

AMERICAN UNIVERSITY OF BEIRUT

DBpediaSearch: An Effective Search Engine for
DBpedia

by

HIBA KHALED ARNAOUT

A thesis

submitted in partial fulfillment of the requirements
for the degree of Master of Science
to the Department of Computer Science
of the Faculty of Arts and Sciences
at the American University of Beirut

Beirut, Lebanon
February 2017

AMERICAN UNIVERSITY OF BEIRUT

DBpediaSearch: An Effective Search Engine for DBpedia

by
HIBA KHALED ARNAOUT

Approved by:

Dr. Shady Elbassuoni, Assistant Professor

Advisor

Computer Science



Dr. Wassim El Hajj, Chairperson & Associate Professor

Member of Committee

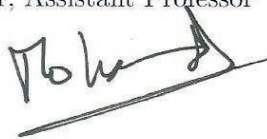
Computer Science



Dr. Mohamad I Jaber, Assistant Professor

Member of Committee

Computer Science



Date of thesis defense: January 20, 2017

AMERICAN UNIVERSITY OF BEIRUT

THESIS, DISSERTATION, PROJECT RELEASE FORM

Student Name: _____
Last First Middle

Master's Thesis Master's Project Doctoral Dissertation

I authorize the American University of Beirut to: (a) reproduce hard or electronic copies of my thesis, dissertation, or project; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.

I authorize the American University of Beirut, to: (a) reproduce hard or electronic copies of it; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes after: **One ___ year from the date of submission of my thesis, dissertation or project.**
Two ___ years from the date of submission of my thesis , dissertation or project.
Three ___ years from the date of submission of my thesis , dissertation or project.

Signature

Date

This form is signed when submitting the thesis, dissertation, or project to the University Libraries

Acknowledgements

Firstly, I would like to thank my thesis advisor Prof. Shady Elbassuoni for supporting me throughout the course of this experience. It all started with an Information Retrieval course, during which I started my research journey with Dr.Elbassuoni. His patience, motivation, and positivity made everything looks easier. I learned a lot from his vast experience in two domains, namely, Information Retrieval and Machine Learning. I would like to thank him for teaching me how to write a proper research paper.

I would like to thank my committee members Prof. Mohamad Jaber and Prof. Wassim ElHajj for their comments and suggestions. Additionally, I am very thankful for Dr.ElHajj's constant support since day one.

I am grateful for the American University of Beirut's research board (URB) for funding my research.

Finally, a special thanks to my family. Words cannot express how grateful I am to my mother who encouraged me. My appreciation to my father for his moral and financial support, and for his ineterest in having a look at the results of my thesis.

An Abstract of the Thesis of

Hiba Arnaout for Master of Science
Major: Computer Science

Title: DBpediaSearch: An Effective Search Engine for DBpedia

The active progress of knowledge-sharing communities like Wikipedia have made it achievable to build large knowledge-bases, such as DBpedia. These knowledge-bases use the Resource Description Framework (RDF) as a flexible data model for representing the information in the Web. The semantic query language for RDF is known as SPARQL. Although querying with SPARQL gives very specific results, the users knowledge of the underlying data is a must. In this thesis, we propose an effective search engine for DBpedia. The search system takes SPARQL queries augmented with keywords as input and gives the most relevant results as output. To be able to do this, we develop a novel ranking model based on statistical machine translation for both triple-pattern queries and keyword-augmented triple-pattern queries. Our system supports automatic query relaxation in case no results were found. Our ranking model also takes into consideration result diversity in order to ensure that the user is provided with a wide range of aspects of the query results. We develop a diversity-aware evaluation metric based on the Discounted Cumulative Gain to evaluate diversified result sets. Finally, we build an evaluation benchmark on DBpedia, which we use to evaluate the effectiveness of our search engine.

Contents

Acknowledgements	v
Abstract	vi
1 Introduction	1
1.1 Motivation	1
1.2 Challenges	3
1.3 Contributions	3
2 Indexing and Query Processing	6
2.1 Knowledge Graph: DBpedia	6
2.2 Triple Patterns	6
2.3 Keyword-augmented Triple Patterns	7
2.4 Query Relaxation	8
3 Ranking	11
3.1 Ranking Model	11
3.1.1 Result Ranking	11
3.1.2 Triple Weight	14
3.1.3 Keyword Augmentation	15
3.2 Relaxation Model	17
3.3 Related Work	18
3.3.1 Unstructured Queries on Unstructured Data	18
3.3.2 Unstructured Queries on Structured Data	19
3.3.3 Structured Queries on Structured Data	20
3.3.4 Query Relaxation	20
3.4 Experimental Evaluation	22
3.4.1 Setup	22
3.4.2 Results	22
4 Diversity	26
4.1 Diversity Problem	26
4.2 Maximal Marginal Relevance	28

4.3	Diversity Notions in RDF Setting	28
4.3.1	Resource-based Diversity	29
4.3.2	Term-based Diversity	29
4.3.3	Text-based Diversity	30
4.4	Diversity-aware Re-ranking Algorithm	31
4.5	Diversity-aware Evaluation Metric	32
4.5.1	Resource-based Novelty	32
4.5.2	Text-based Novelty	32
4.6	Related Work	33
4.7	Experimental Evaluation	34
4.7.1	Evaluation Using Diversity-aware Metric	34
4.7.2	Evaluation Using Crowdsourcing	37
5	Efficiency	42
5.1	Database Statistics	42
5.2	Running Time Statistics	43
5.3	Running Time Challenges	44
A	Guidelines: Choosing the Better Result Set	45
B	Guidelines: Assessing a Subgraph	48
B.1	Purely Structured Queries Guidelines	48
B.2	Keyword-augmented Queries Guidelines	48
B.3	Requiring Relaxation Queries Guidelines	48
B.4	Keyword-augmented and Requiring Relaxation Queries Guidelines	49
B.5	Gold Queries	49
C	Abbreviations	53

List of Figures

1.1	An example RDF graph on movies	2
2.1	Facts table in PostgreSQL	7
2.2	Sample answer of a SPARQL query in PostgreSQL	7
2.3	Witnesscounts associated with DBpedia triples	8
2.4	Triples with keywords and keywords weights	8
3.1	A subgraph of the "WikiLinks" graph	15
4.1	A subgraph and its description	38
5.1	Percentage of time needed per step	44
A.1	Which set is better?	46
A.2	One query and two sets of answers	47
B.1	Purely structured queries guidelines	49
B.2	Sample assessment question (purely structured)	49
B.3	Keyword-augmented queries guidelines	50
B.4	Sample assessment question (keyword-augmented)	51
B.5	Requiring relaxation queries guidelines	51
B.6	Sample assessment question (requiring relaxation)	51
B.7	Keyword-augmented and requiring relaxation queries guidelines	52
B.8	Sample assessment question (keyword-augmented and requiring relaxation)	52

List of Tables

1.1	The set of RDF triples for the RDF graph in Figure 1.1	2
1.2	Un-ranked result set for: director/star of the same show.	4
1.3	Non-diversified result set for: director couples.	5
2.1	List of relaxed queries for movies starred by Allen and directed by Madonna	9
2.2	List of possible subgraphs for queries in Table 2.1	9
3.1	Top-10 un-ranked results: producer/director of the same show. . .	12
3.2	Top-10 ranked results: producer/director of the same show. . . .	13
3.3	Top-10 ranked results: producer/director of the same show, with the keyword: "Chaplin".	14
3.4	Top-10 ranked results: people who graduated from Oxford and were born in Agatha Christie's death place.	18
3.5	Top-10 ranked results of: people who worked with Frank Sinatra. .	19
3.6	Results for 100 evaluation queries: ranking algorithm	23
3.7	Ours vs. No-rank explanation samples.	23
3.8	Ours vs. Indegrees explanations samples	24
3.9	Ours vs. LMRQ explanations samples	25
4.1	Top-10 ranked results of: Democratic politicians who had roles in military events or crises.	27
4.2	Average $DIV - NDCG_{10}$ of the training queries for different values of λ	35
4.3	Average $DIV - NDCG_{10}$ of the test queries	35
4.4	Top-5 results: director-actor couple query with and without diversity	36
4.5	Top-5 results for the Disney query with and without diversity . .	37
4.6	Top-5 results for the Columbia Pictures query with and without diversity	37
4.7	Results for 100 evaluation queries: diversity algorithm	38
4.8	No Diversity vs. Resource-based Diversity	39
4.9	No Diversity vs. Term-based Diversity	40
4.10	No Diversity vs. Text-based Diversity	41

5.1	Database information	43
5.2	Running time per query category	43
5.3	Running time per diversity notion	43

Chapter 1

Introduction

1.1 Motivation

The continuous growth of knowledge-sharing communities like Wikipedia and the active progress in modern information extraction technologies have made it achievable to build large knowledge-bases (KBs) such as DBpedia [1], Freebase [2], and YAGO [3]. In addition, some domain-specific Knowledge-bases such as DBLife [4], and Libra [5] have been also constructed. These knowledge bases use the Resource Description Framework (RDF) [6] as a flexible data model for representing the information extracted and for publishing such information of the web.

The RDF data model is based on representing binary facts as triples of form: **Subject-Predicate-Object**, where **Subjects** are URIs for resources, **Predicates** are URIs for properties, and **Objects** are URIs for resources or literals. A certain fact can be represented in a table store as a triple, or in a graph as two nodes with one edge relating them. As an example, the following statement: “The sky has the color blue” can be converted into a triple form: `<Subject:Sky, Predicate:hasColor, Object:Blue>`. It also can be transformed into two nodes: `Sky` and `Blue` and one edge labeled: `hasColor`.

Unlike the traditional web search, querying this type of structured data permits apprehending the underlying semantics of the data. Figure 1.1 shows a larger example of a partial RDF graph of a movie knowledge base. Table 1.1 shows the corresponding RDF triples.

The semantic query language for RDF is known as SPARQL [7]. It allows to compose structured queries consisting of triple patterns and conjunctions. In general, a SPARQL query Q consists of n -triple patterns, where a triple pattern is an *SPO triple* with variables. A variable occurring in one of the triple patterns can be seen again in another triple in the same query Q , denoting a join condition. For example, if the user’s information need is to extract *shows that were created and starred by the same person*, the following

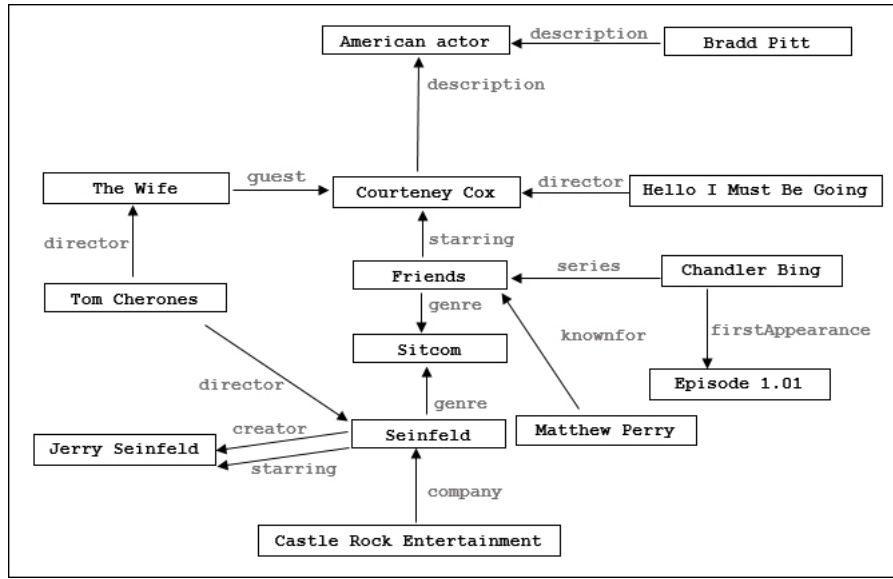


Figure 1.1: An example RDF graph on movies

Subject (S)	Predicate (P)	Object (O)
Brad_Pitt	description	American_actor
Courteney_Cox	description	American_actor
Hello_I_Must_Be_Going	director	Courteney_Cox
Chandler_Bing	series	Friends
Chandler_Bing	firstAppearance	Episode1.01
Matthew_Perry	knownfor	Friends
Friends	starring	Courteney_Cox
The_Wife	guest	Courteney_Cox
The_Wife	director	Tom_Cheronos
Seinfeld	director	Tom_Cheronos
Seinfeld	genre	Sitcom
Friends	genre	Sitcom
Seinfeld	creator	Jerry_Seinfeld
Seinfeld	starring	Jerry_Seinfeld
Seinfeld	company	Castle_Rock_Entertainment

Table 1.1: The set of RDF triples for the RDF graph in Figure 1.1

triple-pattern query can be used: $\langle ?s \text{ creator } ?x; ?s \text{ starring } ?x \rangle$. Given this 2-triple-pattern query, the results are all the subgraphs with 2 triples that are isomorphic to the query Q when compared to the triples in the knowledge-base. For the above example, the predicates in a subgraph of size two (2 triples), should be **creator** and **starring**, and both the subjects and objects in both triples should be identical. One relevant result for the query is: $\langle \text{Seinfeld creator Jerry_Seinfeld; Seinfeld starring Jerry_Seinfeld} \rangle$. Another result

is: `<The_Cosby_Show creator Bill_Cosby; The_Cosby_Show starring Bill_Cosby>`.

1.2 Challenges

Although querying with SPARQL gives very specific results, we face a number of problems that we need to address in order to build our effective search engine:

- **Result ranking:** Running a SPARQL query over a knowledge base, such as DBpedia, would return a set of unranked results. Users usually prefer seeing a ranked result-list rather than a list of un-ranked matches [8]. An example of a subset of an un-ranked set of results for the following query: `<?m starring ?x; ?m director ?x>` which requires movies directed and starred by the same person, is shown in Table 1.2. This example query produces 1793 subgraphs, making it difficult to choose the most relevant ones.
- **In-expressiveness of triple-pattern queries:** In some cases, the user's query need cannot be completely written in terms of triple patterns. As an example: Movies starred and directed by the same person, and related to New York and love? With the absence of triples which describe the plot of the movie, the second part of the question can only be expressed as a set of keywords: `<?m starring ?x[new york love]; ?m director ?x>`.
- **Exact matching of triple-pattern queries:** The users knowledge of the underlying data is a must. That is, the user must use the exact URIs for resources and predicates to construct a query. Triple-pattern queries are very rigid and a query might have few or no results in many occasions. The following query: `<?m producer Woody_Allen; ?m starring Scarlett_Johansson>` would return zero results, since Scarlett Johansson never acted in a movie that was produced by Woody Allen.
- **Results diversity:** It is often the case that the top-ranked results are dominated by one aspect of the query. For example, the query `<?m director ?x; ?m director ?y; ?x spouse ?y>` which asks for director couples. The ranked but non-diversified top-10 set of results is shown in Table 1.3. The set is dominated by Madonna and the two men she was married to. The user would prefer to see a variety of names instead of a homogeneous set of results.

1.3 Contributions

In this thesis, we develop a search engine that allows users to retrieve information from DBpedia.

Position	Subject (S)	Predicate (P)	Object (O)
1	45_(film) 45_(film)	starring director	Peter_Coster Peter_Coster
2	Mothers_&_Daughters Mothers_&_Daughters	starring director	David_Conolly David_Conolly
3	A_Daughter_of_Australia A_Daughter_of_Australia	starring director	Lawson_Harris Lawson_Harris
4	Out_In_Fifty Out_In_Fifty	starring director	Scott_Leet Scott_Leet
5	A_Man_Among_Giants A_Man_Among_Giants	starring director	Rod_Webber Rod_Webber
6	Single Single	starring director	Wilder_Shaw Wilder_Shaw
200	City_Lights City_Lights	starring director	Charlie_Chaplin Charlie_Chaplin
1500	Annie_Hall Annie_Hall	starring director	Woody_Allen Woody_Allen

Table 1.2: Un-ranked result set for: director/star of the same show.

