

# Keyword Search with Multiple Interactive Perspectives and Question Answering over RDF Datasets

*Christos Nikas*

Thesis submitted in partial fulfillment of the requirements for the  
*Masters' of Science degree in Computer Science and Engineering*

University of Crete  
School of Sciences and Engineering  
Computer Science Department  
Voutes University Campus, 700 13 Heraklion, Crete, Greece

Thesis Advisor: Associate Prof. *Yannis Tzitzikas*

---

This work has been performed at the University of Crete, School of Sciences and Engineering, Computer Science Department.

The work has been supported by the Foundation for Research and Technology - Hellas (FORTH), Institute of Computer Science (ICS).



UNIVERSITY OF CRETE  
COMPUTER SCIENCE DEPARTMENT

**Keyword Search with Multiple Interactive Perspectives and Question  
Answering over RDF Datasets**

Thesis submitted by  
**Christos Nikas**  
in partial fulfillment of the requirements for the  
Masters' of Science degree in Computer Science

THESIS APPROVAL

Author: 

Christos Nikas

**IOANNIS TZITZIKAS**

Digitally signed by IOANNIS  
TZITZIKAS  
Date: 2021.03.04 12:40:36 +02'00'

Committee approvals:

Yannis Tzitzikas  
Associate Professor, Thesis Supervisor

**DIMITRIOS  
PLEXOUSAKIS**

Digitally signed by DIMITRIOS  
PLEXOUSAKIS  
Date: 2021.03.04 13:27:21 +02'00'

Dimitris Plexousakis  
Professor, Committee Member

**GEORGIOS  
FLOURIS**

Digitally signed by GEORGIOS  
FLOURIS  
Date: 2021.03.04 10:58:13  
+02'00'

Giorgos Flouris  
Principal Researcher, Committee Member

**Polyvios  
Pratikakis**

Digitally signed by Polyvios  
Pratikakis  
Date: 2021.03.09 10:28:27  
+02'00'

Departmental approval:

Polyvios Pratikakis  
Assistant Professor, Director of Graduate Studies

Heraklion, February 2021



# Keyword Search with Multiple Interactive Perspectives and Question Answering over RDF Datasets

## Abstract

Since the task of accessing RDF datasets through structured query languages like SPARQL is rather demanding for ordinary users, there are various approaches that attempt to exploit the simpler and widely used keyword-based search paradigm. However, this task is challenging since there is no clear unit of retrieval and presentation, the user information needs are in most cases not clearly formulated, the underlying RDF datasets are in most cases incomplete, and there is not a single presentation method appropriate for all kinds of information needs. As a means to alleviate these problems, in this thesis we investigate a multi-perspective interaction approach that offers to the user multiple interactive views of the search results, allowing the user to easily switch between these perspectives and thus exploit the added value that each such perspective offers.

We present a set of fundamental perspectives, we discuss the benefits from each one, we compare the proposed approach with related existing systems and report the results of a task-based evaluation with users. The key finding of the task-based evaluation is that users not familiar with RDF (a) managed to complete the information-seeking tasks with performance very close to that of the experienced users, and (b) they rated positively the approach.

We also focus on the Question Answering Perspective and to this end we introduce a QA pipeline that involves a general purpose entity search service over RDF, SPARQL, and pre-trained neural networks, including a two-stage method for semantic answer type prediction using BERT and class-specificity rewarding, and finally we report very promising quantitative results over well-known benchmarks.



# Αναζήτηση μέσω Λέξεων-Κλειδιών με Πολλαπλές Διαδραστικές Προβολές και Απάντηση Ερωτήσεων επί Συνόλων Δεδομένων RDF

## Περίληψη

Η πρόσβαση σε σύνολα δεδομένων RDF μέσω δομημένων γλωσσών επερωτήσεων, όπως η SPARQL, είναι μια δύσκολη διαδικασία για τους απλούς χρήστες. Για το λόγο αυτό έχουν προταθεί διάφορες προσεγγίσεις που επιχειρούν να εκμεταλλευτούν το πιο απλό, και ευρέως χρησιμοποιημένο, υπόδειγμα της αναζήτησης μέσω λέξεων-κλειδιών. Ωστόσο, αυτή η διαδικασία αποτελεί πρόκληση καθώς δεν υπάρχει ξεκάθαρη μονάδα ανάκτησης και παρουσίασης, τις περισσότερες φορές οι πληροφοριακές ανάγκες του χρήστη δεν είναι διατυπωμένες ξεκάθαρα, τα υποκείμενα σύνολα δεδομένων RDF έχουν ελλείψεις, και δεν υπάρχει μία μοναδική μέθοδος παρουσίασης που να είναι κατάλληλη για όλα τα είδη πληροφοριακών αναγκών.

Για να απαλύνουμε αυτό το πρόβλημα, σε αυτή την εργασία προτείνουμε και αξιολογούμε μια προσέγγιση αλληλεπίδρασης σε πολλαπλές προοπτικές που παρέχει στο χρήστη πολλαπλές διαδραστικές προβολές των αποτελεσμάτων της αναζήτησης, επιτρέποντας του να εναλλάσσει εύκολα τις προβολές και έτσι να αξιοποιεί την προστιθέμενη αξία που παρέχει κάθε μία από αυτές. Παρουσιάζουμε ένα σύνολο από θεμελιώσεις διαδραστικές προβολές, εξετάζουμε τα πλεονεκτήματα της κάθε μιας, συγκρίνουμε την προτεινόμενη προσέγγιση με σχετικά υπάρχοντα συστήματα και σχολιάζουμε τα αποτελέσματα μίας αξιολόγησης με πραγματικούς χρήστες. Το βασικό συμπέρασμα της αξιολόγησης είναι ότι ακόμα και οι χρήστες που δεν είναι εξοικειωμένοι με την RDF (α) κατάφεραν να ολοκληρώσουν τις εργασίες αναζήτησης σχεδόν το ίδιο καλά με τους έμπειρους χρήστες και (β) αξιολόγησαν θετικά την όλη προσέγγιση.

Στην εργασία αυτή επίσης μελετάμε την διαδραστική προβολή που αντιστοιχεί στην Απάντηση Ερωτήσεων (Question Answering), και για το σκοπό αυτό προτείνουμε μια διαδικασία απάντησης ερωτήσεων η οποία εμπλέκει την υπηρεσία αναζήτησης οντοτήτων, επερωτήσεις SPARQL, και μια μέθοδο δύο σταδίων για την πρόβλεψη του σημασιολογικού τύπου της επιδιωκόμενης απάντησης χρησιμοποιώντας το νευρωνικό δίκτυο BERT με επιβράβευση βασισμένη στην εξειδίκευση των κλάσεων. Τέλος, σχολιάζουμε τα πολύ θετικά ποσοτικά αποτελέσματα αποτελεσματικότητας επί γνωστών συλλογών αξιολόγησης.





## Ευχαριστίες

Ευχαριστώ τον επόπτη καθηγητή μου Γιάννη Τζιτζικα για την πολύτιμη υποστήριξη και καθοδήγηση του από την πρώτη μου επαφή με το Πανεπιστήμιο Κρήτης μέχρι και την ολοκλήρωση αυτής της εργασίας. Ακόμη θέλω να εκφράσω τις ευχαριστίες μου στον κ. Δημήτρη Πλεξουσάκη και στον κ. Γιώργο Φλουρή για την προθυμία τους να συμμετέχουν στην τριμελή επιτροπή. Επίσης, ευχαριστώ τον Παύλο Φαφαλιό, τον Παναγιώτη Παπαδάκο, τον Γιάννη Μαρκετάκη και τον Γιώργο Καντηλιεράκη για τη συνεισφορά τους σε αυτή την εργασία και το Ινστιτούτο Πληροφορικής του ΙΤΕ για την υποτροφία που μου προσέφερε κατά τη διάρκεια της μεταπτυχιακής μου εργασίας.

Τέλος, θα ήθελα να ευχαριστήσω την οικογενειά μου για τη συνεχή στήριξη και εμπιστοσύνη τους.







# Contents

<b>Table of Contents</b>	<b>i</b>
<b>List of Tables</b>	<b>iii</b>
<b>List of Figures</b>	<b>v</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Publications related to this thesis . . . . .	4
<b>2 Background &amp; Related Work</b>	<b>5</b>
2.1 Background: RDF . . . . .	5
2.2 Keyword Search over RDF Datasets . . . . .	5
2.3 Visualization of RDF Search Results . . . . .	6
2.4 BERT . . . . .	8
2.5 Question Answering over Keyword search upon Knowledge Graphs	9
<b>3 Multi-Perspective Presentation of Search Results</b>	<b>11</b>
3.1 Rationale and Architecture . . . . .	11
3.1.1 Rationale . . . . .	11
3.1.2 Architecture . . . . .	12
3.2 The Fundamental Perspectives of Keywords Search Results . . . . .	12
3.2.1 Triples Tab . . . . .	13
3.2.2 Entities Tab . . . . .	13
3.2.3 Graph Tab . . . . .	14
3.2.4 Schema Tab . . . . .	15
3.2.5 Question Answering Tab . . . . .	17
3.2.6 Tabs' Roles and Extra Tabs . . . . .	18
3.3 Implementation . . . . .	23
<b>4 Question Answering</b>	<b>25</b>
4.1 Introduction . . . . .	25
4.2 Overview of the Approach . . . . .	26
4.3 Answer Type Prediction . . . . .	27
4.3.1 Overview . . . . .	27

4.3.2	Question Category & Literal Type Prediction . . . . .	28
4.3.3	Resource Answer Type Prediction . . . . .	29
4.3.4	Tuning of the k Parameter . . . . .	30
4.3.5	Model Selection . . . . .	30
4.4	Entities Retrieval and Expansion . . . . .	30
4.4.1	Retrieval . . . . .	30
4.4.2	Expansion . . . . .	31
4.5	Answer Extraction . . . . .	31
4.6	Related Work . . . . .	32
<b>5</b>	<b>Evaluation</b>	<b>33</b>
5.1	User Survey . . . . .	33
5.1.1	Comparing the Functionality with Related Systems . . . . .	33
5.1.2	Efficiency . . . . .	34
5.1.3	Evaluation of Effectiveness . . . . .	35
5.1.4	Evaluation with Users . . . . .	35
5.1.4.1	Information Seeking Tasks . . . . .	35
5.1.4.2	Participants, Questionnaire and Results . . . . .	36
5.1.4.3	Results Analysis and Discussion . . . . .	37
5.1.4.4	Log Analysis . . . . .	40
5.1.4.5	Discussion: Related Systems . . . . .	40
5.2	Answer Type Prediction Evaluation . . . . .	41
5.2.1	Evaluation Metrics . . . . .	41
5.2.2	Results on split of the DBpedia training set . . . . .	41
5.2.3	Results over the final DBpedia test set . . . . .	43
5.2.4	Efficiency . . . . .	43
5.3	Question Answering Evaluation . . . . .	44
5.3.1	Experiment 1: Webquestions . . . . .	44
5.3.2	Without Answer Type Prediction . . . . .	46
5.3.3	Experiment 2: DBpedia Entity: QA . . . . .	46
5.3.4	Experiment 3: DBpedia Entity: QA+RANKING . . . . .	47
5.3.5	Executive Summary . . . . .	48
5.3.6	Efficiency . . . . .	49
<b>6</b>	<b>Conclusion &amp; Future Work</b>	<b>51</b>
	<b>Bibliography</b>	<b>53</b>

# List of Tables

4.1	Results for different values of $k$ . . . . .	30
4.2	Answer examples . . . . .	32
5.1	Search Systems over DBpedia . . . . .	34
5.2	Average load times for each perspective . . . . .	34
5.3	Evaluation Tasks . . . . .	36
5.4	Evaluation results . . . . .	42
5.5	Results for different values of $k$ . . . . .	42
5.6	Confusion matrices for <i>category</i> (top left), <i>literal</i> (top right), and <i>resource</i> (bottom) type prediction. . . . .	43
5.7	Evaluation results over the final test set . . . . .	43
5.8	F1 and Exact scores over WebQuestions . . . . .	45
5.9	Precision @1, @3, @5 for varying answer score threshold over DBpedia Entity . . . . .	47
5.10	NDCG scores over Natural Language Questions of the DBpedia Entity collection for approach I: Keep Initial Scores . . . . .	48
5.11	NDCG scores over Natural Language Questions of the DBpedia Entity collection for approach II: Sum Scores . . . . .	48
5.12	Average time cost for each stage of the pipeline . . . . .	49





