Trading Off Popularity for Diversity in the Results Sets of Keyword Queries on Linked Data

Ananya Dass(✉) and Dimitri Theodoratos(✉)

New Jersey Institute of Technology, Newark, USA
{ad292,dth}@njit.edu

Abstract. Keyword search is the most popular technique for querying the ever growing repositories of RDF graph data on the Web. However, keyword queries are ambiguous. As a consequence, they typically produce on linked data a huge number of candidate results corresponding to a plethora of alternative query interpretations. Current approaches ignore the diversity of the result interpretations and might fail to satisfy the users who are looking for less popular results. In this paper, we propose a novel approach for keyword search result diversification on RDF graphs. Our approach instead of diversifying the query results per se, diversifies the interpretations of the query (i.e., pattern graphs). We model the problem as an optimization problem aiming at selecting k pattern graphs which maximize an objective function balancing relevance and diversity. We devise metrics to assess the relevance and diversity of a set of pattern graphs, and we design a greedy heuristic algorithm to generate a relevant and diverse list of k pattern graphs for a given keyword query. The experimental results show the effectiveness of our approach and proposed metrics and also the efficiency of our algorithm.

1 Introduction

Keyword search is the most popular technique for querying RDF data on the Web because it frees the user from knowing a complex structured query language (e.g., SPARQL) and allows querying the data without having full or partial knowledge of its structure/schema. The convenience and flexibility of keyword search comes with a cost. Keyword queries are ambiguous. As a consequence, there is usually a huge number of candidate results of which very few are relevant to the user intent. Several approaches try to exploit structural or semantic characteristics of the data and/or query results in order to filter out irrelevant results. A better technique ranks the results in descending order of their estimated relevance [7,13]. The relevance is usually estimated based on scoring functions which employ statistics-based IR-style metrics for flat documents (e.g., tf*idf or PageRank) adapted to the structural characteristics of the data. Ranking can be complemented with top-k processing, wherein gains can be achieved in the processing time by avoiding the computation of results which are not expected to be in the top-k positions [20].

DOI: 10.1007/978-3-319-60131-1_9

Even though, the statistics-based metrics can be effective in returning the most popular results, they fail to capture the diversity of the result set and may dissatisfy the users who look for less popular results [11,12]. For instance, a user issuing the query "python" could be interested in searching about the snake "python", the "Python" programming language, or the Monty "Python" comedy group. If results are returned to the user based on the most plausible interpretation of the query (in this case "Python" programming language) then there is an inherent risk of leaving the user who is interested in "python" snake or in Monty "Python" comedy group. This problem is known as the over-specialization problem [18]. Diversifying the results retrieved for a keyword query could be a meaningful solution to this problem. By introducing diversity in the result set, the search mechanism can maximize the user's chance of finding at least one of the retrieved results relevant to her intent [6]. Additionally, even if a keyword query has a single, clearly defined interpretation, it can still be under-specified to some extent. For example, a user searching for "apple electronics" may be interested in laptops, desktops, or the best selling apple electronics, or sale on apple electronics, or service centers for apple electronics. Therefore, another motive for diversifying search results is to cover different aspects of the entire result space and enable the user to explore and find desired results [12].

Search result diversification is a well-studied problem in Information Retrieval and Recommendation Systems [12,14]. However, the problem of diversifying the results of keyword search over RDF graph data has remained under-addressed. In recent years, there is a proliferation of RDF repositories on the Web, and keyword search is commonly used for retrieving data from these repositories. While ranking ensures that the most popular results of a given keyword query are ranked on top, it is often the case that the top results tend to be homogeneous, making it difficult for users interested in less popular aspects to find results relevant to their intent. Thus, result diversity can play a big role in ensuring that the users get a broad view of the different aspects of the results and in satisfying a maximum number of users who are interested in different interpretations of the query.

Our Approach. In this paper, we propose a novel technique for diversifying keyword search results on RDF graph data. We formulate the diversification problem as an optimization problem over pattern graphs. Pattern graphs are structured queries which cluster together results with the same structural and semantic characteristics and represent alternate interpretations of a keyword query. By diversifying pattern graphs instead of query results, we address the data scalability problem of diversification since pattern graphs can be computed efficiently by exploiting a structural summary of the RDF data without exhaustively computing the query results. Further, by diversifying pattern graphs we diversify the alternative interpretations of the query. Given a positive integer k, our diversification approach aims at selecting a k-size set of pattern graphs which maximizes the number of pattern graphs which are relevant to at least one user intent. In order to do so, our approach trades off popularity for diversity.

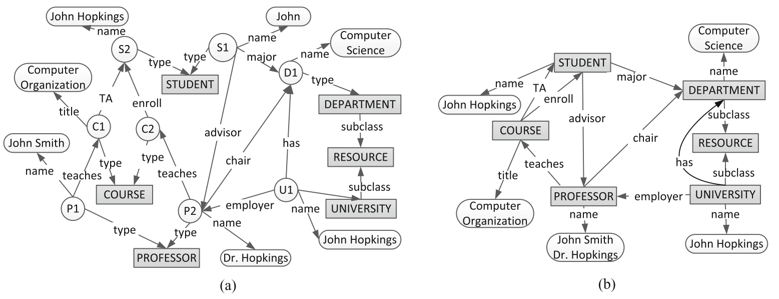


Fig. 1. (a) An RDF graph D, (b) The Structural Summary S of D.

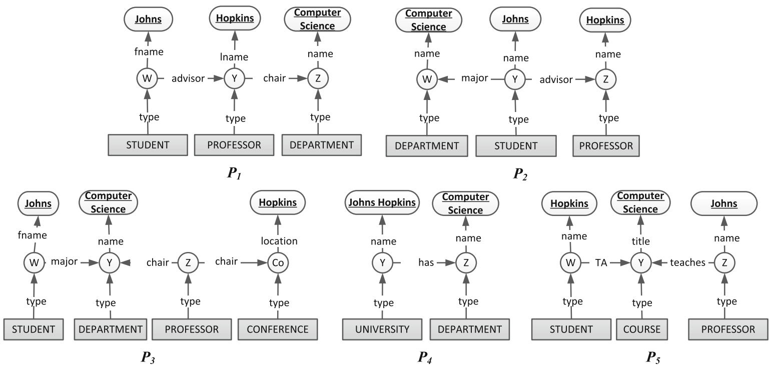


Fig. 2. Patterns graphs of the query $Q = \{$`Johns, Hopkins, Computer, Science`$\}$.