- We develop a novel ranking model based on statistical machine translation for triple-pattern queries.
- To address the in-expressiveness of triple-pattern queries, we allow keyword-augmented triple-pattern queries.
- Our system supports automatic query relaxation (similar to query modification in traditional information retrieval) in case no results were found.
- Our ranking model takes into consideration result diversity in order to ensure that the user is provided with a wide range of aspects of the query results. We develop the first result diversity algorithm in the RDF setting.
- We propose a novelty-aware evaluation metric to assess the result set with respect to both relevance as well as diversity.
- Finally, we create a DBpedia benchmark that consists of 100 queries: purely structured queries, keyword-augmented queries, queries that require relaxation, and keyword-augmented queries that require relaxation. We use the benchmark to evaluate our various models.

Subject (S)	Predicate (P)	Object (O)
W.E. The_Pledge_(film) Madonna_(entertainer)	director director spouse	Madonna_(entertainer) Sean_Penn Sean_Penn
Secretprojectrevolution The_Pledge_(film) Madonna_(entertainer)	director director spouse	Madonna_(entertainer) Sean_Penn Sean_Penn
W.E. The_Crossing_Guard Madonna_(entertainer)	director director spouse	Madonna_(entertainer) Sean_Penn Sean_Penn
W.E. The_Platinum_Collection_(video) Madonna_(entertainer)	director director spouse	Madonna_(entertainer) Sean_Penn Sean_Penn
Secretprojectrevolution The_Crossing_Guard Madonna_(entertainer)	director director spouse	Madonna_(entertainer) Sean_Penn Sean_Penn
Secretprojectrevolution The_Platinum_Collection_(video) Madonna_(entertainer)	director director spouse	Madonna_(entertainer) Sean_Penn Sean_Penn
W.E. Snatch_(film) Madonna_(entertainer)	director director spouse	Madonna_(entertainer) Guy_Ritchie Guy_Ritchie
Secretprojectrevolution Snatch_(film) Madonna_(entertainer)	director director spouse	Madonna_(entertainer) Guy_Ritchie Guy_Ritchie
W.E. RocknRolla Madonna_(entertainer)	director director spouse	Madonna_(entertainer) Guy_Ritchie Guy_Ritchie
Secretprojectrevolution RocknRolla Madonna_(entertainer)	director director spouse	Madonna_(entertainer) Guy_Ritchie Guy_Ritchie

Table 1.3: Non-diversified result set for: director couples.

Chapter 2

Indexing and Query Processing

The Resource Description Framework (RDF) represents information in the Web. An RDF knowledge graph is a collection of RDF triples, such as DBpedia [1]. In this chapter, we discuss setting up the appropriate environment for dealing with such data. In Section 2.1, we go over DBpedia [1] and its datasets. In Section 2.2, we discuss running triple-pattern queries over the stored facts, then in Section 2.3, running keyword-augmented queries. Finally, in Section 2.4, we discuss how to compose a set of relaxed queries in case the user’s query did not produce any results.

2.1 Knowledge Graph: DBpedia

DBpedia [1] is a large structured knowledge base retrieved from the information in Wikipedia. The English version of DBpedia, which we use in this thesis, includes 5.8 million things: persons, places, creative works, organizations, species, and diseases. DBpedia [1] has 1337 unique relations, and 15778883 facts. In order to manipulate the facts (i.e, triples), we use PostgreSQL [9], an object-relational database. We store the collection of facts in a 3-column table: the `Subject` column, the `Predicate` column, the `Object` column. A sample facts table is shown in Figure 2.1. To enhance the retrieval performance, we construct B-tree indexes on all columns: "subject", "predicate", and "object", as well as all the combinations of the columns: "subject, predicate", "subject, object", "predicate, object", and "subject, predicate, object".

2.2 Triple Patterns

Querying DBpedia is accomplished using SPARQL [7]. A SPARQL query is a set of triple patterns and conjunctions. The same variable can be seen in more than one triples of the query, indicating a join condition. For example: `<?m starring ?x; ?m director ?x>` requires the names of shows or movies that were directed

	subject character varying	predicate character varying	object character varying
1	Aman Verma (footballer)	http://dbpedia.org/ontology/birthDate	1987-01-03 http://www.w3.org/2001/XMLSchema#date
2	Gertrude Noone	http://dbpedia.org/ontology/deathYear	2009 http://www.w3.org/2001/XMLSchema#Year
3	Admiral Dewey (tugboat)	http://xmlns.com/foaf/0.1/name	ADMIRAL DEWEY (tugboat)
4	The Devil Does Exist	http://dbpedia.org/ontology/type	Manga
5	Gloc-9	http://dbpedia.org/ontology/associatedBand	Francis Magalona
6	Anopina phaeopina	http://dbpedia.org/ontology/class	Insect
7	Des Coe	http://dbpedia.org/ontology/birthYear	1958 http://www.w3.org/2001/XMLSchema#Year
8	Act of Necessity	http://xmlns.com/foaf/0.1/name	Act of Necessity
9	Scopula nesciaria	http://dbpedia.org/ontology/phylum	Arthropod
10	Mimeresia moreelsi	http://dbpedia.org/ontology/phylum	Arthropod
11	Matt Derbyshire 11	http://dbpedia.org/ontology/numberOfGoals	0
12	Fort de Buade	http://xmlns.com/foaf/0.1/name	Fort de Buade
13	Bellante	http://xmlns.com/foaf/0.1/name	Comune di Bellante
14	T. J. Graham	http://dbpedia.org/ontology/team	Buffalo Bills
15	Connor Undercover	http://dbpedia.org/ontology/openingTheme	Rex Goudie

Figure 2.1: Facts table in PostgreSQL

subject_0 character varying	predicate_0 character varying	object_0 character varying	subject_1 character varying	predicate_1 character varying	object_1 character varying
1 I Killed My Mother	http://dbpedia.org/ontology/starring	Xavier Dolan	I Killed My Mother	http://dbpedia.org/ontology/director	Xavier Dolan
2 Minnesota Cuke and the Search for Noah's Umbrel	http://dbpedia.org/ontology/starring	Mike Nawrocki	Minnesota Cuke and the Search for Noah's Umbrel	http://dbpedia.org/ontology/director	Mike Nawrocki
3 Heart and Soul (1948 film)	http://dbpedia.org/ontology/starring	Vittorio De Sica	Heart and Soul (1948 film)	http://dbpedia.org/ontology/director	Vittorio De Sica
4 Walkover (film)	http://dbpedia.org/ontology/starring	Jerry Seinfeld	Walkover (film)	http://dbpedia.org/ontology/director	Jerry Seinfeld
5 As I Lay Dying (film)	http://dbpedia.org/ontology/starring	James Franco	As I Lay Dying (film)	http://dbpedia.org/ontology/director	James Franco
6 The Scarecrow (1920 film)	http://dbpedia.org/ontology/starring	Buster Keaton	The Scarecrow (1920 film)	http://dbpedia.org/ontology/director	Buster Keaton
7 Lady in the Lake	http://dbpedia.org/ontology/starring	Robert Montgomery (actor)	Lady in the Lake	http://dbpedia.org/ontology/director	Robert Montgomery (actor)
8 Reminiscences of Yearning	http://dbpedia.org/ontology/starring	Rousbeh Rashedi	Reminiscences of Yearning	http://dbpedia.org/ontology/director	Rousbeh Rashedi
9 Marudhanayagam	http://dbpedia.org/ontology/starring	Kamal Haasan	Marudhanayagam	http://dbpedia.org/ontology/director	Kamal Haasan
10 Fatty's Jonah Day	http://dbpedia.org/ontology/starring	Roscoe Arbuckle	Fatty's Jonah Day	http://dbpedia.org/ontology/director	Roscoe Arbuckle
11 City of Boys	http://dbpedia.org/ontology/starring	Frank Rosta	City of Boys	http://dbpedia.org/ontology/director	Frank Rosta
12 Men of Crisis: The Harvey Wallinger Story	http://dbpedia.org/ontology/starring	Woody Allen	Men of Crisis: The Harvey Wallinger Story	http://dbpedia.org/ontology/director	Woody Allen
13 Love Is Colder Than Death (film)	http://dbpedia.org/ontology/starring	Rainer Werner Fassbinder	Love Is Colder Than Death (film)	http://dbpedia.org/ontology/director	Rainer Werner Fassbinder
14 The Adventure of Taurus Bldgood	http://dbpedia.org/ontology/starring	Andy Jones (comedian)	The Adventure of Taurus Bldgood	http://dbpedia.org/ontology/director	Andy Jones (comedian)
15 Brother (2000 film)	http://dbpedia.org/ontology/starring	Takeshi Kitano	Brother (2000 film)	http://dbpedia.org/ontology/director	Takeshi Kitano

Figure 2.2: Sample answer of a SPARQL query in PostgreSQL

and starred by the same person. Since we are using PostgreSQL to store our data, an SQL query should be created and executed for each SPARQL query. We develop a script that would take a SPARQL query as input, and produce its corresponding SQL query. For our example, the SQL translation is: "SELECT * FROM facts f1, facts f2 WHERE f1.subject=f2.subject and f1.object=f2.object and f1.predicate='starring' and f2.predicate='director';" A snapshot of the answer is shown in Figure 2.2. Each row represents one answer, which is a subgraph isomorphic to the query. In this case a 2-triple-pattern query would produce a 2-triple subgraph.

In our ranking model, which we discuss in Chapter 3, we need to associate a weight with each triple. The weight of the fact indicates the importance of this fact. The computation of the weight is proposed in Section 3.1.2. To store the triples weights, we augment our facts table with a new column we call "witness-count". A sample table is shown in Figure 2.3.

2.3 Keyword-augmented Triple Patterns

Another type of queries that our search engine accepts are keyword-augmented structured queries. It has the same format as the queries in Section 2.2, except the user is allowed to add one or more keywords to the triple patterns. For example: <?m starring ?x[romance new york comedy]; ?m director ?x>. This example asks for the name of a movie that was directed and starred by the

	subject character varying	predicate character varying	object character varying	witnesscount integer
1	Mona de Momma	http://dbpedia.org/ontology/trainer	John W. Sadler	84
2	Artificial Joy NYC	http://dbpedia.org/ontology/formerBandMember	Alex Elena	5
3	Robert Rosen (theoretical	http://xmlns.com/foaf/0.1/name	Rosen, Robert	24
4	Marko DeSantis	http://dbpedia.org/ontology/associatedMusica	Popsicko	22
5	That Girl	http://dbpedia.org/ontology/executiveProduce	Sam Denoff	224
6	Definition (song)	http://dbpedia.org/ontology/writer	Hi-Tek	164
7	Mircea Florian	http://dbpedia.org/ontology/deathPlace	Bucharest	6254
8	Wellsville, Kansas	http://dbpedia.org/ontology/populationDensit	515.8324838127078 http://www.w3.org/2001/XMLSchema	23
9	Abdou N'Daffa Faye	http://dbpedia.org/ontology/deathYear	1967 http://www.w3.org/2001/XMLSchema#Year	5
10	Hauschka	http://dbpedia.org/ontology/recordLabel	FatCat Records	19
11	Davariyeh	http://dbpedia.org/ontology/timeZone	Iran Standard Time	28061
12	Schinder (mountain)	http://dbpedia.org/ontology/locatedInArea	Germany	121827
13	Concord, Arkansas	http://dbpedia.org/ontology/country	United States	434204
14	Richard Collier	http://dbpedia.org/ontology/activeYearsEndYe	2008 http://www.w3.org/2001/XMLSchema#Year	14
15	Helicobacter anseris	http://dbpedia.org/ontology/genus	Helicobacter	31
16	Ryieish Green	http://dbpedia.org/ontology/country	England	136425
17	Theodore William Moody	http://dbpedia.org/ontology/birthYear	1907 http://www.w3.org/2001/XMLSchema#Year	58
18	Howard M. Teshner	http://dbpedia.org/ontology/race	Top Flight Handicap	91
19	Necklace and Calabash	http://dbpedia.org/ontology/clic	26129085	10

Figure 2.3: Witnesscounts associated with DBpedia triples

	subject character varying	predicate character varying	object character varying	keywords character varying
1	Annie Hall	http://dbpedia.org/ontology/director	Woody Allen	standup: 1, 2004: 367, fret: 1, maintain: 1, documentari: 186, b
2	Friends	http://dbpedia.org/ontology/creator	David Crane (producer)	2006: 269, klarik: 13, 2004: 257, bright: 115, treatment: 52, nov
3	Agatha Christie	http://dbpedia.org/ontology/birthPlace	Torquay	stori: 492, fish: 35, bodley: 25, thirti: 20, wrote: 24, cru00e8
4	AGL Desktop	http://dbpedia.org/ontology/programmingLangu	C++	integr: 1, client: 177, aol: 49, player: 197, desktop: 204, upgr
5	Friends	http://dbpedia.org/ontology/creator	Marta Kauffman	2004: 260, bright: 114, treatment: 54, novemb: 24, phenomenon: 1
6	Agatha Christie	http://dbpedia.org/ontology/birthPlace	Devon	tamar: 196, stori: 614, histor: 11, deriv: 1, fish: 174, bodley:

Figure 2.4: Triples with keywords and keywords weights

same person. It also indicates the user's preference in receiving romantic comedy movies that happened in New York. The execution of this type of queries is handled the same way as the queries in Section 2.2. All the sgraphs that are isomorphic to the query would be retrieved. In order to handle the keywords, each triple is associated with a set of keywords. In addition, each keyword associated with a certain triple would also have a weight. The keyword weight is used to indicate the relevance of the triple to the keywords in the query. The details of keyword-augmentation is discussed in details in Section 3.1.3. A sample of triples associated with keywords and keywords weights is shown in Figure 2.4.

2.4 Query Relaxation

The user must know the exact URIs for resources and predicates in order to construct a query that will return a set of results. Given a structured query Q , which consists of triple patterns q_1, q_2, \dots, q_n , with the possibility of associating one or more triple patterns with a set of keywords: w_1, w_2, \dots, w_l . The query can return no results in many occasions:

Conjunction has no results: In this case, the user knows the exact URIs for the query, and the triple patterns are written correctly. However, the combination of those triple patterns has no result subgraphs. The following query is an example for such a case: `<?m starring Woody_Allen; ?m director Madonna_(entertainer)>`. The query requires the name of a movie that was starred by Woody Allen and

directed by Madonna. This query produces no results since the knowledge base doesn't contain any movies that were directed by Madonna and starred by Allen. The queries that have such a problem will be handled in the following manner: The query is transformed into a number of relaxed queries. And the number of relaxed queries is equal to the number of constants in the main query. For our example, 4 queries are produced and displayed in Table 2.1. The table shows that for each relaxed query, one constant is replaced by a variable, and the constant (URI) is moved to the keywords space.

Query#	Relaxed Query
Q1	?m ?A Woody_Allen [starring] ?m director Madonna_(entertainer)
Q2	?m starring ?A [woody allen] ?m director Madonna_(entertainer)
Q3	?m starring Woody_Allen ?m ?A Madonna_(entertainer) [director]
Q4	?m starring Woody_Allen ?m director ?A [madonna entertainer]

Table 2.1: List of relaxed queries for movies starred by Allen and directed by Madonna

Table 2.2 shows a number of un-ranked subgraphs for the relaxed queries in Table 2.1.