# List of Figures

1.1	Access Methods over RDF . . . . .	2
1.2	Search results for the query “El Greco museum”. . . . .	3
2.1	The user interface of LOTUS Tab . . . . .	7
2.2	The user interface of DBpedia Precision Search & Find . . . . .	7
2.3	BERT classification example . . . . .	8
3.1	The Triples Tab . . . . .	13
3.2	The Entities Tab . . . . .	14
3.3	The Graph Tab . . . . .	15
3.4	The Schema Tab (Tesla) . . . . .	17
3.5	The Schema Tab (Crete and Mars) . . . . .	17
3.6	The QA Tab . . . . .	18
3.7	The QA Tab . . . . .	18
3.8	The Added Value of each Perspective . . . . .	19
3.9	Search results for the query “El Greco and Kazantzakis”. . . . .	20
3.10	Search results for the query “Paintings with dogs”. . . . .	21
3.11	Search results for the query “Which cities does the Weser flow through?”. . . . .	22
3.12	Search results for the query “Greek philosopher from Athens who is credited as one of the founders of Western philosophy”. . . . .	23
4.1	QA Pipeline . . . . .	27
4.2	Elas4RDF user interface for Question Answering . . . . .	32
5.1	Age distribution of participants . . . . .	36
5.2	‘Very Useful’ and ‘Useful’ preference percentages per tab and category of users. . . . .	38
5.3	Success rates for experienced and inexperienced users. . . . .	39



# Chapter 1

## Introduction

The Web of Data contains thousands of RDF datasets available online (see [44] for a recent survey), including cross-domain knowledge bases (KBs) (e.g., DBpedia and Wikidata), domain specific repositories (e.g., DrugBank [64], ORKG [29], and recently COVID-19 related datasets [16]), as well as Markup data through `schema.org`. These datasets are queried mainly through structured query languages, i.e. SPARQL. Faceted Search is a user-friendlier paradigm for interactive query formulation and exploratory search, however the systems that support it (see [61] for a survey) also need a keyword search engine as a flexible entry point to the information space. Consequently, and since plain users are acquainted with web search engines, an effective method for keyword search over RDF is indispensable. Moreover, keyword search allows for multiple-word (even paragraph-long) queries that can address many topics, and such information needs could be difficult to formulate even in structured query languages. The results of such queries allow users to detect associations of entities that they were not aware of, thus favoring the discovery of new information.

In general we could say that *structured queries* (e.g., using SPARQL) and *unstructured queries* (keyword search) are fundamental components of all access methods over RDF. Figure 1.1 shows the general picture of access services over RDF. Apart from *Structured Query Languages* and *Keyword Search* we can see the category *Interactive Information Access*. That refers to access methods that are beyond the simple “query-and-response” interaction, i.e. methods that offer more interaction options to the user and exploit also the *interaction session*. In this category, we have methods for browsing, methods for faceted search [61], methods for formulating OLAP queries (e.g. [51]), and methods for assistive query building (e.g. [34]). Finally, in the category *natural language interfaces* we have methods for question answering, dialogue systems, and conversational interfaces. As the figure shows, both *interactive information access* and *natural language interfaces* pre-suppose effective and efficient support of structured and unstructured queries.

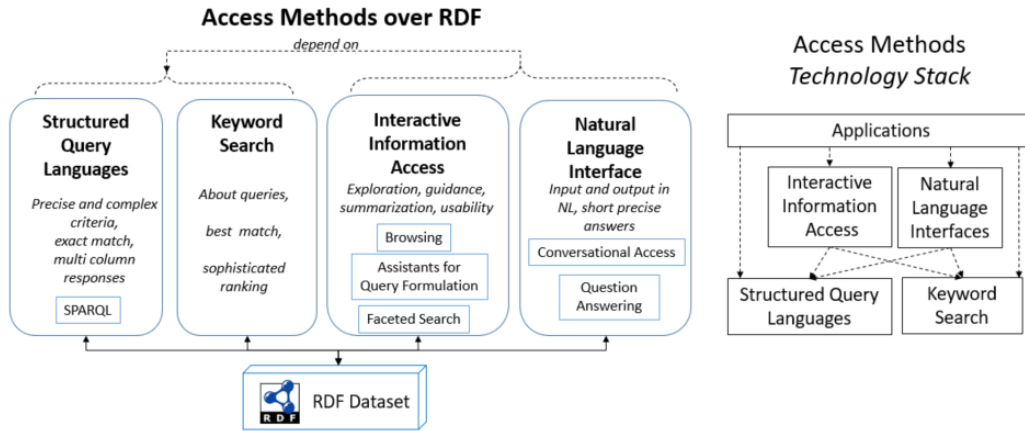


Figure 1.1: Access Methods over RDF

However, keyword search over RDF datasets is a challenging task since (a) in RDF there is no clear unit of retrieval and presentation, (b) it is difficult to understand, from a usually small keyword query, the intent of the user, (c) the data are in most of the cases incomplete (making the provision of effective retrieval difficult), and (d) there is not a single presentation method appropriate for all kinds of information needs.

To tackle these challenges, in this thesis we focus on the value stemming from offering *multiple-perspectives* of the search results, i.e., multiple presentation methods, each presented as a separate *tab*, and allowing the user to easily *switch* between these perspectives, and thus exploit the added value that each such perspective offers. To grasp the idea, Figure 1.2 shows the search results for the query “*El Greco museum*”, as presented in each of the five currently-supported tabs.

As basic keyword-search retrieval method, we assume the *triple-centered* approach proposed in [31] (which in turn relies on `Elasticsearch`) because it is schema-agnostic (and thus general-purpose), and it offers efficient and scalable retrieval services with effectiveness comparable (as evaluated using DBpedia-Entity test collection for entity search [23]) to the effectiveness of dedicated systems for keyword search over RDF (more in [31]). Over this basic service, in this thesis we motivate the provision of certain *fundamental perspectives*, we showcase the benefits from each one, and we evaluate what users can achieve if they have all of them at their disposal.

We also focus on the QA perspective. In comparison to the other perspectives (tabs) that are offered by this approach, i.e. triple tab, entity tab, graph tab, schema tab, the QA tab is supposed to provide a short and concise answer, if that is feasible. To this end, in this thesis we investigate such an approach for open-domain Question Answering over Knowledge Bases that could complement general purpose keyword search over RDF.

In in this thesis: we motivate the multi-perspective approach, we discuss the added value of each perspective, we introduce additional perspectives, we compare

the functionality of the implemented system with other systems over DBpedia, we report the results of a *task-based evaluation with users* that provides interesting insights related to the validation of the main research hypothesis of this work, i.e. whether the provision of more than one tab is helpful for the users. The key finding is that the success rate of all users was very high even of those users not familiar with RDF. We also introduce and evaluate an approach for Question Answering over the search results

Figure 1.2: Search results for the query “El Greco museum”.

This thesis is organized as follows: Chapter 1 introduces the topic of this thesis, the challenges of keyword search and question answering over RDF and our approach to tackle these challenges. Chapter 2 includes background about the topics discussed in this thesis and presents related work Chapter 3 focuses on the Multi-Perspective Presentation of Search Results and specifically the Triples, Entities, Graph and Schema Tab. Chapter 4 presents our approach for the question answering component which consists of the following stages Answer Type Prediction, Entity Retrieval and Expansion and Answer Extraction. Chapter 5 discusses the evaluation of the work performed for this thesis. This includes the task-based evaluation with users of the Multi-Perspective interaction and numeric evaluation for question answering including results specifically for the answer type prediction component. Finally, Chapter 6 concludes the thesis and discusses ideas for future work.

## 1.1 Publications related to this thesis

In this section we list publications written during the development of this thesis.

- G. Kadilierakis, C. Nikas, P. Fafalios, P. Papadacos, Y. Tzitzikas, *Elas4RDF: Multi-perspective Triple-centered Keyword Search over RDF using Elastic-search*, (Demo Paper), Proceedings of the 17th Extended Semantic Web Conference (ESWC'2020), June 2020, Heraklion, Crete

In this work [32], we present a demo including the initial implementation of the multi-perspective keyword search system. This work was presented as a demo paper in ESWC 2020.

- C. Nikas, G. Kadilierakis, P. Fafalios, and Y. Tzitzikas, *Keyword Search over RDF: Is a Single Perspective Enough?*, *Big Data and Cognitive Computing*, vol. 4, no. 3, p. 22, Aug. 2020

In this work [47], we present more extensively each perspective, we add the schema perspective and perform a task-based evaluation with users.

- Christos Nikas, Pavlos Fafalios and Yannis Tzitzikas, *Two-stage Semantic Answer Type Prediction for Question Answering using BERT and Class-Specificity Rewarding*, *SeMantic Answer Type prediction (SMART) Challenge*, 2020 International Semantic Web Conference (ISWC'2020) Challenge, Nov 2020.

In this work [46], we present the Answer Type Prediction component of the Question Answering perspective. This work was published on ISWC 2020 for the SeMantic Answer Type prediction task [41] and achieved second place.

- Christos Nikas, Pavlos Fafalios and Yannis Tzitzikas, *Open Domain Question Answering over Keyword Search upon Knowledge Graphs*

In this work that is currently under preparation, we present the question answering perspective in detail and we evaluate it over popular benchmarks.

## Chapter 2

# Background & Related Work

At first we provide some background about RDF (in Section 2.1), then we discuss the existing approaches for keyword search over RDF (in Section 2.2), then we discuss the visualization of RDF search results (in Section 2.3), and finally we provide some background information about the language model BERT (in Section 2.4).

### 2.1 Background: RDF

RDF stands for *Resource Description Framework* and it is a framework for describing resources on the web. Essentially it is a structurally object-oriented model. RDF uses Uniform Resource Identifiers (URIs), or anonymous nodes, to denote resources, and literals to denote constants. Every statement in RDF can be represented as a *triple*. A triple is a statement of the form subject-predicate-object  $\langle s, p, o \rangle$ , and it is any element of  $T = (U \cup B) \times (U) \times (U \cup B \cup L)$ , where  $U$ ,  $B$  and  $L$  are the sets of URIs, blank nodes and literals, respectively. Any finite subset of  $T$  corresponds to an RDF graph (or dataset). We can divide the URIs in three disjoint sets: entities (e.g., <http://dbpedia.org/resource/Aristotle>), properties (e.g., <http://dbpedia.org/property/dateOfBirth>) and RDF classes (e.g., <http://dbpedia.org/ontology/Philosopher>).

### 2.2 Keyword Search over RDF Datasets

Keyword search over RDF data can be supported either by translating keyword queries to structured (SPARQL) queries (like in [20, 36]), or by building or adapting a dedicated information retrieval system using classical IR methods for indexing and retrieval. This work builds upon approaches that follow the second direction. In general systems of that kind construct the required indexing structures either from scratch or by employing existing IR engines (like Lucene and Solr), adapt the notion of a virtual document for the RDF data, and rank the results (entities, triples or subgraphs) according to commonly used IR ranking functions. There

are various systems that fall in this category, like [8, 12, 37]. Most such systems rely on adaptations of the TD-IDF weighting, as in [11] where the keyword query is translated to a logical expression that returns the ids of the matching entities. Another direction is to return ranked subgraphs instead of relevant entity URIs, like in [50], while in [19] the returned subgraphs are computed using statistical language models.

Ranking is usually based on extensions of the BM25 model, e.g., in [7, 52]. [17] introduces the TSA+VDP keyword search system, where the system first builds offline an index of documents over a set of subgraphs via a breadth-first search method, and at query-time, it returns a ranked list of these documents based on a BM25 model. Regarding the retrieval unit, most works return either URIs or subgraphs, except [27] and [31] that follow a triple-centered approach.

With respect to works that rely on document-centric information retrieval systems, LOTUS [27] makes use of `Elasticsearch` and provides a keyword-search entry point to the Linked Data cloud, focusing on issues of scalability. `Elasticsearch` has been also used for indexing and querying Linked Bibliographic Data in JSON-LD format [30]. Finally, [31] adapts `Elasticsearch` for supporting keyword search over arbitrary RDF datasets. Through an extensive evaluation, the authors studied questions related to the selection of the triple data to index, the weighting of the indexed fields, and the structuring and ranking of the retrieved results. In our work, we make use of the approach proposed in [31] because it is schema-agnostic and returns ranked lists of triples, which offers us the flexibility to provide different visualisations of the search results.

