computing semantic relatedness using Wikipedia-based explicit semantic analysis. *Proceedings of the 20th international joint conference on artificial intelligence* (pp. 1606–1611). .

Gravano, L., & Garcia-Molina, H. (1995). Generalizing GlOSS to vector-space databases and broker hierarchies. *Proceedings of the 21th international conference on very large data bases* (pp. 78–89). .

Guisado-Gámez, J., Dominguez-Sal, D., & Larriba-Pey, J.-L. (2014). Massive query expansion by exploiting graph knowledge bases for image retrieval. *Proceedings of international conference on multimedia retrieval* (pp. 33). .

Hoffart, J., Suchanek, F. M., Berberich, K., & Weikum, G. (2013). YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia. *Artificial Intelligence, 194*, 28–61.

Hulpus, I., Hayes, C., Karnstedt, M., & Greene, D. (2013). Unsupervised graph-based topic labelling using dbpedia. *Proceedings of the 6th ACM international conference on web search and data mining* (pp. 465–474). .

Kim, Y., Callan, J., Culpepper, J. S., & Moffat, A. (2016). Load-balancing in distributed selective search. *Proceedings of the 39th international ACM SIGIR conference on research and development in information retrieval* (pp. 905–908). .

Kulkarni, A., & Callan, J. (2010). Document allocation policies for selective searching of distributed indexes. *Proceedings of the 19th ACM international conference on information and knowledge management* (pp. 449–458). .

Kulkarni, A., & Callan, J. (2015). Selective search: Efficient and effective search of large textual collections. *ACM Transactions on Information Systems, 33*, 17.

Kulkarni, A., Tigelaar, A. S., Hiemstra, D., & Callan, J. (2012). Shard ranking and cutoff estimation for topically partitioned collections. *Proceedings of the 21st ACM international conference on information and knowledge management* (pp. 555–564). .

Mendes, P. N., Jakob, M., García-Silva, A., & Bizer, C. (2011). DBpedia spotlight: Shedding light on the web of documents. *Proceedings of the 7th international conference on semantic systems* (pp. 1–8). .

Mendoza, M., Marín, M., Gil-Costa, V., & Ferrarotti, F. (2016). Reducing hardware hit by queries in web search engines. *Information Processing and Management, 52*, 1031–1052.

Navigli, R., & Ponzetto, S. P. (2012). BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence, 193*, 217–250.

Ni, Y., Xu, Q. K., Cao, F., Mass, Y., Sheinwald, D., & Zhu, H. J. (2016). Semantic documents relatedness using concept graph representation. *Proceedings of the 9th ACM international conference on web search and data mining* (pp. 635–644). .

Paltoglou, G., Salampasis, M., & Satratzemi, M. (2011). Modeling information sources as integrals for effective and efficient source selection. *Information Processing and Management, 47*, 18–36.

Schuhmacher, M., & Ponzetto, S. P. (2014). Knowledge-based graph document modeling. *Proceedings of the 7th ACM international conference on web search and data mining* (pp. 543–552). .

Shokouhi, M. (2007). Central-rank-based collection selection in uncooperative distributed information retrieval. *Proceedings of the European conference on information retrieval* (pp. 160–172). .

Shokouhi, M., Zobel, J., Scholer, F., & Tahaghoghi, S. M. (2006). Capturing collection size for distributed non-cooperative retrieval. *Proceedings of the 29th international ACM SIGIR conference on research and development in information retrieval* (pp. 316–323). .

Shokouhi, M., Zobel, J., Tahaghoghi, S., & Scholer, F. (2007). Using query logs to establish vocabularies in distributed information retrieval. *Information Processing and Management, 43*, 169–180.

Si, L., & Callan, J. (2003). Relevant document distribution estimation method for resource selection. *Proceedings of the 26th international acm sigir conference on research and development in informaion retrieval* (pp. 298–305). .

Strohman, T., Metzler, D., Turtle, H., & Croft, W. B. (2005). Indri: A language model-based search engine for complex queries. *Proceedings of international conference on intelligent analysis* (pp. 2–6). .

Thomas, P., & Shokouhi, M. (2009). SUSHI: Scoring scaled samples for server selection. *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 419–426). .

Wauer, M., Schuster, D., & Schill, A. (2011). Integrating explicit semantic analysis for ontology-based resource selection. *Proceedings of the 13th International Conference on Information Integration and Web-based Applications and Services* (pp. 519–522). .

Witten, I., & Milne, D. (2008). An effective, low-cost measure of semantic relatedness obtained from Wikipedia links. *Proceeding of AAAI Workshop on Wikipedia and Artificial Intelligence* (pp. 25–30). .

Xu, J., & Croft, W. B. (1999). Cluster-based language models for distributed retrieval. *Proceedings of the 22nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 254–261). .

Yuwono, B., & Lee, D. L. (1997). Server ranking for distributed text retrieval systems on the internet. *Proceedings of international conference on database systems for advanced applications* (pp. 41–49). .