Isomorphic to Q#	Subgraph
Q3	Shadows_and_Fog starring Woody_Allen Shadows_and_Fog starring Madonna_(entertainer)
Q4	Annie_Hall starring Woody_Allen Annie_Hall director Woody_Allen
Q4	New_York_Stories starring Woody_Allen New_York_Stories director Martin_Scorsese
Q4	Fading_Gigolo starring Woody_Allen Fading_Gigolo director John_Turturro
Q2	W.E. starring Abbie_Cornish W.E. director Madonna_(entertainer)

Table 2.2: List of possible subgraphs for queries in Table 2.1

No matches for one or more triple patterns: This case can be divided into two subcases:

- *Subject, Predicate, or Object in a triple pattern does not exist in DBpedia.* For example: `<?m director Woody_Allan; ?m starring ?x>` has no results, because the object of the first triple pattern does not exist in the knowledge base. More precisely, "Allan" must be "Allen". This kind of problem is handled as follows: The constant causing the problem is replaced by a variable then moved

to the keyword space. In this example, one of the relaxed queries produced is: `<?m director ?A [woody allan]; ?m starring ?x>`. One sample result for the relaxed query is: `<Helena_from_the_Wedding director Joseph_Infantino; Helena_from_the_Wedding starring Dagmara_Dominczyk>`. Another sample result is: `<Shadows_and_Fog director Woody_Allen; Shadows_and_Fog starring Madonna_(entertainer)>`.

- *Subject, Predicate, and Object exist in DBpedia but the triple does not.* For example, `<?book author Jacques_Chirac; ?book publisher ?p>`, which requires the name of a book written by former president Chirac, and the name of the publisher of the book. In DBpedia, there exists a relation "author" and an entity "Jacque.Chirac", but there are no triples that include both of them at the same time. This problem is solved by relaxing the triple which is causing the issue: `?book ?A Jacques_Chirac [author]; ?book publisher ?p`, and `?book author ?A [jacques chirac]; ?book publisher ?p`.

One sample answer is: `<The_Silent_World_(book) author Jacques_Cousteau; The_Silent_World_(book) publisher Harper_(publisher)>`.

The subgraphs produced by all the relaxed queries of the main query are ranked based on our proposed model in Section 3.2.

Chapter 3

Ranking

In this chapter, we discuss the problem of producing top-k subgraphs for a user’s structured or keyword-augmented structured query. In Section 3.1, we propose a ranking model based on statistical machine translation. We also explain how to calculate the weight of a triple, augment a triple with keywords, and give weights to those keywords. In section 3.2, we discuss the solution of the ”no results found” problem by proposing a query relaxation model. We revise the ranking and query relaxing related work in Section 3.3. In Section 3.4, we finally use a benchmark of 100 queries in order to compare the effectiveness of our models with other state-of-art models.

3.1 Ranking Model

3.1.1 Result Ranking

Often, triple-pattern queries will return too many results, making it difficult for users to find the most relevant ones. Querying RDF graphs using SPARQL will result in subgraphs that match the query structure without taking into consideration which subgraph is more relevant than others. We develop a ranking model based on statistical language models (LMs).

Given a query Q which consists of n triple patterns q_1, q_2, \dots, q_n . Assume G is a result subgraph which consists of n triples t_1, t_2, \dots, t_n . To rank G with respect to Q , we use the following LM-based ranking:

$$P(Q|G) = \prod_{i=1}^n P(q_i|G) \quad (3.1)$$

We then use a translation model as follows:

$$P(q_i|G) = \sum_{j=1}^m P(q_i|t_j)P(t_j|G) \quad (3.2)$$

where $P(t_j|G)$ is defined as the probability of generating triple t_j given result subgraph G :

$$P(t_j|G) = \begin{cases} \frac{1}{|G|}, & \text{if } t_j \text{ is part of } G \\ 0, & \text{otherwise} \end{cases} \quad (3.3)$$

The probability of translating triple pattern q_i into triple t_j is defined in two ways depending on whether q_i is keyword augmented or not.

In case q_i is not keyword-augmented, we have:

$$P(q_i|t_j) = \frac{wc(t_i)}{\sum_{t \in \hat{q}_i} wc(t)} \quad (3.4)$$

where \hat{q}_i is the set of all triples matching q_i and $wc(t)$ is the witnesscount(i.e weight) of a triple t .

Example: Consider the following query Q : $\langle ?s \text{ producer } ?x; ?s \text{ director } ?x \rangle$ which asks for a person who produced and directed the same show, his/her name and the name of the show. Running this query without a ranking model would produce the results shown in Table 3.1.

Subject (S)	Predicate (P)	Object (O)
Koi_Tujh_Sa_Kahan	producer	Reema_Khan
Koi_Tujh_Sa_Kahan	director	Reema_Khan
Chanda_(film)	producer	S._Narayan
Chanda_(film)	director	S._Narayan
The_Bling_Ring	producer	Sofia_Coppola
The_Bling_Ring	director	Sofia_Coppola
Tadpole_(film)	producer	Gary_Winick
Tadpole_(film)	director	Gary_Winick
Barbed_Wire_(1927_film)	producer	Rowland_V._Lee
Barbed_Wire_(1927_film)	director	Rowland_V._Lee
Tarnation_(film)	producer	Jonathan_Caouette
Tarnation_(film)	director	Jonathan_Caouette
Rock_of_Ages_(2012_film)	producer	Adam_Shankman
Rock_of_Ages_(2012_film)	director	Adam_Shankman
Crocodile_Hunter_(film)	producer	Wong_Jing
Crocodile_Hunter_(film)	director	Wong_Jing
Stoic_(film)	producer	Uwe_Boll
Stoic_(film)	director	Uwe_Boll
The_Rosebud_Beach_Hotel	producer	Harry_Hurwitz
The_Rosebud_Beach_Hotel	director	Harry_Hurwitz

Table 3.1: Top-10 un-ranked results: producer/director of the same show.

Now we run the same query, but we rank the results using our LM-based ranking model. The results are shown in Table 3.2.

Subject (S)	Predicate (P)	Object (O)
W.E.	producer	Madonna_(entertainer)
W.E.	director	Madonna_(entertainer)
None_But_the_Brave	producer	Frank_Sinatra
None_But_the_Brave	director	Frank_Sinatra
Snoop_Dogg's_Hustlaz:_Diary_of_a_Pimp	producer	Snoop_Dogg
Snoop_Dogg's_Hustlaz:_Diary_of_a_Pimp	director	Snoop_Dogg
Lincoln_(2012_film)	producer	Steven_Spielberg
Lincoln_(2012_film)	director	Steven_Spielberg
A.I._Artificial_Intelligence	producer	Steven_Spielberg
A.I._Artificial_Intelligence	director	Steven_Spielberg
Munich	producer	Steven_Spielberg
Munich	director	Steven_Spielberg
Amistad_(film)	producer	Steven_Spielberg
Amistad_(film)	director	Steven_Spielberg
Always_(1989_film)	producer	Steven_Spielberg
Always_(1989_film)	director	Steven_Spielberg
The_Karnival_Kid	producer	Walt_Disney
The_Karnival_Kid	director	Walt_Disney
The_Opry_House	producer	Walt_Disney
The_Opry_House	director	Walt_Disney

Table 3.2: Top-10 ranked results: producer/director of the same show.

In case q_i is keyword-augmented, with keywords w_1, w_2, \dots, w_l :

$$P(q_i|t_j) = \prod_{k=1}^l P(q_i, w_k|t_j) \quad (3.5)$$

where $P(q_i, w_k|t_j)$ is defined as follows:

$$P(q_i, w_k|t_j) = \alpha \frac{wc(t_j, w_k)}{\sum_{t \in \hat{q}_i} wc(t, w_k)} + (1 - \alpha) \frac{wc(t_j)}{\sum_{t \in \hat{q}_i} wc(t)} \quad (3.6)$$

where α is a smoothing parameter, and $wc(t_j, w_k)$ is the witnesscount of keyword w_k associated with triple t_j . α is set to 0.8.

Example: Consider the following query Q: <?s producer ?x[Chaplin]; ?s director ?x> which asks for a person, preferably Charlie Chaplin, who produced and directed the same show, his/her name and the name of the show. Running this query using our ranking model which allows keyword-augmented queries would produce the results shown in Table 3.3.

Subject (S)	Predicate (P)	Object (O)
Modern_Times_(film)	producer	Charlie_Chaplin
Modern_Times_(film)	director	Charlie_Chaplin
Limelight_(1952_film)	producer	Charlie_Chaplin
Limelight_(1952_film)	director	Charlie_Chaplin
The_Kid_(1921_film)	producer	Charlie_Chaplin
The_Kid_(1921_film)	director	Charlie_Chaplin
A_Countess_from_Hong_Kong	producer	Charlie_Chaplin
A_Countess_from_Hong_Kong	director	Charlie_Chaplin
A_King_in_New_York	producer	Charlie_Chaplin
A_King_in_New_York	director	Charlie_Chaplin
A_Dog's_Life	producer	Charlie_Chaplin
A_Dog's_Life	director	Charlie_Chaplin
A_Day's_Pleasure	producer	Charlie_Chaplin
A_Day's_Pleasure	director	Charlie_Chaplin
Nashville_(film)	producer	Robert_Altman
Nashville_(film)	director	Robert_Altman
A_Busy_Day	producer	Mack_Sennett
A_Busy_Day	director	Mack_Sennett
Cruel,_Cruel_Love	producer	Mack_Sennett
Cruel,_Cruel_Love	director	Mack_Sennett

Table 3.3: Top-10 ranked results: producer/director of the same show, with the keyword: "Chaplin".

3.1.2 Triple Weight

In order to give each fact a certain value of importance, we associated each triple with a corresponding weight that denotes how important this triple is. These weights are pre-computed, indexed and stored along the actual triples in the knowledge base.

We calculate the weight of a certain triple by leveraging external corpora to estimate the importance of a given triple (i.e., fact). For instance, for a knowledge base constructed from Wikipedia such DBpedia, we use the number of links pointing to the subject and object of a given triple as a measure of the weight of the triple. The intuition behind this is that the more articles that point to the subject and object of the triple, the more important a triple is.

The following example presents two triples of TV shows and their companies. The triples have the same structure of $t = (s, p, o)$ + the new witnesscount denoting the importance of that triple, $t = (s, p, o, w)$:

<Entertainment_USA company BBC 25552>
 <Designer_Guys company WestWind_Pictures 16>

In the case of DBpedia, the link information is present as part of the knowledge base in the form of additional triples. For example, the following triple states

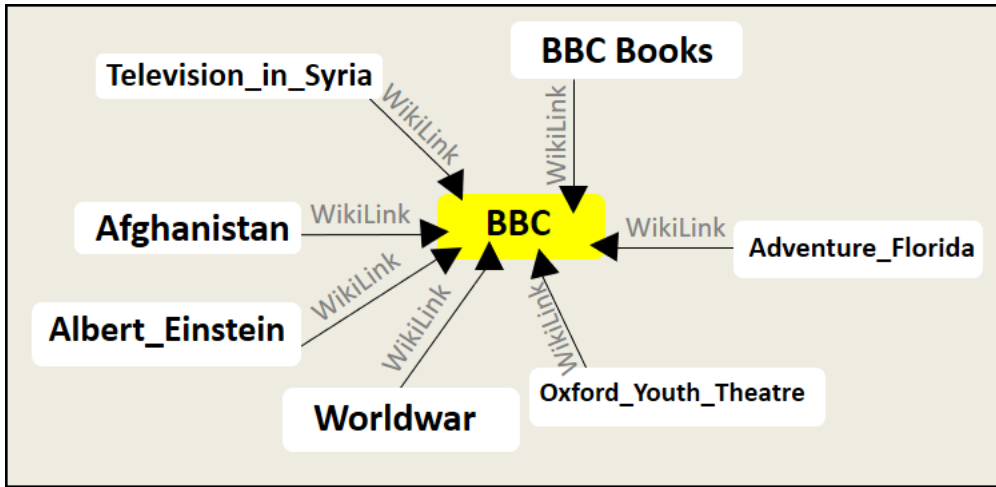


Figure 3.1: A subgraph of the "WikiLinks" graph

that the Wikipedia article of Afghanistan has a link to the Wikipedia article of BBC:

`<Afghanistan WikiLink BBC>`

An example of links pointing to the "BBC" entity is shown in Figure 3.1.

More generally, to compute the weight of a triple in any knowledge base, co-occurrence statistics from external corpora such the Web can be used. However, in this thesis, we build our search engine on top of a particular RDF knowledge base: DBpedia, which is based on Wikipedia and thus we leverage the link structure to compute the weights of the triples in this knowledge base. In particular, the weight of a triple $t = (s, p, o)$ is computed as shown in Equation 3.7.

$$WitnessCount(t) = |links(l, s)| + |links(l, o)| \quad (3.7)$$

where t is a triple, $|links(l, s)|$ is the number of WikiLinks pointing to the Subject s in triple t , and $|links(l, o)|$ is the number of WikiLinks pointing to the Object o in triple t .

3.1.3 Keyword Augmentation

While RDF represents a flexible means to represent a variety of information or facts, some information are better suited as free text. For instance, in the domain of movies, a movie plot is typically text-based and cannot be represented naturally using RDF triples. To accommodate this, we augment the triples in DBpedia with keywords. This allows the user to combine SPARQL queries with keyword conditions to find information that cannot be found using triple-pattern queries only.

Example: The user is interested in finding movies/shows whose directors also starred in them, and in addition are movies/shows about war and whose directors were nominated for Oscars, the user can issue the following keyword-augmented query: `<?m starring ?x [war] ; ?m director ?x [oscars]>`.

One of the results for such keyword-augmented query would be:
`<Henry_V_(1944_film) starring Laurence_Olivier; Henry_V_(1944_film) director Laurence_Olivier>`, whose description in Wikipedia consists of the following snippet: *Henry_V is a 1944 war movie. This was Laurence Olivier's third Oscar-nominated performance*

This step can be handled differently depending on the knowledge base searched. For example, in the case of DBpedia, each entity is associated with an "abstract", which is the first paragraph of the Wikipedia page representing the entity. Thus, for each triple in the knowledge base, we extract the abstracts of the subject and the object, stem those paragraphs, remove stop words and duplicates, and then associate the triple with the resulting list of keywords.

For example, the triple `<Henry_V_(1944_film) director Laurence_Olivier>` is associated with the following keywords: [oscar, academi, 1944, shakespeare, hamlet, war, cinema].

In the case of keyword-augmented queries, ranking the results should be based not only on the importance or the weight of the triples only, but also on the relevance of the triple to the keywords in the query.

To be able to do this, we associate each keyword for a given triple with a weight reflecting the relevance of the triple with respect to the keyword. This score is expressed as the intersection of all the links pointing to the Subject entity and the Keyword entity, plus the intersection of all the links pointing to the Object entity and the Keyword entity.

After augmenting the data with keywords, the keywords are given weights based on the following equation:

$$\begin{aligned} WitnessCount(k, t) = & |inter(links(l, s), links(l, k))| \\ & + |inter(links(l, o), links(l, k))| \end{aligned} \quad (3.8)$$

where $WitnessCount(k, t)$ is the witnesscount of keyword k associated with triple t , $|inter(links(l, s), links(l, k))|$ is the number of WikiLinks pointing at both the subject s and any other resource or literal containing the keyword k , and $|inter(links(l, o), links(l, k))|$ is the number of WikiLinks pointing at both the object o and any other resource or literal containing the keyword k .

For the above example, the triple:
`<Henry_V_(1944_film) director Laurence_Olivier>` is associated with the following weight-augmented keywords with weights: [oscar: 175, academi: 1072, 1944: 245, shakespeare: 595, hamlet:458, war: 903, cinema: 165].

3.2 Relaxation Model

The problem of "no results found" is discussed in Section 2.4, as well as the reasons causing such a problem, and a solution is provided for each case. In this section, we discuss ranking the subgraphs produced by a set of relaxed queries. The goal of ranking subgraphs produced by the relaxed queries is to push to the top the results that are as related to the given query as possible.