## 2.3 Visualization of RDF Search Results

There are several approaches for browsing, exploring and visualizing *RDF datasets in general*, e.g. see the surveys [6] and [10]. Regarding the *visualization of SPARQL results*, there are a few works, however since the form of the results of such queries is essentially that of a relational table, these approaches provide amenities for the visualization of tabular data, i.e. various plots and charts for analytics [59, 35, 62].

As regards the visualization of *keyword search results over RDF*, which is the main focus of our work, DBpedia Precision Search & Find (<http://dbpedia.org/fct/>) returns entities and for each one it shows its URI, its title, the URI of the named graph it belongs to, as well as a description with highlighted the query terms. Also, the user can browse on the Linked Data by clicking on the shown resources. The keyword search systems LOTUS [27, 26] and [31] do not focus on presentation and visualisation. LOTUS returns triples by providing the full URIs of the resources, while [31] returns triples and/or entities using an API. In general, most works (including [54, 18]) do not pay attention to the presentation of results; they focus on the ranking of entities/subgraphs that they compute.

Finally, [60] and [33] investigate the exploitation of semantics in the visualisation of search results. [60] utilizes visualization techniques for offering visual



feedback about the *reasons* a set of search results was retrieved and ranked as relevant. [33] performed an analytical inspection and a user study of the interface offered by two semantic search engines: *Knigine* and *Sigma* (both are not active anymore). In particular, the authors investigated if the exploitation of semantics enables a better visualization of search results and thus a better user experience.

To our knowledge, our work is the first that investigates and evaluates (with real users) a multi-perspective interactive approach to present the search results of a keyword search system over RDF.

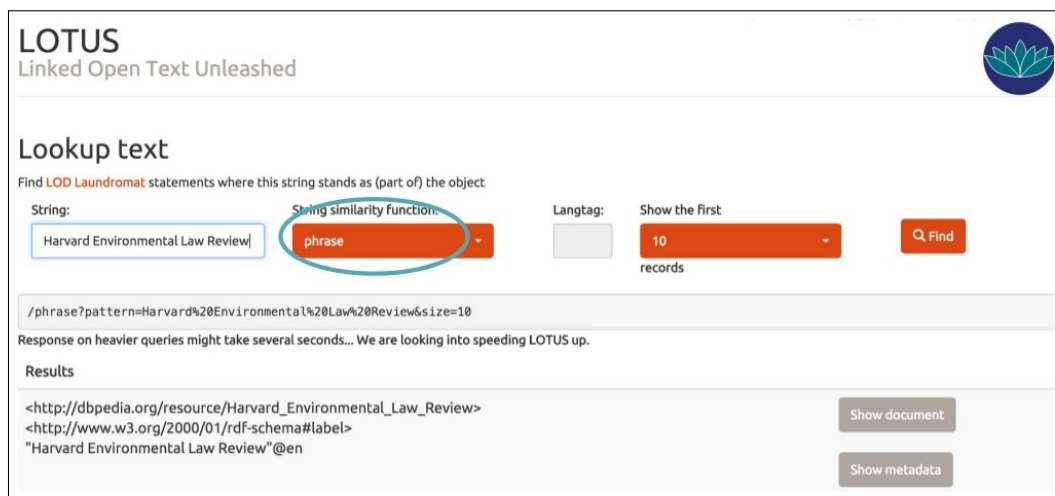


Figure 2.1: The user interface of LOTUS Tab

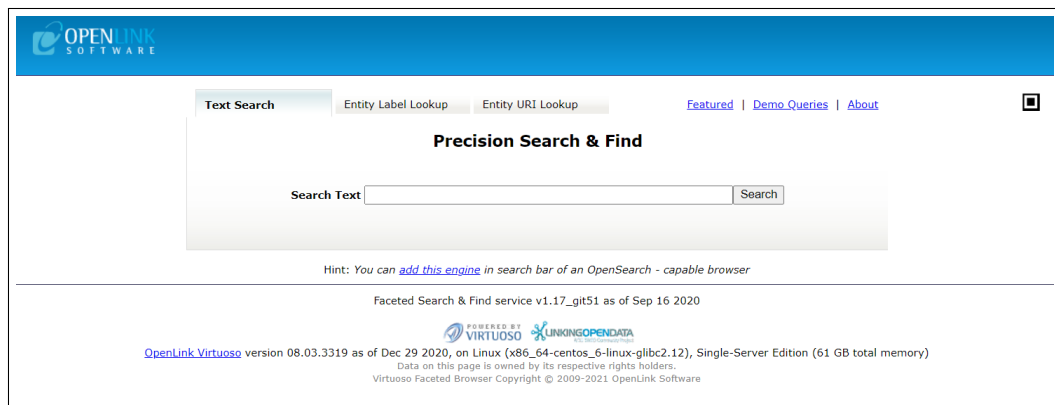


Figure 2.2: The user interface of DBpedia Precision Search & Find

## 2.4 BERT

BERT [14], or Bidirectional Encoder Representations from Transformers, is a language representation model based on the Transformer model architecture of [63]. A pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications. Because of BERT’s massive success and popularity, several methods have been presented to improve BERT on its prediction metrics, by using more data and computational speed [38, 66], or by creating lighter and faster models that compromise on prediction metrics [55].

The use of BERT as a sequence classifier can be summarized in figure 2.3<sup>1</sup>. The input sequence is tokenized, then special tokens are added to separate sentences and apply padding and each token is mapped to a numeric ID. Then the tokenized input is passed through BERT and finally through a Classifier layer. The output of the classifier is a probability for each possible class and the class with the maximum probability is chosen as the final output.

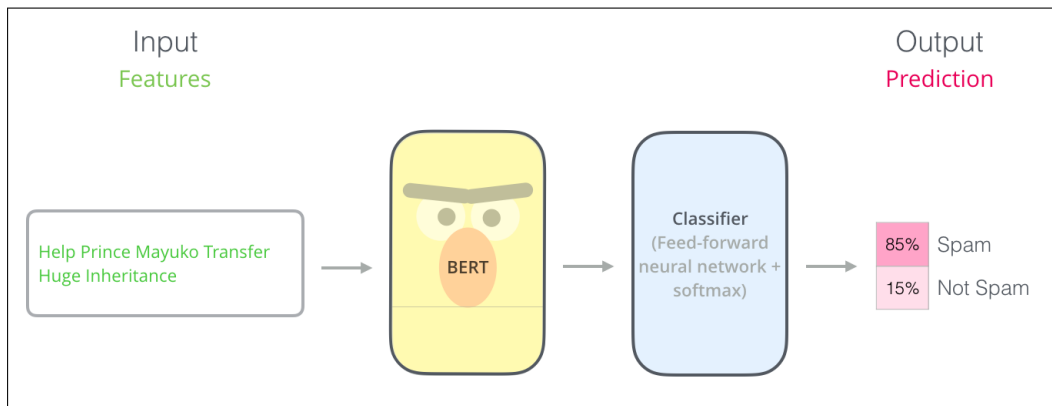


Figure 2.3: BERT classification example

In this work, we use 2 variations of BERT for different purposes. Specifically, we use DistilBERT (more in section 4.3.5) in the Answer Type Prediction component of the Question Answering tab (section 4.3). This model has a sequence classification/regression final layer which allows it to work as a classifier. It receives a question as input and produces an integer as output which indicates the class that the question belongs to.

We also use RoBERTa (more in section 4.3.5) in the Answer Extraction component of the Question Answering tab to perform extractive Question Answering. This model has a span classification layer on top for extractive question-answering tasks. It receives a question and a text that contains the answer to the question as

<sup>1</sup>Taken from <https://jalammar.github.io/illustrated-bert/>

input and produces the indices in the text that correspond to the start and end of the answer.

In this thesis, we make use of all BERT-based models using HuggingFace’s Transformers library [65].

## 2.5 Question Answering over Keyword search upon Knowledge Graphs

In comparison to related work, e.g. see [58] for a recent overview of QA approaches over DBpedia, the most related works are: [25] which converts the natural language question into two subqueries: SPARQL query and keyword search. That work uses a keyword index for special keywords rather than a whole knowledge graph for keyword search and produces the final answer using an algorithm to combine SPARQL results and keyword search results.

Another work regarding Question Answering and Keyword Search is SINA [57]. This system performs query preprocessing to tokenize, remove stopwords and lemmatize terms in the query, then groups keywords into segments and generates conjunctive federated SPARQL queries to retrieve answers. In contrast to our approach, this work relies fully on a SPARQL endpoint instead of using a dataset-specific index for keyword search. Also, it does not use any neural network based methods to perform answer extraction.

To the best of our knowledge, no other work has investigated the effect of Answer Type Prediction in Question Answering over knowledge graphs.



## Chapter 3

# Multi-Perspective Presentation of Search Results

### 3.1 Rationale and Architecture

#### 3.1.1 Rationale

The rationale for the *multi-perspective* (and tabs-switching interaction) approach that we propose can be summarized as:

- *No Clear Unit of Retrieval and Presentation.* In RDF data, there is not the notion of document or web page as it is the case in web searching. Therefore the retrieval, presentation and visualisation of RDF data is challenging due to the complex, interlinked, and multi-dimensional nature of this data type [10].
- *No Clear Information Need.* The user query is just an attempt to formulate his/her information need. Some user needs require a single fact, others a list of entities or a set of facts, other how a set of entities are connected, other have an exploratory nature, and so on.
- *Incomplete Data.* The underlying dataset is in most cases incomplete [43] (also evidenced by the number of papers that aim at completing the missing data [4]), therefore the retrieved triples cannot be considered neither complete, nor appropriately ranked. However the provision of more than one method, each consuming different proportions of the list of top hits (and of their context), increases the probability that one method achieves to return something that is useful for the user's information need.
- *There is not a single presentation method appropriate for all kinds of information needs.* An established method on how to present RDF results for arbitrary query types does not exist yet, and it seems that a single approach

can not suit all possible requirements. Different kinds of information needs need different ways to present the results.

For the above reason we propose a *multi-perspective* approach, where each perspective is presented in a different *tab*, stressing a different aspect (and proportion) of the hits. The user can inspect all tabs and get a better overview and understanding of the search results. The *tabs-switching interaction* that we propose is easy to understand and perform by the user, just like plain Web search engines offer various such tabs (for images, videos, news, etc). Below, in Section 3.2, we shall discuss the rationale (added value) of each particular tab and how it is defined. An orthogonal but important challenge is how to provide several such presentation methods *at real time*, for enabling the user to switch fast between the different perspectives, i.e. the multi-perspective and tab-switching approach should not add a noticeable latency to the responses.

### 3.1.2 Architecture

As keyword search service we adopt the approach proposed in [31] because it is schema agnostic, directly applicable, has good evaluation results, and its triple-centered approach facilitates the multi-perspective approach. Specifically, we exploit the REST API that is offered by that service which accepts keyword queries and returns results in JSON format (code available at <https://github.com/SemanticAccessAndRetrieval/Elas4RDF-search>). On top of this search service we build the multi-perspective approach.

The full DBpedia 2015-10 dataset has been indexed using 2 approaches (i.e., *baseline* and *extended*, described in [31]). We have used that version of DBpedia because it is the version used in the DBpedia-Entity test collection for entity search [23], which allowed us to get comparable results related the effectiveness of the approach (as detailed in [31]). The number of virtual documents (triples) in both cases is 395,569,688. In our setup and experiments, the average query execution time is around 0.7 sec for the baseline method and 1.6 sec for the extended, and depends on the query type.

## 3.2 The Fundamental Perspectives of Keywords Search Results

Below we describe each individual perspective (for short *tab*) and then (in Section 3.2.6) we discuss the role of each in tab in the general search process. In the description of each perspective we consider the DBpedia 2015-10 dataset and the query  $q_{run} = \text{“El Greco museum”}$  as our running example.

### 3.2.1 Triples Tab

**Rationale:** This tab is generally the most useful one since the user can inspect *all components of each triple*, and understand the reason *why* that triple is returned. The addition of images help the user to easily understand which triples involve the same entities.

