Effective Searching of RDF Knowledge Graphs

Article *in* SSRN Electronic Journal · January 2018

DOI: 10.2139/ssrn.3199315

CITATIONS
5

READS
247

2 authors:

Hiba Arnaout
Max Planck Institute for Informatics
7 PUBLICATIONS **26** CITATIONS
SEE PROFILE

Shady Elbassuoni
American University of Beirut
54 PUBLICATIONS **1,384** CITATIONS
SEE PROFILE

Effective Searching of RDF Knowledge Graphs

Hiba Arnaout, Shady Elbassuoni*

Computer Science Department
American University of Beirut
Riad El-Solh 1107 2020
Beirut, Lebanon

Abstract

RDF knowledge graphs are typically searched using triple-pattern queries. Often, triple-pattern queries will return too many or too few results, making it difficult for users to find relevant answers to their information needs. To remedy this, we propose a general framework for effective searching of RDF knowledge graphs. Our framework extends both the searched knowledge graph and triple-pattern queries with keywords to allow users to form a wider range of queries. In addition, it provides result ranking based on statistical machine translation, and performs automatic query relaxation to improve query recall. Finally, we also define a notion of result diversity in the setting of RDF data and provide mechanisms to diversify RDF search results using Maximal Marginal Relevance. We evaluate the effectiveness of our retrieval framework using various carefully-designed user studies on DBpedia, a large and real-world RDF knowledge graph.

Keywords: RDF, Ranking, Diversity, Relaxation

1. Introduction

In recent years, the Web has evolved from a network of linked documents to one where both documents and data are linked, resulting in what is commonly known as the Web of Data. Underpinning this evolution is a set of best practices known as Linked Data[1], which provides mechanisms for publishing and connecting structured data on the Web in a machine-readable form with explicit semantics. The increasing adoption of Linked Data is turning the Web into a global data space that connects data from diverse domains and enables genuinely novel applications.

Recently, Linked Data has grown from an academic endeavor into one that has been embraced by numerous governments and industrial stakeholders, in domains such as business and finance, geography, governance, media, digital libraries, life sciences, in addition to general-purpose and user-generated content datasets. All such data is typically represented as sets of RDF (Resource Description Framework)[2] triples. An RDF triple is a triple with three fields: a *subject*, a *predicate* and an *object* where subjects and predicates are URIs and objects are either URIs or literals. Alternatively, an RDF dataset can be also viewed as a labeled graph, which we refer to as an RDF knowledge graph. In an RDF knowledge graph, node labels are *either* URIs representing resources *or* literals, and edge labels are URIs representing predicates. Figure 1 shows an excerpt of DBpedia [1], an RDF knowledge graph that is automatically constructed from Wikipedia, and Table 1 shows the corresponding RDF triples. We omit the prefix of all URIs for readability.

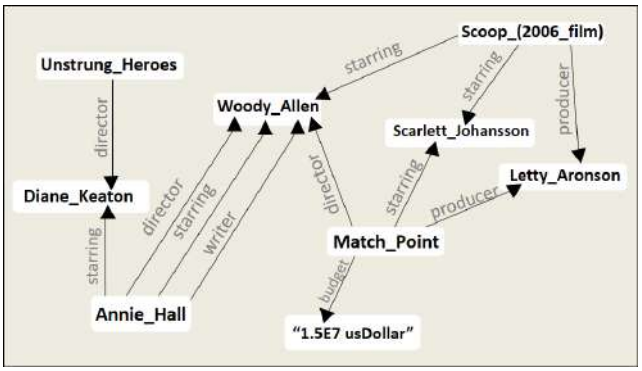

Figure 1: An excerpt of DBpedia's RDF knowledge graph.

The semantic query language for RDF is known as SPARQL [3]. SPARQL allows users to compose structured queries consisting of triple patterns, where a triple pattern is an RDF triple with one or more variables. A variable occurring in one of the triple patterns can be used again in another triple pattern in the query, denoting a join condition. For example, if the user's information need is to find

*Corresponding author
Email addresses: hka22@aub.edu.lb (Hiba Arnaout), se58@aub.edu.lb (Shady Elbassuoni)

[1] http://linkeddata.org/

[2] http://www.w3.org/RDF/

[3] https://www.w3.org/TR/rdf-sparql-query/

Subject	Predicate	Object
Annie_Hall	starring	Woody_Allen
Annie_Hall	writer	Woody_Allen
Annie_Hall	director	Woody_Allen
Annie_Hall	starring	Diane_Keaton
Unstrung_Heroes	director	Diane_Keaton
Scoop_(2006_film)	starring	Woody_Allen
Scoop_(2006_film)	starring	Scarlett_Johansson
Scoop_(2006_film)	producer	Letty_Aronson
Match_Point	director	Woody_Allen
Match_Point	starring	Scarlett_Johansson
Match_Point	producer	Letty_Aronson
Match_Point	budget	1.5E7 usDollar

Table 1: The set of RDF triples corresponding to the RDF knowledge graph in Figure 1.

movies that were directed and starred by the same person, the following triple-pattern query can be used:

```
?x director ?y
?x starring ?y
```

where `?x` and `?y` denote variables. Given this 2 triple-pattern query, the results are all the subgraphs of the underlying knowledge graph that are isomorphic to the query graph. For the above example, the result subgraphs should consist of two triples whose subjects and objects are the same, and whose predicates are `director` and `starring`, respectively. For instance, one such subgraph is:

```
Annie_Hall director Woody_Allen
Annie_Hall starring Woody_Allen
```

RDF knowledge graphs equipped with triple-pattern querying such as SPARQL querying provides a very powerful tool for knowledge discovery. However, in order to fully utilize RDF Knowledge graphs, the following challenges must be addressed.

1. **Data Incompleteness:** while large RDF knowledge graphs contain a vast amount of information in the form of RDF triples, the majority of information on the Web is available in the form of free text. Thus, extending RDF knowledge graphs with text can increase the scope of such knowledge graphs making them very rich sources of information. For example, the RDF knowledge graph in Figure 1 contains information about movies, their directors, actors and producers. While this covers a wide range of interesting information, there is still information that cannot be easily captured in the form of RDF triples such as movie plots, taglines, users comments and so on. Such information naturally appears as free text and by omitting them altogether, we lose a lot of valuable information.

2. **Flexible Querying:** even though triple-pattern queries are highly expressive, they are also very restrictive since they deploy Boolean matching (i.e., a result is either a match to a query or not). It is thus crucial to equip triple-pattern search with flexible querying mechanisms to allow for a more effective searching of RDF data. For example, consider an example query asking for movies that were directed and starred by Woody Allen. This query should be formulated as the following triple-pattern query when run against DBpedia:

```
?x director Woody_Allen
?x starring Woody_Allen
```

However, if the user fails to express her query this way, say by not using the exact URI referring to Woody Allen or by using the URI `Woody_Allen` as the subject of the triple patterns rather than their object, the query will not yield any result. In such cases, query relaxation, which is similar to query modification in traditional Information Retrieval (IR), is crucial to address queries that cannot be answered as exactly formulated. On the other hand, some information needs cannot be entirely expressed using triple-pattern queries due to the lack of appropriate resources in the knowledge graph. For example, assume that the user is interested in finding movies that are based on a novel, or that have won an Oscar. It is not possible for the user to express such information need using a triple-pattern query only. However, if RDF knowledge graphs were extended with text, and keyword conditions were allowed in queries, this can go a long way in addressing a wider range of information needs such as the ones just mentioned.

3. **Result Ranking:** RDF knowledge graphs may produce too many results for many queries. It will thus be highly beneficial to present users with a ranked list of results rather than a mere list of unranked ones. This is particularly crucial when RDF knowledge graphs and triple-pattern queries are extended with text and when query relaxation is deployed. To this end, some notion of relevance or importance must be defined over the RDF knowledge graph that can be utilized for result ranking. For example, consider the information need of finding movies that were directed and starred by the same person. It might be highly desirable to rank the results based on, say, movie popularity. If the information need was to find those movies that also won an Oscar or were based on a novel, these should constitute additional criteria for result ranking.

4. **Result Diversity:** while ranking ensures that the most relevant results are ranked on top, it is often the case that the top results tend to be homogeneous, making it difficult for users to truly explore the knowledge graph. For example, considering our

example query, it might be undesirable to have all the top results about movies directed and starred by one person, say Woody Allen, or all about movies of the same genre. Result diversity can thus play a big role in ensuring that the users get a broad view of the different aspects of the results of their queries, and ensures that, on average, almost all users can find relevant results to their queries in the top ranks.

In this paper, we address all the above challenges and develop a number of novel models and algorithms to effectively search RDF knowledge graphs. Our contributions can be summarized as follows.

- We develop a novel ranking model based on statistical machine translation for triple-pattern queries. Our ranking model operates on top of traditional RDF knowledge graphs as well as on keyword-extended ones that allow keyword conditions in triple-pattern queries.
- We develop a framework for query relaxation that automatically relaxes triple-pattern queries that yield no results in such a way that the original query intention is preserved. We use the constants provided by the user to augment the relaxed queries, by turning them into a set of keywords. This has the advantage of improving the recall of such queries without unduly sacrificing precision. We also develop a principled mechanism to combine the results of relaxed queries using our ranking model.
- We define a notion of result diversity in the RDF setting and develop a general technique based on the Maximal Marginal Relevance [2] in order to provide diverse results to queries over RDF knowledge graphs.
- We propose a novel evaluation metric to evaluate the results of triple-pattern queries over RDF knowledge graphs. Our evaluation metric takes into consideration both the relevance of the results as well as their diversity.
- Finally, we conduct a case study to evaluate the different aspects of our framework on a large real-world RDF knowledge graph, namely DBpedia [1] using various tasks on CrowdFlower [4], a crowdsourcing platform.

The rest of the paper is organized as follows. In Section 2, we describe our data model and query processing. In Section 3, we outline our ranking model for triple-pattern queries, possibly extended with keywords. Section 4 describes our query relaxation framework and Section 5 describes our result diversity approach. Related work is reviewed in Section 6, followed by our case study on DBpedia in Section 7 . Before concluding, we discuss the complexity of the system in Section 8. Finally, in Section 9, we summarize our findings and provide future directions.

[4] https://https://www.crowdflower.com/

2. Data Model and Query Processing

Definition 1 (RDF Knowledge Graph). *Let U be a set of URIs and L be a set of literals. An RDF knowledge graph is a set of RDF triples $\{t_1, t_2, \ldots, t_m\}$ such that each $t_i \in U \times U \times \{U \cup L\}$ for $1 \leq i \leq m$.*

In other words, an RDF knowledge graph is a labeled multi-digraph where the node labels are *either* URIs corresponding to resources *or* literals, and the edge labels are URIs corresponding to predicates. Figure 1 shows an example RDF knowledge graph and Table 1 displays the corresponding RDF triples.

Definition 2 (Extended RDF Knowledge Graph). *Let U be a set of URIs, L be a set of literals and V be a set of keywords. An extended RDF knowledge graph is a set of RDF triples $\{t_1, t_2, \ldots, t_m\}$ such that each triple $t_i \in U \times U \times \{U \cup L\}$ for $1 \leq i \leq m$. Moreover, each triple t_i is associated with a weight $w(t_i) \in R$ for $1 \leq i \leq m$. In addition, t_i is associated with a set of keywords $\{v_1, v_2, \ldots, v_l\}$ such that each $v_j \in V$ for $1 \leq j \leq l$. Finally, each triple-keyword pair (t_i, v_j) is associated with a weight $w(t_i, v_j) \in R$ for $1 \leq i \leq m$ and $1 \leq j \leq l$.*

Informally, an RDF knowledge graph is extended by associating each triple with a weight reflecting the importance of the triple. The triple weights can be computed in various ways depending on the nature of the RDF knowledge graph. For example, in case the knowledge graph was constructed using automatic Information Extraction techniques, the weight of the triple can reflect the confidence in the triple (i.e., its accuracy). It can also be measured based on the authority of the resources referenced in the triple, or a combination of both criteria. These weights will be used to rank the results of queries as we explain in the next section.

Moreover, in an extended RDF knowledge graph, each triple is associated with a set of keywords that represent additional context for the triple. In addition, each one of these keywords is associated with a weight that reflects the importance of the keyword in the context of the triple. Again, these weights will be a key factor in result ranking. For example, an RDF knowledge graph can be extended by using the labels of the subjects and objects and textual patterns of the predicates. It can also be extended by using key phrases or summaries describing the subjects or objects of the triples.

Table 2 shows an excerpt of DBpedia's extended RDF knowledge graph in the form of RDF triples. The graph was extended with keywords extracted from the *abstracts* of the resources in the triples. The abstract of a resource is the first paragraph of the Wikipedia article of the resource.

Subject	Predicate	Object	Weight	Keyword:Weight
Annie_Hall	director	Woody_Allen	2032	oscar:1, comedy:251, york:1449, romance:13, ...
Annie_Hall	starring	Diane_Keaton	758	psychoanalysis:8, york:567, love:494, sex:8, jew:72, ...
Match_Point	writer	Woody_Allen	1866	thriller:121, greed:9, lust:32, money:195, allen:2063, ...
Match_Point	starring	Emily_Mortimer	665	london:216, pregnancy:3, thriller:53, love:365, lust:20, ...
Match_Point	producer	Letty_Aronson	138	thriller:12, money:20, london:67, crime:14, sister:1, ...
Unstrung_Heroes	director	Diane_Keaton	517	1995:115, comic:30, memoir:9, journalist:47, ...

Table 2: An excerpt of DBpedia's extended RDF knowledge graph in the form of RDF triples.

More precisely, for each triple in the knowledge graph, we extracted all the keywords in the abstracts of both the subject and the object of the triple, in case they were URIs, stemmed these keywords, removed stop words and duplicates, and then extended the triple with the resulting keywords. In case the object of the triple was a literal, we just used the literal itself as a keyword.