Given Q , a structured query or a keyword augmented structured query with zero results.

1. Q is relaxed based on the causes and solutions discussed in Section 2.4.
2. The relaxed queries are executed.
3. All subgraphs are retrieved and scored. More formally, the score of a subgraph G produced by a set of relaxed queries R is computed as shown in equation 3.9.

$$P(R|G) = \frac{1}{k} \sum_{i=1}^k P(R_i|G) \quad (3.9)$$

where k is the size of R and $P(R_i|G)$ is defined in 3.10.

$$P(R_i|G) = \begin{cases} 0, & \text{if } G \notin \text{Result Set of } R_i \\ \text{Equation 3.1}, & \text{otherwise} \end{cases} \quad (3.10)$$

The intuition behind this scoring is that the duplicated subgraphs must be given higher weights.

4. Rank the subgraphs in descending order of score.
5. Display top- k results.

Example: Consider the query `<Agatha_Christie deathPlace ?place; ?person birthPlace ?place; ?person almaMater Oxford>`. The query returns zero results. The relaxation strategy is applied and the top-10 subgraphs are shown in Table 3.4.

Example: Consider the query `<?m starring Frank_Senatra; ?m producer ?p>`. The entity "Frank_Senatra" must be "Frank.Sinatra". Because of this mistake, the query produces no results. The relaxation strategy is applied and the top-10 subgraphs are shown in Table 3.5.

Subject (S)	Predicate (P)	Object (O)
Agatha_Christie	deathPlace	Oxfordshire
Roundell_Palmer,_1st_Earl_of_Selborne	birthPlace	Oxfordshire
Roundell_Palmer,_1st_Earl_of_Selborne	almaMater	University_of_Oxford
Viola_Keats	deathPlace	United_Kingdom
Robert_Bridges	birthPlace	United_Kingdom
Robert_Bridges	almaMater	Oxford
Agatha_Christie	deathPlace	Oxfordshire
David_Cameron	birthPlace	Oxfordshire
David_Cameron	almaMater	Eton_College
Agatha_Christie	deathPlace	Oxfordshire
Roger_Norrington	birthPlace	Oxfordshire
Roger_Norrington	birthPlace	Oxford
Agatha_Christie	deathPlace	Oxfordshire
Martin_Keown	birthPlace	Oxfordshire
Martin_Keown	birthPlace	Oxford
Agatha_Christie	deathPlace	Oxfordshire
Orlando_Gibbons	birthPlace	Oxfordshire
Orlando_Gibbons	birthPlace	Oxford
Agatha_Christie	deathPlace	Oxfordshire
David_Oyelowo	birthPlace	Oxfordshire
David_Oyelowo	birthPlace	Oxford
Agatha_Christie	deathPlace	Oxfordshire
Dexter_Blackstock	birthPlace	Oxfordshire
Dexter_Blackstock	birthPlace	Oxford
Agatha_Christie	deathPlace	Oxfordshire
Dexter_Blackstock	birthPlace	Oxfordshire
Dexter_Blackstock	birthPlace	Oxford
Agatha_Christie	deathPlace	Oxfordshire
Heather_Angel_(actress)	birthPlace	Oxfordshire
Heather_Angel_(actress)	birthPlace	Oxford
Agatha_Christie	deathPlace	Oxfordshire
Garry_Parker	birthPlace	Oxfordshire
Garry_Parker	birthPlace	Oxford

Table 3.4: Top-10 ranked results: people who graduated from Oxford and were born in Agatha Christie’s death place.

3.3 Related Work

3.3.1 Unstructured Queries on Unstructured Data

The traditional information retrieval technique was based on running a keyword query over unstructured data, mostly text-based documents. For works such as [10], [11], and [12], given a query Q (a set of keywords), and a result (a text

Subject (S)	Predicate (P)	Object (O)
None_But_the_Brave	starring	Frank_Sinatra
None_But_the_Brave	producer	Frank_Sinatra
On_the_Town_(film)	starring	Frank_Sinatra
On_the_Town_(film)	producer	Arthur_Freed
The_Amazing_Mr._Bickford	starring	Frank_Zappa
The_Amazing_Mr._Bickford	producer	Frank_Zappa
From_Here_to_Eternity	starring	Frank_Sinatra
From_Here_to_Eternity	producer	Buddy_Adler
Take_Me_Out_to_the_Ball_Game_(film)	starring	Frank_Sinatra
Take_Me_Out_to_the_Ball_Game_(film)	producer	Arthur_Freed
The_Pride_and_the_Passion	starring	Frank_Sinatra
The_Pride_and_the_Passion	producer	Stanley_Kramer
On_the_Town_(film)	starring	Frank_Sinatra
On_the_Town_(film)	producer	Roger_Edens
Double_Dynamite	starring	Frank_Sinatra
Double_Dynamite	producer	Irwin_Allen
High_Society_(1956_film)	starring	Frank_Sinatra
High_Society_(1956_film)	producer	Sol_C._Siegel
None_But_the_Brave	starring	Frank_Sinatra
None_But_the_Brave	producer	William_H._Daniels

Table 3.5: Top-10 ranked results of: people who worked with Frank Sinatra.

document), two probability distributions are computed for both the query and the document. If the probability distributions are close enough, the document is considered highly relevant to the query. In this thesis, our corpus is a knowledge graph, instead of a set of documents, our query is a keyword-augmented set of triple patterns, instead of keywords, and our output is a set of ranked subgraphs, instead of a set of relevant documents.

3.3.2 Unstructured Queries on Structured Data

The work in [13], [14], [15], and [16] is keyword search on XML data or graphs. The output of a keyword search in this case is a ranked list of trees or graphs. The authors in [17] and [5] proposed a model of entity search on records. The output is a list of ranked entities. Our work rank subgraphs consisting of triples instead of ranking entities. A closer work is [18], where the authors proposed a ranking model of keyword search over RDF graphs. The output in this case is a ranked list of subgraphs. Content-based measure such as $tf*idf$, which were applied in [17], [15], and [19] would be irrelevant in our work since we don't deal with terms. Finally, the graphs which are proposed as results in [13] and [18] do not have a specific structure or size. On the other hand, in our system, the results(subgraphs) and the query are isomorphic, so aggregation over the nodes

and edges [17], [15], and [19] would not work for us. Our approach handles keyword-augmented structured queries which is different from all works in this category.

3.3.3 Structured Queries on Structured Data

The work in [20] deals with SPARQL queries on RDF graphs. The result graphs are estimated based on confidence of the result, which is irrelevant in our work, because all the triples are extracted from the same knowledge-base using the same method, thus having the same confidence values. It is also based on compactness, which is also unrelated since the size of the outputted subgraphs is determined by the users query. (i.e the number of triple patterns of the query). The third element of result estimation is the informativeness component which we evaluate using witnesscount. Unlike our work, NAGA does not support relaxed queries and keyword-augmented queries. It also does not address diversifying the set of results.

The authors in [21] proposed extending 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. In our work, we don't modify the triples. The queries in this work are SPARQL plus text phrases, where the subject, predicate, or object can be augmented with one or more keywords. In our work, we allow augmenting the whole triple with one or more keywords. They handle query relaxation, which we do as well. However, they do not address the diversity problem.

One of the closest related work to ours is [22, 23], where the authors propose searching RDF graphs with SPARQL and keywords. They introduce a language model based approach to ranking the results of exact, relaxed and keyword-augmented queries over RDF graphs such as graphs. They are different from our work in the following four aspects: our work develop a new ranking model based on statistical machine translation that utilizes Wikipedia link structure. Second, the authors work was evaluated using two relatively small datasets unlike our case where it will be evaluated on DBpedia. Third, our approach performs a careful and efficient relaxation of queries to ensure that the relaxed queries are as close to the original query intention as possible. Finally, our approach will not only take into consideration the relevance of results but also their diversity based on Maximal Marginal Relevance [24].

3.3.4 Query Relaxation

On relaxation, there is [25]. The authors develop a comprehensive set of query relaxation techniques, where the relaxation candidates can be collected from both the RDF data itself as well as from external ontological and textual sources. Their query relaxation techniques consists of four types of relaxations. The first one is

replacing entities and relations with some other entities and relations that have the same meaning. For example: `bornIn` can be substituted with `citizenOf`. The second type is replacing the entities and relations with variables. This type has a lot to do with our relaxation techniques, in which we don't just add variables randomly but according to a certain study of why the query is not giving any results. The third type that the authors present is removing a certain triple entirely, thus converting an n-triple query into an (n-1)-triple query. In our work, we don't play with the size of the main query, we don't remove nor add a triple pattern. The last type is dropping all or some of the keywords associated with a triple. Additionally, expansion terms could be also considered for each keyword. To rank their subgraphs, each relaxed query has its own weight. The higher the weight the greater the similarity between the relaxed query and the main query.

Also on relaxation, there is [26]. The authors address the issue of relaxing the user query on RDF databases and computing the most relevant answers. They measure the similarities of the relaxed queries with the original query and design the algorithm to retrieve the most relevant answers as soon as possible. Not all the relaxed queries produced will be executed on the data, they will disregard the unnecessary ones. For our work, we are only producing relaxed queries that we think are suitable and close enough to the user's main query.

The authors in [21] propose two strategies for query relaxation: structural relaxation and predicate paraphrasing. Structural relaxation is done by replacing a triple pattern by a set of triple patterns that denote a path. For example: `<?x bornIn UK>` is replaced by `<?x bornIn ?y; ?y locatedIn UK>`. The other type of query relaxation is done by generating paraphrases for each predicate. For example, the predicate `graduatedFrom` would be transformed into the textual predicate `"went to"`.

In [27], the authors work on investigating techniques to find the parts in the main query that are responsible of its failure. They propose producing a set of relaxed queries, called Maximal Succeeding Subqueries (XSSs). These generated subqueries must have the maximal number of triple patterns of the initial query. They continue to improve their work in [28], by proposing three relaxation strategies. In their first strategy, they deal with the failure causes of the initial query. In their second strategy, they proceed to deal with the failure causes in the set of relaxed queries. Since some of the failure causes might not be discovered in the second strategy, they propose a third strategy. This strategy is an optimization of the second one, where the gaps will be filled.

3.4 Experimental Evaluation

3.4.1 Setup

In order to evaluate our ranking algorithm. We construct a 100 queries benchmark over DBpedia, and which can be broadly divided into 4 categories: structured, keyword-augmented, requiring relaxation, and keyword-augmented and requiring relaxation.

We run the benchmark over four models, where we collect the top-10 subgraphs for each query. The first model is ours. The second model is the no-rank model, where the order of the subgraphs is random. The third model is similar to ours for the queries with no keywords, however, the witnesscounts of the facts are calculated differently, Equation 3.11. This model does not support keyword-augmented queries. The fourth and last model is [22], which has been proven to outperform [13], [29], and [5].

$$WitnessCount(t) = deg(s) + deg(o) \quad (3.11)$$

where $deg(s)$ and $deg(o)$ are the number of edges in the knowledge graph G that have s and o as objects respectively.

To gather the relevance assessments, we relied on the crowdsourcing platform CrowdFlower [30] as follows. We run each query four times over four models. We collect the top-10 results of each model. For each query, we submit two sets to be compared, one is produced by our ranking model, and the other one is a set produced by a competitor. More particularly, each set of ours is going through 3 comparisons: one with the unranked set, one with the set produced by the "indegrees" model, and one with the set produced by the model in [22], which we call LMRQ. The assessor in each comparison must choose the set that she feels it answers the query better. Appendix A shows more details about the guidelines.

The assessors are also asked to pass a test and get at least a 90% score to be able to continue assessing. Moreover, a set of gold queries are embedded in the benchmark to catch and exclude any bad behavior or random answer.

3.4.2 Results

The results of our evaluation are shown in Table 3.6. The reported numbers are out of 100 queries. In the case of no-rank and in-degrees, our model outperforms both competitors by winning 70% and 60% of the queries respectively. For the thirs model, we proved to be as good as the state-of-art model, by having the same quality of result sets for 95% of the queries. Moreover, we did not loose any query against [22] (LMRQ), and we won 5% of the queries. Each assessor had to provide a valid explanation of her choice. In Tables 3.7, 3.8, and 3.9, we show some of these explanations.

Ours vs.:	No-rank	Indegrees	LMRQ
Won	70	60	5
Tie	29	36	95
Lost	1	2	0

Table 3.6: Results for 100 evaluation queries: ranking algorithm

Ours vs. No-rank
<p>WON Q:List of music composers in Disney movies ? X <i>ours</i>, Y <i>no-rank</i> Comments: "Y does not include any of music composers in Disney movies." "X lists all the movies from Disney."</p>
<p>Q:List of NBA players who are also actors ? X <i>no-rank</i>, Y <i>ours</i> Comments: "Y lists includes more famous NBA players who are also actors." "X list does not answer to the question."</p>
<p>TIE Q:List of couples who graduated from the same universities ? X <i>ours</i>, Y <i>no-rank</i> Comments: "They both are good answers." "Different lists, both followed instruction."</p>
<p>Q: List of scientists who were born and died in the same place ? X <i>no-rank</i>, Y <i>ours</i> Comments: "They both are good answers." "Everything OK in both lists."</p>
<p>LOST Q:List of married actors who are influenced by the same people ? X <i>no-rank</i>, Y <i>ours</i> Comments: "More results in X." "There are more actors in the X list." "Same." "Only correct list is X."</p>

Table 3.7: Ours vs. No-rank explanation samples.

The inter-rater agreement produced by CrowdFlower was 76%, 88%, and 90%, for comparing with no-rank model, indegrees model, and [22] respectively.

We outperformed No-rank since it does not support ranking. The subgraphs are chosen randomly. In addition, the no-rank model does not take into con-

<p>Ours vs. Indegrees</p> <p>WON</p> <p>Q:List of death dates of indian actors ? <i>X indegrees, Y ours</i></p> <p>Comments: "X list are not Indian." "Y More indian actors."</p>
<p>Q:List of black comedy movies ? <i>X indegrees, Y ours</i></p> <p>Comments: "Y list includes black comedies." "List X is irrelevant."</p>
<p>TIE</p> <p>Q:List of late actresses from Los Angeles; and their death dates ? <i>X ours, Y indegrees</i></p> <p>Comments: "Both Y and X list correct answers". "Both lists contain information requested".</p>
<p>Q:List of doctors who graduated from the University of Oxford ? <i>X ours, Y indegrees</i></p> <p>Comments: "Both have the same accuracy." "X and Y both answer the question."</p>
<p>LOST</p> <p>Q:List of comedy movies starred by a couple and directed by Frank Sinatra ? <i>X indegrees, Y ours</i></p> <p>Comments: "X list includes Sinatra's movies." "X is good and lists the correct answers."</p>

Table 3.8: Ours vs. Indegrees explanations samples

sideration the keywords associated when ranking a subgraph. With the second model, the indegrees, we outperformed it because it does not handle the keywords associated with the query. In addition, the witnesscount computation in this model does not reflect the quality of a certain fact very well. That appears obviously in the case of the purely structured queries. In the case of the ties, where our sets and the no-rank sets, and our sets and the in degrees sets were considered the same, the logic of the assessors was that both sets answered the question, and that the question was somehow specific enough that almost all subgraphs are good enough answers.

For the model in [22], it was expected that we will do as good as they did, since we both use statistical language models in building our ranking models. We both support keyword-augmented queries, and we both support query relaxation.