**Description:** A ranked list of triples is displayed to the user (as fetched from the search service described in Section 3.1.2), where each triple is shown in a separate row. For visualising a triple, we create a *snippet* for each triple element (subject, predicate, object). The snippet is composed of: i) a title (the text indexed by the baseline method), ii) a description (the text indexed by the extended index; if any), and iii) the URI of the resource (if the element is a resource). If the triple element is a resource, its title is displayed as a hyperlink, allowing the user to further explore it. We also retrieve and show an image of this resource (if any). For the query  $q_{run} = \text{“El Greco museum”}$ , more than 4.2 millions triples are retrieved. The first two triples are about the Museum of El Greco in Crete, the third about the El Greco Museum in Toledo, the fourth about the entity El Greco, the fifth is a triple about a list of works by El Greco, and so on. A screenshot of this tab for the query ”Crete and Mars” is displayed on figure 3.1.

The screenshot shows the 'Elas4RDF' interface with the search query 'crete and mars'. The 'Triples' tab is selected. The search results are displayed in a table-like format. Each row represents a triple, with columns for the subject, predicate, and object. The subject column contains a small image and the title 'Icaria Planum'. The predicate column contains the word 'comment' or 'abstract'. The object column contains a snippet of text describing the resource. The results are for 'Icaria Planum' and 'Stato da Mar'.



Subject	Predicate	Object
 Icaria Planum	abstract	" Icaria Planum is a region on Mars in the Thaumasia quadrangle of Mars that is 566.59 km across and is located at 43.27 S and 253.96E. It was named after a classic albedo feature that was approved in 1979. The name of the classic feature was based on the land where Icarus live..."
 Icaria Planum	comment	" Icaria Planum is a region on Mars in the Thaumasia quadrangle of Mars that is 566.59 km across and is located at 43.27 S and 253.96E. It was named after a classic albedo feature that was approved in 1979. The name of the classic feature was based on the land where Icarus live..."
Stato da Mar	comment	" The Stato da Mar or Domini da Mar (State/Domains of the Sea) was the name given to the Republic of Venice's maritime and overseas possessions, including Istria, Dalmatia, Albania, Negroponte, the Morea (the Kingdom of the Morea), the Aegean islands of the Duchy of the A..."
Stato da Mar	abstract	" The Stato da Mar or Domini da Mar (State/Domains of the Sea) was the name given to the Republic of Venice's maritime and overseas possessions, including Istria, Dalmatia, Albania, Negroponte, the Morea (the Kingdom of the Morea), the Aegean islands of the Duchy of the A..."

Figure 3.1: The Triples Tab

### 3.2.2 Entities Tab

**Rationale:** If the user is interested in *entities*, and not in particular facts, this view provides the *main entities*.

**Description:** Here the retrieved triples are *grouped* based on entities (subject and object URIs), and the entities are *ranked* following the approach described in [31] which considers the weighted gain factor of the ranking order of the triples in

which the entities appear. Then, a ranked list of entities is displayed to the user, where each entity is shown in a different row. For visualising an entity, we create the same snippet like previously. The title is displayed as a hyperlink, since the entities are resources, allowing the user to further explore the entity. For  $q_{run}$  the returned entities include “El Greco”, the two museums of El Greco (in Crete and Toledo), particular paintings, like “Saint Peter and Saint Paul”, the music album “El Greco” by Vangelis, the film “El Greco (2007)”, and so on. A screenshot of this tab for the query ”Crete and Mars” is displayed on figure 3.2.

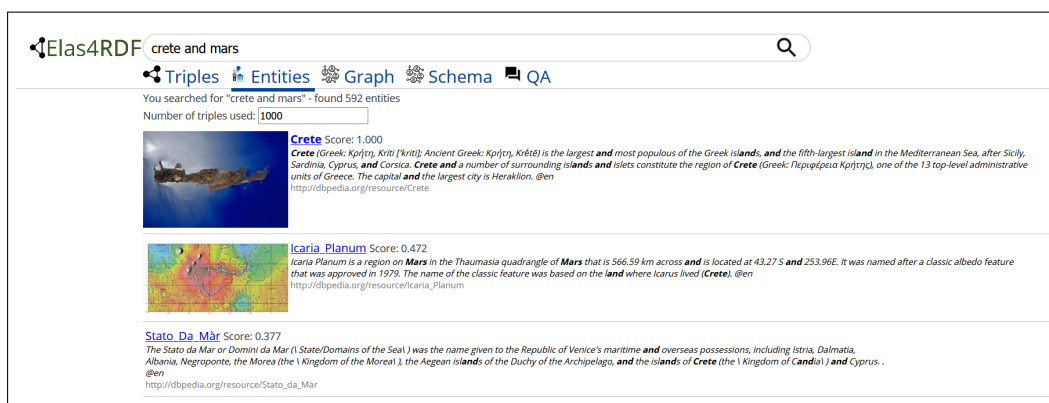


Figure 3.2: The Entities Tab

### 3.2.3 Graph Tab

**Rationale:** This tab allows the user to inspect a larger number of triples without having to scroll down. Most importantly, this view reveals the *grouping* of triples, how they are *connected*, and whether there is one or more poles and interesting connections.

**Description:** The retrieved triples are visualised as a graph for stressing how the triples are connected. By default, the graph shows the top-15 triples, however the user can increase or decrease this number, while the nodes are clickable, pointing to the corresponding resource in DBpedia. In our implementation we use JavaScript InfoVis Toolkit (<https://phillogb.github.io/jit/>). For  $q_{run}$  the user can see how the top ranked triples are connected and can spot easily the nodes that have high connectivity. A screenshot of this tab for the query ”Crete and Mars” is displayed on figure 3.3.



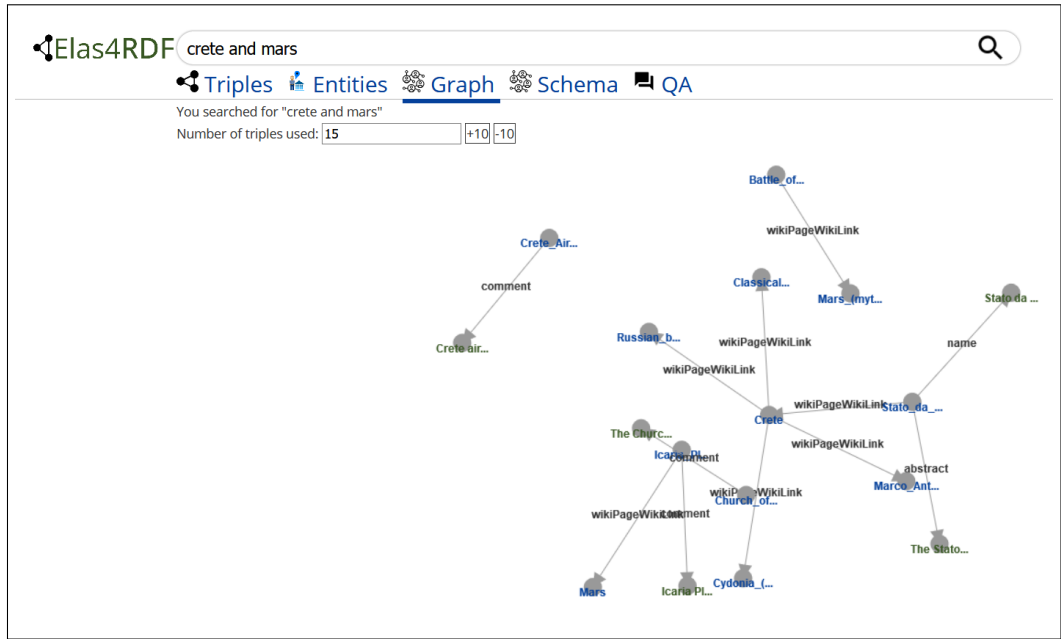


Figure 3.3: The Graph Tab

### 3.2.4 Schema Tab

**Rationale:** The objective is to show which are the *more frequent schema elements* of the retrieved triples. This is useful for (a) understanding the *conceptual context* of the hits, (b) for *exploring (restricting) interactively the triples or entities* of the answer (by filtering with respect to class or property), and (c) for helping an experienced user to inspect which classes and properties occur in the answer, if after the keyword search, the user would like to formulate a SPARQL query (directly or through a faceted search system, or through a query builder in general like [49, 34]).

**Description:** The schema tab is divided in four frames as shown in Figure 3.4.

Upper Left Frame: It shows the more frequent classes and properties, accompanied by their frequency. Let  $A$  be the top- $K$  triples retrieved for the current query,  $P$  the properties in  $A$ , i.e.  $P = \{p \mid (s, p, u) \in A\}$ , and  $C$  the classes of the URIs in the triples of  $A$ , i.e.  $C = \{c \mid (s, rdf : type, c), s \in SP\}$ . For each  $c \in C$ , its frequency is defined as  $freq(c) = |\{o \in SP \mid (o, rdf : type, c) \in KB\}|$ , while for each  $p \in P$ ,  $freq(p) = |\{(s, p, o) \in A\}|$ . Through a parameter  $F$  we control the number of visible elements, i.e. initially the user can see only the  $F$  in number elements of  $C$  with the highest frequency, and the  $F$  in number elements of  $P$  with the highest frequency (however the user can expand the visible elements to see all of them). By clicking a class or a property the user can see the corresponding triples and entities in the frames at the right side that will be described later on.

Bottom Left Frame: It shows graphically the more frequent classes and properties.

A parameter  $K$  (just like in the graph tab) controls the number of triples that feed the schema tab (the user can increase decrease it as she wishes to). In particular, the graph  $\Gamma = (Nodes, Edges)$  that is visualized is defined as  $Nodes = C$ , and  $Edges = \{(c, c') \in C \times C' \mid (s, p, o), (s, rdf : type, c), (o, rdf : type, c') \in A\}$ , i.e. an edge connects two classes  $c$  and  $c'$  if there is at least one triple in  $A$  that connects an instance of  $c$  with one instance of  $c'$ . Ideally the graph visualization should make evident the frequencies, i.e. the more frequent classes and properties should be visualized with bigger boxes and arrows. It is not hard to see that the number of edges, i.e.  $|E|$ , can be higher than the number of distinct properties that occur in  $A$ , e.g. if  $(s, p, o) \in A$  and  $s$  is classified to two classes  $c1$  and  $c2$ , and  $o$  to two classes  $c3$  and  $c4$ , then the graph will contain the four edges  $\{(n(c1), n(c3)), (n(c1), n(c4)), (n(c2), n(c3)), (n(c2), n(c4))\}$ . The reverse is also possible, i.e.  $|E|$  can be less than the number of distinct properties, e.g. if  $(s, p1, o)$  and  $(s, p2, o)$  belong to  $A$ , and each of  $s$  and  $o$  is classified to one class, then only one edge will be visible between these two classes. Note that several variations and extensions are possible from the area of semantic model visualization and summarization.

Right Upper and Right Bottom Frames: These frames show the *triples* and *entities*, related with the user's click. Suppose the user has *clicked on a frequent class* "c1(18)". The triples frame will show all triples  $\{(s, p, o) \in A \mid (s, rdf : type, c1) \wedge (o, rdf : type, c1)\}$ , and let call this set  $T$ . The entities frame will show the more frequent entities that occur in  $T$ . If the user *clicks on a frequent property* "p2(10)", the triple frame will show the 10 triples  $A$  that have  $p2$  as property, let call this set  $T$ , and the entity frame will show the more frequent entities of those occurring in  $T$ . The above behaviour is supported also by the graph, i.e. clicking on a node is interpreted as if the user had clicked on the corresponding frequent class.

Returning to  $q_{run}$ , we can see the classes **Person**, **Agent**, **Location**, **Work**, etc. and various properties. The right frames show the triples and entities after having clicked on "Architectural Structure", i.e. triples and entities that are related to the query *and* classified under the class "Architectural Structure" (we can see information about a museum in Florina, another in Bilbao, etc.).

As another example, for the query "Tesla", the user is getting what is shown in Figure 3.4, enabling him to focus to the desired triples or entities, i.e. to those related to: Tesla Motors (Organization), Nicola Tesla (Agent), Tesla Model X (Mean Of Transportation), Tesla West Virginia (Place). By increasing the number of triples he can also find Tesla Band (Group). By clicking on the property "author" the user can directly see the triple related to works authored by Nicola Tesla. In general in this tab the user can increase a lot the number of consumed triples: although more classes and properties will appear their number is not high, hence in most cases they will not clutter the diagram (in the example of Figure 3.4 the schema tab consumes 75 triples).

A screenshot of this tab for the query "Crete and Mars" is displayed on figure 3.5.