To set the weights of the triples and the triple-keyword pairs, the Wikipedia link-structure was used. To compute the weight $w(t)$ of triple $t = (s, p, o)$, we used the number of Wikipedia articles that contain a hyperlink to the subject or the object of triple t as follows:

$$w(t) = |links(s)| + |links(o)| \quad (1)$$

where $links(s)$ is the set of Wikipedia articles that contain a hyperlink to the subject s of triple t, and $links(o)$ is the set of Wikipedia articles that contain a hyperlink to the object o of triple t. The intuition behind this is that the more Wikipedia articles that link to the subject and the object of the triple, the more important the triple is (i.e., a measure of authority).

To compute the weight $w(t, v)$ of a triple $t = (s, p, o)$ with respect to keyword v, we again used the Wikipedia link-structure as follows:

$$\begin{aligned} w(t, v) = &|links(s) \cap links(v)| \\ &+ |links(o) \cap links(v)| \end{aligned} \quad (2)$$

where $links(s)$ is the set of Wikipedia articles that contain a hyperlink to the subject s, $links(o)$ is the set of Wikipedia articles that contain a hyperlink to the object o, and $links(v)$ is the set of Wikipedia articles that contain a hyperlink to any article that contains keyword v. By taking the intersection between $links(s)$ and $links(v)$, and similarly $links(o)$ and $links(v)$, we are effectively computing the weight of triple t with respect to keyword v as the number of Wikipedia articles that mention both the subject of t and the keyword v (and similarly for the object o). This way, a triple will have a high weight with respect to a keyword v if both its subject and object co-occur frequently with the keyword v in Wikipeda.

Note that the way a knowledge graph is extended with keywords and how the weights of triples and triple-keyword pairs are set highly depend on the nature of the knowledge graph. We have shown one example of how to do this for DBpedia, which we base our case study on. This procedure can be also applied to any other Wikipedia-based knowledge graph such as YAGO [3]. For other types of knowledge graphs, an appropriate method for extending the knowledge graph must be defined based on how the knowledge graph is constructed.

Definition 3 (Triple-Pattern Query). *Let U be a set of URIs, L be a set of literals, and X be a set of variables. A triple-pattern query is a set of triple patterns $\{q_1, q_2, \ldots, q_n\}$ where $q_i \in \{U \cup X\} \times \{U \cup X\} \times \{U \cup L \cup X\}$ for $1 \leq i \leq n$.*

A triple-pattern query is a thus a graph with multiple edges where node labels are either URIs representing resources, literals or variables, and edge labels are either URIs representing predicates or variables. For example, the following triple-pattern query can be used to find movies directed and starred by the same person:

```
?x starring ?y
?x director ?y
```

Definition 4 (Extended Triple-Pattern Query). *Let U be a set of URIs, L be a set of literals, X be a set of variables and S be a set of keywords. An extended triple-pattern query is a set of triple patterns $\{q_1, q_2, \ldots, q_n\}$ where $q_i \in \{U \cup X\} \times \{U \cup X\} \times \{U \cup L \cup X\}$ for $1 \leq i \leq n$. Moreover, each triple pattern q_i for $1 \leq i \leq n$ can be optionally associated with a set of keywords $\{v_1, v_2, \ldots, v_l\}$ where $v_j \in V$ for $1 \leq j \leq l$.*

In other words, an extended triple-pattern query is also a graph with multiple edges. However, in the case of an extended triple-pattern query, one or more edges can be associated with keyword conditions that cannot be expressed using triple patterns. For example, the following extended triple-pattern query can be used to find movies directed and starred by the same person *that are based on a novel*:

```
?x starring ?y [novel]
?x director ?y
```

Definition 5 (Result Subgraph). *Let U be a set of URIs, L be a set of literals and X be a set of variables. A result subgraph to a triple-pattern query or an extended triple-pattern query $\{q_1, q_2, \ldots, q_n\}$ is a set of triples $\{t_1, t_2, \ldots, t_n\}$ such that $t_i \in U \times U \times \{U \cup L\}$ for $1 \leq i \leq n$. Moreover, each triple $t_i = (s_i, p_i, o_i)$ corresponds to exactly one triple-pattern $q_i = (s'_i, p'_i, o'_i)$ for $1 \leq i \leq n$ and such that 1) $s_i = s'_i$ or $s'_i \in X$, $p_i = p'_i$ or $p'_i \in X$, and 3) $o_i = o'_i$ or $o'_i \in X$.*

Subject	Predicate	Object
Bordellet	starring	Ole_Ege
Bordellet	director	Ole_Ege
Marx_Reloaded	starring	Jason_Barker
Marx_Reloaded	director	Jason_Barker
The_Kid_(1921_film)	starring	Charlie_Chaplin
The_Kid_(1921_film)	director	Charlie_Chaplin
The_Almost_Guys	starring	Eric_Fleming
The_Almost_Guys	director	Eric_Fleming
Kranti	starring	Manoj_Kumar
Kranti	director	Manoj_Kumar
Purab_Aur_Paschim	starring	Manoj_Kumar
Purab_Aur_Paschim	director	Manoj_Kumar
The_Count_(film)	starring	Charlie_Chaplin
The_Count_(film)	director	Charlie_Chaplin
Annie_Hall	starring	Woody_Allen
Annie_Hall	director	Woody_Allen
Love_and_Death	starring	Woody_Allen
Love_and_Death	director	Woody_Allen
Hollywood_Ending	starring	Woody_Allen
Hollywood_Ending	director	Woody_Allen

Table 3: 10 result subgraphs for the triple-pattern query asking for movies starred and directed by the same person.

That is, the results of a triple-pattern query (whether extended with keywords or not) is a *connected* subgraph of the searched knowledge graph that is isomorphic to the triple-pattern query. In other words, the variables in the query triple patterns are bound to resources or literals from the knowledge graph. Note that in the case where the same variable is used multiple times in different triple patterns, the variable must be bound to the same resource or literal within each result subgraph. Also note that keyword conditions do not play any role in query processing. They are only crucial in ranking the results as we explain in the next section. Table 3 shows an example triple-pattern query asking for movies starred and directed by the same person and 10 corresponding result subgraphs from DBpedia's RDF knowledge graph.

3. Result Ranking

Often, triple-pattern queries will return too many results, making it difficult for users to navigate through them to find what they are looking for. Moreover, in the case of extended triple-pattern queries, it is crucial to rank the result subgraphs based on their relevance to the keyword conditions in the query. To address these two challenges, we build a novel ranking model for triple-pattern queries, possibly extended with keywords. Our ranking model is very generic and can be applied to any extended RDF knowledge graph as defined in Section 2.

Our model is based on statistical machine translation and was inspired by the following observation. When searching RDF knowledge graphs, a query consists of triple patterns or triple patterns extended with keywords. On the other hand, the result subgraphs are composed of RDF triples. This is a common problem in cross-lingual Information Retrieval (IR) that deals with retrieving information in a language different from the user queries. Many approaches have been proposed to perform cross-lingual IR, including statistical language-modeling-based approaches (LM approaches for short). We adopt one such approach [4] commonly used to the context of RDF as follows.

Given a query Q which consists of n triple patterns $q_1, q_2, \ldots, q_n$ and a result subgraph G which consists of n triples $t_1, t_2, \ldots, t_n$, we first compute a probability of subgraph G generating the query Q as follows:

$$P(Q|G) = \prod_{i=1}^{n} P(q_i|G) \tag{3}$$

That is, we assume independence between the triple patterns and estimate the probability of G generating Q as the product of G generating each triple pattern $q_i \in Q$. This independence assumption is commonly deployed in IR ranking approaches to overcome the curse of dimensionality [4].

Since a sugbraph G consists of triples whereas q_i is a triple pattern (possibly extended with keywords), we apply statistical machine translation to estimate the probability of subgraph G generating triple pattern q_i as follows:

$$P(q_i|G) = \sum_{j=1}^{m} P(q_i|t_j)P(t_j|G) \tag{4}$$

where $P(q_i|t_j)$ is the probability of translating triple pattern q_i into triple t_j, $P(t_j|G)$ is the probability of generating triple t_j given subgraph G, and m is the number of triples in the knowledge graph. The probability of generating triple t_j given result subgraph G is computed as follows:

$$P(t_j|G) = \begin{cases} \frac{1}{n} \text{ , if } t_j \in G \\ 0 \text{ , otherwise} \end{cases} \tag{5}$$

Recall that subgraph G consists of n triples $t_1, t_2, \ldots, t_n$. We assume that G generate any of its triples $t_1, t_2, \ldots, t_n$ with uniform probability (i.e., $1/n$). For any other triple $t_j \notin G$, its probability of being generated by G would be equal to *zero* since it is not part of the subgraph G.

The probability of translating triple pattern q_i into triple t_j is defined in two ways depending on whether q_i is extended with keywords or not. In case q_i does not consist of any keywords, it is computed as follows:

$$P(q_i|t_j) = \frac{w(t_i)}{\sum_{t \in \hat{q_i}} w(t)} \tag{6}$$

where $\hat{q_i}$ is the set of all triples matching (i.e., isomorphic to) q_i and $w(t)$ is the weight of a triple t as defined in Definition 2.

Subject	Predicate	Object
None_But_the_Brave	starring	Frank_Sinatra
None_But_the_Brave	director	Frank_Sinatra
Cruel_Summer_(film)	starring	Kanye_West
Cruel_Summer_(film)	director	Kanye_West
The_Bond	starring	Charlie_Chaplin
The_Bond	director	Charlie_Chaplin
A_Dog's_Life	starring	Charlie_Chaplin
A_Dog's_Life	director	Charlie_Chaplin
Citizen_Kane	starring	Orson_Welles
Citizen_Kane	director	Orson_Welles
The_Prince_of_Tides	starring	Barbra_Streisand
The_Prince_of_Tides	director	Barbra_Streisand
Gran_Torino	starring	Clint_Eastwood
Gran_Torino	director	Clint_Eastwood
CSNY/Dj_vu	starring	Neil_Young
CSNY/Dj_vu	director	Neil_Young
Annie_Hall	starring	Woody_Allen
Annie_Hall	director	Woody_Allen
City_Lights	starring	Charlie_Chaplin
City_Lights	director	Charlie_Chaplin

Table 4: Top-10 ranked result subgraphs for the triple-pattern query asking for movies starred and directed by the same person. As can be seen, all the movies in this list are popular ones by famous people.

Subject	Predicate	Object
Lady_from_Shanghai	starring	Orson_Welles
Lady_from_Shanghai	director	Orson_Welles
The_Trial	starring	Orson_Welles
The_Trial	director	Orson_Welles
Journey_into_Fear	starring	Orson_Welles
Journey_into_Fear	director	Orson_Welles
The_Prince_of_Tides	starring	Barbra_Streisand
The_Prince_of_Tides	director	Barbra_Streisand
Mr._Arkadin	starring	Orson_Welles
Mr._Arkadin	director	Orson_Welles
Black_Magic	starring	Orson_Welles
Black_Magic	director	Orson_Welles
Moby_Dick	starring	Orson_Welles
Moby_Dick	director	Orson_Welles
Outlaw_Josey_Wales	starring	Clint_Eastwood
Outlaw_Josey_Wales	director	Clint_Eastwood
Firefox	starring	Clint_Eastwood
Firefox	director	Clint_Eastwood
The_Town	starring	Ben_Affleck
The_Town	director	Ben_Affleck

Table 5: 10 ranked result subgraphs for the triple-pattern query asking for movies starred and directed by the same person that are based on a novel. As can be seen, all the movies in this list are based on novels.

Our ranking model thus ranks the result subgraphs of a given triple-pattern query based on the weights of the triples in the result subgraphs. Given that the weights reflect the importance of the triples with respect to each other and given that we normalize these weights by the sum of weights of all triples matching the corresponding triple-pattern in the query as can be seen in Equation 6, our ranking model effectively ranks the results based on some notion of relevance to the query. Table 4 shows the top-10 ranked results in DBpedia for the query:

```
?x starring ?y
?x director ?y
```

which can be used to find movies starred and directed by the same person.

In case q_i is extended with keywords $s_1, s_2, \ldots, s_l$, the probability of translating the extended triple pattern q_i into triple t_j is computed as follows:

$$P(q_i|t_j) = \prod_{k=1}^{l} P(q_i, s_k|t_j) \quad (7)$$

That is, we assume independence between the keywords $s_1, s_2, \ldots, s_l$ and compute the probability of translating q_i extended with keywords $s_1, s_2, \ldots, s_l$ into t_j as the product of translating q_i extended with each individual keyword s_k into t_i for $1 \leq k \leq l$. The probability of translating a triple pattern q_i extended with a single keyword s_k, which we denote by $P(q_i, s_k|t_j)$ is computed as follows:

$$P(q_i, s_k|t_j) = \alpha \frac{w(t_j, s_k)}{\sum_{t \epsilon \hat{q_i}} w(t, s_k)} + (1 - \alpha) \frac{w(t_j)}{\sum_{t \epsilon \hat{q_i}} w(t)} \quad (8)$$

where α is a smoothing parameter, and $w(t_j, s_k)$ is the weight of the triple-keyword pair (t_j, s_k). Note that it is crucial to smooth the probability $P(q_i, s_k|t_j)$ since we rely on the weight of the triple-keyword pair $w(t_i, s_k)$ to estimate this probability. Without smoothing, $P(q_i, s_k|t_j)$ would be zero if triple t_j is not extended with the keyword s_k. This in turn would mean that the probability of translating an extended triple pattern into any triple that is not extended with all of the keywords in the triple pattern would be zero, even if the triple is extended with some of the keywords in the triple pattern. Smoothing ensures that these triples would still have a probability mass and would be considered when ranking the query results.

In the case of extended triple patterns, our ranking model thus ranks the result subgraphs based on the weights of the triples with respect to the keywords in the query. Given that the triple-keyword weights reflect the importance of the triples in the context of the keywords, our ranking model will rank the result subgraphs based on their relevance to both the triple patterns, the structured part of the query, and the keywords, the unstructured part of the query. Table 5 shows the top-10 ranked results in DBpedia for the query:

```
?x starring ?y [novel]
?x director ?y
```

which can be used to find movies based on a novel that are starred and directed by the same person.