<p>Ours vs. LMRQ</p> <p>WON</p> <p>Q:List of directors of comedy shows or movies about school and friends ? X <i>LMRQ</i>, Y <i>ours</i></p> <p>Comments: "ONLY Y HAS SHOW ABOUT SCHOOL." "Y is more accurate."</p>
<p>Q:List of people who were born in Agatha Christie's death place; who graduated from Oxford ? X <i>ours</i>, Y <i>LMRQ</i></p> <p>Comments: "X fully answers the question. Y features random people's death places." "y doesn't cover topic."</p>
<p>TIE</p> <p>Q:List of award-wining authors ? X <i>LMRQ</i>, Y <i>ours</i></p> <p>Comments: "Both fully answer the question." "lists are identical."</p>
<p>Q:List of people who graduated from Harvard University and their nationalities ? X <i>LMRQ</i>, Y <i>ours</i></p> <p>Comments: "X and Y both include people who graduated from Harvard University and their nationalities. So X and Y are the same." "same content."</p>
<p>LOST</p> <p>No Loss.</p>

Table 3.9: Ours vs. LMRQ explanations samples

However, we improve our search engine later on to support query diversification, which they do not have.

Chapter 4

Diversity

It is often the case that the top-ranked results are homogeneous. First, we discuss the problem of result diversity in Section 4.1. We then discuss the Maximal Marginal Relevance (MMR) approach in Section 4.2. In Section 4.3, we define different notions for result diversity in the setting of RDF. In Section 4.4, we develop an approach for result diversity based on the MMR. Finally, in Section 4.5, we develop a diversity-aware evaluation metric based on the Discounted Cumulative Gain [31] and use it, in Section 4.7, to evaluate our 100 queries benchmark over DBpedia.

4.1 Diversity Problem

Although result ranking improves the user satisfaction, it is often the case that the top-ranked results are dominated by one aspect of the query. This is a common problem in IR in general [24]. For example, consider the query `<?p01 party Democratic_Party_(United_States); ?event commander ?p01>` which requires the name of a Democratic politician who had a role in a military event or crisis. As can be seen from Table 4.1, a large number of the top-10 results are events by the same commander, namely Barack Obama.

The goal of a diversity-aware ranking model is to produce an ordering 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 is an optimization problem where the objective is to produce an ordering that maximizes both the relevance of the top- k results and their diversity. This objective function is hard to solve. This is why most approaches address it as an approximation problem. The problem is known as the top- k set selection problem [32]. It can be formulated as follows: 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 $S \subseteq U$ with respect to Q and let DIV be a function that measures the diversity of a subset of results $S \subseteq U$. Finally, let f be a function that combines both relevance and diversity. The top- k

set selection problem can be solved by finding S^* in Equation 4.1.

$$S^* = \operatorname{argmax}_{S \subseteq U} f(Q, S, REL, DIV), \text{ such that } |S^*| = k. \quad (4.1)$$

Subject (S)	Predicate (P)	Object (O)
Jefferson_Davis American_Civil_War	party commander	Democratic_Party_(United_States) Jefferson_Davis
Barack_Obama Operation_Odyssey_Dawn	party commander	Democratic_Party_(United_States) Barack_Obama
Barack_Obama New_York_Air_National_Guard	party commander	Democratic_Party_(United_States) Barack_Obama
Barack_Obama Texas_Air_National_Guard	party commander	Democratic_Party_(United_States) Barack_Obama
Barack_Obama Ohio_Air_National_Guard	party commander	Democratic_Party_(United_States) Barack_Obama
Barack_Obama Arkansas_Army_National_Guard	party commander	Democratic_Party_(United_States) Barack_Obama
Barack_Obama Georgia_Air_National_Guard	party commander	Democratic_Party_(United_States) Barack_Obama
Barack_Obama Alabama_Air_National_Guard	party commander	Democratic_Party_(United_States) Barack_Obama
Barack_Obama Michigan_Air_National_Guard	party commander	Democratic_Party_(United_States) Barack_Obama
Barack_Obama Massachusetts_Air_National_Guard	party commander	Democratic_Party_(United_States) Barack_Obama

Table 4.1: Top-10 ranked results of: Democratic politicians who had roles in military events or crises.

In order to solve this optimization problem, one must clearly specify both the relevance function REL and diversity function DIV , as well as how to combine them. Gollapudi and Sharma [32] proposed a set of axioms to guide the choice of the objective function $f(Q, S, REL, DIV)$ and they showed that for most natural choices of the 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 shown in Equation 4.2.

$$f(Q, S, REL, DIV) = (k - 1) \sum_{r \in S} rel(r, Q) + 2\lambda \sum_{r, r' \in S} d(r, r') \quad (4.2)$$

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

The objective function, Equation 4.2, clearly trades off both relevance of results in the top- k set with their diversity (as measured by their average distance).

Solving such objective function is again NP-hard, however there exists known approximation algorithms to solve the problem that mostly rely on greedy heuristics [32].

4.2 Maximal Marginal Relevance

Carbonell and Goldstein introduced the Maximal Marginal Relevance (MMR) method [24] which they use to re-rank a set of pre-retrieved documents U given a query Q .

Given a query Q , a set of results U and a subset $S \subset U$, the marginal relevance of a result $r \in U \setminus S$ is defined in Equation 4.3.

$$MR(r, Q, S) = \lambda rel(r, Q) + (1 - \lambda) \min_{r' \in S} d(r, r') \quad (4.3)$$

where $rel(r, Q)$ is a measure of how relevant r is to Q , $d(r, r')$ is a symmetric distance measure between r and r' 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 S , the marginal relevance of a result r is a measure of how much do we gain in terms of both relevance and diversity by adding the result r to the selected set S . To measure how much the result r would contribute to the relevance aspect of S , it is straight forward and we can use the result's relevance to Q . On the other hand, measuring how much result r would contribute to the diversity of S is more involved. The most natural thing to do is to compare r with all the results $r' \in S$ and compute a similarity (or rather dissimilarity) between r and every other result $r' \in S$ and then aggregate these similarities over all the results in S . We do exactly this by assuming there is a distance function that can measure how result r is different from any other result r' and then we use the minimum of the distances of r from all the results $r' \in S$ as a measure of the overall contribution of result r to the diversity of set S . By maximizing this minimum over a set of results $r \notin S$, we can find the result that when added to S would render it most diverse as compared to any other result.

Given all these considerations, we set the relevance $rel(r, Q)$ to the score of the result r obtained from our search engine.

4.3 Diversity Notions in RDF Setting

We propose three different notions of diversity and we explain how we build a subgraph representation that allows us to achieve each such notion.

4.3.1 Resource-based Diversity

In this notion of diversity, the goal is to diversify the different resources (i.e., entities and relations) that appear in the results. This ensures that no one resource will dominate the result set. Recall our example query asking for the name of a Democratic politician who had a role in a military event or crisis. Table 4.1 shows the top-10 subgraphs retrieved for the query using our ranking model.

In order to diversify the top-k results of a certain query, we define a language model for each result as follows:

Resource-based Language Model: the resource-based language model of result r is a probability distribution over all resources in the DBpedia.

The parameters of the result language model are estimated using a smoothed maximum likelihood estimator as shown in Equation 4.4.

$$P(w|r) = \alpha \frac{c(w;r)}{|r|} + (1 - \alpha) \frac{1}{|Col|} \quad (4.4)$$

where w is a resource, Col is the set of all unique resources in DBpedia, $c(w;r)$ is the number of times resource w occurs in r , $|r|$ is the number of times all resources occur in r , and $|Col|$ is the number of unique resources in DBpedia. Finally, α is the smoothing parameter.

For example, consider the subgraph:

```
<Jefferson_Davis party Democratic_Party_(United_States);  
    American_Civil_War commander Jefferson_Davis>
```

from Table 4.1. The resource-based language model of this subgraph is:

$P(Jefferson_Davis)=0.266$, $P(party)=0.133$, $P(commander)=0.133$,
 $P(American_Civil_War)=0.133$ and $P(Democratic_Party_(United_States))=0.133$.,
with $|Col| = 5771699$, and $\alpha=0.8$.

4.3.2 Term-based Diversity

In this notion of diversity, we are only interested in diversifying the results in terms of the variable bindings. To be able to do this, we define a language model for each result as follows.

Term-based Language Model: the term-based language model of result r is a probability distribution over all terms (unigrams) in DBpedia.

The parameters of the result language model are estimated using a smoothed maximum likelihood estimator as shown in Equation 4.5.

$$P(w|r) = \alpha \frac{c(w;r)}{|r|} + (1 - \alpha) \frac{1}{|Col|} \quad (4.5)$$

such that $w \notin QTerms$, where w is a term, $QTerms$ is the list of the terms in the query, Col is the set of all unique terms in DBpedia, $c(w; r)$ is the number of times term w occurs in r , $|r|$ is the number of times all terms occur in r , and $|Col|$ is the number of unique terms in DBpedia. Finally, α is the smoothing parameter.

This notion of diversity is particularly important in the case of keyword-augmented queries and when performing query relaxation. By excluding terms that appear in the original query when representing each result, we ensure that when these representations are later used for diversity, the top-ranked results still stay close to the original user query.

For example, the query `<?m starring ?x [Woody Allen]; ?m musicComposer ?y>`. The term-based language model for the subgraph:

```
<Crimes_and_Misdemeanors starring Woody.Allen;
Crimes_and_Misdemeanors musicComposer Franz_Schubert>
```

is as follows:

$P(crimes)=0.4$, $P(misdemeanors)=0.4$, $P(franz)=0.2$ and $P(schubert)=0.2$, with $|Col| = 2424955$, and $\alpha=0.8$. As the term-based language model does not include any of the terms in the query, this will ensure that we are only diversifying the results in terms of variable bindings. The relations (predicates) are also considered terms in this notion of diversity.

4.3.3 Text-based Diversity

In our search engine, each triple is also associated with a text snippet which can be used to process keyword-augmented queries. A text snippet can be directly utilized to provide diversity among the different results using the MMR measure. We define a language model for each result as follows.

Text-based Diversity: the text-based language model of a result r is a probability distribution over all the keywords in all the text snippets of all the triples in DBpedia. The parameters of the text-based language model is computed using a smoothed maximum-likelihood estimator as shown in Equation 4.6.

$$P(w|r) = \alpha \frac{c(w; D(r))}{|D(r)|} + (1 - \alpha) \frac{1}{|Col|} \quad (4.6)$$

where $c(w; D(r))$ is the number of times keyword w occurs in $D(r)$ (the text snippet of subgraph r), $|D(r)|$ is the number of occurrences of all keywords in $D(r)$, and $|Col|$ is the number of unique keywords in the text snippets of all triples in DBpedia. Finally, α is the smoothing parameter.

For example, consider the following subgraph:

```
<Sweet_and_Lowdown writer Woody.Allen;
Sweet_and_Lowdown distributor Sony_Pictures>
```


with the following two sets of keywords associated with triple 1 and triple 2, respectively:

$[need, story, standup, 2004, samantha, playwright, oscar]$ and $[story, need, fox]$. The keywords in this subgraph will then be the union of the keywords of the two triples in the subgraph as follows: $need, story, standup, 2004, samantha, playwright, oscar, fox$. Consequently, the text-based language model of this subgraph is as follows: $P(need)=0.123, P(story)=0.123, P(standup)=0.061, P(2004)=0.061, P(playwright)=0.061, P(oscar)=0.061$ and $P(fox)=0.061$, with $\alpha = 0.8$ and $|Col| = 2937745$.

4.4 Diversity-aware Re-ranking Algorithm

We explain how the marginal relevance can be used to provide a diverse-aware ranking of results given a query Q . Let U be the set of ranked results using any regular ranking model (i.e., that depends only on relevance without taking into consideration diversity). The algorithm to re-rank the results works as follows:

Maximal Marginal Relevance Re-Ranking Algorithm

1. Initialize the top- k set S with the highest ranked result $r \in U$
2. Iterate over all the results $r \in U \setminus S$, and pick the result r^* with the maximum marginal relevance $MR(r^*, Q, S)$. That is,

$$r^* = \operatorname{argmax}_{r \in U \setminus S} [\lambda \operatorname{rel}(r, Q) + (1 - \lambda) \min_{r' \in S} d(r, r')] \quad (4.7)$$

3. Add r^* to S
4. If $|S| = k$ or $S = U$ return S otherwise repeat steps 2, 3 and 4

The distance between two results r and r' is computed in Equation 4.8.

$$d(r, r') = \sqrt{JS(r||r')} \quad (4.8)$$

where $JS(r||r')$ is the Jensen-Shannon Divergence [33] between the language models of results r and r' and is computed as shown in Equation 4.9.

$$\begin{aligned} JS(r||r') &= \frac{KL(r||M) + KL(r'||M)}{2} \\ &= \frac{\sum_i^{terms} r(i) \log \frac{r(i)}{M(i)} + r'(i) \log \frac{r'(i)}{M(i)}}{2} \end{aligned} \quad (4.9)$$

where $KL(x||y)$ is the Kullback-Leibler Divergence between two language models x and y , and $M = \frac{1}{2}(r + r')$ is the average of the language models of r and r' .

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 measure.

4.5 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 [34, 35, 36]. We adopt a similar strategy and propose a novel evaluation metric that takes into consideration both aspects we are concerned with here, namely relevance and diversity, to evaluate a result set for a given query.

We introduce an adjustment to the Discounted Cumulative Gain (DCG) [31] 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 in Equation 4.10.

$$DCG_p = rel_1 + nov_1 + \sum_{i=2}^p \left(\frac{rel_i}{\log_2(i)} + nov_i \right) \quad (4.10)$$

where rel_i is the relevance score of the result at position i and nov_i is its novelty.

4.5.1 Resource-based Novelty

Concerning our first two diversity notions: the resource-based and term-based, the novelty of a result at position i can be computed as follows:

$$nov_i = \frac{\#unseen_i}{\#variables} \quad (4.11)$$

where $\#unseen_i$ is the number of resources that are bound to variables in result at position i that have not yet been seen, and $\#variables$ is the total number of variables in the query. Our goal is to diversify the results with respect to the variable bindings.

4.5.2 Text-based Novelty

The computation of the text-based novelty metric is very similar in spirit to the resource-based one. The only difference is that in the case of text-based diversity,

our goal is to diversify the results with respect to their text snippets. 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, the text-based novelty can be computed as follows:

$$nov_i = \frac{|keywords_i \setminus (keywords_i \cap (\cup_{j=1}^{i-1} keywords_j))|}{|keywords_i|} \quad (4.12)$$

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 $keywords_i$.

4.6 Related Work

Result diversity for document retrieval has gained much attention in recent years. The work in this area deals primarily with unstructured and semi-structured data [37, 24, 38, 35, 32, 36]. Most of the techniques perform diversification by optimizing a bi-criteria objective function that takes into consideration both result relevance as well as result novelty with respect to other results. Gollapudi and Sharma [32] presented an axiomatic framework for this problem and studied various objective functions that can be used to define such optimization problem. They proved that in most cases, such problem is hard to solve and proposed several approximation algorithms to solve such problem. Carbonell and Goldstein introduced the Maximal Marginal Relevance (MMR) method [24] which is one approximation solution to such optimization problem. Zhai et al. [36] studied a similar approach within the framework of language models and derived an MMR-based loss function that can be used to perform diversity-aware ranking. Aragwal et al. [37] assumed that query results belong to different categories and they proposed an objective function that tries to trade off the relevance of the results with the number of categories covered by the selected results.

In an RDF setting, where results are constructed at query time by joining triples, we do not have an explicit notion of result categories. We thus adopted the Maximal Marginal Relevance approach [24] to the setting of RDF data since it directly utilizes the results to perform diversity rather than explicitly taking the categories of the results into consideration.

Apart from document retrieval, there is very little work on result diversity for queries over structured data. In [39] the authors propose to navigate SQL results through categorization, which takes into account user preferences. In [40], the authors introduce a pre-indexing approach for efficient diversification of query results on relational databases. However, they do not take into consideration the relevance of the results to the query.