As an example, consider the RDF data graph D of Fig. 1(a), and its structural summary S in Fig. 1(b). Consider also the keyword query $Q = \{$`Johns, Hopkins, Computer, Science`$\}$ on D. Figure 2 shows five pattern graphs of Q computed over S. These pattern graphs are alternative interpretations of Q. For instance, the pattern graph P_1 interprets "Johns" as a student advised by Professor "Hopkins" who is the chair of the "Computer Science" department. Among those pattern graphs P_1, P_2, P_4 and P_5 provide meaningful interpretations while, pattern graph P_3 does not seem to be relevant to a user intent. Let us assume that the pattern graphs are ranked on popularity in descending order as follows: P_1, P_2, P_3, P_4, P_5. Let us also assume that we are required to return only three pattern graphs. If we select the relevant pattern graphs based on popularity, we are going to return the list (P_1, P_2, P_3). Therefore, we are returning only two pattern graphs which are relevant to a user intent. We can now try to also diversify the pattern graphs. We can do so by taking into account semantic dissimilarities between them. For instance, P_1 and P_2 interpret the keywords in the same way: "Johns" is a student, "Hopkins" is a professor and "Computer Science" is a department and they link these interpretations in a quite similar way. Pattern graph P_3 shares three out of four keyword interpretations with P_1 and P_2 (for the keywords "Johns", "Computer", and "Science") and some connections ("major"), therein displaying a certain degree of semantic similarity to P_1 and P_2. Further, the interpretations of the keywords in P_4 and P_5 and the connections between them have very little or nothing in common with those of the rest of the patterns graphs (or between them). Therefore, if we try to balance popularity and diversity in the selected three pattern graph set, most probably the pattern graphs P_1, P_4 and P_5 will be selected. In this case, all three returned pattern graphs are relevant to a user intent. We formalize this intuition in the following sections.

Contribution. The main contributions of the paper are the following:

- We formalize the problem of diversifying the pattern graphs returned by a keyword query on an RDF data graph. Exploiting pattern graphs addresses the scalability problem faced by keyword search approaches on RDF graphs. We define the problem as an optimization problem which aims at selecting a k-size set of pattern graphs that maximizes an objective function on relevance and diversity.
- In order to measure the relevance of a pattern graph to a keyword query, we devise a relevance metric. This metric exploits the *tf*idf* measure and popularity scores of the different semantic components of the pattern graph.
- We express the diversity of a set of pattern graphs as the average pairwise semantic distance between pattern graphs. To assess the pairwise pattern graphs semantic distance, we introduce an original metric. based on the similarities between the semantic interpretations of the query keywords in the pattern graphs and the way they are semantically connected but also on the dissimilarity between the concepts involved in the pattern graphs.
- To cope with a high complexity of the diversification problem, we design a greedy heuristic algorithm for computing a list of top-k pattern graphs trading off popularity for diversity.

We ran extensive experiments to evaluate and fine-tune the effectiveness of the approach and the proposed metrics and the efficiency of the algorithm.

2 Data Model and Pattern Graph Computation

Data Model. The Resource Description Framework (RDF) provides a framework for representing information about Web resources in a graph form. The RDF vocabulary includes elements that can be broadly classified into Classes, Properties, Entities and Relationships. All the elements are resources. Similarly to [7,8], our data model is an RDF graph defined as follows:

Definition 1 (RDF Graph). An *RDF graph* is a quadruple $G = (V, E, L, l)$:

V is a finite set of vertices, which is the union of three disjoint sets: V_E (representing entities), V_C (representing classes) and V_V (representing values).

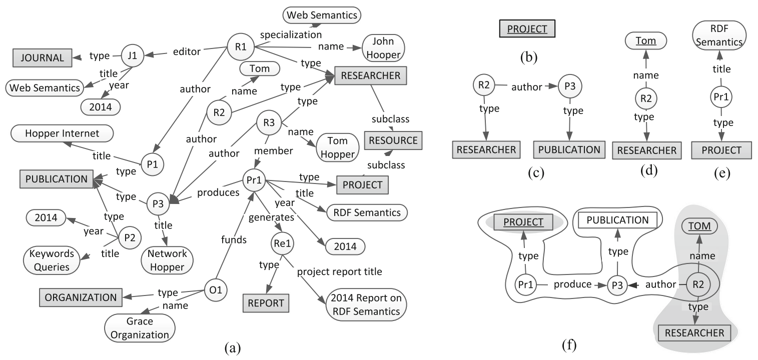


Fig. 3. (a) An RDF graph, (b), (c), (d) and (e) class, relationship, value and property matching constructs, respectively, (f) inter-construct connection and result graph.

E is a finite set of directed edges, which is the union of four disjoint sets: E_R (inter-entity edges called *Relationship* edges which represent entity relationships), E_P (entity to value edges called *Property* edges which represent property assignments), E_T (entity to class edges called *type* edges which represent entity to class membership) and E_S (class to class edges called *subclass* edges which represent class-subclass relationship).

L is a finite set of labels that includes the labels “type” and “subclass”.

l is a function from $V_C \cup V_V \cup E_R \cup E_P$ to L. That is, l assigns labels to class and value vertices and to relationship and property edges.

Entity and class vertex and edge labels are Universal Resource Identifiers (URIs). Vertices are identified by IDs which in the case of entities and classes are URIs. Every entity belongs to a class. Figure 3(a) shows an example RDF graph. For simplicity, vertex and edge identifiers are not shown in this example.

Query Language Semantics. A *query* Q on an RDF graph G is a set of keywords. A *keyword instance* of a keyword k in Q is a vertex or edge label in G containing k. The *answer* of Q on G is a set of result graphs of Q on G. Each result graph is a minimal subgraph of G involving at least one instance of every keyword in Q and is formally defined below. In order to facilitate the interpretation of the semantics of the keyword instances, every instance of a keyword in Q is matched against a small subgraph of G which involves this keyword instance and the corresponding class vertices. This subgraph is called *matching construct*. Figures 3(b), (c), (d) and (e) show a class, relationship, value and property matching construct, respectively, for different keyword instances in the RDF graph of Fig. 3(a). Underlined labels in a matching construct denote the keyword instances. Each matching construct provides information about the semantic context of the keyword instance under consideration. For instance, the matching construct of Fig. 3(d) shows that `Tom` is the name of an entity $R2$ of type Researcher.

A *signature* of Q is a function that matches every keyword k in Q to a matching construct of k in G. Given a query signature S, an *inter-construct connection* between two distinct matching constructs C_1 and C_2 in S is a simple path augmented with the class vertices of the intermediate entity vertices in the path (if not already in the path) such that: (a) one of the terminal vertices in the path belongs to C_1 and the other belongs to C_2, and (b) no vertex in the connection except the terminal vertices belong to a construct in S. Figure 3(f) shows an inter-construct connection between the matching constructs for keywords `Project` and `Tom` in the RDF graph of Fig. 3(a). The matching constructs are shaded and the inter-construct connection is circumscribed.

A subgraph of G is said to be *connection acyclic* if there is no cycle in the graph obtained by viewing its matching constructs as vertices and its inter-construct connections between them as edges. Given a signature S for Q on G, a *result graph* of S on G is a connected, connection acyclic subgraph of G which contains only the matching constructs in S and possibly inter-construct connections between them. A *result graph* for Q on G is a result graph for a signature of Q on G. Figure 3(f) shows a result graph for the query {`Project, Tom`} on the RDF graph of Fig. 3(a).