4. Query Relaxation

In Section 3, we proposed a ranking model to rank the results of triple-pattern queries, possibly extended with keywords. This addresses the case when queries return too many results and ensures that the user is provided with a ranked list of result subgraphs based on their relevance to the query. On the other hand, some triple-pattern queries would return no results due to the exact-matching semantics of triple-pattern queries (i.e., a result subgraph must be entirely isomorphic to the triple-pattern query). To address this issue and improve the recall of such queries, we propose to perform automatic query relaxation to retrieve result subgraphs that *partially* match a given user query. This is done in judicious manner to ensure that the relaxed queries are close in spirit to the original query, thus improving recall without unduly sacrificing precision.

Our query relaxation approach works as follows. Given a triple-pattern query $Q = \{q_1, q_2, \ldots, q_n\}$, where q_i is a triple pattern, possibly extended with keywords. In case the query Q does not yield any results, we generate r relaxed queries $R_1, R_2, \ldots, R_r$ from Q. The way we generate these relaxed queries depend on the reason why the original query Q did not yield any results. There are three cases which can be enumerated as follows.

Case I:. One or more constants (i.e., URIs or literals) in the query do not exist in the knowledge graph. For example, the query:

```
?x director Woody_Allan
?x starring Woody_Allan
```

would not yield any results when run on DBpedia because the object of the first and second triple pattern, `Woody_Allan`, does not exist in the knowledge graph (`Woody_Allen` is misspelled as `Woody_Allan` in the query). In this case, we identify the constants that do not exist in the knowledge graph, relax each such constant into a variable, and extend the triple pattern that contains the problematic constant with keywords extracted from the relaxed constant. For our example query, since there is only one constant that does not exist in the knowledge graph, we generate the following relaxed query:

```
?x director ?z [woody allan]
?x starring ?z [woody allan]
```

Note that since both triple patterns contain the same constant `Woody_Allan`, it will be relaxed into the same variable in both triple patterns.

Case II:. One or more triple patterns do not have any matches in the knowledge graph even though all the constants in the query exist. For example, the query:

```
?x director Woody_Allen
?x producer Scarlett_Johansson
```

asks for movies directed by Woody Allen and produced by Scarlett Johansson. This query would not yield any results when run on DBpedia since the second triple pattern does not have any matching triples in the knowledge graph, even though the constants mentioned in the triple pattern exist in the knowledge graph. In this case, we first identify the triple patterns which do not have any matching triples, then relax only those by replacing each constant in the identified triple patterns with a variable and extending the corresponding triple patterns with keywords extracted from the relaxed constants. For our example query above, the following two relaxed queries will thus be generated:

```
?x director Woody_Allen
?x ?y Scarlett_Johansson [producer]

?x director Woody_Allen
?x producer ?y [scarlett johansson]
```

Case III:. The join condition in the query cannot be satisfied even though each triple pattern individually matches some triples in the knowledge graph. For example, consider the query:

```
?x starring Ryan_Reynolds
?x director Woody_Allen
```

This query would not yield any results when run on DBpedia since there exists no movies in the knowledge graph that were directed by Woody Allen and starred by Ryan Reynolds. However, each triple pattern individually would yield results, namely movies starred by Ryan Reynolds and movies directed by Woody Allen, respectively. In this case, we generate r relaxed queries where r is the number of constants in the query by replacing one constant at a time with a variable and extending the corresponding triple pattern with keywords extracted from the relaxed constant. For our example query, we will thus generate the following four relaxed queries:

```
?x starring ?y [ryan reynolds]
?x director Woody_Allen

?x ?y Ryan_Reynolds [starring]
?x director Woody_Allen

 ?x starring Ryan_Reynolds
 ?x ?y Woody_Allen [director]

 ?x starring Ryan_Reynolds
 ?x director ?y [woody allen]
```

Once a set of relaxed queries $R_1, R_2, \ldots, R_r$ are generated, they are run on the searched knowledge graph and their result subgraphs are retrieved. Our final step is to combine the results of these queries and provide them to the user in a ranked manner. To do this, we extend our

Subject	Predicate	Object
Celebrity_(film)	starring	Kenneth_Branagh
Celebrity_(film)	director	Woody_Allen
Shadows_and_Fog	starring	Madonna
Shadows_and_Fog	director	Woody_Allen
Annie_Hall	starring	Woody_Allen
Annie_Hall	director	Woody_Allen
Blade:_Trinity	starring	Ryan_Reynolds
Blade:_Trinity	director	David_S._Goyer
Crimes_&_Misdemeanors	starring	Woody_Allen
Crimes_&_Misdemeanors	director	Woody_Allen
Sleeper_(1973_film)	starring	Woody_Allen
Sleeper_(1973_film)	director	Woody_Allen
New_York_Stories	starring	Woody_Allen
New_York_Stories	director	Woody_Allen
Broadway_Danny_Rose	starring	Woody_Allen
Broadway_Danny_Rose	director	Woody_Allen
Stardust_Memories	starring	Woody_Allen
Stardust_Memories	director	Woody_Allen
Annie_Hall	starring	Paul_Simon
Annie_Hall	director	Woody_Allen

Table 6: Top-10 results for the triple-pattern query asking for movies starred by Ryan Reynolds and directed by Woody Allen. This query would not yield any results when run on DBpedia. The query will thus be relaxed to retrieve movies directed by Woody Allen only or movies starred by Ryan Reynolds only.

Subject	Predicate	Object
Annie_Hall	starring	Woody_Allen
Annie_Hall	director	Woody_Allen
Shadows_and_Fog	starring	Woody_Allen
Shadows_and_Fog	director	Woody_Allen
Sleeper_(1973_film)	starring	Woody_Allen
Sleeper_(1973_film)	director	Woody_Allen
New_York_Stories	starring	Woody_Allen
New_York_Stories	director	Woody_Allen
Hollywood_Ending	starring	Woody_Allen
Hollywood_Ending	director	Woody_Allen
Stardust_Memories	starring	Woody_Allen
Stardust_Memories	director	Woody_Allen
Broadway_Danny_Rose	starring	Woody_Allen
Broadway_Danny_Rose	director	Woody_Allen
Small_Time_Crooks	starring	Woody_Allen
Small_Time_Crooks	director	Woody_Allen
Love_and_Death	starring	Woody_Allen
Love_and_Death	director	Woody_Allen
Scoop_(2006_film)	starring	Woody_Allen
Scoop_(2006_film)	director	Woody_Allen

Table 7: 10 ranked result subgraphs for the triple-pattern query asking for *comedy* movies starred and directed by the same person.

ranking model described in Section 3 as follows. Given a subgraph G which is a result of *one or more* relaxed query, we compute its score as the probability of the subgraph generating the original query Q as follows:

$$P(Q|G) = \frac{1}{r}\sum_{i=1}^{r} P(R_i|G) \quad (9)$$

where r is the number of relaxed queries generated for query Q and $P(R_i|G)$ is the probability of subgraph G generating relaxed query R_i, which is in turn computed using our ranking model described in Section 3.

The intuition behind the above weighted sum scoring is that a sugbraph G might be a result to multiple relaxed queries. That is, the more relaxed queries a subgraph G is a result of, the higher its score should be. Table 6 shows the top-10 results in DBpedia for our triple-pattern query asking for movies directed by *Woody Allen* and starring *Ryan Reynolds*. Since this query will not return any results, it will be relaxed as explained above and the results of the relaxed queries will then be combined and ranked based on Equation 9.

5. Result Diversity

While our ranking model described in Section 3 ranks the results of queries based on some notion of relevance, it is often the case that the top-ranked results are very homogenous. This is a common problem in IR in general [2], commonly referred to as result diversity. For example, consider the following example query:

```
?x starring ?y [comedy]
?x director ?y
```

As can be seen from Table 7, the top-10 ranked results of such query are dominated by Woody Allen movies. In such a case, it might be more interesting to show the user a diversified set of results, say movies by different directors, or movies with different genres or a mix of blockbuster and independent movies. To be able to achieve this goal, a diversity-aware ranking model must be used to rank the results of a given query such that the top-k results are most relevant to the query and at the same time as diverse from each other as possible. This problem can be cast as an optimization problem known as the *top-k set selection* problem [5], which is defined as follows.

Definition 6 (Top-k Set Selection Problem). *Let Q be a query and U be its result set. Furthermore, let REL be a function that measures the relevance of a subset of results $V \subseteq U$ with respect to Q and let DIV be a function that measures the diversity of a subset of results $V \subseteq U$. Finally, let f be a function that combines both relevance and diversity. The top-k set selection problem is the optimization problem solved by finding $V^* \subseteq U$ where:*

$$V^* = \underset{V \subseteq U}{\operatorname{argmax}} \; f(Q, V, REL, DIV), \; such \; that |V^*| = k \quad (10)$$

In order to solve the top-k set selection problem, one must clearly specify both the relevance function REL and

the diversity function DIV, as well as how to combine them. Gollapudi and Sharma [5] proposed a set of axioms to guide the choice of the objective function f and they showed that for most natural choices of relevance and diversity functions, and the combination strategies between them, the above optimization problem is NP-hard. For instance, one such choice of the objective function is as follows:

$$f(Q,V,REL,DIV) = (k-1)\sum_{G\in V} rel(G,Q) + 2\lambda \sum_{G,G'\in V} d(G,G') \quad (11)$$

where $rel(G,Q)$ is a (positive) score that indicates how relevant result G is with respect to query Q (the higher this score is, the more relevant G is to Q) and $d(G,G')$ is a discriminative and *symmetric* distance measure between two results G and G', and λ is a weighting parameter.

The objective function described in Equation 11 clearly trades off both relevance of results in the top-k set with their diversity (as measured by their average distance). As mentioned earlier, solving such objective function is NP-hard, however there exists some known greedy algorithms that approximately solve this problem such as the Maximal Marginal Relevance algorithm [2] proposed by Carbonell and Goldstein, where marginal relevance is defined as follows.

Definition 7 (Marginal Relevance). *Given a query Q, a set of results U and a subset $V \subset U$, the marginal relevance of a result $G \in U \setminus V$ is computed as follows:*

$$MR(G,Q,V) = \lambda rel(G,Q) + (1-\lambda)\min_{G'\in V} d(G,G') \quad (12)$$

where $rel(G,Q)$ is a measure of how relevant result G is to query Q, $d(G,G')$ is a symmetric distance measure between G and G' and λ is a weighting parameter.

The idea behind the marginal relevance metric is very intuitive. Given a query Q and a set of already selected results V, the marginal relevance of a result G is a measure of how much do we gain in terms of both relevance and diversity by adding the result G to the selected set V. To measure how much the result G would contribute to the relevance of V, we can use the result's relevance to Q. On the other hand, measuring how much result G would contribute to the diversity of V is more involved. The most natural thing to do is to compare G with all the results $G' \in V$ and compute a similarity (or rather dissimilarity) between G and every other result $G' \in V$, and then aggregate these similarities over all the results in V. This is exactly what is done in the Maximal Marginal Relevance (MMR) approach [2] by assuming there is a distance function that can measure how result G is different from any other result G' and then using the *minimum* of the distances of G from all the results $G' \in V$ as a measure of the overall contribution of result G to the diversity of set V. By maximizing this minimum over the set of results not yet in V, we can find the result that when added to V would render it most diverse as compared to any other result.

We thus adopt the above described MMR approach to the setting of RDF and use it to diversify the results of triple-pattern queries, whether extended with keywords or not. To be able to do this, the following two aspects must be precisely defined. First, we need to define how to measure the relevance of a given result subgraph G with respect to a triple-pattern query Q, or $rel(G,Q)$ in Equation 12. Second, we must also define how to measure the distance between two result subgraphs, i.e., $d(G,G')$ in Equation 12.

The answer of the first question above is straight forward. To compute the relevance of a subgraph G with respect to query Q, we can rely on our ranking model described in Section 3, which computes the relevance of subgraph G with respect to query Q as the probability of subgraph G generating Q, or $P(Q|G)$ as defined in Equation 3. To measure the distance between RDF subgraphs, we assume that each result subgraph is represented using a language model (LM), which is a probability distribution over some terms. We then measure the distance between two subgraphs as the distance between their corresponding language models. We first explain how to build the language models for the result subgraphs then how to measure the distance between them.

We propose three different methods to build language models for results subgraphs. Each method can achieve a different level or notion of diversity.

Definition 8 (Knowledge-graph-based LM). *Given a subgraph G, its knowledge-graph-based language model is a probability distribution over all the terms (i.e., URIs and literals) in the RDF knowledge graph. The parameters of this language model are estimated using a smoothed maximum likelihood estimator as follows:*

$$P(x|G) = \alpha\frac{c(x;G)}{|G|} + (1-\alpha)\frac{1}{|KG|} \quad (13)$$

where x is a term, $c(x;G)$ is the number of times term x occurs in G, $|G|$ is the number of times all terms occur in G, $|KG|$ is the number of unique terms in the knowledge graph, and α is a smoothing parameter.

Using the knowledge-graph-based language modeling, the result subgraphs will be diverse in terms of the URIs and literals that appear in the subgraphs. This ensures that no one resource or literal will dominate the result set.

Definition 9 (Query-based LM). *The query-based language model of result G is a probability distribution over all the unigrams in the knowledge graph. The parameters of this language model are estimated using a smoothed maximum likelihood estimator as follows:*

$$P(x|G) = \alpha\frac{c(x;G)}{|G|} + (1-\alpha)\frac{1}{|KG|} \quad (14)$$

such that $x \notin Qgrams$, where x is a unigram, $Qgrams$ is the list of all the unigrams extracted from the query, $c(x;G)$ is the number of times unigram x occurs in G, $|G|$ is the number of times all unigrams occur in G, $|KG|$ is the number of unique unigrams in the knowledge graph, and α is a smoothing parameter.

The unigrams are extracted from the knowledge graph by tokenizing the URIs and literals. Similar to the knowledge-graph-based diversity notion, the query-based one ensures that no one resource or literal will dominate the result set. However, in this notion of diversity, we diversify the results of queries in terms of the variable bindings only (i.e., the resources and literals in the result subgraphs that correspond to variables in the query). By excluding the unigrams that appear in the query when building the language model of each subgraph, we also ensure that when these language models are later used for diversity, the top-ranked subgraphs stay close to the original user query. This is particularly important for the case of keyword-extended queries and when relaxing queries.