4.7 Experimental Evaluation

4.7.1 Evaluation Using Diversity-aware Metric

4.7.1.1 Setup

In order to evaluate our greedy diversity-aware re-ranking algorithm, we use a benchmark of 100 queries which we constructed over DBPedia, and which can be broadly divided into 4 categories: structured, keyword-augmented, requiring relaxation, and keyword-augmented and requiring relaxation. Our query benchmark was used in order to tune the weighting parameter of MMR (λ in Equation 4.7), that is used to trade-off relevance and diversity. It was also used to compare the various notions of diversity and to evaluate the effectiveness of diversity versus no diversity based on our diversity-aware evaluation metric $DIV - DCG$ described in Section 4.5.

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 4.5, we needed to gather a relevance assessment for each result to be able to compute rel_i . To gather these relevance assessments, we relied on the crowdsourcing platform CrowdFlower [30] 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 4.7. 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 [30].

We prepared a set of guidelines for each query category. We explained the idea of a structured result to the contributors. The queries were represented in natural language. The answers were represented as a set of triples (i.e., subgraphs). The guidelines which we submitted to CrowdFlower [30] can be found in details in Appendix B.

4.7.1.2 Parameter Tuning

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 in the training set 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 4.2 shows the

Notion	$\lambda = 0.1$	$\lambda = 0.2$	$\lambda = 0.3$	$\lambda = 0.4$	$\lambda = 0.5$	$\lambda = 0.6$	$\lambda = 0.7$	$\lambda = 0.8$	$\lambda = 0.9$
Resource-based	0.798	0.789	0.780	0.769	0.762	0.758	0.751	0.740	0.727
Term-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 4.2: Average $DIV - NDCG_{10}$ of the training queries for different values of λ

Notion	Diversified Result Set	Non-diversified Result Set
Resource-based	0.80	0.74
Term-based	0.81	0.74
Text-based	0.91	0.68

Table 4.3: Average $DIV - NDCG_{10}$ of the test queries

average $DIV - NDCG_{10}$ for our three notions of diversity for different values of λ . The numbers reported in Table 4.2 is the average over 5 training repetitions using 5 different 80-queries training sets. This is a well known validation technique called the k-fold cross-validation [41]. In this case, the value of k is 5. 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 $DIV - NDCG_{10}$ possible.

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%. In addition, we computed the Fleiss Kappa Coefficient as another measure of agreement which is more reliable measure than that of CrowdFlower as it takes into consideration agreement by chance. We obtained a Kappa Coefficient of 40% which can be interpreted as "Fair Agreement" [42].

4.7.1.3 Comparison of Various Notions of Diversity

Given the optimal values of the parameter λ that were set based on the five 80 training queries in our benchmark, we computed the average $DIV - NDCG_{10}$ for the five 20 test queries for each notion of diversity, as well as for the cases when no diversity was employed. The summary of our findings over the 20 queries is shown in Table 4.3. 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.

4.7.1.4 Qualitative Results

We show some qualitative results that highlight the importance of result diversity for RDF search and the difference between the various notions of diversity we discussed here.

Resource-based Diversity. Consider the query `<?film1 director ?x; ?film2 starring ?y; ?x spouse ?y>` whose corresponding information need is "Give me the name of a director whose partner is an actor (or actress) and the name of a movie for each". Table 4.4 shows the top-5 results with and without diversification, with λ set to 0.1. The table shows that without diversification, the results are dominated by two resources, namely `Madonna` and `Sean_Penn` with different combinations of movies they acted in or directed, but when diversification is involved, we get a set of different director-actor pairs as shown in the second column of Table 4.4.

No Diversity	Resource-based Diversity
W.E. director Madonna Mystic_River starring Sean_Penn Madonna spouse Sean_Penn	W.E. director Madonna Mystic_River starring Sean_Penn Madonna spouse Sean_Penn
Secretprojectrevolution director Madonna Mystic_River starring Sean_Penn Madonna spouse Sean_Penn	Citizen_Kane director Orson_Welles Cover_Girl starring Rita_Hayworth Orson_Welles spouse Rita_Hayworth
W.E. director Madonna The_Tree_of_Life starring Sean_Penn Madonna spouse Sean_Penn	Henry_V director Laurence_Olivier Ship_of_Fools starring Vivien_Leigh Laurence_Olivier spouse Vivien_Leigh
W.E. director Madonna Dead_Man_Walking starring Sean_Penn Madonna spouse Sean_Penn	That_Thing_You_Do! director Tom_Hanks Jingle_All_the_Way starring Rita_Wilson Tom_Hanks spouse Rita_Wilson
W.E. director Madonna This_Must_Be_the_Place starring Sean_Penn Madonna spouse Sean_Penn	Then_She_Found_Me director Helen_Hunt The_Simpsons_Movie starring Hank_Azaria Helen_Hunt spouse Hank_Azaria

Table 4.4: Top-5 results: director-actor couple query with and without diversity

Term-based Diversity. Consider one of the test queries: `<?m director ?x [Disney]; ?m starring ?y>`. The information need is: "Give me the name of a movie, its director, and its star, preferably a *Disney* movie". The top-5 results with the diversity parameter λ set to 0.1 are shown in Table 4.5. In the non-diversified set of results, we have one popular *Disney* movie, `Aladdin`, that is repeated five times. On the other hand, the diversified set of results contains five different *Disney* movies. Note that, if we had used the resource-based diversity notion to diversify the results of this query, we would have ended up with 5 different movies which would not have been necessarily *Disney* movies. The

merit of using the term-based diversity notion is that it diversifies the result set with respect to terms rather than resources, and excludes the terms that appear in the query when considering diversity.

No Diversity	Term-based Diversity
Aladdin_(1992_Disney_film) director John_Musker Aladdin_(1992_Disney_film) starring Robin_Williams	Aladdin_(1992_Disney_film) director John_Musker Aladdin_(1992_Disney_film) starring Robin_Williams
Aladdin_(1992_Disney_film) director Ron_Clements Aladdin_(1992_Disney_film) starring Robin_Williams	102_Dalmatians director Kevin_Lima 102_Dalmatians starring Glenn_Close
Aladdin_(1992_Disney_film) director John_Musker Aladdin_(1992_Disney_film) starring Frank_Welker	Cars_2 director John_Lasseter Cars_2 starring Michael_Caine
Aladdin_(1992_Disney_film) director Ron_Clements Aladdin_(1992_Disney_film) starring Frank_Welker	Leroy_&_Stitch director Tony_Craig Leroy_&_Stitch starring Tara_Strong
Aladdin_(1992_Disney_film) director John_Musker Aladdin_(1992_Disney_film) starring Gilbert_Gottfried	Snow_White_and_the_Seven_Dwarfs director Wilfred_Jackson Snow_White_and_the_Seven_Dwarfs starring Pinto_Colvig

Table 4.5: Top-5 results for the Disney query with and without diversity

Text-based Diversity. Consider one of the test queries: `<?m distributor Columbia_Pictures>`. The corresponding information need is: "Give me the name of a film distributed by the Columbia Pictures company". The top-5 results for both no diversity, and text-based diversity with the diversity parameter λ set to 0.1 are shown in Table 4.6. While the non-diversified set contains different movies, these movies in fact are not very diverse. Unlike the non-diversified set, the diversified set of results contains movies that have totally different genres, locations, casts, and plots. This indirect level of diversity cannot be captured using our first two diversity notions.

No Diversity	Text-based Diversity
Spider-Man_2 distributor Columbia_Pictures	Spider-Man_2 distributor Columbia_Pictures
Close_Encounters distributor Columbia_Pictures	Sharkboy_&_Lavagirl distributor Columbia_Pictures
A_Clockwork_Orange distributor Columbia_Pictures	Jungle_Menace distributor Columbia_Pictures
Spider-Man_3 distributor Columbia_Pictures	Quantum_of_Solace distributor Columbia_Pictures
Lawrence_of_Arabia distributor Columbia_Pictures	The_Da_Vinci_Code distributor Columbia_Pictures

Table 4.6: Top-5 results for the Columbia Pictures query with and without diversity

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 these to 0.8 to focus mainly on the effect of the weighting parameter λ in our experiments.

4.7.2 Evaluation Using Crowdsourcing

4.7.2.1 Setup

We also evaluate our diversity algorithm using crowdsourcing [30]. We follow the same technique in setting up the guidelines as in Section 3.4. The assessor is

Sleepers_(film) starring Kevin_Bacon	Sleepers_(film)
Sleepers_(film) starring Robert_De_Niro	Sleepers_(film)
Arthur_and_the_Invisibles starring Robert_De_Niro	Arthur_and_th
Arthur_and_the_Invisibles starring Madonna_(entertainer)	Arthur_and_th
<small> sleeper 1996 american legal crime drama film written produc direct barri levinson base lorenzo carcaterra s 1995 novel name star jason patric brad pitt robert de niro dustin hoffman vittorio gassmann kevin bacon among other norwood born juli 8 1958 actor musician whose includ music footloos 1984 controversi histor Interview with the Vampire: The Vampire Chronicles starring Tom Cruise Arthur and th</small>	

Figure 4.1: A subgraph and its description

asked to choose between two sets, one is diversified, and the other one is ranked using our ranking model (i.e based on relevance only). The assessor does not know which set is the diversified set. The positions of the sets are randomly set for each query. She is also asked to provide a valid explanation of her choice. We do this test for all three diversity notions, over our 100-query benchmark. The assessors are also asked to pass a test with at least a 90% score to be able to continue assessing. Moreover, a set of gold queries are embedded in the benchmark to catch and exclude any bad behavior or random answer.

4.7.2.2 Results

The results of this evaluation are shown in Table 4.7. The reported numbers are out of 100 queries.

Non-diversified vs.:	Resource-based	Term-based	Text-based
Won	52	57	38
Tie	43	38	57
Lost	5	5	5

Table 4.7: Results for 100 evaluation queries: diversity algorithm

For both the resource-based notion and the term-based notion, we outperform the non-diversified model by 9%. Unlike the text-based notion, the diversity for these two notions is clear and straight forward. The assessor can see the differences between the sets by just looking at the answers. Some sample comments are shown in Tables 4.8 and 4.9 . The assessors who preferred the diversified sets noticed that the reason is they have more than one flavor in them.

For the third notion, the text-based notion, the diversity was hidden. To help them in comparing the sets, we provided a set of keywords for each answer. As shown in Figure 4.1, by mousing over the answer, they can see its description. 57% of the diversified sets were considered as good as the non-diversified ones. 38% of the diversified sets were considered better than the non-diversified ones. Some explanations are shown in Table 4.10.

The inter-rater agreement produced by CrowdFlower was 77%, 71%, and 71%, for comparing with resource-based notion, term-based notion, and text-based notion respectively.

Non-diversified vs. Diversity(Resource-based)
<p>WON Q:List of movies for which the directors and the actors have won awards ? X <i>Non-Diversified</i>, Y <i>Diversified Notion 1</i> Comments: "more actors and movies in Y." "x all orson welles."</p>
<p>Q:List of people who died in WWII and their resting places ? X <i>Diversified Notion 1</i>, Y <i>Non-Diversified</i> Comments: "x various y repeat." "X looks better."</p>
<p>TIE Q:List of writers who had worked with D.W. Griffith ? X <i>Diversified Notion 1</i>, Y <i>Non-Diversified</i> Comments: "both X and Y contain the same number of writer who had worked with D.W. Griffith." "both answer the question right."</p>
<p>Q:List of publishers that published Stephen King books? X <i>Diversified Notion 1</i>, Y <i>Non-Diversified</i> Comments: "same information in both lists." "both right."</p>
<p>LOST Q:List of Bette Davis movies with her late co-stars ? X <i>Diversified Notion 1</i>, Y <i>Non-Diversified</i> Comments: "Y is better because it's more accurate." "X about something else."</p>
<p>Q:List of Brazilian actors ? X <i>Non-Diversified</i>, Y <i>Diversified Notion 1</i> Comments: "X and Y both answer the question with some precision. In Y they speak of an actor and of Brazilian films or news. Both include List of Brazilian actors, but in Y that only sees 1 actor: Meira. So X is better in this case." "X is better."</p>

Table 4.8: No Diversity vs. Resource-based Diversity

Non-diversified vs. Diversity(Term-based)
<p>WON Q:List of award-winning spielberg movies ? X <i>Non-Diversified</i>, Y <i>Diversified Notion 2</i> Comments: "y - less duplicates." "y better than x."</p>
<p>Q:List of award-winning cartoonists ? X <i>Non-Diversified</i>, Y <i>Diversified Notion 2</i> "y has more various results." "Y list more varied."</p>
<p>TIE Q:List of people influenced by Freud ? X <i>Non-Diversified</i>, Y <i>Diversified Notion 2</i> "Both are same list." "Both answer the question right."</p>
<p>Q:List of art directors ? X <i>Non-Diversified</i>, Y <i>Diversified Notion 2</i> Comments: "Both have varied lists." "They are the same."</p>
<p>LOST Q:List of Woody Allen movies that Dick Hyman worked on ? X <i>Diversified Notion 2</i>, Y <i>Non-Diversified</i> "X no Dick." "Y better."</p>
<p>Q:List of producers who had worked with Frank Sinatra ? X <i>Non-Diversified</i>, Y <i>Diversified Notion 2</i> "x is all about frank sinatra but y not." "x gives better answer."</p>

Table 4.9: No Diversity vs. Term-based Diversity

Non-diversified vs. Diversity(Text-based)
<p>WON Q:List of classical music composers ? X <i>Diversified Notion 3</i>, Y <i>Non-Diversified</i> Comments: "X is diversified." "Y all about Mozart."</p>
<p>Q:List of authors who are also actors in drama movies ? X <i>Diversified Notion 3</i>, Y <i>Non-Diversified</i> Comments: "There is more variety in X list" "X has a variety of actors"</p>
<p>TIE Q:List of people who worked with Kevin Bacon ? X <i>Diversified Notion 3</i>, Y <i>Non-Diversified</i> Comments: "X and Y answer the question accurately. X and Y are all varied and diverse and responds with precision. For this reason, the sets X and Y are equal and precise. Therefore, X and Y are the same." "all same."</p>
<p>Q:List of people influenced by Albert Einstein ? X <i>Non-Diversified</i>, Y <i>Diversified Notion 3</i> Comments: "Both X & Y are same lists just in different order." "Both have varied names."</p>
<p>LOST Q:List of rock music singers from Ottawa ? X <i>Diversified Notion 3</i>, Y <i>Non-Diversified</i> Comments: "y has more relevant answers." "y relevant."</p>
<p>Q:List of Bette Davis movies with her late co-stars ? X <i>Non-Diversified</i>, Y <i>Diversified Notion 3</i> Comments: "In the list Y there are films in which they do not star by Bette Davis" "Y is the correct one."</p>

Table 4.10: No Diversity vs. Text-based Diversity

Chapter 5

Efficiency

In this chapter, we discuss the running time of our algorithms. More particularly, the search engine can be divided into two parts. The first part begins with the user entering her query and ends with a list of top-10 subgraphs ranked by relevance only. The second part starts with the full ranked by relevance set and ends by producing a list of top-10 subgraphs ranked by both diversity and relevance.

5.1 Database Statistics

Ranking a query results requires a visit to several PostgreSQL indexed tables. The tables titles, columns, descriptions, examples, and sizes are shown in Table 5.1.