### 3.2. THE FUNDAMENTAL PERSPECTIVES OF KEYWORDS SEARCH RESULTS17

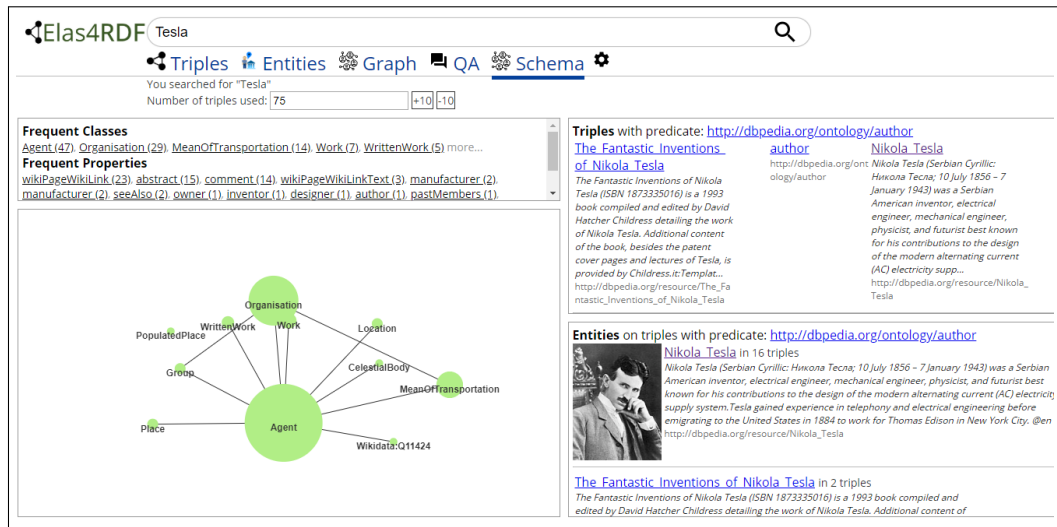


Figure 3.4: The Schema Tab (Tesla)

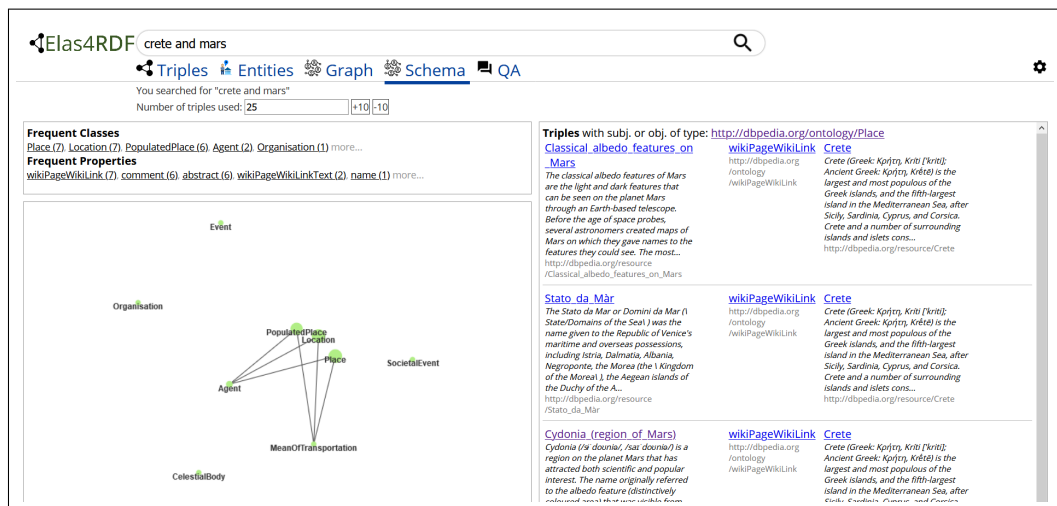


Figure 3.5: The Schema Tab (Crete and Mars)

#### 3.2.5 Question Answering Tab

The task of Question Answering (QA) is to identify the *question type* and the *expected answer type*, therefore, based on the analysis of the QA perspective, a short answer (presented in the appropriate way), could be promoted (just like WSE do), therefore one direction for further research is to investigate the applicability of approaches like [9] and [40] for complex questions. We focus on this tab extensively in chapter 4. Screenshots of this tab for the queries "Who is the father of Queen Elizabeth II?" and "Which country does Greenland belong to?" are displayed on

figures 3.6 and 3.7.

The screenshot shows the Elas4RDF interface with the search query "who is the father of Queen Elizabeth II?". The interface includes navigation tabs for Triples, Entities, Graph, Schema, and QA. The QA tab is active, displaying a table with two columns: Answer Type and Answers. The first answer is for "King George VI", with a score of 0.856. The second answer is for "Prince Charles", with a score of 0.505. Each answer includes a category, a resource link, and a detailed description.

Answer Type	Answers
Category	<b>King George VI.</b>
resource	From entity: <a href="http://dbpedia.org/resource/Coronation_of_Queen_Elizabeth_II">http://dbpedia.org/resource/Coronation_of_Queen_Elizabeth_II</a>
Type	The coronation of Queen Elizabeth II as monarch of the United Kingdom, Canada, Australia, New Zealand, Union of South Africa, Pakistan, and Ceylon took place on 2 June 1953. Elizabeth ascended the thrones of these countries at age 25, upon the death of her father, <b>King George VI</b> , on 6 February 1952, and was proclaimed queen by her various privy and executive councils shortly afterwards.
Person	Score: 0.856
	<b>Prince Charles.</b>
	From entity: <a href="http://dbpedia.org/resource/Monarchy_of_Belize">http://dbpedia.org/resource/Monarchy_of_Belize</a>
	The monarchy of Belize (the Belizean monarchy) is a system of government in which a hereditary monarch is the sovereign of Belize; the incumbent is Queen Elizabeth II, officially called Queen of Belize, who has reigned since 21 September 1981. The heir apparent is Elizabeth's eldest son, <b>Prince Charles</b> , though the Queen is the only member of the royal family with any constitutional role. Monarchy of Belize Link from a Wikipedia to another Wikipedia Sophia of Hanover. Monarchy of Belize Link from a Wikipedia to another Wikipedia James VI and I. Monarchy of Belize Link from a Wikipedia to another Wikipedia Charles, Prince of Wales. Monarchy of Belize Link from a Wikipedia to another Wikipedia Elizabeth II. Monarchy of Belize Link from a Wikipedia to another Wikipedia Colville Young. Monarchy of Belize heir apparent Charles, Prince of Wales. Monarchy of Belize Link from a Wikipedia to another Wikipedia Edward VIII. Monarchy of Belize Link from a Wikipedia to another Wikipedia Anne, Queen of Great Britain. Monarchy of Belize incumbent Elizabeth II. Monarchy of Belize first monarch Elizabeth II
	Score: 0.505

Figure 3.6: The QA Tab

The screenshot shows the Elas4RDF interface with the search query "which country does greenland belong to?". The interface includes navigation tabs for Triples, Entities, Graph, Schema, and QA. The QA tab is active, displaying a table with two columns: Answer Type and Answers. The first answer is for "Kingdom of Denmark", with a score of 0.532. The second answer is for "Denmark", with a score of 0.05. Each answer includes a category, a resource link, and a detailed description.

Answer Type	Answers
Category	<b>Kingdom of Denmark.</b>
resource	From entity: <a href="http://dbpedia.org/resource/List_of_airports_in_Greenland">http://dbpedia.org/resource/List_of_airports_in_Greenland</a>
Type	This is a list of airports in Greenland, grouped by type and sorted by location.Greenland (Greenlandic: Kalaallit Nunaat, Danish: Grønland) is an autonomous country within the <b>Kingdom of Denmark</b> , located between the Arctic and Atlantic Oceans, east of the Canadian Arctic Archipelago.
Country	Score: 0.532
	<b>Denmark.</b>
	From entity: <a href="http://dbpedia.org/resource/Military_of_Greenland">http://dbpedia.org/resource/Military_of_Greenland</a>
	The government of Greenland does not have control of Greenland's military or foreign affairs. The defence of Greenland is the responsibility of <b>Denmark</b> . However, following the November 2008 referendum on increased autonomy, which attracted significant popular support (72% turnout, 75% vote in favor), the governments of Greenland and Denmark have agreed to a 30-point package that will begin to reverse this position. Military of Greenland Link from a Wikipedia to another Wikipedia United States. Military of Greenland Link from a Wikipedia to another Wikipedia Norway. Military of Greenland Link from a Wikipedia to another Wikipedia Denmark. Military of Greenland Link from a Wikipedia to another Wikipedia Iceland. Military of Greenland Link from a Wikipedia to another Wikipedia Russia. Military of Greenland Link from a Wikipedia to another Wikipedia Nazi Germany. Military of Greenland Link from a Wikipedia to another Wikipedia Soviet Union. Military of Greenland Link from a Wikipedia to another Wikipedia China
	Score: 0.05

Figure 3.7: The QA Tab

### 3.2.6 Tabs' Roles and Extra Tabs

There are several other tabs that could be supported and could be useful in certain kinds of information needs, e.g. *image* tab, *geo* tab, *time* tab, etc. Each can be construed as a tool that could aid the user to focus on a particular aspect, based on the task/information need at hand, each enacted by a simple click (therefore the required effort is minimal). One rising question is how to provide an *overview* of these in an effortless manner, and/or how to rank them if that is desired. For reasons of transparency and exploration, it is beneficial to make the user aware of the existence of these, instead of promoting and showing only one, as some Web Search Engines (WSE) do.

In this work we confine ourselves on the previous five tabs since we believe that they are both *KB-independent* and *task-independent*, hence they can be considered

as fundamental. The added value from each of these basic perspectives is summarized in Figure 3.8. The diagram also shows some main paths that indicate *why* a user may decide, in a tab-switching interaction, to *move* from a tab to another (of course the user is free to follow any order). Below we provide a few additional examples showcasing the benefits from using more than one tab.

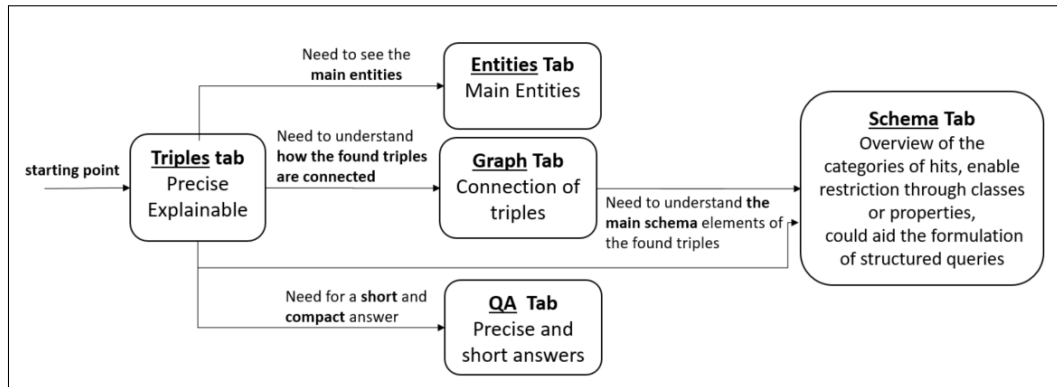


Figure 3.8: The Added Value of each Perspective

For the query  $q$ ="El Greco and Kazantzakis" in the Entities Tab, as shown in Figure 3.9, the user can find in the first two positions the two main entities of the query, i.e. "El Greco" (the painter), and "Nikos Kazantzakis" (the writer and philosopher), while in the Triples Tab the user can find a triple that connects these two entities. From the Graph Tab the user can see the triples grouped in two poles (one for each entity) and the user can realize that there is only one triple that connect these two poles (in the top-35 triples). Finally, with the Schema Tab the user can refine to Location and find entities whose name is related to the main entities, like "El Greco Apartments" and "Nikos Kazantzakis (municipality)".