Definition 10 (Text-based LM). *The text-based language model of a subgraph G is a probability distribution over all the keywords in the extended knowledge graph. The parameters of the this language model are estimated using a smoothed maximum-likelihood estimator as follows:*

$$P(x|G) = \alpha \frac{c(x;G)}{|G|} + (1-\alpha)\frac{1}{|KG|} \tag{15}$$

where $c(x;G)$ is the number of times keyword x occurs in G, $|G|$ is the number of occurrences of all keywords in G, $|KGl|$ is the number of unique keywords in the extended knowledge graph, and α is a smoothing parameter.

In the text-based diversity notion, our goal is to diversify the result subgraphs with respect to the keywords used to extend the triples in the subgraph. By doing so, we ensure that the subgraphs would have different textual context and would thus represent different aspects of the query results.

Once the language models of the result subgraphs have been built using one of the two methods described above, our goal next is to measure the distance between the subgraphs using their corresponding language models. In order to do this, we rely on the Jensen-Shannon divergence [6] as follows:

$$d(G,G') = \sqrt{JS(G||G')} \tag{16}$$

where $JS(G||G')$ is the Jensen-Shannon divergence between the language models of subgraphs G and G' which is computed as follows:

$$JS(G||G') = \frac{KL(G||M) + KL(G'||M)}{2} \tag{17}$$

Algorithm 1: MMR-based re-ranking algorithm

Input : U = Set of results ranked by relevance only
Output: S = Set of results ranked by relevance and diversity

```
S(0) = U(0)
i = 0
min = ∞
while i < k AND i < |U| do
    for result r ϵ U\S do
        for result r' ϵ S do
            M = r ∪ r'
            d(r,r')=(KL(r,M)+KL(r',M))/2
            if d(r,r') < min then
                min = d(r,r')
            end
            Score(r)=λ rel(r)+(1-λ)min
        end
        S(i) = max(U\S) ; // Pick the result with
         the highest new score
        i++
    end
    return S
end
```

where $M = \frac{1}{2}(G+G')$ is the average of the language models of G and G', and $KL(G||M)$ is the Kullback-Leibler divergence between the language models of G and M, which can be computed as follows:

$$KL(G||M) = \sum_i P(x_i|G) log \frac{P(x_i|G)}{P(x_i|M)} \tag{18}$$

We opted for using the Jensen-Shannon Divergence as its square root is a symmetric distance measure which is exactly what is required in the MMR approach.

To summarize, we diversify the results of a given triple-pattern query by first using our ranking model described in Section 3 to retrieve a ranked list of subgraphs for the given query. Next, we re-rank the results using the MMR approach as described above. Algorithm 1 displays the pseudo-code of our MMR-based re-ranking algorithm.

Table 9 show the top-10 results in DBpedia, re-ranked by MMR using the knowledge-graph-based language models, for the query:

```
?x director ?y [comedy]
?x starring ?y
```

which can be used to find comedy movies starred and directed by the same person. Without re-ranking using MMR, the result set was dominated by one resource, namely Woody Allen, as can be seen in Table 7. On the other hand, the re-ranked results are diverse and consist of comedy movies by different people, as can be seen in Table 9.

Table 8 shows the top-10 result subraphs for the keyword-extended query:

Knowledge-graph-based Diversified Result Set			Query-based Diversified Result Set		
Subject	**Predicate**	**Object**	**Subject**	**Predicate**	**Object**
City_Lights	starring	Charlie_Chaplin	City_Lights	starring	Charlie_Chaplin
City_Lights	director	Charlie_Chaplin	City_Lights	director	Charlie_Chaplin
Braveheart	starring	Mel_Gibson	Limelight	starring	Charlie_Chaplin
Braveheart	director	Mel_Gibson	Limelight	director	Charlie_Chaplin
Pinched	starring	Harold_Lloyd	Braveheart	starring	Mel_Gibson
Pinched	director	Harold_Lloyd	Braveheart	director	Mel_Gibson
Annie_Hall	starring	Woody_Allen	Pay_Day	starring	Charlie_Chaplin
Annie_Hall	director	Woody_Allen	Pay_Day	director	Charlie_Chaplin
Boxes	starring	Jane_Birkin	The_Tramp	starring	Charlie_Chaplin
Boxes	director	Jane_Birkin	The_Tramp	director	Charlie_Chaplin
Awara	starring	Raj_Kapoor	The_Bond	starring	Charlie_Chaplin
Awara	director	Raj_Kapoor	The_Bond	director	Charlie_Chaplin
Cruel_Summer	starring	Kanye_West	The_Rink	starring	Charlie_Chaplin
Cruel_Summer	director	Kanye_West	The_Rink	director	Charlie_Chaplin
The_Alamo	starring	John_Wayne	The_Count	starring	Charlie_Chaplin
The_Alamo	director	John_Wayne	The_Count	director	Charlie_Chaplin
Sleeper	starring	Woody_Allen	Cupid's_Rival	starring	Billy_West
Sleeper	director	Woody_Allen	Cupid's_Rival	director	Billy_West
Citizen_Kane	starring	Orson_Welles	The_Kid	starring	Charlie_Chaplin
Citizen_Kane	director	Orson_Welles	The_Kid	director	Charlie_Chaplin

Table 8: top-10 results for the triple-pattern query asking for movies starred and directed by the same person that have something to do with Chaplin after re-ranking by the MMR algorithm using the knowledge-graph-based language models and the query-based language models.

```
?x starring ?y [chaplin]
?x director ?y
```

asking for movies that were directed and starred by the same person, and have something to do with Chaplin. The first column shows the top-10 results after re-ranking by the MMR algorithm using knowledge-graph-based language models. The second column shows the top-10 results after re-ranking by the MMR algorithm using the query-based language models. As can be seen from the table, in the knowledge-graph-based diversity notion, the results are diversified with respect to the resources and thus movies by different are displayed. On the other hand, the query-based diversity approach takes into consideration the unigrams in the query and thus trades off diversity for relevance to the original query. In our case, the query-based diversity notion would thus ensure that the diversified results would still consist of movies that have something to do with Chaplin, as is intended by the query.

Table 10 shows the top-10 results for the query:

```
?x distributor Warner_Bros
```

asking for movies that were distributed by Warner_Bros, with and without diversity. In the first column of the table, we show the non-diversified top-10 results (i.e., ranked by relevance only), and in the second column we display the top-10 results after re-ranking by the MMR algorithm using the text-based language models. While both result sets are not diverse in terms of resources, the first result set is actually a lot more homogeneous than the second one. In the first result set (first column of Table 10), most of the movies are either science fiction movies or super hero movies. Moreover, almost half of these movies are directed by Christopher Nolan. On the other hand, the movies in

Subject	**Predicate**	**Object**
Annie_Hall	starring	Woody_Allen
Annie_Hall	director	Woody_Allen
City_Lights	starring	Charlie_Chaplin
City_Lights	director	Charlie_Chaplin
A_Bronx_Tale	starring	Robert_De_Niro
A_Bronx_Tale	director	Robert_De_Niro
The_Nutty_Professor	starring	Jerry_Lewis
The_Nutty_Professor	director	Jerry_Lewis
That_Thing_You_Do!	starring	Tom_Hanks
That_Thing_You_Do!	director	Tom_Hanks
Project_A	starring	Jackie_Chan
Project_A	director	Jackie_Chan
Harlem_Nights	starring	Eddie_Murphy
Harlem_Nights	director	Eddie_Murphy
Bill_Cosby:_Himself	starring	Bill_Cosby
Bill_Cosby:_Himself	director	Bill_Cosby
Tropic_Thunder	starring	Ben_Stiller
Tropic_Thunder	director	Ben_Stiller
Blazing_Saddles	starring	Mel_Brooks
Blazing_Saddles	director	Mel_Brooks

Table 9: Top-10 result subgraphs for the triple-pattern query asking for *comedy* movies starred and directed by the same person. The results were re-ranked by the MMR algorithm using the knowledge-graph-based language models.

Non-Diversified Result Set			Diversified Result Set		
Subject	**Predicate**	**Object**	**Subject**	**Predicate**	**Object**
The_Dark_Knight	distributor	Warner_Bros	The_Dark_Knight	distributor	Warner_Bros
Tom_and_Jerry	distributor	Warner_Bros	Sweeney_Todd	distributor	Warner_Bros
Blade_Runner	distributor	Warner_Bros	The_Coo-Coo_Nut_Grove	distributor	Warner_Bros
2001:_A_Space_Odyssey	distributor	Warner_Bros	Dragnet	distributor	Warner_Bros
Dragnet	distributor	Warner_Bros	2001:_A_Space_Odyssey	distributor	Warner_Bros
Slumdog_Millionaire	distributor	Warner_Bros	Dive_Bomber	distributor	Warner_Bros
Citizen_Kane	distributor	Warner_Bros	Blade_Runner	distributor	Warner_Bros
Batman_Begins	distributor	Warner_Bros	Battlefield_Earth	distributor	Warner_Bros
Inception	distributor	Warner_Bros	Mindscape	distributor	Warner_Bros
Batman	distributor	Warner_Bros	Adventures_of_Road_Runner	distributor	Warner_Bros

Table 10: Top-10 ranked result subgraphs for the triple-pattern query asking for movies distributed by Warner Bros without diversity and with diversity using the text-.based language models.

the second result set (second column of Table 10) are more diverse in terms of genre, movie plot, cast, etc. This would only be achievable if we take into consideration the keywords associated with the result subgraphs as is done in the case of the text-based diversity notion.

6. Related Work

Utilizing RDF data for information retrieval and knowledge discovery is an ongoing research endeavor in the Semantic Web community. A recent survey by Bast et al. [7] provides a comprehensive review of a large body of systems and approaches in this area. Instead, we only review works that address one of the challenges we tackled here when it comes to searching RDF knowledge graphs. We assess such works and contrast them to ours based on three criteria, namely 1) effective querying (i.e., keyword support and query relaxation), 2) result ranking, and 3) result diversity.

One of the motivation of our work was the incompleteness of RDF knowledge graphs and the rigidity of triple-pattern-based query languages such as SPARQL. To address these issues, we proposed extending both knowledge graphs and triple-pattern queries with keywords, as well as performing automatic query relaxation. Other approaches addressed the issue of inflexibility of triple-pattern search by enabling users to search RDF knowledge graphs using some form of natural language. This family of works can be broadly categorized into three main families: natural language question answering, keyword search and hybrid search that uses a combination of triple patterns and keywords.

The approaches in the first group [8, 9, 10, 11, 12, 13, 14, 14, 15, 16, 17] mostly work by translating the natural language question into a set of candidate triple-pattern queries. This is mainly done by analyzing the structure of the question and mapping parts of the question to resources in the knowledge graph. They then rank the candidate triple-pattern queries and execute the top query to retrieve the result subgraphs. While these approaches provide a user-friendly approach to search RDF knowledge graphs, they do not provide any *result ranking* for the generated triple-pattern query. They also do not overcome the issue of data incompleteness that all RDF knowledge graphs suffer from, with the exception of [9, 10] which uses a similar framework to ours that we review later in this section when we discuss hybrid querying approaches. In addition, none of the aforementioned approaches address the issue of result diversity either. Our work is thus complementary to these approaches and can be combined with them by allowing users to pose their queries as natural language questions, which can be translated into a set of *extended* triple-pattern queries that are processed using our framework to provide users with ranked and diverse result subgraphs.

The approaches in the second group [18, 19, 20, 21, 22, 23] allow users to search RDF knowledge graphs using keyword queries. The keyword query is then mapped into one or more triple-pattern query [18, 19, 20, 21, 22]. Again in all these approaches, result ranking and diversity for the generated candidate queries are not considered, as is the case with the approaches that deal with natural language questions.

The third group of works that deals with flexible querying of RDF knowledge graphs usually use a combination of text and RDF triples, either on the data side or the querying side or both. The framework proposed here is inspired by few such approaches, which are earlier individual works of ours on result ranking [24, 25], query relaxation [26, 24, 25], and result diversity [27] for triple-pattern queries. However, none of our previous work presented a general framework that combines all three aspects. Moreover, our framework proposed here uses a novel ranking model for result ranking based on statistical machine translation, which is more intuitive than our previous result ranking model. Finally, all our previous attempts were evaluated on small RDF knowledge graphs in contrast to our framework proposed here, which is evaluated on a large real-world RDF knowledge graph, namely DBpedia.

Other related approaches that use combined RDF data

and text [28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38] usually rely on keyword queries or natural language questions as means of searching and retrieve a set of ranked resources (i.e., entities). One example is the Fielded Sequential Dependence work [36] that proposed augmenting the RDF entities with weighted fields. These fields are retrieved from DBpedia [1] and Wikipedia. As an example, the entity `Barack_Obama` would be associated with the following set of fields: `names, attributes, categories, etc`. Each one of these fields contains a number of keywords describing the entity. This is close to what we do when it comes to augmenting our RDF triples. However, in their work, the input is a set of keywords, and the output is an entity or a list of ranked entities. In our case, the input is a triple-pattern query, possibly augmented with keywords, and the output is a a set of *ranked* RDF subgraphs. A similar work [39] embraces the idea of entity features retrieved from Wikipedia. They use them later on in their learn-to-rank algorithm, which is a modified algorithm for tensor factorization. In this work, the input is also a set of keywords and the output is a list of relevant entities. Another work also focuses on a learning-to-rank approach [37], where the authors consider different features that use documents that are query-related, entity mentions, and knowledge base entities. They combine documents and entity retrieval, given an unstructured query. In addition, there is a work [38] that proposes a framework that depends on structured and unstructured information to produce a list of ranked entities, given an unstructured query. Their model depends on the RDF graph, textual and link analysis, which is similar to what we do (using DBpedia links and abstracts), and external information from popular search engines (for example, Google), however again, the output is a list of ranked entities rather than RDF subgraphs as in our case.