The Structural Summary and Pattern Graphs. In order to construct pattern graphs we use the structural summary of the RDF graph as in [10,20]. Intuitively, the structural summary is a graph that summarizes the RDF graph.

Definition 2 (Structural Summary). The *structural summary* of an RDF graph G is a vertex and edge labeled graph constructed from G as follows:

1. Merge every class vertex and its entity vertices into one vertex labeled by the class vertex label and remove all the type edges from G.
2. Merge all the value vertices which are connected with a property edge labeled by the same label to the same class vertex into one vertex labeled by the union of the labels of these value vertices. Merge also the corresponding edges into one edge labeled by their label.

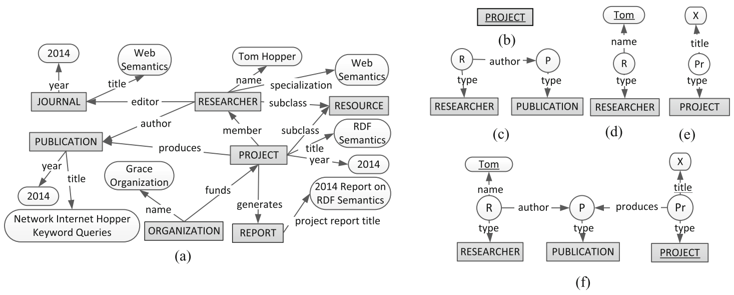


Fig. 4. (a) Structural Summary G', (b), (c), (d) and (e) are matching constructs for keywords in the keyword query `Q1={Tom, author, Project, title}` on G' (f) Pattern Graph of Q on G'.

3. Merge all the relationship edges between the same class vertices which are labeled by the same label into one edge with that label.

Figure 4(a) shows the structural summary for the RDF graph G of Fig. 3(a). Similarly to matching constructs on the data graph we define matching constructs on the structural summary. Since the structural summary does not have entity vertices, a matching construct on a structural summary possess one distinct entity variable vertex labeled by a distinct variable for every class vertex and a distinct value variable for every value vertex label which does not contain a keyword instance. Figure 4(b), (c), (d), and (e) show the class, relationship, value and property matching constructs for the keywords "Project", "author", "Tom", and "title", respectively, on the structural summary of Fig. 4(a).

Pattern graphs are the subgraphs of the structural summary, strictly consisting of one matching construct for every keyword in the query Q and the connections between them without these connections forming a cycle.

Definition 3 (Pattern Graph). A *(result) pattern graph* for a keyword query Q is a graph similar to a result graph for Q, with the following two exceptions:

(a) The labels of the entity vertices in the result graph, if any, are replaced by distinct variables in the pattern graph. These variables are called *entity variables* and they range over entity labels.
(b) The labels of the value vertices are replaced by distinct variables whenever these labels are not the keyword instances in the result graph. These variables are called *value variables* and they range over value labels in the RDF graph.

Figure 4(f) shows an example of a pattern graph, for the keyword query $Q = \{$`Tom, author, project, title`$\}$ on the RDF graph of Fig. 3(a). This pattern graph is computed over the structural summary of Fig. 4(a) combing the matching constructs of Fig. 4(b), (c), (d) and (e). Labels R, P, and Pr are entity variables and X is a value variable.

Given a keyword query Q over an RDF data graph G, we first find all the matching constructs for all the keywords in Q on the structural summary G' and then generate all the pattern graphs on G' for all possible signatures of Q.

3 Balancing Relevance and Diversity

We provide in this section a formal definition of the problem we address and then elaborate on its components: how to assess the relevance and the diversity of sets of pattern graphs.

3.1 Problem Statement

Our goal is to provide the user with a set of pattern graphs which is relevant and diverse. To this end, we define the problem as an optimization problem. Let

G denote an RDF data graph, Q be a keyword query on G, $\mathcal{P}$ be the set of pattern graphs of Q on G and k be a positive integer. Given a subset $\mathcal{S}$ of $\mathcal{P}$, let $relevance(\mathcal{S}, Q)$ denote the relevance of S with respect to Q, and $diversity(\mathcal{S})$ denote the diversity of set $\mathcal{S}$. We aim at selecting a subset $\mathcal{S}$ of $\mathcal{P}$ which maximizes the objective function $\alpha * relevance(\mathcal{S}, Q) + (1 - \alpha) * diversity(\mathcal{S})$, where $\alpha \in [0, 1]$, is a parameter which tunes the importance of relevance and diversity. In other words,

$$\mathcal{S} \in \text{ arg max}_{\mathcal{S}' \subseteq \mathcal{S},\ |\mathcal{S}'|=k}(\alpha * relevance(\mathcal{S}', Q) + (1 - \alpha) * diversity(\mathcal{S}'))$$

The tuning parameter α allows to give more importance to the relevance or diversity of the pattern graph set to be selected. If $\alpha = 1$, the selected pattern graph set will have the most relevant pattern graphs without considering diversity. If $\alpha = 0$, the pattern graph set will be selected solely based on its diversity.

We assume that the relevance of one pattern graph is independent of the relevance of another pattern graph to Q. The relevance of a set of pattern graphs $\mathcal{S}'$ of size k to a keyword query Q is the average relevance of its pattern graphs:

$$relevance(\mathcal{S}', Q) = 1/k * (\textstyle\sum_{P \in \mathcal{S}'} relevance(P, Q)\ )$$

where $relevance(P, Q) \in [0, 1]$ and denotes the relevance of pattern graph P to Q. The diversity of $\mathcal{S}'$ is defined as:

$$diversity(\mathcal{S}') = \textstyle\sum_{P_i, P_j \in \mathcal{S}', P_i \neq P_j} dist(P_i, P_j)/k(k-1)$$

where $dist(P_i, P_j)$ denotes the semantic distance between pattern graphs P_i and P_j; $dist(P_i, P_j) \in [0, 1]$. Dividing the sum by the total number of pattern graph pairs, normalizes $diversity(\mathcal{S}')$ in the [0, 1] range. We define in the next sections metrics for assessing $relevance(P, Q)$ and $dist(P_i, P_j)$.

3.2 Assessing the Relevance of a Pattern Graph

Our approach exploits statistical information for the popularity (frequency) of the class and value vertices and the property and relationship edges of the pattern graphs in the RDF graph. In doing so, it also takes into account structural and semantic information of the pattern graphs. In this sense, two edges with the same label are different if they involve entity variables of different types. For assessing the popularity of value vertex vertices with keyword instances in the pattern graph, we employ the well known *tf*idf* metric of Information Retrieval (IR) adapted to the syntactic and semantic features of the RDF data.