An example of the "Facts" table entry is:

```
<Annie_Hall,starring,Woody_Allen,2032,1141107>.
```

An example of the "Keywords" table entry is:

```
<1141107: standup: 1, playwright: 119, role: 164, humor: 39...>.
```

An example of the "op_var" table entry is:

```
<A.C._Trumbo_House, 476, histor=76, percent=3, 2010=53, place=124>.
```

An example of the "sp_var" table entry is:

```
<King_George_VI_Chase,1364,mellor=15, hors=1289, 2006=20, bright=3>.
```

An example of the "so_var" table entry is:

```
<starring,26913826, tokiwa=111, paramita=2, agimat=225, fantasporto=47>.
```

An example of the "s_var" table entry is:

```
<birthPlace,North_Elmham,212,govern=24, singl=4, lynn=93, novemb=1>.
```

An example of the "p_var" table entry is:

```
<Rod_Carrillo,The_Cataracs,248, danc=116, california=126, project=18>.
```

An example of the "o_var" table entry is:

```
<Roxanne_Jones,termPeriod,4,jone=5, 19=1, die=1, state=8>.
```

Title	Columns	Description	Number of Rows
Facts	S, P, O, WC, ID	DBpedia triples	15.7 million
Keywords	ID, Keywords	Weighted keywords of a triple	15.7 million
op_var	S, \sum WC, \sum KWC	\sum of WCs and KWCs for all facts with S=x	2 million
sp_var	O, \sum WC, \sum KWC	\sum of WCs and KWCs for all facts with O=x	4.6 million
so_var	P, \sum WC, \sum KWC	\sum of WCs and KWCs for all facts with P=x	1337
s_var	P, O, \sum WC, \sum KWC	\sum of WCs and KWCs for all facts with P=x and O=y	5.4 million
p_var	S, O, \sum WC, \sum KWC	\sum of WCs and KWCs for all facts with S=x and O=y	15.4 million
o_var	S, P, \sum WC, \sum KWC	\sum of WCs and KWCs for all facts with S=x and P=y	12.6 million

Table 5.1: Database information

5.2 Running Time Statistics

The average running time for each query category is shown in Table 5.2.

Category	Average Running Time
Purely Structured	18 s
Keyword-augmented	83 s
Requiring Relaxation	115 s

Table 5.2: Running time per query category

The average ranking running time per query is 47 s, and the standard deviation is 0.2 s. The minimum running time per query is 0.093 s (93 ms), and the maximum running time per query is 431 s (7 minutes).

We follow this by reporting the average diversity running time for each of the three diversity notions in Table 5.3. Moreover, the average of the diversity running time for resource-based and term-based notions is 33 s.

Notion	Average Running Time
Resource-based	32.8 s
Term-based	33.1 s
Text-based	145.9 s

Table 5.3: Running time per diversity notion

We record the running time for each step in the process over the 100-query benchmark, and then we do the average to get the values in Figure 5.1 as percentages indicating the percentage of time each step needs. The percentage of time spent in processing the input, scoring the subgraphs, and re-scoring them is reasonable. However, 60% of the time is spent on running the SQL queries needed on PostgreSQL [9]. Another step that is slowing down the system is the 37% of the time spent on computing the language model for each subgraph in the result set.

In addition, we record the running time of the scoring loop of the keyword-augmented queries versus the scoring loop of the purely structured queries. The job for this type of queries is slightly more complicated since we need to take care of the keywords associated with the queries. On average, the running time

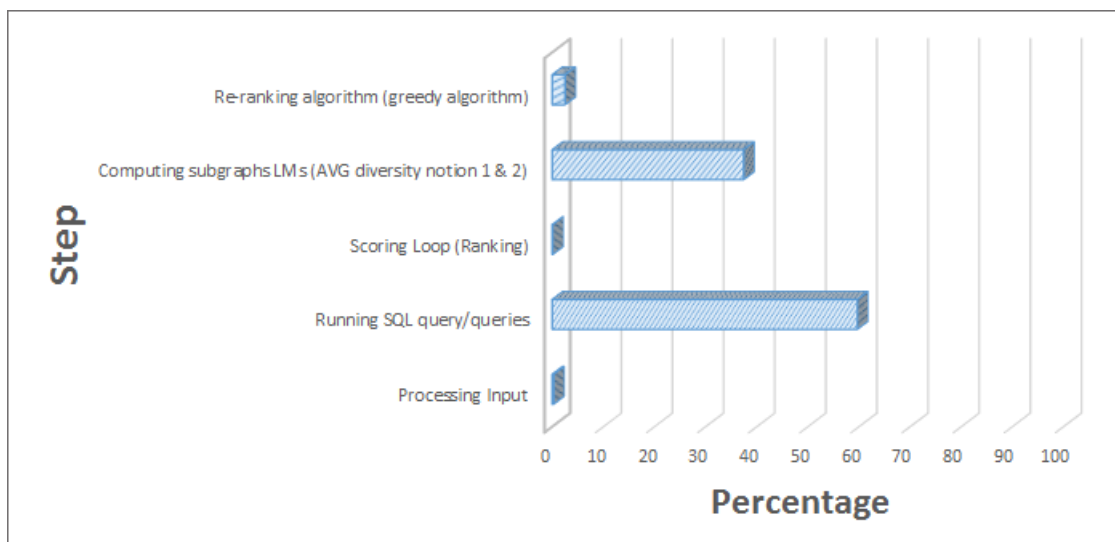


Figure 5.1: Percentage of time needed per step

needed for scoring a keyword-augmented set of subgraphs is 0.095 s. This value is greater than the needed time to score a purely structured set of subgraphs: 0.084s.

Finally, we observe 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.

5.3 Running Time Challenges

The main two issues in our system, in terms of execution time, are running SQL queries on PostgreSQL and computing the language models for all the subgraphs for the purpose of diversifying the results. For the first problem, we can consider implementing the system using a NoSQL database, MongoDB [43] for instance. However, this kind of databases has some difficulty in implementing self joins. The "\$lookup" command, which MongoDB introduced in their newer version, can be used to join collections. Another solution is to use a rank-aware join algorithm such as the one proposed in [44]. Unlike the traditional materialize-then-sort method for retrieving the results of a SPARQL query, the authors propose a rank-aware join algorithm optimized for native RDF stores. As for the second issue, one solution is to go for re-ranking a *subset* of the result set. So instead of computing the language models of n subgraphs, where n is the number of subgraphs returned by the SQL query, we can consider computing k language models for top- k subgraphs (ranked by relevance).

Appendix A

Guidelines: Choosing the Better Result Set

We have developed a set of guidelines for comparing two sets of top-10 subgraphs. This will help the assessor in understanding how a subgraph (one or more triples) can describe the answer of a natural language question.

Figure A.1 shows the guidelines for choosing between set X, set Y, or same. X and Y are anonymous sets, which means that our model in each query can be X or Y. This way the user would not stick on voting for the same side every time. In addition, the assessor is asked to provide a valid explanation of her choice. Figure A.2 shows a sample query. 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.

Given a question in *natural language* and two possible sets of results, **choose the set that answers the question better**. Each set contains a number of results, and **each result consists of one or more lines, where each line reveals a fact**. You are also asked to explain your choice.

A **valid explanation** is a one that clearly states the reason behind your choice. Note that very generic explanations can affect your job acceptance.

Examples:

- List of basketball players ?

X	Y
LeBron James type Basketball Player	Abdulwahab Al-Hamwi type Basketball Player
Michael Jordan type Basketball Player	Wang Zhizhi type Basketball Player
Shaquille O'Neal type Basketball Player	Toon van Helfteren type Basketball Player

X and Y both answer the question by listing names of players. However, X is better because the names are more popular.

- List of politicians, Alma maters, and their birth years ?

X	Y
Ellen DeGeneres almaMater University of New Orleans	Bill Clinton almaMater Yale University
Ellen DeGeneres birthYear 1958	Bill Clinton birthYear 1946
Melinda Gates almaMater Duke University	Che Guevara almaMater University of Buenos Aires
Melinda Gates birthYear 1964	Che Guevara birthYear 1928
Bill Lawrence almaMater University of William& Mary	Bernie Sanders almaMater University of Chicago
Bill Lawrence birthYear 1968	Bernie Sanders birthYear 1941

Y is better because it's more accurate. It includes politicians. X partially answered the question.

- List of books made into movies ?

X	Y
Meshwar lehad el hait author Omar Taher	Eat, Pray, Love author Elizabeth Gilbert
Aarafouh belhozn author Omar Taher	Eat, Pray, Love director Ryan Murphy
Brigada Ligeira author Antonio Candido	The Devil Wears Prada author Lauren Weisberger
Na Sala de Aula author Antonio Candido	The Devil Wears Prada director David Frankel
Yawarakana hoho author Natsuo Kirino	Into the Wild author Jon Krakauer
Metabora author Natsuo Kirino	Into the Wild director Sean Penn

Y is better. X is irrelevant, it does not include any books that were also movies.

- List of rivers in Africa ?

X	Y
Nile length 6853km	SenegalRiver length 1086km
CongoRiver length 4700km	Nile length 6853km
Senegal_River length 1086km	Zambezi length 2574km

X and Y both answer the question, and both include popular rivers in Africa. So X and Y are the same.

Figure A.1: Which set is better?

Question: List of stars of NBC shows ?

X	Y
H.R._Pufnstuf network NBC	The_Singing_Bee_(U.S._game_show) network NBC
H.R._Pufnstuf starring Sharon_Baird	The_Singing_Bee_(U.S._game_show) starring NBC
The_Bill_Dana_Show channel NBC	The_Singing_Bee_(U.S._game_show) format NBC
The_Bill_Dana_Show starring Bill_Dana_(comedian)	The_Singing_Bee_(U.S._game_show) starring NBC
The_Rerun_Show channel NBC	The_Singing_Bee_(U.S._game_show) starring NBC
The_Rerun_Show starring Candy_Ford	The_Singing_Bee_(U.S._game_show) starring NBC
Awake_(TV_series) network NBC	The_Singing_Bee_(U.S._game_show) presenter NBC
Awake_(TV_series) starring Dylan_Minnette	The_Singing_Bee_(U.S._game_show) starring NBC
Mister_Roberts_(TV_series) channel NBC	The_Dick_Powell_Show network NBC
Mister_Roberts_(TV_series) starring Richard_X_Slattery	The_Dick_Powell_Show starring Frank_Sinatra
Kristin_(TV_series) network NBC	A_Man_and_His_Music_+_Ella_+_Jobim network NBC
Kristin_(TV_series) starring Kristin_Chenoweth	A_Man_and_His_Music_+_Ella_+_Jobim starring Frank_Sinatra
NewsRadio network NBC	Law_&_Order network NBC
NewsRadio starring Jon_Lovitz	Law_&_Order starring George_Dzundza
No_One_Would_Tell distributor NBC	Seinfeld network NBC
No_One_Would_Tell starring Fred_Savage	Seinfeld starring Jerry_Seinfeld
The_Debbie_Reynolds_Show channel NBC	Friends network NBC
The_Debbie_Reynolds_Show starring Patricia_Smith_(actress)	Friends starring Jennifer_Aniston
Love;_Sidney network NBC	Friends distributor NBC
Love;_Sidney starring Tony_Randall	Friends starring Jennifer_Aniston

Which result set is a better answer for the question?

X
 Y
 same

Explain your choice:(You must write a valid explanation)

Enter text here

Figure A.2: One query and two sets of answers

Appendix B

Guidelines: Assessing a Subgraph

In Section 4.7, we have mentioned our 100 queries benchmark, which we submitted its results to CrowdFlower [30] for relevance assessments. Since our benchmark’s results are subgraphs, we needed to explain the concept of subgraphs and triples to the workers (judges). In order to accomplish this, we have developed a set of guidelines for each query category. This will help the judge in understanding how a subgraph (one or more triples) can describe the answer of a natural language question.

B.1 Purely Structured Queries Guidelines

Figure B.1 shows the guidelines for purely structured queries and Figure B.2 shows a sample query for this category. We had 17360 unique subgraphs (results) for 62 purely structured queries.

B.2 Keyword-augmented Queries Guidelines

Figure B.3 shows the guidelines for keyword-augmented structured queries and Figure B.4 shows a sample query for this category. We had 7280 unique subgraphs (results) for 26 keyword-augmented queries.

B.3 Requiring Relaxation Queries Guidelines

Figure B.5 shows the guidelines for queries that need relaxation and Figure B.6 shows a sample query for this category. We had 2240 unique subgraphs (results) for 8 requiring relaxation queries.

Given a query in natural language, and a possible result, assess how well the result answers the query on a two-level scale:

- **Popular result:** you are familiar with the names in the result.
 - **NOTE:** "You are familiar with the names" means you know them **WITHOUT** doing any web search.
- **Unpopular result:** you are not familiar with the names in the result.

- A result for a query consists of one or more lines and each line consists of three terms. Each line reveals a certain fact.
- For example, consider the query: Give me the name of a movie directed by one of its stars?
- The following result:
 - *AnnieHall* director *WoodyAllen*
 - *AnnieHall* starring *WoodyAllen*
- The first line states that Woody Allen directed the movie Annie Hall and the second line states that Woody Allen starred in Annie Hall. This is a "Popular result" because the director Woody Allen is a well-known director in the movie industry.

- On the other hand, the following result:
 - *Sins(film)* director *VinodPande*
 - *Sins(film)* starring *VinodPande*
- also provides the name of a movie that was directed by one of its stars, however, neither the film nor the director are famous in the movie industry. Hence, this is an "Unpopular result".

Figure B.1: Purely structured queries guidelines

Query:
Give me the name of a person who died or fought in WW2, and his/her resting place ?

Result:

George_Brett_(general) battle World_War_II
George_Brett_(general) restingPlace Winter_Park,Florida

Assess the quality of the result:

Popular result
 Unpopular result

Figure B.2: Sample assessment question (purely structured)

B.4 Keyword-augmented and Requiring Relaxation Queries Guidelines

Figure B.7 shows the guidelines for queries that need relaxation and Figure B.8 shows a sample query for this category. We had 1120 unique subgraphs (results) for 4 requiring relaxation queries.

B.5 Gold Queries

To ensure a high quality of the relevance assessments gathered and a careful understanding of the guidelines, we constructed a set of gold queries that were embedded within the benchmark queries. This enabled us to exclude assessments from untrusted workers.

Given a query in natural language, a related subquery, and a possible result, assess how well the result answers the query on a four-level scale:

- **Popular result and answers the subquery:** the given result perfectly answers the related subquery, and you are familiar with the names in the result.
- **Unpopular result but answers the subquery:** the given result perfectly answers the related subquery. However, you are not familiar with the names in the result.
- **Popular result but DOES NOT answer the subquery:** You are familiar with the names in the result. However, the result does not answer the related subquery.
- **Unpopular result and DOES NOT answer the subquery:** the given result neither answers the related subquery nor you are familiar with the names in the result.

NOTE: "You are familiar with the names" means you know them **WITHOUT** doing any web searching.

- A result for a query consists of one or more lines and each line consists of three terms. Each line reveals a certain fact.

For example, consider the following query and its subquery: **Give me the name of a movie directed by one of its stars? Preferably comedy movies?**

- One possible result for this query is:
 - `The_Adventurer` director `Charlie_Chaplin`
 - `The_Adventurer` starring `Charlie_Chaplin`
- The first line states that Charlie Chaplin directed the movie `The_Adventurer` and the second line states that he also starred in it. This is a **"Popular result and answers the subquery"** result, because 1) it answers the subquery as the `adventurer` is a comedy movie, and 2) the name of Charlie Chaplin is well known in the movie industry.
- An example of an **"An unpopular result but answers the subquery"** result is:
 - `A_Mix_Up_in_Hearts` starring `Oliver_Hardy`
 - `A_Mix_Up_in_Hearts` director `Oliver_Hardy`
- because 1) it is a silent comedy movie so answers the subquery. However, neither the movie nor the actor/director are famous.
- An example of a **"Popular result but DOES NOT answer the subquery"** result is:
 - `A_Bronx_Tale` director `Robert_De_Niro`
 - `A_Bronx_Tale` starring `Robert_De_Niro`
- because 1) the actor/director is well known in the movie business. However, the movie is an action-drama film and not a comedy movie so it does not answer the subquery.
- An example of an **"Unpopular result and DOES NOT answer the subquery"** result is:
 - `Sins_(film)` director `Vinod_Pande`
 - `Sins_(film)` starring `Vinod_Pande`
- because 1) the movie is not a comedy movie so it does not answer the subquery and 2) neither the movie nor star are well-known or famous.