## 20 CHAPTER 3. MULTI-PERSPECTIVE PRESENTATION OF SEARCH RESULTS

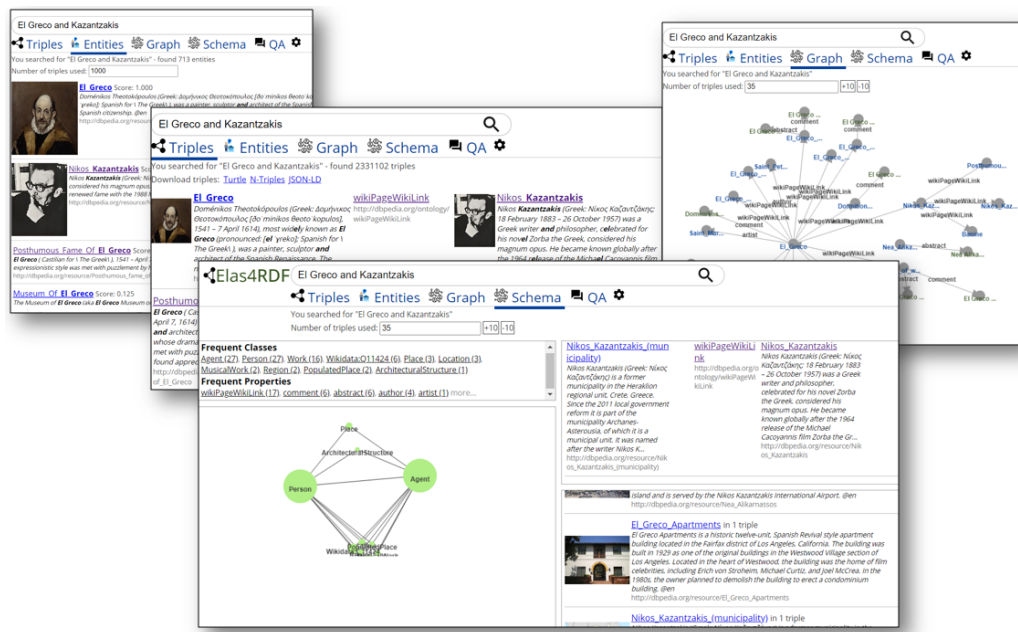


Figure 3.9: Search results for the query “El Greco and Kazantzakis”.

As another example, for the query “Paintings with dogs” in the Triples Tab, as shown in Figure 3.10, the user can find relevant specific information including information about “Painted Dog Conservation” (a non profit organization for the protection of the painted dog, or African wild dog), information about particular paintings, information about “Greg Rasmussen” the founder of the “Painted Dog Conservation”, etc. In the Entities tab the user can find the main entities, including the “Painted Dog Conservation”, the species “African Wild Dog”, one painting of Goya (The Dog), the “Dogs Playing Poker” (the series of sixteen oil paintings by C. M. Coolidge), etc. The Schema Tab shows the classes and properties of the found triples, through which the user can understand that there are related: species, (art) works, locations, etc. Moreover the user can refine/explore the information space as she wishes to. In Figure 3.10 the user has refined using the class “Work” and in the right bottom frame he can find various paintings with dogs including: “The Dog (Goya)”, “The Sentry (painting)”, “The Hunt In The Forest”, “Interior With A Young Couple And A Dog” “Portrait Of Charles V With A Dog” etc. Finally, the QA Tab returns two entities “Francisco Goya” (the painter of the painting “The Dog”), and “Coenraad Jacob Temminck” (a Dutch aristocrat, zoologist, and museum director who first described scientifically in 1820 the species African Wild Dog).

### 3.2. THE FUNDAMENTAL PERSPECTIVES OF KEYWORDS SEARCH RESULTS<sup>21</sup>

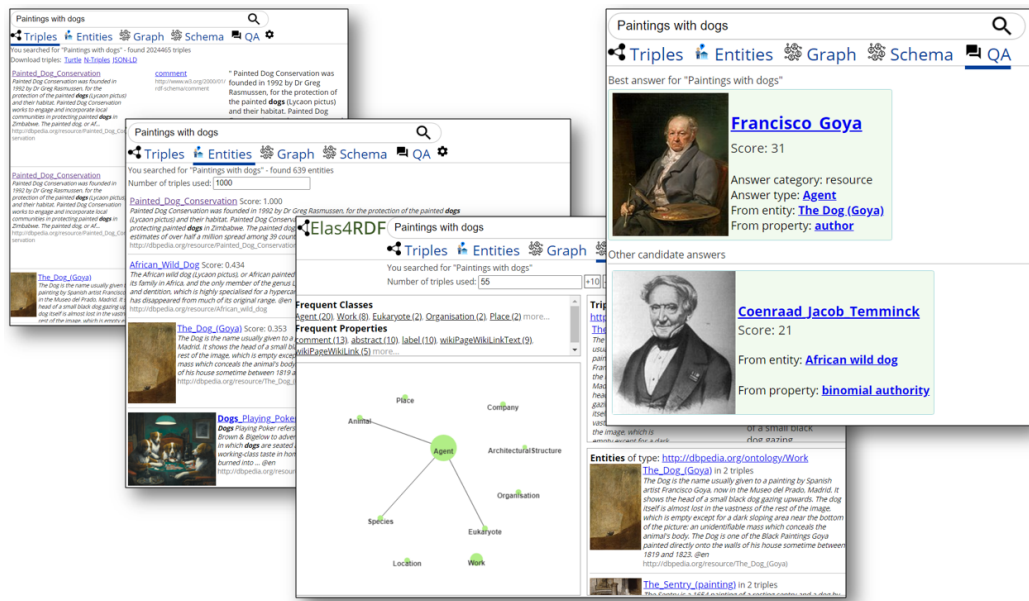


Figure 3.10: Search results for the query “Paintings with dogs”.

For list questions, i.e. questions with a set of elements as the correct response, like “Which cities does the Weser flow through?” the user may decide to inspect only the QA Tab and the Entities Tab as shown in Figure 3.11.

## 22 CHAPTER 3. MULTI-PERSPECTIVE PRESENTATION OF SEARCH RESULTS

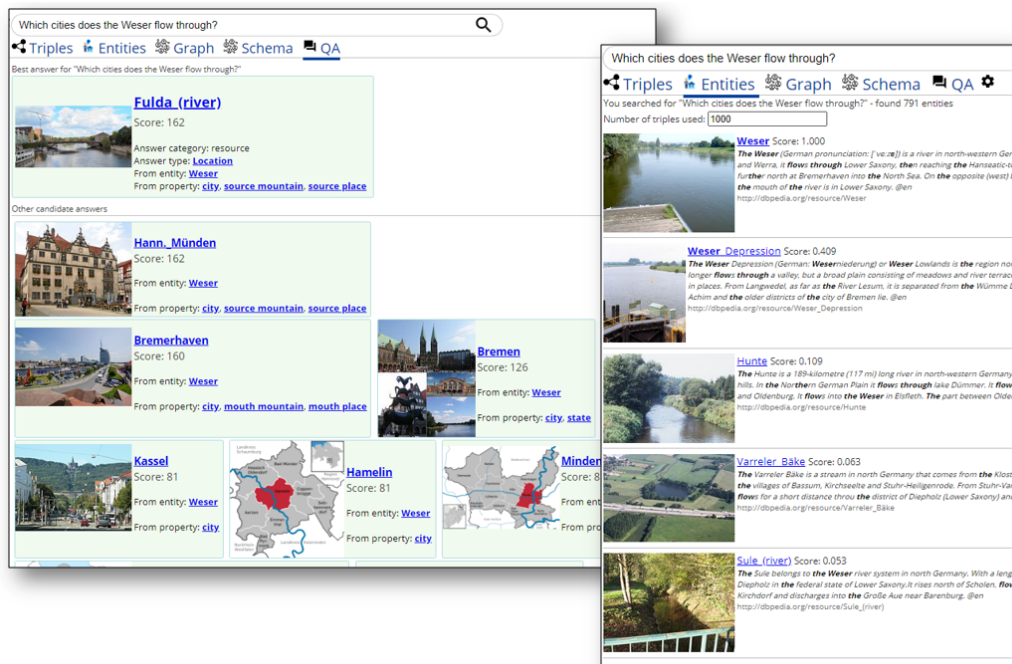


Figure 3.11: Search results for the query “Which cities does the Weser flow through?”.

Longer queries are also possible, for instance for the query “Greek philosopher from Athens who is credited as one of the founders of Western philosophy”, from the Entity Tab (as shown in Figure 3.12) the user we can see that Socrates received the higher score, while from the QA tab the user can see various other philosophers as candidate answers.



The screenshot displays the search results for the query "Greek philosopher from Athens who is credited as one of the founders of Western philosophy". The interface includes a search bar with the query, navigation tabs (Triples, Entities, Graph, Schema, QA), and search results for "Western Philosophy", "Socrates", "Aristippus", and "Other candidate answers" including Aristotle, Plato, and Antisthenes.

**Western Philosophy** Score: 1.000  
*Western philosophy* **ass**="highlighted">is the philosophical thought and work of the Western world. **Has**="highlighted">historically, the term re- beginning with Hellenic (i.e. **ass**="highlighted">Greek) philosophy, and eventually covering a large area of the globe. @en  
[http://dbpedia.org/resource/Western\\_philosophy](http://dbpedia.org/resource/Western_philosophy)

**Socrates** Score: 0.790  
 Socrates (/səˈkræɪt.ɪz; **ass**="highlighted">Greek: Σωκράτης; **ass**="highlighted">credited as one of the founders of Western philosophy. **has**="highlighted">is students Plato and Xenophon as his students.  
<http://dbpedia.org/resource/Socrates>

**Ancient **ass**="highlighted">Greek Philosophy** Score: 0.59  
 Ancient **ass**="highlighted">Greek philosophy arose in the 6th century BCE Roman Empire. It dealt with a wide variety of subjects including political philosophy, ethics, metaphysics, and epistemology. **ass**="highlighted">concede that **ass**="highlighted">Greek philosophy has influenced much of modern Western thought.  
[http://dbpedia.org/resource/Ancient\\_Greek\\_philosophy](http://dbpedia.org/resource/Ancient_Greek_philosophy)

**Zenodotus (**ass**="highlighted">philosopher)** Score: 0.201  
 Zenodotus (/zəˈnɒdɒt.əs; **ass**="highlighted">Greek: Ζηνόδοτος; fl. late 5th c. BCE; **ass**="highlighted">Athens. He was described as the darling (παίδικα) of the school (c. 485). He was a teacher of Damascius when he came to Asia Minor.  
[http://dbpedia.org/resource/Zenodotus\\_\(philosopher\)](http://dbpedia.org/resource/Zenodotus_(philosopher))

**Aristippus**  
 Score: 51  
 Answer category: resource  
 Answer type: Agent  
 From entity: Socrates  
 From property: influenced

**Aristotle**  
 Score: 51  
 From entity: Socrates  
 From property: influenced

**Plato**  
 Score: 51  
 From entity: Socrates  
 From property: influenced

**Antisthenes**  
 Score: 51  
 From entity: Socrates  
 From property: influenced

Figure 3.12: Search results for the query “Greek philosopher from Athens who is credited as one of the founders of Western philosophy”.

### 3.3 Implementation

The Elas4RDF web application has been developed using Java, the Spring Boot framework for the Web API and Thymeleaf to render the front end templates. For the graph tab, we also used the JavaScript InfoVis Toolkit<sup>1</sup> The Question Answering component is developed using Python. We used HuggingFace’s Transformers library [65] for the language models. The Question Answering component communicates with the Elas4RDF web application through a REST API built using Flask<sup>2</sup>.

<sup>1</sup><https://philogb.github.io/jit/>

<sup>2</sup><https://flask.palletsprojects.com>



## Chapter 4

# Question Answering

### 4.1 Introduction

Question answering over knowledge bases (KBQA) is an important NLP task because of the rapid growth of knowledge bases (KBs) on the web and the commercial value they bring for real-world applications [48]. In knowledge bases, where data is represented as a graph, e.g. using the Resource Description Framework (RDF), methods relying on Graph Processing and SPARQL Query Generation are adopted in order to extract the desired information [15]. At the same time, neural network-based (NN-based) Question Answering methods have received increasing attention in recent years and have already achieved impressive results [1].

Nevertheless in vague or complex information needs and questions, that require considering and joining facts, QA methods are not that good. the same time, there is not a single QA component per QA task that is perfect, and the performance of a QA component varies based on questions with different features [58]. For this method in this work, we refine and investigate an approach where at its core has a *keyword search* service, since keyword search can provide relevant hits for any kind of information need. Specifically, we consider the *multi-perspective* keyword search over RDF presented in [47], and focus on one of these perspectives: the QA perspective. In comparison to the rest perspectives (tabs) that are offered by that approach, i.e. triple tab, entity tab, graph tab, schema tab, the QA tab is supposed to provide a short and concise answer, if that is feasible.