Consider a pattern graph P over an RDF data graph G. Let $C_1, \ldots, C_n$ be the class vertex labels in P. Let also $|V_{C_i}|$ denote the number of entities of type C_i in the RDF graph G, and $|V_E|$ denote the total number of entities in G. The popularity of the class vertices of P is given by the formula:

$$pop_c(P) = 1/n * (\textstyle\sum_{C_i \in \{C_1, \ldots, C_n\}} |V_{C_i}|/|V_E|)$$

Let $P_1, \ldots, P_m$ denote the distinct (owner class vertex, property edge label) pairs in P. Let also $|E_{P_i}|$ denote the number of property edges complying with P_i in the RDF graph G, and $|E_P|$ denote the total number of property edges in G. The popularity of the property edges of P is defined as:

$$pop_p(P) = 1/m * (\sum_{P_i \in \{P_1, \ldots, P_m\}} |E_{P_i}|/|E_P|)$$

Let $R_1, \ldots, R_u$ denote the distinct (domain class vertex, relationship edge label, range class vertex) triples in P. Let also $|E_{R_i}|$ denote the number of relationship edges complying with R_i in the RDF graph G, and $|E_R|$ denote the total number of relationship edges in G. The popularity of the relationship edges of P is given by the formula:

$$pop_r(P) = 1/u * (\sum_{R_i \in \{R_1, \ldots, R_u\}} |E_{R_i}|/|E_R|)$$

For defining the popularity of value vertices with keyword instances in a pattern graph, we modify the *tf*idf* metric so that it applies to RDF graphs. The metric *tf*idf* (term frequency, inverse document frequency) used in IR reflects how important a term is to a document in a corpus of documents. *tf(t, d)* denotes the frequency of a term *t* in a document *d* while *idf(t)* is the logarithmically scaled inverse fraction of the documents that contain the term. In the context of an RDF graph G, the set of property edges in G which have the same label L and are incident to entity vertices of a type C correspond to a document. This set of property edges is denoted by $E(C, L)$. Given a keyword k_i and a set of property edges $E(C, L)$, let $E(k_i, C, L)$ be the subset of $E(C, L)$ which contains only those property edges whose value comprises k_i. Then:

$$tf(k_i, E(C, L)) = |E(k_i, C, L)|/|E(C, L)|$$

Let W denote the set of all property edge sets $E(C, L)$ in G. For a given keyword k_i, let W_i be the subset of W consisting of those property edge sets $E(C, L)$ such that $tf(k_i, E(C, L)) > 0$ (that is, property edge sets where k_i occurs in the value of at least one of their property edges). Then:

$$idf(k_i) = log(|W|/|W_i|)$$

Let $k_1, \ldots, k_j$ denote the keywords which appear in the labels of value vertices in a pattern graph P. Note that, multiple keywords can appear in the label of a value vertex in P. Let v_i denote the value vertex whose label contains the keyword k_i and L_i is the label of the property edge connecting v_i to an entity variable vertex of type C in P. This means that k_i also appears in the values for the set of property edges for the (C_i, L_i) pair. Then, the popularity of value vertices containing keywords in P is given by the formula:

$$pop_v(P) = 1/j * (\sum_{k_i \in \{k_1, \ldots, k_j\}} tf(k_i, (C_i, L_i)) * idf(k_i))$$

We define the relevance of pattern graph P to keyword query Q as the sum of the popularity of the components of P as follows:

$$relevance(P, Q) = 1/4 * (\sum_{i \in \{c,p,r,v\}} pop_i(P))$$

Clearly, the values of $pop_i(P)$ are in the range $[0, 1]$. By dividing the sum by 4 we guarantee that $relevance(P, Q)$ also ranges between 0 and 1.

3.3 Assessing the Semantic Distance Between Two Pattern Graphs

In order to measure the semantic distance of two pattern graphs, we consider both structural and semantic features of the pattern graphs.

The first factor we consider in assessing the distance of two pattern graphs is the similarity of their matching constructs. Remember that the matching constructs are small graphs that involve only a single keyword instance and provide a context for interpreting the keywords. Given a pattern graph P for a keyword query $Q = \{k_1, \ldots k_n\}$, let $mc(P)$ denote the set of matching constructs of Q—one for every keyword in Q. The larger the number of keywords which are interpreted in the same way in the two pattern graphs, the more similar the pattern graphs are. The similarity of the matching constructs in the two pattern graphs is given by the formula

$$mc\_sim(P_1, P_2) = (|mc(P_1) \cap mc(P_2)|)/n$$

where n is the number of keywords in Q. Note that $n = |mc(P_1)| = |mc(P_2)|$. Clearly, $mc\_sim(P_1, P_2) = 1$ if P_1 and P_2 share the same matching constructs, and $mc\_sim(P_1, P_2) = 0$ if they have no common matching constructs.

For instance, Fig. 5 shows 5 pattern graphs of a query with 5 keywords. Intuitively, P_2 and P_3 are more similar to P_1 than P_4 and P_5 because P_4 and P_5 interpret the keyword `semantics` differently. Metric mc_sim catches this intuition since $mc\_sim(P_1, P_2) = mc\_sim(P_1, P_3) = 5/5$ while $mc\_sim(P_1, P_4) = mc\_sim(P_1, P_5) = 4/5$.

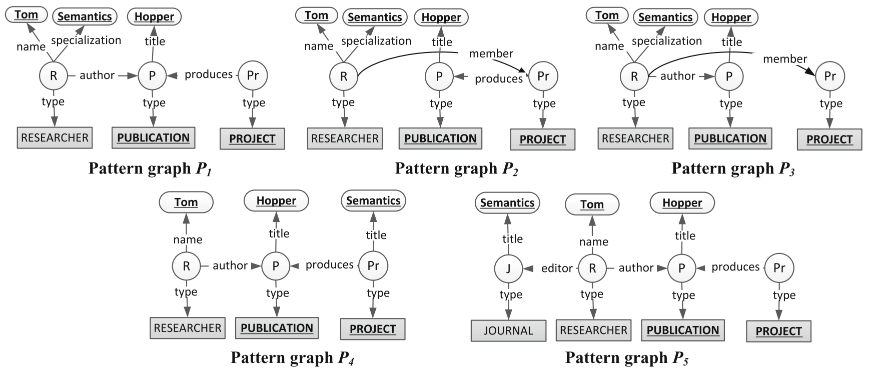


Fig. 5. Pattern graphs of `Q={Tom, semantics, publication, hopper, project}`.