Note that in order to be able to find more information about certain names mentioned in a result, you hover over these names and a brief summary will be provided.

Figure B.3: Keyword-augmented queries guidelines

An example of a gold answer (an obvious answer) is the answer to the following query:

"Give me the name of an operating system and its developer ?"
 <Windows_10 developer Microsoft>

This 1-triple subgraph answers the query and the contains popular names like "Windows" and "Microsoft".

On the other hand, the following answer is obviously irrelevant to the query:

"Give me the name of an Asian country and its capital ?"
 <Spain haspopulation 46,423,064>

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%.

Query:
Give me the name of a movie that was directed by one of its stars, and the death place and date of the director ? Preferably a director who does comedy and Broadway ?

Result:

The_Merchant_of_Venice_(unfinished_film) starring Orson_Welles
 The_Merchant_of_Venice_(unfinished_film) director Orson_Welles
 Orson_Welles deathPlace Los_Angeles
 Orson_Welles deathDate 1985-10-10 <http://www.w3.org/2003/01/19/rdf-syntax-ns#type> merchant | venice | frequently | cited | unfinished | film | directed | orson | welles | based | william | shakespeare | s | pla actually | completed | better | described | partially | lost | film | welles | played | role | shylock | addition | producing | dir | well | writing | adaptation | charles | gray | featured | antonio | irina | maleeva | jessica | cast | members | dorian | bond croushaw | mauro | bonnani | nina | palinkas | bonnani | professional | actor | editor | working | welles | s | don | quixote younger | sister | oja | kodar | whose | real | name | olga | palinkas | differing | sources | give | film | s | running | time | 31 minutes | welles | started | work | film | 1969 | originally | produced | part | abandoned | 90 | minute | television | special bag | made | cbs | later | year | project | close | completion | cbs | withdrew | funding | welles | long | running | disputes authorities | regarding | tax | status |...

Assess the quality of the result:

- Popular result and answers the subquery
- Unpopular result but answers the subquery
- Popular result but DOES NOT answer the subquery
- Unpopular result and DOES NOT answer the subquery

Figure B.4: Sample assessment question (keyword-augmented)

Given a query in natural language and a possible result, assess the quality of the result on a four-level scale:

- **Popular result and answers the query:** the given result perfectly answers the query and you are familiar with the names in the result.
- **Unpopular result but answers the query:** the given result answers the query. However, you are not familiar with the names in the result.
- **Partially answers the query:** the given result partially answers the query.
- **DOES NOT answer the query:** the given result does not answer the query.

NOTE: "You are familiar with the names" means you know them WITHOUT doing any web searching.

- A result for a query consists of one or more lines and each line consists of three terms. Each line reveals a certain fact. For example, consider the query: Give me the name of shows Kelsey Grammer starred in?
- The following is a possible result to the query:
 - [Frasier starring Kelsey_Grammer](#)
- This result states that Kelsey Grammer starred in the show Frasier. This is an "popular result and answers the query" result because 1) it gives the name of a show that Kelsey Grammer starred in as the query requested and 2) the show Frasier is a very popular and well-known sitcom.

- An example of an "Unpopular result but answers the query" result is:
 - [Roman's_Empire starring Kelsey_Grammer](#)
- because it gives you a show that Kelsey Grammer starred in as the query requested. However, the show is not popular and it was cancelled after five episodes.

- An example of a "Partially answers the query" result is:
 - [Girlfriends executiveProducer Kelsey_Grammer](#)
- because it gives you a show that Kelsey Grammer had worked on. However, he did not star in it as the query requires.

- An example of a "DOES NOT answer the query" result is:
 - [Camille_Grammer spouse Kelsey_Grammer](#)
- because it does not give you any show that Kelsey Grammer was involved in at all.

Figure B.5: Requiring relaxation queries guidelines

Query:
Give me the name of a movie starred by Woody Allen and the name of the movie's writer ?

Result:

New_York_Stories starring Woody_Allen
 New_York_Stories writer Woody_Allen

Assess the quality of the result:

- Popular result and answers the query
- Unpopular result but answers the query
- Partially answers the query
- DOES NOT answer the query

Figure B.6: Sample assessment question (requiring relaxation)

Given a query in natural language and a possible result, assess the quality of the result on a four-level scale:

- **Popular result that answers both the query and the subquery:** the given result perfectly answers both the query and the related subquery, and you are familiar with the names in the result.
- **Unpopular result that answers both the query and the subquery:** the given result answers both the query and the related subquery. However, you are not familiar with the names in the result.
- **A result that answers either the query OR the subquery:** the given result answers the query. However, it **does not** answer the related subquery. Or, it answers the related subquery. However, it **does not** answer the query.
- **A result that partially answers the query:** the given result **partially** answers the query.
- **A result that DOES NOT answer the query:** the given result **does not** answer the main query.

NOTE: "You are familiar with the names" means you know them WITHOUT doing any web searching.

- A result for a query consists of one or more lines and each line consists of three terms. Each line reveals a certain fact.

For example, consider the query: **Give me the name of an actor and his birth year? Preferably an American actor?**

- The following is a possible result to the query:
 - TomCruise occupation Actor
 - TomCruise birthYear 1962
- This result states that Tom Cruise is an actor that was born in 1962. This is a "Popular result that answers both the query and the subquery" result because 1) it gives the name of an actor and his birth year, 2) the actor is American so it answers the related subquery, and 3) Tom Cruise is a well-known actor
- An example of an "Unpopular result that answers both the query and the subquery" result is:
 - PatrickKerr occupation Actor
 - PatrickKerr birthYear 1956
- because it gives you the name of an American actor and his birth year. However, the actor is not popular.
- An example of a "A result that answers either the query OR the subquery" result is:
 - DanielRadcliffe occupation Actor
 - DanielRadcliffe birthYear 1989
- because it gives you the name of an actor, his birth year. However, the actor is English (not American). So the result does not answer the related subquery.
- An example of an "A result that partially answers the query" result is:
 - JaneLeeves starring Miracleon34th_Street
 - Seinfeld guest JaneLeeves
- because although it does give you the name of an actor, it does not give you her birth year. In addition, she is not an American actress.
- An example of an "A result that DOES NOT answer the query" result is:
 - LeBronJames occupation BasketballPlayer
 - LeBronJames birthplace Ohio
- Note that in order to be able to find more information about certain names mentioned in a result, you can hover over these names and a brief summary will be provided.

Figure B.7: Keyword-augmented and requiring relaxation queries guidelines

Query:
Give me the name of an educational institution? its location and the number of students in it? Preferably the institution is a law school?

Result:

Texas_Tech_University state Texas
Texas_Tech_University numberOfStudents 33111 <http://www.w3.org/2001/XMLSchema#nonNegativeInteger>

Assess the quality of the result:

- ◉ Popular result and answers the subquery
- ◉ Unpopular result but answers the subquery
- ◉ Popular result but DOES NOT answer the subquery
- ◉ Unpopular result and DOES NOT answer the subquery

texas | tech | university | often | referred | texas | tech | ttu | public | research | univer
established | february | 10 | 1923 | originally | known | texas | technological | college
system | seventh | largest | student | body | state | texas | school | texas | house | und
school | location | university | offers | degrees | 150 | courses | study | 13 | colleges |
tech | university | awarded | 200 | 000 | degrees | since | 1927 | including | 40 | 000 | g
foundation | classifies | texas | tech | high | research | activity | research | projects | a
computing | nanophotonics | atmospheric | sciences | wind | energy | among | pro

Figure B.8: Sample assessment question (keyword-augmented and requiring relaxation)

Appendix C

Abbreviations

KB	Knowledge-Base
KG	Knowledge-Graph
KWC	Keyword Witnesscount
LMRQ	Language-Model-based Ranking for Queries on RDF-graph
O	Object
P	Predicate
RDF	Resource Description Framework
S	Subject
WC	Witnesscount

Bibliography

- [1] S. Auer, C. Bizer, R. Cyganiak, G. Kobilarov, J. Lehmann, and Z. Ives, “Dbpedia: A nucleus for a web of open data,” in *ISWC/ASWC*, 2007.
- [2] “Freebase: A social database about things you know and love.” <http://www.freebase.com>.
- [3] F. M. Suchanek, G. Kasneci, and G. Weikum, “Yago: A large ontology from wikipedia and wordnet,” *J. Web Sem.*, vol. 6, no. 3, 2008.
- [4] P. DeRose, X. Chai, B. J. Gao, W. Shen, A. Doan, P. Bohannon, and X. Zhu, “Building community wikipedias: A machine-human partnership approach,” in *ICDE*, 2008.
- [5] Z. Nie, Y. Ma, S. Shi, J. Wen, and W. Ma, “Web object retrieval,” in *WWW*, 2007.
- [6] “W3c: Resource description framework (rdf).” www.w3.org/RDF/, 2004.
- [7] “W3c: Sparql query language for rdf.” www.w3.org/TR/rdf-sparql-query/, 2008.
- [8] S. Chaudhuri, G. Das, V. Hristidis, and G. Weikum, “Probabilistic information retrieval approach for ranking of database query results,” *SIGMOD Record*, vol. 35, no. 4, 2006.
- [9] “A powerful, open source object-relational database system..” <https://www.postgresql.org/>.
- [10] D. Petkova and W. Croft, “Hierarchical language models for expert finding in enterprise corpora,” *Int. J. on AI Tools*, vol. 17, no. 1, 2008.
- [11] H. Fang and C. Zhai, “Probabilistic models for expert finding,” in *ECIR*, 2007.
- [12] J. D. Lafferty and C. Zhai, “Document language models, query models, and risk minimization for information retrieval,” in *SIGIR*, 2001.

- [13] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan, “Keyword searching and browsing in databases using banks,” in *ICDE*, 2002.
- [14] V. Kacholia, S. Pandit, S. Chakrabarti, S. Sudarshan, R. Desai, and H. Karambelkar, “Bidirectional expansion for keyword search on graph databases,” in *VLDB*, 2005.
- [15] V. Hristidis, H. Hwang, and Y. Papakonstantinou, “Authority-based keyword search in databases,” *TODS*, vol. 33, no. 1, 2008.
- [16] K. Golenberg, B. Kimelfeld, and Y. Sagiv, “Keyword proximity search in complex data graphs,” in *SIGMOD*, 2008.
- [17] T. Cheng, X. Yan, and K. C.-C. Chang, “Entityrank: Searching entities directly and holistically,” in *VLDB*, 2007.
- [18] S. Elbassuoni and R. Blanco, “Keyword search over rdf graphs,” in *CIKM*, CIKM ’11, 2011.
- [19] G. Li, B. Ooi, J. Feng, J. Wang, and L. Zhou, “Ease: an effective 3-in-1 keyword search method for unstructured, semistructured and structured data,” in *SIGMOD*, 2008.
- [20] G. Kasneci, F. M. Suchanek, G. Ifrim, M. Ramanath, and G. Weikum, “Naga: Searching and ranking knowledge,” in *ICDE*, 2008.
- [21] M. Yahya, D. Barbosa, K. Berberich, Q. Wang, and G. Weikum, “Relationship queries on extended knowledge graphs,” in *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, WSDM ’16*, (New York, NY, USA), pp. 605–614, ACM, 2016.
- [22] S. Elbassuoni, M. Ramanath, R. Schenkel, M. Sydow, and G. Weikum, “Language-model-based ranking for queries on RDF-graphs,” in *CIKM*, 2009.
- [23] S. Elbassuoni, M. Ramanath, R. Schenkel, and G. Weikum, “Searching rdf graphs with SPARQL and keywords,” *IEEE Data Engineering Bulletin*, vol. 33, no. 1, 2010.
- [24] J. Carbonell and J. Goldstein, “The use of mmr, diversity-based reranking for reordering documents and producing summaries,” in *SIGIR*, 1998.
- [25] S. Elbassuoni, M. Ramanath, and G. Weikum, “Query relaxation for entity-relationship search,” in *Proc. of the 8th Extended Semantic Web Conference(ESWC)*, 2010.

- [26] H. Huang, C. Liu, and X. Zhou, “Computing relaxed answers on rdf databases,” in *WISE*, 2008.
- [27] G. Fokou, S. Jean, A. Hadjali, and 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*, (New York, NY, USA), pp. 237–252, Springer-Verlag New York, Inc., 2015.
- [28] G. Fokou, S. Jean, A. Hadjali, and M. Baron, *RDF Query Relaxation Strategies Based on Failure Causes*, pp. 439–454. Cham: Springer International Publishing, 2016.
- [29] G. Kasneci, F. Suchanek, G. Ifrim, S. Elbassuoni, M. Ramanath, and G. Weikum, “NAGA: Harvesting, Searching and Ranking Knowledge,” in *The 2008 ACM SIGMOD 2008 International Conference on Management of Data (SIGMOD)*, 2008.
- [30] “Crowd flower: A crowd sourcing company..” <https://www.crowdfunder.com/>.
- [31] K. Järvelin and J. Kekkonen, “Cumulated gain-based evaluation of ir techniques,” *ACM Transactions on Information Systems (TOIS)*, pp. 422–446, 2002.
- [32] S. Gollapudi and A. Sharma, “An axiomatic approach for result diversification,” in *Proceedings of the 18th international conference on World wide web, WWW '09*, (New York, NY, USA), pp. 381–390, ACM, 2009.
- [33] J. Lin., “Divergence measures based on the shannon entropy,” *IEEE Transactions on Information Theory*, pp. 145–151, 1991.
- [34] J. Allan, C. Wade, and A. Bolivar, “Retrieval and novelty detection at the sentence level,” in *SIGIR*, pp. 314–321, 2003.
- [35] C. L. Clarke, M. Kolla, G. V. Cormack, O. Vechtomova, A. Ashkan, S. Büttcher, and I. MacKinnon, “Novelty and diversity in information retrieval evaluation,” in *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '08*, (New York, NY, USA), pp. 659–666, ACM, 2008.
- [36] C. X. Zhai, W. W. Cohen, and 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*, (New York, NY, USA), pp. 10–17, ACM, 2003.

- [37] R. Agrawal, S. Gollapudi, A. Halverson, and S. Ieong, “Diversifying search results,” in *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, WSDM ’09, (New York, NY, USA), pp. 5–14, ACM, 2009.
- [38] H. Chen and 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, (New York, NY, USA), pp. 429–436, ACM, 2006.
- [39] Z. Chen and 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, (New York, NY, USA), pp. 641–652, ACM, 2007.
- [40] E. Vee, U. Srivastava, J. Shanmugasundaram, P. Bhat, and S. A. Yahia, “Efficient Computation of Diverse Query Results,” in *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, (Washington, DC, USA), pp. 228–236, IEEE Computer Society, 2008.
- [41] R. Kohavi, “A study of cross-validation and bootstrap for accuracy estimation and model selection,” in *IJCAI*, pp. 1137–1143, Morgan Kaufmann, 1995.
- [42] J. L. Fleiss, “Measuring nominal scale agreement among many raters,” *Psychological Bulletin*, vol. 76, no. 5, pp. 378 – 382, 1971.
- [43] K. Chodorow and M. Dirolf, *MongoDB: The Definitive Guide*. O’Reilly Media, Inc., 1st ed., 2010.
- [44] S. Magliacane, A. Bozzon, and E. Della Valle, “Efficient execution of top-k sparql queries,” in *Proceedings of the 11th International Conference on The Semantic Web - Volume Part I*, ISWC’12, (Berlin, Heidelberg), pp. 344–360, Springer-Verlag, 2012.