Most of the above approaches assume the query results to be single resources and thus fail to address more complex information needs where the goal of the user is to find subgraphs connecting multiple resources from the knowledge graph. There are only very few exceptions to this. The first is our previous work [40] where the results of keyword queries were assumed to be RDF subgraphs. However, in this work, the result subgraphs were ranked based on how well they match the keywords in the query rather than taking into consideration the relative importance of the triples that constitute the subgraphs. In addition, result diversity was not tackled in this work. The second is a recent work [41] where the authors propose to extend the knowledge graph to account for missing entities, classes, predicates, or facts. They combine the knowledge graph with textual corpora that might or might not include facts that already exist in the knowledge graph. To search the extended knowledge graph, they propose an extension to triple-pattern queries where the subject, predicate, or object can be expressed using keywords instead of URIs. However, in their approach, result ranking is based on how well the result subgraphs match the keywords expressed in the query without taking into consideration the relative importance of the triples in the result subgraphs as in our case. They also provide query relaxation which is very similar in spirit to the query relaxation techniques we proposed here, however they do not address the issue of result diversity like we do.

Another major challenge that we addressed in this work is result ranking for triple-pattern queries, possibly extended with keywords. One of the earliest work on this is the NAGA system [42] , which contains a ranking model for triple-pattern queries over RDF knowledge graphs. However, NAGA does not support neither extended triple-pattern queries nor query relaxation. It also does not address the issue of result diversity for triple-pattern queries as we do here.

On the other hand, some related work [43, 44, 44] focused on the problem of query relaxation in RDF knowledge graphs. The first such work [43] proposed a framework to generate a set of relaxed queries for a given triple-pattern query. However, the results of each relaxed query are all scored equally based on how close the relaxed query is to the original query. In contrast, our framework only generates relaxed queries that are close to the original query by extending triple patterns with keywords and then takes into consideration the keywords as an additional means of scoring. Thus, the result subgraphs for each relaxed query might be scored differently.

The other related works specifically on query relaxation for RDF search [45, 44] focused on investigating techniques to find the parts in the original query that are responsible for its failure. In our work, we also do this and produce different types of relaxed queries based on a thorough analysis of why the original query did not yield any results. More importantly, we also combine and rank the results of all the relaxed queries we generate and we ensure the results to be diverse, thus showing different facets of all possible relaxations. Another work that tries to define the reason of the failing query is DebEAQ [46], a debugging tool. The authors combines two approaches, namely discovering missing subgraphs of a data graph, and rewriting the queries such that they deliver some results.

Another work [47] proposed rewriting part of the query. The "optional clause" is the part that allows the query to find matching that fail to match some conditions in the query. For example, consider they transform the following query:

```
?x name ?z
?x proceedingsEditorOf ?y
```

to:

```
?x name ?z
?x editorOf ?y
```

where the second triple pattern is an optional one. Moreover, the predicate `proceedingsEditorOf` is a sub-property of the predicate `editorOf`. They proceed to order their

relaxed versions of the original query from less to more general from a logical standpoint.

One work [48] combined query relaxation techniques with query approximation, which is based on edit distance and RDFS-based inference rules. The querying system can perform both approximate matching and relaxation of the user's query. The system rank the results of the relaxed queries by assessing how closely they match the original query. Moreover, the approach on approximate querying over RDF data [49, 50] proposed a measure to evaluate the similarity between a query and an answer. They define a scoring function to compute the similarity distance, which is based on quality (how much the paths retrieved align with the paths in the query) and conformity (how much the combination of the paths in the answer is similar to the combination of the paths in the query). Almost all of the relaxation works focused on finding the reason of the query failure and rewriting, or reformulating, the structured part of the query. On the other hand, in our work, we do not change, drop, or add triple patterns. We benefit from the keyword augmentation feature that we provide to our user. More particularly, the constants of the query with zero results are transformed into a set of keywords accompanying the relaxed query. By following this technique, we do not sacrifice any part of the user's query, which in turn allows us to find results as close as possible to the original information need. We also generate multiple relaxed queries to cover as many aspects of the original query as possible, and combine the results of such relaxed query in a principled manner using our ranking model based on statistical machine-translation.

Finally, when it comes to result diversity in the context of RDF search, to the best of our knowledge, no prior work addressed this challenge. There is a wealth of work on search result diversity in general [51, 2, 52, 53, 5, 54]. However none of these approaches are applicable in the RDF setting. Most of the techniques work for document retrieval and diversify the results by optimizing a bi-criteria objective function that takes into consideration both result relevance as well as result diversity. Typically, result diversity is measured by computing similarities between results. However, in the RDF search setting, since results are constructed at query time by joining triples, we do not have an explicit notion of result similarity that can be easily utilized. In our framework, we defined this notion of result similarity or diversity and showed how we can incorporate it in the well established Maximal Marginal Relevance framework [2] to diversify the results of triple-pattern queries.

Apart from document retrieval, there is very little work on result diversity for queries over structured data in general [55, 56, 57, 58]. For instance some of these approaches deal with diversifying the results of SQL queries [55, 58, 57] by making use of predefined result categories while others deal with Web resources for instance [56]. None of these approaches are again applicable to the case of triple-pattern queries whose results are constructed on the fly at query time. In addition, none of these approaches take into consideration the relevance of the results but instead focus solely on result diversity. On the other hand, in our framework, our goal is provide users with *diverse* and *relevant* results to their queries.

7. Case Study: DBpedia

7.1. Dataset

To evaluate the effectiveness of our framework, we conducted a case study on a large real-world RDF knowledge graph, namely DBpedia [1]. DBpedia has 1337 unique predicates and 15.8 million triples. The knowledge graph was stored as a set of RDF triples in a relational database managed by PostgreSQL [59].

The knowledge graph was extended with keywords extracted from the *abstracts* of the resources in the triples. The abstract of a resource is the first paragraph of the Wikipedia article of the resource. More precisely, for each triple in the knowledge graph, we extracted all the keywords in the abstracts of both the subject and the object of the triple, in case they were URIs, stemmed these keywords, removed stop words and duplicates, and then extended the triple with the resulting keywords. In case the object of the triple was a literal, we just used the literal itself as a keyword.

To set the weights of the triples and the triple-keyword pairs, the Wikipedia link-structure was used. To compute the weight $w(t)$ of triple $t = (s, p, o)$, we used the number of Wikipedia articles that contain a hyperlink to the subject or the object of triple t as follows:

$$w(t) = |links(s)| + |links(o)| \tag{19}$$

where $links(s)$ is the set of Wikipedia articles that contain a hyperlink to the subject s of triple t, and $links(o)$ is the set of Wikipedia articles that contain a hyperlink to the object o of triple t. The intuition behind this is that the more Wikipedia articles that link to the subject and the object of the triple, the more important the triple is (i.e., a measure of authority).

To compute the weight $w(t, v)$ of a triple $t = (s, p, o)$ with respect to keyword v, we again used the Wikipedia link-structure as follows:

$$\begin{aligned} w(t, v) = &|links(s) \cap links(v)| \\ &+ |links(o) \cap links(v)| \end{aligned} \tag{20}$$

where $links(s)$ is the set of Wikipedia articles that contain a hyperlink to the subject s, $links(o)$ is the set of Wikipedia articles that contain a hyperlink to the object o, and $links(v)$ is the set of Wikipedia articles that contain a hyperlink to an article containing keyword v. By taking the intersection between $links(s)$ and $links(v)$, and similarly $links(o)$ and $links(v)$, we are effectively computing the weight of triple t with respect to keyword v as the number of Wikipedia articles that mention both the subject of

Information Need	Query
Triple-pattern queries with many results	
Movies starred and produced by the same person	?m starring ?x ?m executiveProducer ?x
Romance novelists and their birth places	?x genre Romance_novel ?x birthPlace ?p
Extended triple-pattern queries with many results	
Music composers of movies about gunfight and battle	?m musicComposer ?c [gunfight battle]
Books written by award-winning feminist authors	?b author ?x [feminist] ?x award ?a
Triple-pattern queries with zero results	
Movies produced by Woody Allen and starring Scarlett Johansson	?m producer Woody_Allen ?m director Scarlett_Johansson
Oxford graduates born in Agatha Christie's death place	Agatha_Christie deathPlace ?p ?x birthPlace ?p ?x almaMater Oxford
Extended triple pattern queries with zero results	
People who influenced Egyptian writers	?w birthPlace Egypt ?w occupation ?o [writer] ?w influencedBy ?x
Comedy movies starring a couple and directed by Sinatra	?m starring ?x [comedy] ?m starring ?y ?x spouse ?y ?m director Frank_Sinatra

Table 11: Example queries from our query benchmark.

t and the keyword v (and similarly for the object o). This way, a triple will have a high weight with respect to a keyword v if both its subject and object co-occur frequently with the keyword v in Wikipeda.

7.2. Benchmark

To evaluate the different components of our framework, we constructed a benchmark consisting of 139 queries, which can be broadly divided into 4 categories: 63 triple-pattern queries with many results, 64 extended triple-pattern queries with many results, 9 triple-pattern queries with zero results, and 3 extended triple-pattern queries with zero results. Table 11 shows two example queries from each category. In the whole benchmark, the average number of triple patterns per query was 2.2 and the average number of keywords per an extended query was 1. 100 queries in the benchmark were used to evaluate the various aspects of our framework, which are result ranking and query relaxation (Section 7.3), and result diversity (Section 7.4). The remaining 39 queries were used for quality assurance, since all our user studies were conducted on CrowdFlower [60], a crowdsourcing platform. The query benchmark as well as any assessments gathered are publicly available at `https://github.com/HibaArnaout/sup_material` [5].

Note that when creating our benchmark, we had the *ranking*, *relaxation*, and *diversity* algorithms in mind. We tried coming up with information needs that would help showing the difference between a well ranked set of results and a badly ranked set of results. The same goes for well diversified and well relaxed sets. Most of the existing standard benchmarks were created for entity search. One carefully constructed example is a collection for entity search in DBpedia [61]. These benchmarks contain a majority of monotone queries. On the other hand, a minority of these queries were suitable for our framework, which we use queries similar to them in our benchmark.

7.3. Result Ranking and Query Relaxation

Our first experiment was used to evaluate our ranking model described in Section 3 and our query relaxation approach outlined in Section 4. We compared our ranking model to three different models. The first model is a random model, which does not provide any result ranking for the result subgraphs. This model is used to verify that result ranking is indeed crucial in the case of RDF search as we hypothesized. The second model we compare our ranking model to uses the same statistical machine translation approach as ours, however, it uses a different weighting paradigm than the one we outlined in the beginning of this section to extend our knowledge graph (Equations 19 and 20). In this model, the weights of the triples were computed based on the in-degrees of the nodes in the knowledge graph. That is, given a triple $t = (s, p, o)$ with subject s, predicate p, and object o, the weight of the

[5] The full benchmark and all assessments are also submitted as supplementary material

triple was set as follows:

$$w(t) = deg(s) + deg(o) \quad (21)$$

where $deg(s)$ and $deg(o)$ are the number of edges in the knowledge graph that are incident on s and o, respectively. This model was used to validate the weighting scheme we used to extend our knowledge graph.

The third and final model that we compare to is the language-modeling ranking approach proposed by Elbassuoni et al. [25], which is the closest approach to ours.Note that our ranking approach was a similar in spirit to the model proposed in [25]. However, our approach is much less complex and more intuitive as it models the problem of ranking result subgraphs to triple-pattern queries as a machine translation problem. At the same time, our approach still preserves the high performance when compared to other approaches, as indicated by our experimental results. Their ranking model estimates a query language model and a result subgraph language model. The result subgraphs are then ranked in increasing order of the Kullback-Leibler (KL) divergence between the query language model and the result subgraph language model. More formally, given a query Q with n triple patterns, the query language model and the result subgraphs language models will be defined over all n-tuples T (i.e., tuples of n triples). The KL-divergence between the query language model P_Q and result subgraph language model P_G is computed as follows:

$$KL(Q \,||\, G) = \sum_i P_Q(T_i) log \frac{P_Q(T_i)}{P_G(T_i)} \quad (22)$$

The parameters of the above language models were estimated using what is called witness counts. The witness count of a triple is the number of Web pages that mention both the subject and object of the triples. In case of keyword-extended triple patterns, the witness count of a triple-keyword pair will be the number of Web pages that mention the subject and object of the triple as well as the keyword. In our experiment, we estimate the witness counts using the Wikipedia link-structure by using Equations 19 and 20.

We have chosen these models to compare to since no other work deals with ranking of results to triple-pattern queries or extended triple-pattern queries. As explained in Section 6, all other related works either deal with the problem of keyword search, natural language question answering, or relationship queries where triple patterns are constructed using keywords.

To evaluate the different models, we relied on the crowdsourcing platform CrowdFlower as follows. We ran 100 queries from our benchmark four times using the four different models. We then retrieved the top-10 results returned by each model. For each query, we presented two result sets to *five different* workers, one containing the top-10 results retrieved using our ranking model, and the other one containing the top-10 results retrieved by a competitor. We also presented the workers with the information need underlying the query for which the result sets were provided. We then asked the workers to choose which result set they believed provide better answers to the given information need. The workers were also given a "same" option, which states that they do not prefer one result set over the other. This way, the top-10 results returned by our model for each query were compared to three different result sets: the *non-ranked* result set, the result set returned by the *in-degrees* model, and the one returned by the model in [25], which we refer to as *KL-DIV* model.