Although P_2 and P_3 have the same common matching constructs with P_1, P_2 looks more similar to P_1 than P_3 does. Therefore, the second factor we consider is to what extent matching constructs for the same keywords are connected in the same way in the two pattern graphs. The higher the number of pairs of keywords in P_1 and P_2 whose matching constructs are connected in the same way in the two pattern graphs, the more similar P_1 and P_2 are. Of course, if the matching constructs of two keywords are not the same in P_1 and P_2, their connections cannot be the same in the two pattern graphs and this pair of keywords does not contribute to the similarity of P_1 and P_2. We define a connection between two keywords k_i and k_j of Q in a pattern graph P of Q as a graph consisting of the matching constructs of k_i and k_j, respectively, and a simple path between these matching constructs in P augmented with type edges and class vertices for every entity variable vertex in the path. Let z be the number of unordered pairs of query keywords which have the same connection in the two pattern graphs. The similarity of the keyword pair connections in P_1 and P_2 is given by:

$$conn\_sim(P_1, P_2) = \frac{z}{(n(n-1))/2}$$

where n is the number of keywords in Q. The denominator reflects the number of unordered keyword pairs for the keywords in Q. Similarly to $mc\_sim(P_1, P_2)$, $conn\_sim(P_1, P_2)$ ranges between [0,1], with 1 indicating that the matching constructs for all the keywords and all the connections between them are the same.

In the example of Fig. 5 both pattern graphs P_2 and P_3 have five common matching constructs with P_1. However, $conn\_sim(P_1, P_2) = 6/10$ and $conn\_sim(P_1, P_3) = 4/10$. Intuitively, P_2 looks more similar to P_1 than P_3 to P_1.

Measuring the similarity of two pattern graphs P_1 and P_2 based solely on the similarity of matching constructs and matching construct connections, $mc\_sim(P_1, P_2)$ and $conn\_sim(P_1, P_2)$, cannot entirely capture their semantic closeness. Compare, for instance, the pattern graphs P_4 and P_5 with the pattern graph P_1 in Fig. 3. Both P_4 and P_5 have 4 keyword matching constructs and 6 pairs of matching construct connections in common with P_1. However, our intuition suggests that P_5 is less similar (more dissimilar) to P_1 than P_4 as it has the class vertex (concept) "Journal" which does not appear in P_1. In contrast, P_1 and P_4 have the same class vertices. Therefore, we introduce the metric of concept dissimilarity to capture the dissimilarity of two pattern graphs. Let $c(P)$ denote the set of class vertices in a pattern graph P. Given two pattern graphs P_1 and P_2 of a keyword query,

$$concept\_dsim(P_1, P_2) = \frac{|(c(P_1) \cup c(P_2)) - (c(P_1) \cap c(P_2))|}{|c(P_1) \cup c(P_2)|}$$

$concept\_dsim(P_1, P_2)$ ranges between 0 (when P_1 and P_2 have all their class vertices in common) and 1 (when P_1 and P_2 do not have common class vertices). The higher the value of $concept\_dsim(P_1, P_2)$, the more distant the pattern graphs P_1 and P_2 are.

Taking into account all the factors, we define the distance $dist(P_1, P_2)$ of two pattern graphs P_1 and P_2 as shown next

$$dist(P_1, P_2) = \frac{1 - [(mc\_sim(P_1, P_2) + conn\_sim(P_1, P_2))/2 - concept\_dsim(P_1, P_2)]}{2}$$

Note that $concept\_dsim(P_1, P_2)$ is considered with a negative sign since it expresses dissimilarity.

$dist(P_1, P_2) = 0$ when the two pattern graphs are the same and $dist(P_1, P_2) = 1$ when $concept\_dsim(P_1, P_2) = 1$.

4 Algorithm

In this section, we present an algorithm for the problem of balancing the relevance and diversity of sets of pattern graphs stated in Sect. 3.1. Exhaustively generating all size-k subsets of a set of pattern graphs and computing their relevance and diversity in order to find an optimal one has exponential complexity in the number of the pattern graphs. In fact, different versions of the diversification

Algorithm 1. PGDiversification (Pattern Graph Diversification)

```
Input: $Q = \{k_1, \dots, k_n\}$: a keyword query with $n$ keywords,
       $S$: structural summary of the data graph,
       $\alpha$: tuning factor, $k$: size of the output list.
Output: $\mathcal{P}_{div}$: set of diversified pattern graphs of size $k$.
 1: for all $k_i \in Q$ do
 2:     $L_i \leftarrow$ {set of all matching constructs of $k_i$ on $S$};
 3: $\mathcal{P} \leftarrow ComputePatternGraphs(\{\times_{i=1}^{n} L_i\}, S)$;
 4: $\mathcal{P} \leftarrow SortByRelevance(\mathcal{P})$;
 5: $\mathcal{P}_{div} \leftarrow \mathcal{P}[0]$;
 6: $i = 1$;
 7: while $i < k$ do
 8:     $j = i$;
 9:     $NextIndex = -1$;
10:     $NextScore = 0$;
11:     while $j \leqslant |\mathcal{P}|$ do
12:         $distance = 0$;
13:         for all $p_i \in \mathcal{P}_{div}$ do
14:             $distance = distance + dist(p_i, \mathcal{P}[j])$
15:         $CurrentScore = \alpha * relevance(\mathcal{P}[j], Q) + (1 - \alpha) * distance/|\mathcal{P}_{div}|$;
16:         if $CurrentScore > NextScore$ then
17:             $NextScore = CurrentScore$;
18:             $NextIndex = j$;
19:         $j = j + 1$;
20:     $\mathcal{P}_{div}.add(\mathcal{P}[NextIndex])$;
21:     $swapPGs(\mathcal{P}[i], \mathcal{P}[NextIndex])$;
22:     $i = i + 1$;
```

problem have previously been shown to be NP-hard [1,5,12,14]. Therefore, we design a heuristic algorithm, called $PGDiversification$, which greedily selects a new pattern graphs at every iteration and incrementally computes the relevance and diversity of pattern graph sets. Algorithm $PGDiversification$ takes as input a keyword query Q, the structural summary S of an RDF graph, the tuning parameter α, and a positive integer k. The output is a subset of the set of pattern graphs of Q on S of size k.

The algorithm starts by finding all the matching constructs of the keywords in query Q on S (lines 1–2) and then generates the set $\mathcal{P}$ of pattern graphs for all possible signatures of Q (line 3). The pattern graphs are generated as r-radius Steiner graphs [16]. Different algorithms can be used for generating the pattern graphs (for instance, in [8,20]). The way the pattern graphs are computed is orthogonal to our approach for pattern graph diversification. The pattern graphs of $\mathcal{P}$ are ranked in descending order of their relevance (line 4). The variable $\mathcal{P}_{div}$ represents the output set of size k which is a subset of the set of pattern graphs $\mathcal{P}$. Initially, the set $\mathcal{P}_{div}$ contains a pattern graph with the highest relevance (line 5). Subsequently, at every iteration, a pattern graph is chosen for inclusion in $\mathcal{P}_{div}$ so that the new $\mathcal{P}_{div}$ set maximizes the objective function (line 8–22). The process terminates when $|\mathcal{P}_{div}| = k$.