To this end, in this work we investigate such an approach for open-domain Question Answering over Knowledge Bases that could complement general purpose keyword search over RDF.

We present a QA approach over DBpedia that relies on: (i) an entity search system to retrieve unstructured textual descriptions for entities, (ii) a Semantic Answer Type prediction component to predict the answer type, (iii) SPARQL to retrieve structured information matching the predicted answer type, (iv) an entity expansion component to expand the textual description with the information retrieved from the triple store, and (v) a powerful language model fine-tuned for

question answering to extract the final answers.

In brief, given a natural language question, we first retrieve the top-k entities and their textual descriptions (through keyword search), then we get the triples only of these entities that have the predicted answer type, then we generate natural language sentences and we apply extractive question answering using pre-trained neural networks.

Related research questions are: (a) How good the QA pipeline over DBpedia can be, in comparison to approaches and benchmarks freebase?, (b) how the Answer Type Prediction affects the quality of QA? (c) how answers from this QA pipeline can contribute to the entity retrieval task over DBpedia Entity dataset, and entity ranking in general?

The results of our evaluation indicate that the answers generated by this approach provide additional value for entity search when combined with the initial entities retrieved by the search service used. Our approach can also perform well on difficult question answering datasets ( $> 52\%$  Accuracy), without having been trained on the specific datasets, but relying on the structured and unstructured information retrieval methods that we use.

In brief, the main contribution of this work are: (a) we investigate a process for QA in the context of an keyword search access paradigm, (b) we detail the QA pipeline that comprises components for Answer Type Prediction, Entity Retrieval & Expansion, and neural network based approach for Answer Extraction, (c) we evaluate the pipeline over multiple datasets, showcasing the value added by our approach for different information needs.

The rest of this chapter is organized as follows: Section 4.2 provides an overview of the approach, Section 4.3 describes Answer Type Prediction, Section 4.4 describes Entity Retrieval and Expansion Section 4.5 describes Answer Extraction, Finally, Section 4.6 describes related work.

## 4.2 Overview of the Approach

We introduce and investigate a question answering pipeline, which can be summarized as follows: we retrieve the top-k entities and their textual descriptions (through search), then we get the triples only of these entities that have the predicted answer type, then from these triples we generate natural language sentences, and finally we apply extractive question answering using neural networks.

Consequently the pipeline is supported by 3 main components: Answer Type Prediction, Entity Retrieval and Expansion, and Answer Extraction, as illustrated in Figure 4.1.

First we predict the answer by extending and improving a previous work on Answer Type Prediction [46] (this step is described in §4.3).

We also retrieve a set of entities relevant to the question, from DBpedia, along with a short description of each entity using the Elas4RDF search service [31]. Then we extend the description of each entity with information from RDF nodes

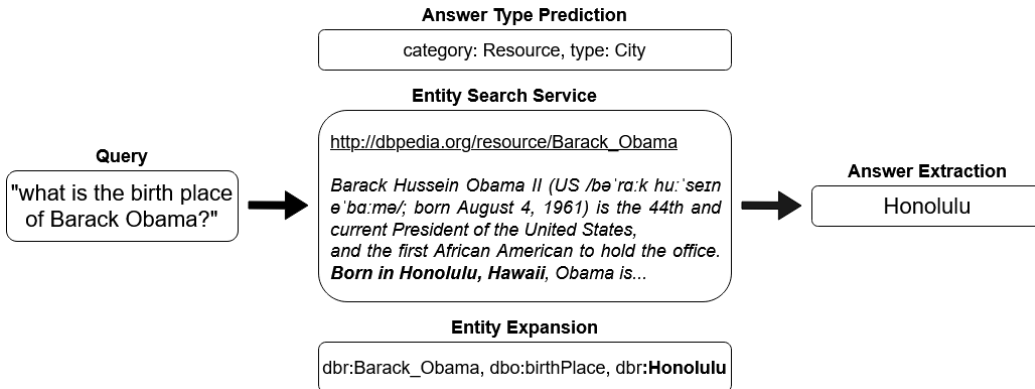


Figure 4.1: QA Pipeline

matching the predicted answer type by running SPARQL queries at real-time (more in §4.4).

Finally, we use a RoBERTa [38] model fine-tuned on the SQuAD 2 dataset [53] to perform extractive question answering for the question using the extended description of each entity. Therefore, we obtain an answer from each retrieved entity. Finally, we rank the answers using the score from the output of the model and present them on the user interface of the Elas4RDF web application. (more in §4.5).

The input of the task performed by this component is a set of top-K entities returned by the search service, where each entity consists of a URI and a short textual description. Using the Answer Type Prediction, Entity Expansion and Answer Extraction techniques that are described in more detail below, the component produces a natural language answer extracted by the textual description.

## 4.3 Answer Type Prediction

### 4.3.1 Overview

Answer Type Prediction is the task of predicting the type of the answer to a natural language question, given the question. Our approach for answer type prediction is based on the work [46] which was submitted on the SMART Task [41] and gained second place. The task is split in 2-stages: Category prediction and Type prediction. In particular, the problem is modeled as a two-stage classification task: in the first step the task is to predict the general category of the answer (*resource*, *literal*, or *boolean*), while in the second step the task is to predict the particular answer type (*number*, *date*, *string*, or a particular *resource class* from a target ontology).

Two datasets are provided for this task, one using the DBpedia ontology and the other using the Wikidata ontology. Both follow the below structure: Each

question has a (a) question id, (b) question text in natural language, (c) an answer category (*resource/literal/boolean*), and (d) answer type. If the category is *resource*, answer types are ontology classes from either the DBpedia ontology ( $\sim 760$  classes) or the Wikidata ontology ( $\sim 50\text{K}$  classes). If the category is *literal*, answer types are either *number*, *date*, or *string*. Finally, if the category is *boolean*, answer type is always *boolean*.

An excerpt from this dataset is shown below:

```
[ {
  "id": "dbpedia_14427",
  "question": "What is the name of the opera based on Twelfth Night?",
  "category": "resource",
  "type": ["dbo:Opera", "dbo:MusicalWork", "dbo:Work" ]
}, {
  "id": "dbpedia_23480",
  "question": "Do Prince Harry and Prince William have the same parents?",
  "category": "boolean",
  "type": ["boolean"]
} ]
```

With respect to the size of the datasets, the DBpedia dataset contains 21,964 questions (train: 17,571, test: 4,393) and the Wikidata dataset contains 22,822 questions (train: 18,251, test: 4,571). The DBpedia training set consists of 9,584 resource, 2,799 boolean, and 5,188 literal questions. The Wikidata training set consists of 11,683 resource, 2,139 boolean, and 4,429 literal questions.

For question category and type prediction we use 2 DistilBERT for sequence classification models. We choose DistilBERT instead of BERT to reduce memory footprint and time required to answer a question.

### 4.3.2 Question Category & Literal Type Prediction

A question can belong to one of the following three categories: (1) boolean, (2) literal, (3) resource. *Boolean questions* (also referred to as Confirmation questions) only have ‘yes’ or ‘no’ as an answer (e.g. “Does the Owyhee river flow into Oregon?”). Thus, there is no further classification for this category of questions. *Resource questions* have a specific fact as an answer (e.g. “What is the highest mountain in Italy?”) that can be described by a class in an ontology (e.g. <http://dbpedia.org/ontology/Mountain>). *Literal questions* have a literal value as answer, which can be a *number*, *string*, or *date* (e.g. “Which is the cruise speed of the airbus A340?”).

To detect question categories, we fine-tune a DistilBERT model using the Huggingface PyTorch implementation<sup>1</sup>. We choose this model because we approach answer type prediction as a classification problem where each question is a sequence of words.

<sup>1</sup><https://huggingface.co/transformers/>

Because we only use 3 types to classify literal questions, following the approach of [56] we integrate literal type prediction into the same classifier with category prediction. By doing this, we save computing requirements and reduce memory footprint because we avoid using a different BERT classifier for literal type prediction. Therefore, this model classifies each question in one of the following 5 classes: 1) boolean, 2) literal date, 3) literal number, 4) literal string, 5) resource

To fine tune the model we used the training datasets provided for the SMART challenge (described in §4.3.1). Specifically, we used questions from both the DBpedia and the Wikidata dataset. Because the data is imbalanced for categories (13.7% boolean, 26.6% literal, 59.4% resource) we randomly sampled questions for each class so that all classes had the same number of samples.

As we will see below, this model achieves 97.7% accuracy on our test set in this prediction task.

### 4.3.3 Resource Answer Type Prediction

The prediction of the answer type of questions in the *resource* category is a more fine-grained (and thus more challenging) classification problem, because of the large number of types a question can be classified to ( $\sim 760$  classes on DBpedia and  $\sim 50K$  classes on Wikidata). Therefore, it is not effective to train a classifier on all the ontology classes, especially for open-domain tasks.

To reduce the number of possible types for classification, we selected a subset ( $C$ ) of all ontology classes, based on the number of samples of each class in the training set. This subset  $C$  contains classes that have at least  $k$  occurrences in the training set. We set  $k = 10$  as this number provides a good trade-off between number of classes and performance. The choice of this parameter is described more extensively in section 5.2.2.

The final number of classes in  $C$  is 88. Because we chose to train the system on a subset of all the classes, our classifier cannot handle questions with labels that are not included in this subset. To tackle this problem, we replace their labels with the labels of super classes that belong in  $C$ . Then we fine tuned a DistilBERT model on them.

Since most questions in the dataset have several answer types ordered by specificity, according to the semantic hierarchy formed in the ontology, in the fine tuning stage we use these questions multiple times, one with each of the provided types as the label. The goal is to find an answer type that is as specific as possible for the question. However, the model may classify a question to a more general answer type in the ontology. To tackle this problem, we ‘reward’ (inspired by [13]), the predictions of the classes that lie below the top class. The reward of a class  $c$  is measured by the depth of the class in the hierarchy, specifically,  $reward(c) = depth(c)/depth_{Max}$ , where  $depth(c)$  is the depth of  $c$  in its hierarchy, while  $depth_{Max}$  is the maximum depth of the ontology (6 for DBpedia). This means that, after applying normalization and adding the rewards on the output of the model, the top class can be a sub-class that was originally ranked below

a more general class. For example, for the question “*What is the television show whose company is Playtone and written by Erik Jendresen?*” the top 5 classes that the classifier predicts are: 1) Work, 2) TelevisionShow, 3) Film, 4) MusicalWork, 5) WrittenWork. Then rewards are applied to classes that are a subclass of Work. After applying the rewards, the top 5 classes are: 1) TelevisionShow, 2) Work, 3) Film, 4) Book, 5) MusicalWork. We can see that TelevisionShow, is now the top prediction, which is both correct and more specific than the previous top prediction (Work).

#### 4.3.4 Tuning of the k Parameter

To find the optimal value for the parameter k, which is the minimum sample size required to include a class in the subset of classes included in the classifier, we evaluated our system using 4 different values: 5, 10, 30 and 50.

Table 5.5 shows the number of classes included in the classifier for each different value of k and the corresponding performance. We notice that the best results are obtained using k=10, while the results for all other cases are slightly worse.

Table 4.1: Results for different values of k

Value	Classes	NDCG@5	NDCG@10
5	180	0.775	0.765
10	151	0.786	0.778
30	79	0.785	0.772
50	55	0.785	0.748

#### 4.3.5 Model Selection

DistilBERT[55] is a smaller general-purpose language representation model based on BERT[14]. A distilbert model can be 40% smaller in size than an equivalent BERT model, while retaining 97% of its language understanding capabilities and being 60% faster. We chose this model for category classification and answer type classification because the compromise in the language understanding capabilities is not significant for us, since our models perform well enough for the required tasks. At the same time we plan to make this system available through a web application, therefore answer time speed, and memory footprint is important for us.

## 4.4 Entities Retrieval and Expansion

### 4.4.1 Retrieval

We use the elas4rdf search service [31] to retrieve a set of entities that are relevant to the query. We query this service with the input question after removing stop-words. The output of this stage is a list of entities described by their URI and a short textual description of the entity, extracted by the rdfs:comment property.



The number of retrieved entities is set to 10, but it can be adjusted. A higher number of entities could yield more useful answers, but will require more time to be processed.

#### 4.4.2 Expansion

For Resource and Literal questions, we use the dbpedia SPARQL endpoint to find facts about the retrieved entities that match the answer type. Then, we generate natural language sentences from these facts and append the sentences to the entity description.