Table 12 shows the result set returned by our approach as well as the three compared to approaches for the following query:

```
Agatha_Christie deathPlace ?p
?x birthPlace ?p
?x almaMater Oxford
```

The query asks for Oxford graduates born in Agatha Christie's death place. This query did not yield any results when it was run over DBpedia, since the join condition in the query cannot be satisfied. As explained in Section 4, our framework would relax this query by replacing one constant at a time with variable, and extending the triple pattern with keywords extracted from the relaxed constant. That is, our framework thus generated the following five relaxed queries:

```
?a deathPlace ?p [agatha christie]
?x birthPlace ?p
?x almaMater Oxford

Agatha_Christie ?a ?p [death place]
?x birthPlace ?p
?x almaMater Oxford

   Agatha_Christie deathPlace ?p
   ?x ?a ?p [birth place]
   ?x almaMater Oxford

   Agatha_Christie deathPlace ?p
   ?x birthPlace ?p
   ?x ?a Oxford [alma mater]

   Agatha_Christie deathPlace ?p
   ?x birthPlace ?p
   ?x almaMater ?a[oxford]
```

Each one of the queries were then run and the results of all the relaxed queries were pooled together and ranked using on of the four ranking models (ours and the three competitors). As can be seen in Table 12, the top-5 results of the non-ranked and the KL-DIV models all have nothing to do with Agatha Christie, whereas the results of the In-degrees model do not have anything to do with alma mater. This can be mainly attributed to the fact that none of these three models take into consideration

Our Approach			KL-DIV Approach		
Subject	**Predicate**	**Object**	**Subject**	**Predicate**	**Object**
Agatha_Christie	deathPlace	Oxfordshire	Margaret_Copley_Thaw	deathPlace	British_Kenya
Theo_James	birthPlace	Oxfordshire	Tom_Mboya	birthPlace	British_Kenya
Theo_James	almaMater	University_of_Nottingham	Tom_Mboya	almaMater	Oxford
Agatha_Christie	deathPlace	Oxfordshire	Musa_Mwariama	deathPlace	British_Kenya
Wentworth_Miller	birthPlace	Oxfordshire	Tom_Mboya	birthPlace	British_Kenya
Wentworth_Miller	almaMater	Princeton_University	Tom_Mboya	almaMater	Oxford
Agatha_Christie	deathPlace	Oxfordshire	Henry_Harvey	deathPlace	Walmer
Theo_James	birthPlace	Oxfordshire	Robert_Bridges	birthPlace	Walmer
Theo_James	almaMater	Bristol_Old_School	Robert_Bridges	almaMater	Oxford
Viola_Keats	deathPlace	United_Kingdom	Arthur_Wellesley	deathPlace	Walmer
Robert_Bridges	birthPlace	United_Kingdom	Robert_Bridges	birthPlace	Walmer
Robert_Bridges	almaMater	Oxford	Robert_Bridges	almaMater	Oxford
Agatha_Christie	deathPlace	Oxfordshire	Dele_Charley	deathPlace	Freetown
David_Oyelowo	birthPlace	Oxfordshire	Lamina_Sankoh	birthPlace	Freetown
David_Oyelowo	birthPlace	Oxford	Lamina_Sankoh	almaMater	Oxford
Non-ranked Approach			**In-degrees Approach**		
Subject	**Predicate**	**Object**	**Subject**	**Predicate**	**Object**
Peter_Yates	deathPlace	United_Kingdom	Agatha_Christie	deathPlace	Oxfordshire
Robert_Bridges	birthPlace	United_Kingdom	Heather_Angel	birthPlace	Oxfordshire
Robert_Bridges	almaMater	Oxford	Heather_Angel	birthPlace	Oxford
Robert_Sangster	deathPlace	United_Kingdom	Agatha_Christie	deathPlace	Oxfordshire
Robert_Bridges	birthPlace	United_Kingdom	Benjamin_Whitrow	birthPlace	Oxfordshire
Robert_Bridges	almaMater	Oxford	Benjamin_Whitrow	birthPlace	Oxford
Paul_Gonsalves	deathPlace	United_Kingdom	Agatha_Christie	deathPlace	Oxfordshire
Robert_Bridges	birthPlace	United_Kingdom	David_Oyelowo	birthPlace	Oxfordshire
Robert_Bridges	almaMater	Oxford	David_Oyelowo	birthPlace	Oxford
Alexander_Mackenzie	deathPlace	United_Kingdom	Agatha_Christie	deathPlace	Oxfordshire
Robert_Bridges	birthPlace	United_Kingdom	Dexter_Blackstock	birthPlace	Oxfordshire
Robert_Bridges	almaMater	Oxford	Dexter_Blackstock	birthPlace	Oxford
Gregory_Isaacs	deathPlace	United_Kingdom	Agatha_Christie	deathPlace	Oxfordshire
Robert_Bridges	birthPlace	United_Kingdom	Garry_Parker	birthPlace	Oxfordshire
Robert_Bridges	almaMater	Oxford	Garry_Parker	birthPlace	Oxford

Table 12: Four Top-5 ranked result subgraphs for the triple-pattern query asking for Oxford graduates who were born in Agatha Christie's death place.

the keywords in the relaxed queries. In the non-ranked result set, the results are five randomly selected subgraphs that are isomorphic to one of the relaxed queries. In the KL-DIV model, its query relaxation approach does not extend triple patterns with keywords as in our approach, and thus again the keywords in any of the relaxed queries would not play any role in result ranking. Finally, in the in-degrees model, again the keywords do not play any role in ranking since the result subgraphs are ranked based on the in-degrees of their nodes. On the other hand, since our approach takes into consideration the keywords in the relaxed queries while ranking the results, our top-5 results consist of people who were born in Agatha Christie's death place and who graduated from some university close to Oxford or some people who graduated from Oxford and were born in the death place of some one closely related to Agatha Christie such as the British actress Viola Keats.

To ensure reliable results, we required the workers to pass a qualification test on which they must obtain a score of at least 90% to be able to perform the actual the evaluation task. Moreover, a set of gold queries were embedded in the evaluation task to ensure that the workers were actually assessing the results, rather than just randomly picking one of the two result sets displayed. The verification and the gold queries were queries corresponding to very obvious information needs with one very relevant result set and another obviously very irrelevant. For example, one such query asked for the area codes of African countries. We provided two result sets for this query, one displaying area codes of African countries and another set displaying area codes of Asian countries. In the evaluation task, we also asked the workers to provide us with some explanation of why they chose one dataset over the other. Overall, The inter-rater agreement reported by CrowdFlower was *76%* when comparing our approach to the non-ranked results, *88%* when comparing to the in-degrees model, and *90%*

Ours vs.	Non-ranked	In-degrees	KL-DIV
Won	72	63	6
Tied	27	33	94
Lost	1	4	0

Table 13: The results of comparing the different ranking models for 100 evaluation queries.

when comparing to the KL-DIV model. In addition, we computed the Fleiss Kappa Coefficient as another measure of agreement which is a more standard measure than that of CrowdFlower as it takes into consideration agreement by chance. We obtained an average Kappa Coefficient of 44%, which can interpreted as "Moderate Agreement" [62].

The results of our pairwise comparisons are shown in Table 13. The reported numbers are out of 100 queries. The result set retrieved by our model was chosen over the one without any ranking 72% of the time, and over the one retrieved by the in-degrees model 63% of the time. The result set returned by our model was only chosen over the third model, the one based on Kulback-Leibler divergence ([25] (KL-DIV in Table 13), only 6% of the time. However, the result sets by the two models tied in 94% of the time, indicating that the workers on CrowdFlower found them similar in most of the cases. This is very natural given that both models relied in spirit on the same criteria to rank the results, namely the Wikipeida link-structure as a means of weighting triples and triple-keyword pairs.

By outperforming the non-ranked results, we have shown that result ranking is indeed crucial as perceived by users. This is particularly true in the case of extended keyword queries, where our ranking model ensures that the result subgraphs are ranked based on their relevance to the keywords expressed in the query. Moreover, by outperforming the in-degrees model, we have shown that the way we compute the triple weights based on the Wikipedia link-structure and the way we use these weights to rank the results are indeed effective. Recall that the in-degrees model ranks the results subgraphs based on the in-degrees of the nodes, which based on our user study is not as effective in capturing the importance of the triples and their relevance to keywords as are our weights based on the Wikipedia link-structure. Finally, our model seemed to perform equally well when compared to the KL-DIV approach since both approaches use statistical language models to rank the results. Moreover, both approaches rely on the Wikipeida link-structure to estimate the parameters of the language models. However, our approach supports result diversity as well, while the KL-DIV does not. In addition, our approach also handles query relaxation in a more judicious way by extending relaxed triple patterns with keywords.

7.4. Result Diversity

7.4.1. Diversity-aware Evaluation Metric

To be able to evaluate the effectiveness of our result diversity approach, an evaluation metric that takes into consideration both relevance of results as well as their diversity must be used. There is a wealth of work on diversity-aware evaluation metrics for IR systems such as [63, 53, 54]. Our metric is inspired by the alpha-NDCG metric [53] and we adopt it and make it more specific to our setting of RDF data and our notions for relevance and diversity. We propose an evaluation metric that takes into consideration both aspects we are concerned with here, namely relevance and diversity, to evaluate the result set of a given query.

To do so, we introduce an adjustment to the Discounted Cumulative Gain (DCG) [64] metric by adding a component that takes into consideration the novelty of a certain result, which reflects result diversity in a given result set. In other words, for each result r, we assume there exist two scores: a relevance score for the result r, and a novelty score that reflects the novelty of result r with respect to the previously selected results S.

More formally, given a particular result set (a result ordering) of p results, the diversity-aware DCG, which we coin *DIV-DCG* is computed as follows:

$$DIV - DCG_p = rel_1 + nov_1 + \sum_{i=2}^{p}(\frac{rel_i}{log_2(i)} + nov_i) \quad (23)$$

where rel_i is the relevance score of the result at position i and nov_i is its novelty. We next explain how to compute the novelty score of a given result, which depends on the notion of diversity that is being used.

Knowledge-graph-based Novelty.. In our first two diversity notions, the knowledge-graph-based one and the query-based one, our goal is to diversify the results with respect to the variable bindings. We thus define the novelty of a result at position i as follows:

$$nov_i = \frac{\#unseen_i}{\#variables} \quad (24)$$

where $\#unseen_i$ is the number of resources that are bound to variables in the result at position i that have not yet been seen, and $\#variables$ is the total number of unique variables in the query. This way, result i would receive a boost in its novelty for each new resource it binds to a variable in the query.

Text-based Novelty.. The computation of the text-based novelty metric is very similar in spirit to the knowledge-graph-based one. The only difference is that in the case of text-based diversity, our goal is to diversify the results with respect to the keywords they are extended with. To be able to quantify this, we measure for each result, the amount of new keywords that this result contributes to the set of keywords of the previously ranked results. More precisely,

Notion	$\lambda = 0.1$	$\lambda = 0.2$	$\lambda = 0.3$	$\lambda = 0.4$	$\lambda = 0.5$	$\lambda = 0.6$	$\lambda = 0.7$	$\lambda = 0.8$	$\lambda = 0.9$
Knowledge-graph-based	**0.798**	0.789	0.780	0.769	0.762	0.758	0.751	0.740	0.727
Query-based	**0.798**	0.785	0.770	0.759	0.752	0.747	0.741	0.731	0.724
Text-based	**0.932**	0.931	0.928	0.923	0.918	0.915	0.906	0.892	0.875

Table 14: Average $DIV - NDCG_{10}$ over the training queries for the different values of λ.

the text-based novelty can be computed as follows:

$$nov_i = \frac{|keywords_i \setminus (keywords_i \cap (\cup_{j=1}^{i-1} keywords_j))|}{|keywords_i|} \quad (25)$$

where $keywords_i$ is the set of the keywords associated with the subgraph at position i, $\cup_{j=1}^{i-1} keywords_j$ is the set of all the keywords seen so far (up to subgraph $i-1$), and $|keywords_i|$ is the number of keywords in the set $keywords_i$.

7.4.2. Parameter Tuning

We use our benchmark in order to evaluate our greedy diversity-aware re-ranking algorithm described in Section 5. First, we use the query benchmark to tune the weighting parameter of MMR (λ in Equation 12), which is used to trade-off relevance and diversity. Recall that in order to compute the $DIV - DCG$ of a given query, we need two measures for each result at position i, namely, the relevance of the result rel_i and its novelty nov_i. While nov_i can be directly computed based on the results themselves as explained in Section 7.4.1, we needed to gather a relevance assessment for each result to be able to compute rel_i. To gather these relevance assessments, we again relied on CrowdFlower as follows. Each query was run several times using our diversity-aware re-rank algorithm with different notions of diversity and with different values of the weighting parameter λ in Equation 12. In particular, for each one of our three diversity notions, each query was run 10 times with λ ranging from 0.1 to 1 (i.e., no diversity) and the top-10 results were retrieved. This resulted in 30 sets of possibly overlapping result sets. Finally, the results in each of these 30 sets were pooled together and the unique set of results were assessed on CrowdFlower.

For each query, we presented each one of its result to a worker along with the information need underlying the query. We then asked the worker to assess the relevance of the result with respect to the information need. Recall that we had four categories in our query benchmark. For the first category, the triple-pattern queries with many results, we asked the workers on CrowdFlower to assess the results on a 2-level scale: 1) a result consisting of authoritative resources and 2) a result consisting of non-authoritative resources. The number of unique results we had for this category was *1968*. For the second category, which is the keyword-extended queries with many results, we defined a 4-level scale: 1) an authoritative result and answers the subquery (i.e., the keywords part), 2) a non-authoritative result but answers the subquery, 3) an authoritative result but does not answer the subquery, and 4) a non-authoritative result and does not answer the subquery. For this category, we had *973* unique results. For the third category, the triple-pattern queries with zero results, we defined a 4-level scale: 1) an authoritative result and answers the query, 2) a non-authoritative result but answers the query, 3) a result that partially answers the query, and 4) a result that does not answer the query. For this category, we had *314* unique results. Finally, for the last category consisting of keyword-extended queries with zero results, we had a 5-level scale: 1) an authoritative result that answers both the query and the subquery, 2) a non-authoritative result that answers both the query and the subquery, 3) a result that answers either the query OR the subquery, 4) a result that partially answers the query, and 5) a result that does not answer the query. The number of unique results for this category was *163*.

The total number of results that we got by running all the queries for all the cases mentioned above was 28000. Each result was assessed by *five different* workers on CrowdFlower. To ensure a high quality of the relevance assessments gathered, we constructed a set of gold queries that were embedded within the benchmark queries. Moreover, before a worker was allowed to work on one of our tasks, she had to pass a qualification test that we created to guarantee that she understood the task guidelines. Any worker with a score less than *90%* was not allowed to work on our tasks. The average score of those workers that were allowed to work on our tasks was *93%*. The inter-rater agreement reported by CrowdFlower was *67%*. Moreover, we obtained a Kappa Coefficient of *40%*, which can be interpreted as "Fair Agreement" [62].