5 Experimental Results

We implemented our approach and ran experiments to examine: (a) the effectiveness of our distance metric in assessing the semantic distance of pattern graphs, and the quality of our approach in retrieving a maximum number of relevant pattern graphs, (b) the quality of the approximation of the greedy heuristic algorithm, and (c) the efficiency of the *PGDiversificiation* algorithm in computing the set of pattern graphs that trades off relevance and diversity.

5.1 Datasets and Queries

We used the DBLP[1] and Jamendo[2] real datasets for our experiments. DBLP is a bibliography database of 600 MB of size, containing 8.5 M triples. Jamendo is a repository of Creative Commons licensed music of 85 MB of size, containing 1.1 M triples. The experiments were conducted on a standalone machine with an Intel i7-5600U@2.60GHz processor and 8GB memory. We experimented with a large number of queries and Table 1 reports on 10 of them for each dataset in the interest of space.

5.2 Effectiveness Results

Effectiveness of the Distance Metric. We first want to examine the quality of our distance metric. To this end, for each of the queries in Table 1, we select five

[1] https://datahub.io/dataset/l3s-dblp.
[2] http://dbtune.org/jamendo/.

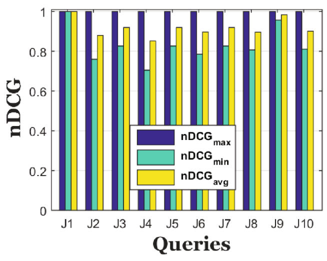
Table 1. Keyword Queries on the Jamendo and DBLP Datasets

Keyword queries on Jamendo		Keyword queries on DBLP	
Q ID	Keywords	Q ID	Keywords
J1	document, teenage, fantasie	D1	concatenable, aspectisation, oliver
J2	nuts, spy4, lemonade	D2	dataflow, quantization
J3	divergence, obsession, lyrics	D3	donatella, intermittent, congestion
J4	reflection, record	D4	balvinder, coscheduling, article
J5	document, cool, divergence	D5	springer, inproceedings
J6	cicada, performance	D6	skogstad, tensorial, morphology
J7	extraordinary, blissful, madness	D7	hierarchical, hybridization
J8	awesome, passion, spy4	D8	person, tolga, coscheduling
J9	guitarist, lemonade	D9	charles, peephole, inproceedings
J10	disgusting, revenge, fantasie	D10	tolga, forward, normalizability

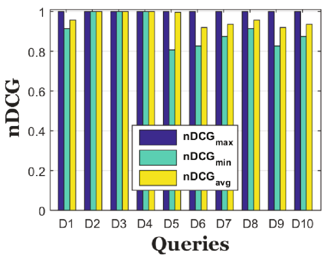
of their pattern graphs. We ask three expert users to score the semantic similarity of each one of these five pattern graphs with another pattern graph (the pattern graph with the highest relevance). The scores are integers in the range [0, 3]. A score of 0 denotes that the two pattern graphs are totally dissimilar. The ground truth is determined by majority vote. We also use our distance metric $dist(P_1, P_2)$ to rank the five pattern graphs in descending order of their distance from the most relevant pattern graph. We assess the quality of the ranking based on $dist(P_1, P_2)$ using the *normalized Discounted Cumulative gain* (nDCG) metric which is defined as follows. The *discounted cumulative gain* (DCG) for position n in a ranked list is given by the following formula:

$$DCG_n = \sum_{i=1}^{n} \frac{2^{rel_i} - 1}{log_2(i+1)}$$

In order to take into account equivalent classes of pattern graphs in the ranked lists (that is, pattern graphs which have the same rank), we have extended nDCG by introducing minimum, maximum and average values for it. The nDCG$_{max}$ value of a ranked list RL_e with equivalence classes corresponds to the nDCG value of a strictly ranked (that is, without equivalence classes) list obtained from RL_e by ranking the pattern graphs in the equivalence classes correctly (that is, in compliance with the scores given by the expert users). The nDCG$_{min}$ value of RL_e is defined analogously. The nDCG$_{avg}$ value of RL_e is the average nDCG value over all strictly ranked lists obtained from RL_e by ranking the pattern graphs in the equivalence classes in all possible ways. The nDCG values range between 0 and 1. Figure 6(a) and (b) show the nDCG$_{min}$, nDCG$_{max}$ and nDCG$_{avg}$ values for the queries on the two datasets. As one can see, all the values are very close to 1. This implies that our distance metric successfully assesses the semantic similarity of two pattern graphs.

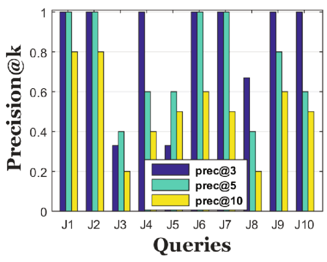


(a) Jamendo dataset

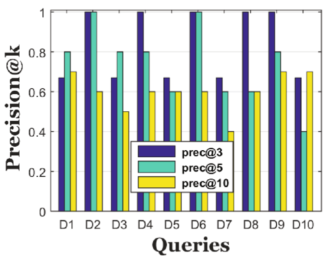


(b) DBLP dataset

Fig. 6. nDCG$_{max}$, nDCG$_{min}$ and nDCG$_{avg}$ for the queries on the two datasets

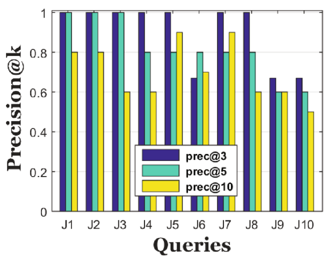


(a) Jamendo dataset

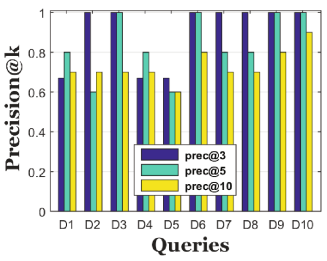


(b) DBLP dataset

Fig. 7. Prec@k for $k = 3$, 5 and 10, for the queries of Table 1 on the two datasets based solely on the relevance metric ($\alpha = 1$).



(a) Jamendo dataset



(b) DBLP dataset

Fig. 8. Prec@k for $k = 3$, 5 and 10, for the queries of Table 1 on the two datasets based on the relevance and diversity metrics ($\alpha = 0.5$).

Effectiveness of the Approach. In order to evaluate the quality of the approach in retrieving relevant results, we measure the relevant results retrieved by Algorithm $PGDiversification$ for different queries when only our relevance metric, and when our metric which balances relevance and diversity is taken into account. Three expert users characterize the retrieved pattern graphs as relevant or not to the query based on whether the pattern graphs express meaningful interpretations of the query and ground truth is determined by majority vote.