The main parameter in our diversity-aware re-ranking algorithm is the weighting parameter λ which trades-off relevance and diversity. To be able to set this parameter, we compute the *normalized* $DIV - DCG_{10}$ for each query varying the value of λ from 0.1 to 0.9. The normalized $DIV - DCG_{10}$ or $DIV - NDCG_{10}$ is computed by dividing the $DIV - DCG_{10}$ by the *ideal* $DIV - DCG_{10}$. To be able to compute the ideal $DIV - DCG_{10}$, we re-ranked the results using a greedy approach. The new ordering pushes the results with the best combination of diversity and relevance gain to the top. Table 14 shows the average $DIV - NDCG_{10}$ for our three notions of diversity for different values of λ. The numbers reported in Table 14 is the average over 5 runs, where the experiment was run each time with 80 randomly selected queries out of the 100 in our benchmark and the average $DIV - NDCG_{10}$ over the selected 80 queries was computed and then averaged over

Notion	Diversified	Non-diversified
Knowledge-graph-based	0.80	0.74
Query-based	0.81	0.74
Text-based	0.91	0.68

Table 15: Average $DIV - NDCG_{10}$ for the test queries using the optimum value of λ.

Non-diversified vs.:	Knowledge-graph-based	Query-based	Text-based
Won	50	54	38
Tied	44	41	56
Lost	6	5	6

Table 16: Comparison of different notions of diversity versus no diversity.

the 5 runs [65]. The values highlighted in bold are the highest values, and their corresponding λ values are the optimum values of the parameter. For our three notions, the best value for λ is 0.1, which means we should give 90% importance to diversity over relevance in our re-ranking algorithm in order to get the best average $DIV - NDCG_{10}$ possible.

Note that our framework consisted of some other parameters, namely the smoothing parameters for the language models used to compute the distance between results, however we opted to pre-set those to 0.8 to focus mainly on the effect of the weighting parameter λ in our experiments.

7.4.3. Comparison of the Various Notions of Diversity

Given the optimal values of the parameter λ that were set based on the 5 sets of training queries (80 queries in each run), we computed the average $DIV - NDCG_{10}$ for the five sets of test queries (20 queries in each run) for each notion of diversity, as well as for the cases when no diversity was employed. The summary of our findings over the test queries is shown in Table 15. As can be seen from the table, the average $DIV - NDCG_{10}$ for all notions is significantly larger than those where no diversification of results took place.

We also evaluated our diversity algorithm using a comparative study on CrowdFlower. We followed the same approach in setting up the guidelines as we did when evaluating result ranking (Section 7.3). For each query, each worker was asked to choose between two result sets, one diversified using one of our three notions of diversity, and the other non-diversified (i.e., results are just ranked using our ranking model). The worker was not aware which set was the diversified one and which was not. Each comparison was assessed by *three different* workers. In addition, the worker was also asked to provide an explanation for her choice. Finally, each worker was also given the option to not pick any of the result sets over the other if she deemed them to be similar in quality. To ensure that workers performed the evaluation tasks carefully, all the workers had to pass a qualification test with a score of at least 90% to be able to perform the actual evaluation tasks. Moreover, a set of gold queries were embedded in the benchmark to ensure that the workers were not randomly choosing between the sets. Similar to the case of result ranking, these qualification and gold queries corresponded to very obvious information needs and we showed the workers two result sets, one obviously answering the query and the other completely irrelevant (say answers to a different query). The inter-rater agreement as reported by CrowdFlower was *77%*, *71%*, and *71%*, for comparing the non-diversified result sets with the knowledge-graph-based diversified set, the query-based diversified set, and text-based diversified set, respectively. We also obtained an average Kappa Coefficient of *33%*, which can interpreted as "Fair Agreement" [62]. The results of this user study can be seen in Table 16. The reported numbers are out of 100 queries. The rest of the queries in the benchmark were used as qualification and gold queries, as in all other experiments.

As can be seen from the table, the result sets consisting of results that were re-ranked using our MMR algorithm using both the knowledge-graph-based language models and the query-based language models were chosen over the non-diversified result sets in 50% of the time and 54% of the time, respectively. On the other hand, the result sets consisting of results that were ranked using the text-based language models were only chosen 38% of the time. They tied with the non-diversified result sets for the majority of the queries (56%). This can be attributed to the fact that in the text-based diversity notion, the results are diversified with respect to the keywords associated with the result subgraphs. These might not be very obvious to users at first glance. To overcome this, we also provided the set of keywords associated with each result subgraph when presenting the subgraphs to the workers. We did not however explicitly ask them to consider them when assessing the results sets in order not to bias their choice. Overall, we can conclude that users in general would hardly prefer non-diversified results over diversified ones, as can be seen in the last row of Table 16.

8. Efficiency

We report some statistics about the data used in our case study. Moreover, we discuss the running time of our

algorithms. As mentioned before, our case study was done using a relational database managed by PostgreSQL [59]. Ranking a set of results, given a query, requires visits to several indexed tables. Some of these tables were prepared as materialized views, so that the system does not spend a lot of time looking for all triples that contains `Woody_Allen` as subject, for example. We repeat that for all the possible combinations. The information about some of these tables, their descriptions, number of rows, and examples, are shown in Tables 17 and 18.

Given the fact that query processing to find isomorphic subgraphs is carried out by means of inner joins between huge tables, the running time of the queries is not yet very satisfactory, even with our extensive indexing. The average *ranking* running time per query is 47s. The standard deviation is 0.2s. The minimum running time per query is 0.093s, and the maximum running time per query is 431s. Moreover, the average time taken to re-rank the query results based on the MMR approach to ensure diversity are as follows. For the knowledge-graph-based and for the query-based diversity, the average is 33s. However, for the text-based diversity, it is 146s. Finally, we record the average running time for *producing a set of relaxed queries*, the average running time for *running a set of relaxed queries*, and the average running time for *scoring the sets of results produced by relaxed queries*, which are 0.1s, 54s, and 0.2s respectively. In future work, we plan to develop top-k query processing techniques that can retrieve the top-k highest scored result subgraphs for a given query. To this end, we will need to implement a top-k rank-join approach [66]. We also plan to investigate using different data management systems such as graph databases or RDF stores for instances.

9. Conclusion & Future Work

In this paper, we proposed a general framework for effective searching of RDF knowledge graphs. Our framework allow users to search the underlying knowledge graph using both triple-pattern queries, as well as triple-pattern queries extended with keywords. Our framework supports result ranking by means of a novel ranking model based on statistical machine translation. In addition, our framework supports automatic query relaxation for queries with no results. Finally, our framework provides various notions of result diversity specifically defined for the setting of RDF and that are achieved using the Maximal Marginal Relevance re-ranking approach. We conducted a case study on DBpedia [1] and evaluated the effectiveness of our framework using a series of carefully-designed user studies on CrowdFlower.

In future work, we plan to evaluate our framework using other large knowledge graphs. We also plan to extend our framework to support searching multiple linked knowledge graphs at the same time. The extended framework will be made publicly available as a service to effectively search linked open data sources such as DBpedia [1], YAGO [3], and perhaps a number of other Wikipedia-based knowledge graphs. This can be accomplished after enhancing the efficiency of our retrieval system using a top-k rank-join approach for instance. Finally, we plan to experiment with various data stores to improve the efficiency of our proposed algorithms, particularly for result ranking and diversity such as graph databases and RDF-specific stores.

Acknowledgments

We would like to thank the American University of Beirut's research board (URB) for funding this project.

References

[1] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, Z. Ives, DBpedia: A Nucleus for a Web of Open Data, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pp. 722–735. doi:10.1007/978-3-540-76298-0_52.
URL http://dx.doi.org/10.1007/978-3-540-76298-0_52

[2] J. Carbonell, J. Goldstein, The use of mmr, diversity-based reranking for reordering documents and producing summaries, in: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '98, ACM, New York, NY, USA, 1998, pp. 335–336. doi:10.1145/290941.291025.
URL http://doi.acm.org/10.1145/290941.291025

[3] F. M. Suchanek, G. Kasneci, G. Weikum, Yago: a core of semantic knowledge, in: Proceedings of the 16th international conference on World Wide Web, ACM, 2007, pp. 697–706.

[4] C. Zhai, Statistical Language Models for Information Retrieval, Synthesis Lectures on Human Language Technologies, Morgan & Claypool Publishers, 2008.
URL http://dx.doi.org/10.2200/S00158ED1V01Y200811HLT001

[5] S. Gollapudi, A. Sharma, An axiomatic approach for result diversification, in: Proceedings of the 18th International Conference on World Wide Web, WWW '09, ACM, New York, NY, USA, 2009, pp. 381–390. doi:10.1145/1526709.1526761.
URL http://doi.acm.org/10.1145/1526709.1526761

[6] J. Lin, Divergence measures based on the shannon entropy, IEEE Trans. Inf. Theor. 37 (1) (2006) 145–151. doi:10.1109/18.61115.
URL http://dx.doi.org/10.1109/18.61115

[7] H. Bast, B. Buchhold, E. Haussmann, Semantic search on text and knowledge bases, Foundations and Trends in Information Retrieval 10 (2-3) (2016) 119–271. doi:10.1561/1500000032.
URL http://dx.doi.org/10.1561/1500000032

[8] C. Unger, L. Bühmann, J. Lehmann, A.-C. Ngonga Ngomo, D. Gerber, P. Cimiano, Template-based question answering over rdf data, in: Proceedings of the 21st International Conference on World Wide Web, WWW '12, ACM, New York, NY, USA, 2012, pp. 639–648. doi:10.1145/2187836.2187923.
URL http://doi.acm.org/10.1145/2187836.2187923

[9] M. Yahya, K. Berberich, S. Elbassuoni, M. Ramanath, V. Tresp, G. Weikum, Natural language questions for the web of data, in: Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL '12, Association for Computational Linguistics, Stroudsburg, PA, USA, 2012, pp. 379–390.
URL http://dl.acm.org/citation.cfm?id=2390948.2390995

[10] M. Yahya, K. Berberich, S. Elbassuoni, G. Weikum, Robust question answering over the web of linked data, in: Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management, CIKM '13, ACM, New York, NY,

Table	Description	Number of Rows
Facts	DBpedia triples	15.8 million
Keywords	Weighted keywords of a triple	15.8 million
op_var	For every subject, an aggregation of weights and weighted keywords	2 million
sp_var	For every object, an aggregation of weights and weighted keywords	4.6 million
so_var	For every predicate, an aggregation of weights and weighted keywords	1337

Table 17: Datasets Statistics

Table	Sample Row
Facts	<*Triple*:Annie_Hall,starring,Woody_Allen, w:2032>
Keywords	<*Triple_ID*:1141107, $w\_keywords$:standup=1, playwright=119, role=164, humor=39>
op_var	<*subject*:A._C._Trumbo_House, $\sum w$:476, $\sum w\_keywords$:histor=76, percent=3, place=124>
sp_var	<*object*:King_George_VI_Chase, $\sum w$:1364, $\sum w\_keywords$:mellor=15, 2006=20, bright=3>
so_var	<*predicate*:starring,$\sum w$:26913826, $\sum w\_keywords$:tokiwa=111, agimat=225>

Table 18: Data Samples

USA, 2013, pp. 1107–1116. doi:10.1145/2505515.2505677.
URL http://doi.acm.org/10.1145/2505515.2505677

[11] K. Xu, S. Zhang, Y. Feng, D. Zhao, Answering Natural Language Questions via Phrasal Semantic Parsing, Springer Berlin Heidelberg, Berlin, Heidelberg, 2014, pp. 333–344. doi:10.1007/978-3-662-45924-9_30.
URL http://dx.doi.org/10.1007/978-3-662-45924-9_30

[12] Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL, ACL, 2013.

[13] J. Berant, P. Liang, Semantic parsing via paraphrasing, in: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers, 2014, pp. 1415–1425.
URL http://aclweb.org/anthology/P/P14/P14-1133.pdf

[14] H. Bast, B. Buchhold, E. Haussmann, Relevance scores for triples from type-like relations, in: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '15, ACM, New York, NY, USA, 2015, pp. 243–252. doi:10.1145/2766462.2767734.
URL http://doi.acm.org/10.1145/2766462.2767734

[15] W. Yih, M. Chang, X. He, J. Gao, Semantic parsing via staged query graph generation: Question answering with knowledge base, in: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers, 2015, pp. 1321–1331.
URL http://aclweb.org/anthology/P/P15/P15-1128.pdf

[16] V. Lopez, M. Pasin, E. Motta, AquaLog: An Ontology-Portable Question Answering System for the Semantic Web, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, pp. 546–562. doi:10.1007/11431053_37.
URL http://dx.doi.org/10.1007/11431053_37

[17] V. Lopez, V. Uren, M. R. Sabou, E. Motta, Cross ontology query answering on the semantic web: An initial evaluation, in: Proceedings of the Fifth International Conference on Knowledge Capture, K-CAP '09, ACM, New York, NY, USA, 2009, pp. 17–24. doi:10.1145/1597735.1597739.
URL http://doi.acm.org/10.1145/1597735.1597739

[18] Y. Lei, V. Uren, E. Motta, Semsearch: A search engine for the semantic web, in: Proceedings of the 15th International Conference on Managing Knowledge in a World of Networks, EKAW'06, Springer-Verlag, Berlin, Heidelberg, 2006, pp. 238–245. doi:10.1007/11891451_22.
URL http://dx.doi.org/10.1007/11891451_22

[19] T. Tran, P. Cimiano, S. Rudolph, R. Studer, Ontology-based interpretation of keywords for semantic search, in: Proceedings of the 6th International The Semantic Web and 2Nd Asian Conference on Asian Semantic Web Conference, ISWC'07/ASWC'07, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 523–536.
URL http://dl.acm.org/citation.cfm?id=1785162.1785201

[20] Q. Zhou, C. Wang, M. Xiong, H. Wang, Y. Yu, Spark: Adapting keyword query to semantic search, in: Proceedings of the 6th International The Semantic Web and 2Nd Asian Conference on Asian Semantic Web Conference, ISWC'07/ASWC'07, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 694–707.
URL http://dl.acm.org/citation.cfm?id=1785162.1785213

[21] G. Zenz, X. Zhou, E. Minack, W. Siberski, W. Nejdl, From keywords to semantic queries-incremental query construction on the semantic web, Web Semantic 7 (3) (2009) 166–176. doi:10.1016/j.websem.2009.07.005.

[22] T. Tran, H. Wang, P. Haase, Hermes: Data web search on a pay-as-you-go integration infrastructure, Web Semant. 7 (3) (2009) 189–203. doi:10.1016/j.websem.2009.07.001.
URL http://dx.doi.org/10.1016/j.websem.2009.07.001

[23] J. Pound, A. K. Hudek, I. F. Ilyas, G. Weddell, Interpreting keyword queries over web knowledge bases, in: Proceedings of the 21st ACM International Conference on Information and Knowledge Management, CIKM '12, ACM, New York, NY, USA, 2012, pp. 305–314. doi:10.1145/2396761.2396803.
URL http://doi.acm.org/10.1145/2396761.2396803

[24] S. Elbassuoni, R. Blanco, Keyword search over rdf graphs, in: Proceedings of the 20th ACM International Conference on Information and Knowledge Management, CIKM '11, ACM, New York, NY, USA, 2011, pp. 237–242. doi:10.1145/2063576.2063615.
URL http://doi.acm.org/10.1145/2063576.2063615

[25] S. Elbassuoni, M. Ramanath, R. Schenkel, M. Sydow, G. Weikum, Language-model-based ranking for queries on rdf-graphs, in: Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM '09, ACM, New York, NY, USA, 2009, pp. 977–986. doi:10.1145/1645953.1646078.
URL http://doi.acm.org/10.1145/1645953.1646078

[26] S. Elbassuoni, M. Ramanath, G. Weikum, Query relaxation for entity-relationship search, in: Proceedings of the 8th Extended Semantic Web Conference on The Semanic Web: Research and Applications - Volume Part II, ESWC'11, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 62–76.
URL http://dl.acm.org/citation.cfm?id=2017936.2017942

[27] H. Arnaout, S. Elbassuoni, Result diversity for RDF search, in: Proceedings of the 8th International Joint Conference on

Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2016) - Volume 1: KDIR, Porto - Portugal, November 9 - 11, 2016., 2016, pp. 249–256. doi:10.5220/0006046402490256.
URL http://dx.doi.org/10.5220/0006046402490256

[28] L. Ding, T. Finin, A. Joshi, R. Pan, R. S. Cost, Y. Peng, P. Reddivari, V. Doshi, J. Sachs, Swoogle: A search and metadata engine for the semantic web, in: Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management, CIKM '04, ACM, New York, NY, USA, 2004, pp. 652–659. doi:10.1145/1031171.1031289.
URL http://doi.acm.org/10.1145/1031171.1031289

[29] G. Cheng, W. Ge, Y. Qu, Falcons: Searching and browsing entities on the semantic web, in: Proceedings of the 17th International Conference on World Wide Web, WWW '08, ACM, New York, NY, USA, 2008, pp. 1101–1102. doi:10.1145/1367497.1367676.
URL http://doi.acm.org/10.1145/1367497.1367676

[30] E. Oren, R. Delbru, M. Catasta, R. Cyganiak, H. Stenzhorn, G. Tummarello, Sindice.com: a document-oriented lookup index for open linked data, Int. J. Metadata Semant. Ontologies 3 (1) (2008) 37–52. doi:10.1504/IJMSO.2008.021204.
URL http://dx.doi.org/10.1504/IJMSO.2008.021204

[31] R. Blanco, P. Mika, S. Vigna, Effective and efficient entity search in rdf data, in: Proceedings of the 10th International Conference on The Semantic Web - Volume Part I, ISWC'11, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 83–97.
URL http://dl.acm.org/citation.cfm?id=2063016.2063023

[32] D. A. Ferrucci, E. W. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. M. Prager, N. Schlaefer, C. A. Welty, Building watson: An overview of the deepqa project, AI Magazine 31 (3) (2010) 59–79.
URL http://www.aaai.org/ojs/index.php/aimagazine/article/view/2303

[33] M. Joshi, U. Sawant, S. Chakrabarti, Knowledge graph and corpus driven segmentation and answer inference for telegraphic entity-seeking queries, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL, 2014, pp. 1104–1114.
URL http://aclweb.org/anthology/D/D14/D14-1117.pdf

[34] W. Yih, X. He, C. Meek, Semantic parsing for single-relation question answering, in: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 2: Short Papers, 2014, pp. 643–648.
URL http://aclweb.org/anthology/P/P14/P14-2105.pdf

[35] A. Bordes, J. Weston, R. Collobert, Y. Bengio, Learning structured embeddings of knowledge bases, in: Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI'11, AAAI Press, 2011, pp. 301–306.
URL http://dl.acm.org/citation.cfm?id=2900423.2900470

[36] N. Zhiltsov, A. Kotov, F. Nikolaev, Fielded sequential dependence model for ad-hoc entity retrieval in the web of data, in: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '15, ACM, New York, NY, USA, 2015, pp. 253–262. doi:10.1145/2766462.2767756.
URL http://doi.acm.org/10.1145/2766462.2767756

[37] M. Schuhmacher, L. Dietz, S. Paolo Ponzetto, Ranking entities for web queries through text and knowledge, in: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, CIKM '15, ACM, New York, NY, USA, 2015, pp. 1461–1470. doi:10.1145/2806416.2806480.
URL http://doi.acm.org/10.1145/2806416.2806480

[38] R. Mirizzi, A. Ragone, T. Di Noia, E. Di Sciascio, Ranking the linked data: The case of dbpedia, in: Proceedings of the 10th International Conference on Web Engineering, ICWE'10, Springer-Verlag, Berlin, Heidelberg, 2010, pp. 337–354.
URL http://dl.acm.org/citation.cfm?id=1884110.1884138

[39] N. Zhiltsov, E. Agichtein, Improving entity search over linked data by modeling latent semantics, in: Proceedings of the 22nd ACM international conference on Conference on information & knowledge management, CIKM '13, ACM, New York, NY, USA, 2013, pp. 1253–1256. doi:10.1145/2505515.2507868.
URL http://doi.acm.org/10.1145/2505515.2507868

[40] S. Elbassuoni, R. Blanco, Keyword search over rdf graphs, in: Proceedings of the 20th ACM International Conference on Information and Knowledge Management, CIKM '11, ACM, New York, NY, USA, 2011, pp. 237–242. doi:10.1145/2063576.2063615.
URL http://doi.acm.org/10.1145/2063576.2063615

[41] M. Yahya, D. Barbosa, K. Berberich, Q. Wang, G. Weikum, Relationship queries on extended knowledge graphs, in: Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, WSDM '16, ACM, New York, NY, USA, 2016, pp. 605–614. doi:10.1145/2835776.2835795.
URL http://doi.acm.org/10.1145/2835776.2835795

[42] G. Kasneci, F. M. Suchanek, G. Ifrim, M. Ramanath, G. Weikum, Naga: Searching and ranking knowledge, in: Proceedings of the 2008 IEEE 24th International Conference on Data Engineering, ICDE '08, IEEE Computer Society, Washington, DC, USA, 2008, pp. 953–962. doi:10.1109/ICDE.2008.4497504.
URL http://dx.doi.org/10.1109/ICDE.2008.4497504

[43] H. Huang, C. Liu, X. Zhou, Computing relaxed answers on rdf databases, in: Proceedings of the 9th International Conference on Web Information Systems Engineering, WISE '08, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 163–175. doi:10.1007/978-3-540-85481-4_14.
URL http://dx.doi.org/10.1007/978-3-540-85481-4_14

[44] G. Fokou, S. Jean, A. Hadjali, M. Baron, RDF Query Relaxation Strategies Based on Failure Causes, Springer International Publishing, Cham, 2016, pp. 439–454. doi:10.1007/978-3-319-34129-3_27.
URL http://dx.doi.org/10.1007/978-3-319-34129-3_27

[45] G. Fokou, S. Jean, A. Hadjali, M. Baron, Cooperative techniques for sparql query relaxation in rdf databases, in: Proceedings of the 12th European Semantic Web Conference on The Semantic Web. Latest Advances and New Domains - Volume 9088, Springer-Verlag New York, Inc., New York, NY, USA, 2015, pp. 237–252. doi:10.1007/978-3-319-18818-8_15.
URL http://dx.doi.org/10.1007/978-3-319-18818-8_15

[46] E. Vasilyeva, T. Heinze, M. Thiele, W. Lehner, Debeaq - debugging empty-answer queries on large data graphs, in: 32nd IEEE International Conference on Data Engineering, ICDE 2016, Helsinki, Finland, May 16-20, 2016, 2016, pp. 1402–1405. doi:10.1109/ICDE.2016.7498355.
URL https://doi.org/10.1109/ICDE.2016.7498355

[47] C. A. Hurtado, A. Poulovassilis, P. T. Wood, A Relaxed Approach to RDF Querying, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 314–328. doi:10.1007/11926078_23.
URL https://doi.org/10.1007/11926078_23

[48] A. Poulovassilis, P. T. Wood, Combining Approximation and Relaxation in Semantic Web Path Queries, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 631–646. doi:10.1007/978-3-642-17746-0_40.
URL https://doi.org/10.1007/978-3-642-17746-0_40

[49] R. De Virgilio, A. Maccioni, R. Torlone, A similarity measure for approximate querying over rdf data, in: Proceedings of the Joint EDBT/ICDT 2013 Workshops, EDBT '13, ACM, New York, NY, USA, 2013, pp. 205–213. doi:10.1145/2457317.2457352.
URL http://doi.acm.org/10.1145/2457317.2457352

[50] R. De Virgilio, A. Maccioni, R. Torlone, Approximate querying of rdf graphs via path alignment, Distributed and Parallel Databases 33 (4) (2015) 555–581. doi:10.1007/s10619-014-7142-1.
URL https://doi.org/10.1007/s10619-014-7142-1

[51] R. Agrawal, S. Gollapudi, A. Halverson, S. Ieong, Diversifying search results, in: Proceedings of the Second ACM International

Conference on Web Search and Data Mining, WSDM '09, ACM, New York, NY, USA, 2009, pp. 5–14. doi:http://doi.acm.org/10.1145/1498759.1498766.
URL http://doi.acm.org/10.1145/1498759.1498766

[52] H. Chen, D. R. Karger, Less is more: probabilistic models for retrieving fewer relevant documents, in: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '06, ACM, New York, NY, USA, 2006, pp. 429–436. doi:http://doi.acm.org/10.1145/1148170.1148245.
URL http://doi.acm.org/10.1145/1148170.1148245

[53] C. L. Clarke, M. Kolla, G. V. Cormack, O. Vechtomova, A. Ashkan, S. Büttcher, I. MacKinnon, Novelty and diversity in information retrieval evaluation, SIGIR '08, ACM, New York, NY, USA, 2008, pp. 659–666. doi:http://doi.acm.org/10.1145/1390334.1390446.
URL http://doi.acm.org/10.1145/1390334.1390446

[54] C. X. Zhai, W. W. Cohen, J. Lafferty, Beyond independent relevance: methods and evaluation metrics for subtopic retrieval, in: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval, SIGIR '03, ACM, New York, NY, USA, 2003, pp. 10–17. doi:http://doi.acm.org/10.1145/860435.860440.
URL http://doi.acm.org/10.1145/860435.860440

[55] Z. Chen, T. Li, Addressing diverse user preferences in SQL-query-result navigation, in: Proceedings of the 2007 ACM SIGMOD international conference on Management of data, SIGMOD '07, ACM, New York, NY, USA, 2007, pp. 641–652. doi:http://doi.acm.org/10.1145/1247480.1247551.
URL http://doi.acm.org/10.1145/1247480.1247551

[56] J. Dalton, R. Blanco, P. Mika, Coreference aware web object retrieval, in: Proceedings of the 20th ACM international conference on Information and knowledge management, CIKM '11, ACM, New York, NY, USA, 2011, pp. 211–220. doi:http://doi.acm.org/10.1145/2063576.2063612.
URL http://doi.acm.org/10.1145/2063576.2063612

[57] E. Vee, U. Srivastava, J. Shanmugasundaram, P. Bhat, S. A. Yahia, Efficient Computation of Diverse Query Results, in: Proceedings of the 2008 IEEE 24th International Conference on Data Engineering, IEEE Computer Society, Washington, DC, USA, 2008, pp. 228–236. doi:10.1109/ICDE.2008.4497431.
URL http://dl.acm.org/citation.cfm?id=1546682.1547168

[58] E. Demidova, P. Fankhauser, X. Zhou, W. Nejdl, DivQ: diversification for keyword search over structured databases, in: Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval, SIGIR '10, ACM, New York, NY, USA, 2010, pp. 331–338. doi:http://doi.acm.org/10.1145/1835449.1835506.
URL http://doi.acm.org/10.1145/1835449.1835506

[59] A powerful, open source object-relational database system., https://www.postgresql.org/.

[60] Crowd flower: A crowd sourcing company., https://www.crowdflower.com/.

[61] K. Balog, R. Neumayer, A test collection for entity search in dbpedia, in: Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '13, ACM, New York, NY, USA, 2013, pp. 737–740. doi:10.1145/2484028.2484165.
URL http://doi.acm.org/10.1145/2484028.2484165

[62] J. Fleiss, et al., Measuring nominal scale agreement among many raters, Psychological Bulletin 76 (5) (1971) 378–382.

[63] J. Allan, C. Wade, A. Bolivar, Retrieval and novelty detection at the sentence level, in: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval, SIGIR '03, ACM, New York, NY, USA, 2003, pp. 314–321. doi:10.1145/860435.860493.
URL http://doi.acm.org/10.1145/860435.860493

[64] K. Järvelin, J. Kekäläinen, Cumulated gain-based evaluation of ir techniques, ACM Trans. Inf. Syst. 20 (4) (2002) 422–446. doi:10.1145/582415.582418.
URL http://doi.acm.org/10.1145/582415.582418

[65] R. Kohavi, A study of cross-validation and bootstrap for accuracy estimation and model selection, in: Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2, IJCAI'95, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1995, pp. 1137–1143.
URL http://dl.acm.org/citation.cfm?id=1643031.1643047

[66] I. F. Ilyas, G. Beskales, M. A. Soliman, A survey of top-k query processing techniques in relational database systems, ACM Computing Surveys (CSUR) 40 (4) (2008) 11.