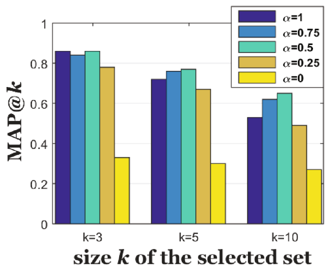
Table 2. MAP@k for the queries of Table 1 on the two datasets

	JAMENDO		DBLP	
	Rel.	Rel. & Div.	Rel.	Rel. & Div.
MAP@10	0.53	0.62	0.51	0.66
MAP@5	0.72	0.77	0.72	0.84
MAP@3	0.86	0.86	0.82	0.91

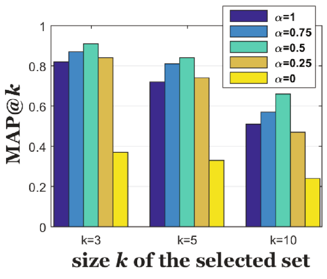
The quality of our approach on a query is expressed by *precision@k* (*prec@k*), which is the ratio of the number of relevant pattern graphs in the set of k pattern graphs returned by our algorithm to k. Figure 7 displays *prec@k* for $k = 3$, 5 and 10 for the queries of Table 1 when only the relevance metric is taken into account (that is, when the tuning parameter $\alpha = 1$). Figure 8 displays *prec@k* for $k = 3$, 5 and 10 for all the queries of Table 1 when both the relevance and diversity metrics are taken into account (that is, when the tuning parameter $\alpha = 0.5$). As we can see, for each value of k and for each dataset, *precision@k* is the same or better in most of the queries. This observation demonstrates the benefit of introducing diversity in the process of selecting the set of k pattern graphs as, when $\alpha = 0.5$, a larger number of the selected pattern graphs can satisfy at least one user.

We also measure the mean average precision at k (MAP@k) for two different values of the tuning parameter: $\alpha = 1$ and $\alpha = 0.5$. MAP@k is the mean of the average precision at k for the queries of Table 1. The values of MAP@k are shown in Table 2 and summarize the measurements displayed in Figs. 7 and 8. As one can see, taking into account diversity ($\alpha = 0.5$) improves MAP@k in all cases. For instance, it improves MAP@10 by 17% on the Jamendo and by 29% on the DBLP dataset. The increase is more pronounced when k is larger. This is expected since a larger k offers more chances to the algorithm to diversify the selected pattern graph set.

Calibrating the Tuning Parameter. In order to select a good value for the tuning parameter α we ran experiments to measure $MAP@k$ for $k = 3$, 5, and

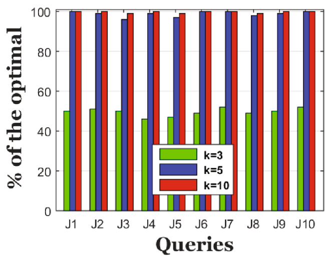


(a) Jamendo dataset

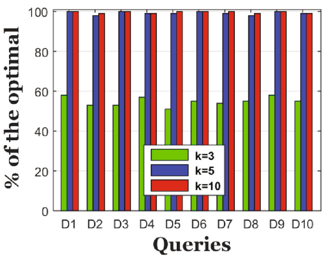


(b) DBLP dataset

Fig. 9. MAP@k for $k = 3$, 5 and 10, for the queries of Table 1 with different α values.



(a) Jamendo dataset



(b) DBLP dataset

Fig. 10. Closeness (%) of the objective function values computed by algorithm *PGDiversification* to the optimal ones, for k = 3, 5, 10, for the queries on the two datasets.

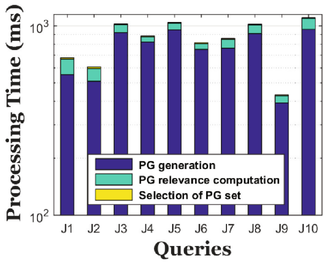
10, for all the queries of Table 1, varying α from 0 to 1 in increments of 0.25. The results are depicted in Fig. 9. As one can see, the highest values for $MAP@k$ are displayed for $\alpha = 0.5$ for all values of k. Therefore, α has been set to 0.5 in all other experiments. Reasonably, the lowest values for $MAP@k$ are displayed for $\alpha = 0$, since all the pattern graphs (with the exception of the initial most popular pattern graph) are selected based on the semantic distance metric and the popularity metric, which is a good criterion for relevance, did not contribute to the selection of relevant pattern graphs.

5.3 Quality of the Approximation by the HeuRistic Algorithm

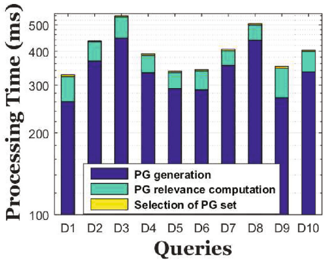
We also wanted to evaluate the quality of the approximation of the optimal solution by the heuristic greedy algorithm. We computed the optimal solution and measured the values of the objective function for $k = 3$, 5, and 10, by running an algorithm which exhaustively enumerates all combinations of k pattern graphs from the generated set of candidate pattern graphs. This was possible since the queries were selected so that they do not generate on the structural summary a huge number of candidate pattern graphs. Figure 10 displays the closeness of the objective function values produced by our algorithm as a percentage to the optimal ones for $k = 3$, 5, and 10, for all the queries of Table 1. Not surprisingly, the closeness is around 50% for $k = 3$ since the selected pattern graph set is always initialized with the most popular pattern graph which might not be part of the optimal solution and the next pattern graph selections are based on their distance from previously selected ones. However, it moves very close to 100% already for $k = 5$ for all queries on both datasets, and it is perfect or almost perfect for $k = 10$. These results suggest that the greedy heuristic produces a good approximation of the optimal solution.

5.4 Efficiency Results

We ran Algorithm $PGDiversification$ for all the queries of Table 1. The execution time is reported in Fig. 11 for pattern graphs sets of size $k = 5$ and tuning



(a) Jamendo dataset



(b) DBLP dataset

Fig. 11. Processing time of *PGDiversification* for the queries on the two datasets.

parameter $\alpha = 0.5$. The execution time for each query consists of three components: (a) the generation of the pattern graphs using the structural summary, (b) the computation of the relevance of the pattern graphs and the selection of one with the highest relevance, and (c) the application of the greedy heuristic for generating the list of k pattern graphs which is both relevant and diversified. One can see that the execution time is dominated by the pattern graph generation process. This is expected since computing the pattern graphs requires accessing the database for finding all the keyword matching constructs.

Overall, our approach displays interactive execution times ($\leqslant$1 s) on both datasets. This performance demonstrates that our approach succeeds in improving the number of relevant pattern graphs selected without suffering from the scalability issue of keyword search approaches.

6 Related Work

The goal of the diversification problem is to solve the over-specialization problem where a highly homogenous set of results is returned to the user due to relevance-based ranking and/or personalization. Result diversification is a way to minimize user dissatisfaction by providing a diverse set of results. In general, the diversification problem is defined as selecting a subset of the retrieved result set with k results such that the diversity among these k results is maximized. In [4], the concept of maximal marginal relevance (MMR) is used to tradeoff between relevance and novelty. In [14] an axiomatic approach for result diversification is adopted. Reference [12] is a review of different definitions for diversity, and of algorithms and evaluation metrics for diversification. It categorizes diversity definitions as content-based [22], novelty-based [21] and coverage-based [1]. Most of these approaches perform diversification as a post-processing or re-ranking step of candidate result retrieval which can incur a huge computation cost since the number of candidate results can be extremely large for a keyword query. In contrast, our diversification process is part of a query disambiguation phase which takes place before extracting any search results. We compute pattern graphs corresponding to alternate interpretations of a given keyword query since they offer clear semantics and quality information for diversification. This way, we also avoid the computation overhead of computing all relevant results.

A number of papers study search result diversification on structured and semi-structured databases [2,11,15,17]. However, there are only few contributions on diversifying keyword search results on RDF data. Previous techniques cannot be directly applied in this context as the semantic information in RDF data graphs requires different criteria and methods. [19] addresses diversification of entity search results using a categorization technique to cluster similar entities. [3,9] addresses diversification issues of keyword query results on RDF data, but do not consider relevance aspects. In this paper, we exploit structural and semantic characteristics of pattern graphs for capturing their relevance to a query, and also the similarity and dissimilarity between pattern graphs.

7 Conclusion

We presented a novel technique which exploits pattern graphs of a keyword query instead of the query results for trading off popularity for diversity in the result sets of keyword queries on RDF data graph. Diversification of pattern graphs addresses the scalability problem of keyword search approaches on large data graphs and also allows diversifying the interpretations of the keyword query. We introduced metrics to assess the relevance of a pattern graph and the semantic distance between patterns graphs. We also designed a greedy heuristic algorithm which maximizes an objective function on pattern graph sets balancing popularity and diversity. Our extensive experimental results show the feasibility of our approach in terms of the efficiency and the effectiveness of the algorithm.

References

1. Agrawal, R., Gollapudi, S., Halverson, A., Ieong, S.: Diversifying search results. In: WSDM, pp. 5–14. ACM (2009)
2. Aksoy, C., Dass, A., Theodoratos, D., Wu, X.: Diversification of keyword query result patterns. In: Cui, B., Zhang, N., Xu, J., Lian, X., Liu, D. (eds.) WAIM 2016. LNCS, vol. 9659, pp. 171–183. Springer, Cham (2016). doi:10.1007/978-3-319-39958-4_14
3. Bikakis, N., Giannopoulos, G., Liagouris, J., Skoutas, D., Dalamagas, T., Sellis, T.: RDivF: diversifying keyword search on RDF graphs. In: Aalberg, T., Papatheodorou, C., Dobreva, M., Tsakonas, G., Farrugia, C.J. (eds.) TPDL 2013. LNCS, vol. 8092, pp. 413–416. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40501-3_49
4. Carbonell, J., Goldstein, J.: The use of MMR, diversity-based reranking for reordering documents and producing summaries. In: SIGIR, pp. 335–336 (1998)
5. Carterette, B.: An analysis of NP-completeness in novelty and diversity ranking. Inf. Retrieval **14**(1), 89–106 (2011)
6. Chen, H., Karger, D.R.: Less is more: probabilistic models for retrieving fewer relevant documents. In: SIGIR, pp. 429–436. ACM (2006)

7. Dass, A., Aksoy, C., Dimitriou, A., Theodoratos, D.: Exploiting semantic result clustering to support keyword search on linked data. In: Benatallah, B., Bestavros, A., Manolopoulos, Y., Vakali, A., Zhang, Y. (eds.) WISE 2014. LNCS, vol. 8786, pp. 448–463. Springer, Cham (2014). doi:10.1007/978-3-319-11749-2_34
8. Dass, A., Aksoy, C., Dimitriou, A., Theodoratos, D.: Keyword pattern graph relaxation for selective result space expansion on linked data. In: Cimiano, P., Frasincar, F., Houben, G.-J., Schwabe, D. (eds.) ICWE 2015. LNCS, vol. 9114, pp. 287–306. Springer, Cham (2015). doi:10.1007/978-3-319-19890-3_19
9. Dass, A., Aksoy, C., Dimitriou, A., Theodoratos, D., Wu, X.: Diversifying the results of keyword queries on linked data. In: Cellary, W., Mokbel, M.F., Wang, J., Wang, H., Zhou, R., Zhang, Y. (eds.) WISE 2016. LNCS, vol. 10041, pp. 199–207. Springer, Cham (2016). doi:10.1007/978-3-319-48740-3_14
10. Dass, A., Dimitriou, A., Aksoy, C., Theodoratos, D.: Incorporating Cohesiveness into keyword search on linked data. In: Wang, J., Cellary, W., Wang, D., Wang, H., Chen, S.-C., Li, T., Zhang, Y. (eds.) WISE 2015. LNCS, vol. 9419, pp. 47–62. Springer, Cham (2015). doi:10.1007/978-3-319-26187-4_4
11. Demidova, E., Fankhauser, P., Zhou, X., Nejdl, W.: DivQ: diversification for keyword search over structured databases. In: SIGIR, pp. 331–338. ACM (2010)
12. Drosou, M., Pitoura, E.: Search result diversification. ACM SIGMOD Rec. **39**(1), 41–47 (2010)
13. Elbassuoni, S., Ramanath, M., Schenkel, R., Weikum, G.: Searching RDF graphs with SPARQL and keywords. IEEE Data Eng. Bull. **33**(1), 16–24 (2010)
14. Gollapudi, S., Sharma, A.: An axiomatic approach for result diversification. In: WWW, pp. 381–390. ACM (2009)
15. Hasan, M., Mueen, A., Tsotras, V., Keogh, E.: Diversifying query results on semi-structured data. In: CIKM, pp. 2099–2103. ACM (2012)
16. Li, G., et al.: Ease: an effective 3-in-1 keyword search method for unstructured, semi-structured and structured data. In: SIGMOD, pp. 903–914 (2008)
17. Li, J., Liu, C., Yu, J.X.: Context-based diversification for keyword queries over XML data. Proc. KDE **27**(3), 660–672 (2015)
18. Radlinski, F., Dumais, S.: Improving personalized web search using result diversification. In: SIGIR, pp. 691–692. ACM (2006)
19. Ruotsalo, T., Frosterus, M.: Semantic entity search diversification. In: ICSC, pp. 32–39 (2013)
20. Tran, T., Wang, H., Rudolph, S., Cimiano, P.: Top-k exploration of query candidates for efficient keyword search on graph-shaped (RDF) data. In: ICDE (2009)
21. Zhang, M., Hurley, N.: Avoiding monotony: improving the diversity of recommendation lists. In Recommender Systems, pp. 123–130 (2008)
22. Ziegler, C.-N., McNee, S.M., Konstan, J.A., Lausen, G.: Improving recommendation lists through topic diversification. In: WWW, pp. 22–32. ACM (2005)