For Resource questions, for each entity, we retrieve all RDF triples where the subject is the entity, and the object has an RDF type that matches the top type returned by the answer type prediction component, or an equivalent class, using the following query:

```
select distinct str(?pl) as ?pLabel ?a where {
  <entity uri> ?p ?a .
  ?p rdfs:label ?pl .
  <answer type> owl:equivalentClass ?eq .
  ?a rdf:type ?eq .
  FILTER(lang(?pl) = 'en' || lang(?pl) = '')
}
```

For Literal date questions we retrieve triples where the property that connects the entity with the candidate answer has an `rdfs:range` equal to `xsd:date`.

For Literal number and string questions we retrieve all triples where the subject is the entity and the object is a literal. Then we check programmatically if the object is numeric or a string depending on the answer type. We follow this process because not all literal RDF Nodes have an XSD Schema data type.

From the retrieved triples we use the label of the corresponding entity, the object, which is a candidate answer and the label of the property that connects the entity with this answer. Then we generate a sentence of the form "*entity\_label + property\_label + object*" and append it to the textual description of the entity.

## 4.5 Answer Extraction

This stage receives a list of entity URIs and their expanded textual description. For each entity in the list, we generate an answer from the expanded entity description using a RoBERTa model for extractive question answering from the huggingface transformers library<sup>2</sup>. Then, we sort the answers by their score and display them on the Question Answering perspective of Elas4RDF, along with the answer category and type.

The model that we use is fine-tuned on the SQuAD dataset by deepset.ai<sup>3</sup>.

<sup>2</sup><https://huggingface.co/transformers/>

<sup>3</sup><https://huggingface.co/deepset/roberta-base-squad2>

RoBERTa (Robustly optimized BERT approach), is a retraining of BERT with improved training methodology, using 10 times more data and compute power. We chose this model over BERT because it obtains higher scores, which is important for us because of the increased difficulty of the task of extractive question answering.

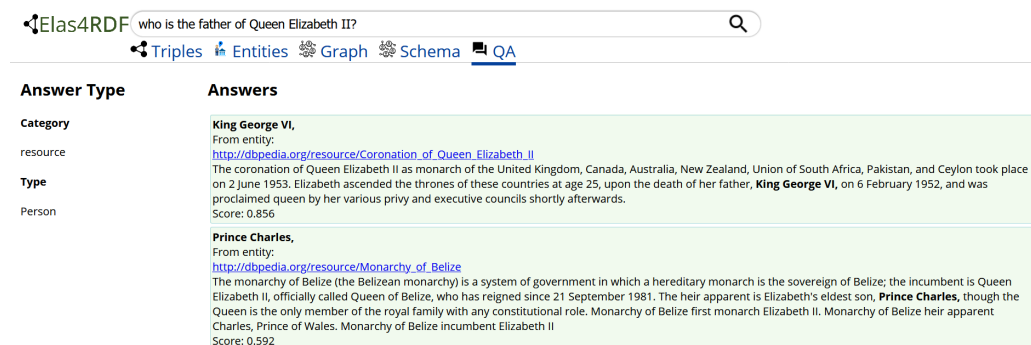


Figure 4.2: Elas4RDF user interface for Question Answering

A few successfully answered question examples are displayed below on table 4.2

Question	Answer
Who did Mozart write his four horn concertos for?	Joseph Leutgeb
What things did Martin Luther King do?	human rights advocate and community activist
When did Charles Goodyear invented rubber?	1839

Table 4.2: Answer examples

## 4.6 Related Work

In comparison to related work, e.g. see [58] for a recent overview of QA approaches over DBpedia, the most related works are:

[25] which converts the natural language question into two subqueries: SPARQL query and keyword search. This work uses a keyword index for special keywords rather than a whole knowledge graph for keyword search and produces the final answer using an algorithm to combine SPARQL results and keyword search results. Based on our current knowledge, no other work has investigated the effect of Answer Type Prediction in Question Answering.

# Chapter 5

## Evaluation

In section 5.1 we evaluate the multi-perspective keyword search and presentation approach and report the results obtained from a task-based evaluation with users. In section 5.2 we evaluate the answer type prediction component of the question answering perspective and report results in the context of the SeMantic AnswER Type prediction task [41]. Finally, in section 5.3 we evaluate the question answering component over popular benchmarks for both question answering and entity search tasks.

### 5.1 User Survey

Below we evaluate the proposed approach by (a) comparing its *functionality* with those of related systems, (b) proving its feasibility by discussing *efficiency*, (c) discussing the retrieval *effectiveness* of the system, and (d) reporting the results of a *task-based evaluation with users* that examines the usefulness of the proposed multi-perspective approach, as well as some results by *log analysis*.

#### 5.1.1 Comparing the Functionality with Related Systems

Since DBpedia is a core dataset of the Linked Open Data cloud [2], we decided to compare with *interactive systems* (not just APIs) that offer a kind of access/search facility over DBpedia. For this reason we considered the following systems: LOTUS [27], GraFa [42] (<http://grafa.dcc.uchile.cl/>), RelFinder [24] (<http://www.visualdataweb.org/refinder.php>), DBpedia Search & Find (<http://dbpedia.org/fct/>), SPARKLIS [22] (<http://www.irisa.fr/LIS/ferre/sparklis/>), and our system Elas4RDF (<https://demos.isl.ics.forth.gr/elas4rdf/>).

The results are summarized in Table 5.1. The table has a column for each of the following features: *triple search*, *entity search*, *graph-view*, *faceted search*, *QA*, *relation finder*, *SPARQL* query support. The last column sums up the number of features each system supports: we count each supported feature with 1, and each partially supported feature with 0.5, as an indicator of the *spectrum of the*

*provided access services*. We can see that most systems focus on only one or two access methods, while our system offers four, hence it provides a wider spectrum of access services.

Table 5.1: Search Systems over DBpedia

System	Triple retrieval	Entity search	Graph view	Faceted search	QA	Relation finder	SPARQL support	SUM
LOTUS [27] (no online demo)	Yes	No	No	No	No	No	No	1/7
GraFa [42]	No	No	No	Yes	No	No	No	1/7
RelFinder [24]	No	Partial (through auto completion)	Partial (only of related entities)	No	No	Yes	No	1/7
DBpedia Search & Find	Yes (no images)	No	No	Partial (simple)	No	No	Partial (query display)	2/7
SPARKLIS [22]	No	No	No	Yes (Very Expressive)	No	No	Yes	2/7
Elas4RDF	Yes	Yes	Yes	No	Yes	No	No	4/7

### 5.1.2 Efficiency

The efficiency of the back-end search service (i.e. of the ranking service) was evaluated in [31]. Here we focus on the cost for providing the multiple perspectives of the search results. The key point is that the implementation of the perspectives on top of the search service, described in Section 3.1.2, *does not add significant overhead*, preserving the real-time interaction. Furthermore, the triples and entities retrieved from the search service are *cached*, further improving load times when the same query is issued on different perspectives.

In Table 5.2, the average load time of each perspective is displayed (with and without caching), considering 10 queries of varying length from 1 to 8 words and using an instance of the system that runs on a machine with 6 physical cores and maximum memory allocation size set to 8GB. We can see that even without caching all responses are returned in less than 3 seconds, while with caching enabled, the average time is around 150 ms.

Table 5.2: Average load times for each perspective

Perspective	Triples	Entities	Graph	Schema	QA
Without caching	980 ms	2,582 ms	1,018 ms	924 ms	2,869 ms
With caching	145 ms	124 ms	91 ms	175 ms	118 ms

### 5.1.3 Evaluation of Effectiveness

Another evaluation aspect is the *effectiveness* of the system, i.e. its capability to fulfill the information needs of the user. Note here that, since one can use his own retrieval, ranking or visualisation method in any of the fundamental perspectives, evaluating the performance of the method used in each different tab is out of the scope of this work. As regards the implementation of the tabs in our prototype (described in Section 3.2), the ranking of the entities in the *entities tab* has been extensively evaluated in [31], demonstrating a high performance. This provides a very positive evidence about the quality of the triples that feed all tabs, in the sense that if triple-ranking were not effective, then it would be hard for the *entities tab* to be effective. More importantly, the results of the user study (that we shall see in Section 5.1.4) validate the good quality of the results shown in each tab. Specifically, the large majority of users managed to find correct answers for most of the requested tasks. That would be impossible if the majority of the results in the tabs were irrelevant (more about the user study below in Section 5.1.4).

### 5.1.4 Evaluation with Users

Since there is no dataset that could be used for evaluating the particular multi-perspective interaction we decided to carry out a task-based evaluation with users. Specifically, we wanted to understand how users would use such a system, whether they find useful and/or like the multi-perspective approach, and for collecting general and specific feedback.

#### 5.1.4.1 Information Seeking Tasks

Since we are in keyword-search setting (and not in a structured query building process), we selected a number of tasks that have IR nature, and at the same time are not trivial (some of them are hard to answer, and/or DBpedia has related but not exactly the requested information). We also tried to capture various kinds of information needs, while keeping the list of tasks short for attracting more participants. The selected 11 tasks are shown in Table 5.3. They include queries of various kinds (entity property queries, entity relation queries, fact checking queries, entity list queries). In total, answering these questions requires at least 30 minutes.

Table 5.3: Evaluation Tasks

ID	Task
T1	Is there any person that is fisherman, writer and poet? Provide at least 3 related names (or URIs).
T2	Is there any writer and astronaut from Russia? Provide related names or URIs.
T3	Find information that relates Albert Einstein with Stephen Hawking.
T4	Find if El Greco was influenced by Michelangelo.
T5	Is there any reference of Freud to the ancient Greece?
T6	How is Mars related to Crete?
T7	Find mathematicians related to Pisa.
T8	Find painters of the Ancient Greece.
T9	Are there drugs that contain aloe?
T10	Which cities does the Weser flow through?
T11	Find at least 5 rivers of Greece.

#### 5.1.4.2 Participants, Questionnaire and Results

We invited by email various persons to participate in the evaluation voluntarily. The users were asked to carry out the tasks and to fill (anonymously) the prepared questionnaire. No training material was given to them, and the participation to this evaluation was optional (invitation by email). Eventually, 25 persons participated (from May 5, 2020 to May 18, 2020). The number was sufficient for our purposes since, according to [21], 20 evaluators are enough for getting more than 95% of the usability problems of a user interface. In numbers, the participants were 32% female and 68% male, with ages ranging from 20 to 54 years; the distribution is almost uniform, only the age of 23 is the more frequent 20%, as shown in Figure 5.1.

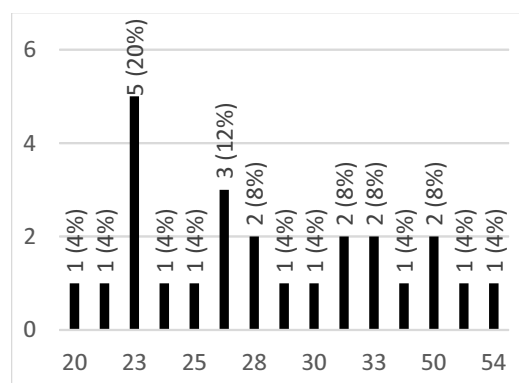


Figure 5.1: Age distribution of participants

As regards occupation and skills, all have studied Computer Science, except one

Physicist. In detail, 20% were undergraduate students, 15% of them postgraduate computer science students, and the rest computer engineers, professionals and researchers. Students came from at least 3 different universities, while 40% of all the participants have never used DBpedia before. The questionnaire is shown below, enriched with the results of the survey in the form of percentages written in bold:

- E1 *How would you rate the Triples tab?:* Very Useful (**40%**), Useful (**44%**), Little Useful (**16%**), Not Useful (**0%**)
- E2 *How would you rate the Entities tab?:* Very Useful (**44%**), Useful (**28%**), Little Useful (**24%**), Not Useful (**4%**)
- E3 *How would you rate the Graph tab?:* Very Useful (**32%**), Useful (**52%**), Little Useful (**12%**), Not Useful (**4%**)
- E4 *How would you rate the Schema tab?:* Very Useful (**16%**), Useful (**40%**), Little Useful (**36%**), Not Useful (**8%**)
- E5 *How would you rate the QA tab?:* Very Useful (**16%**), Useful (**36%**), Little Useful (**40%**), Not Useful (**8%**)
- E6 *Did you find it useful that the system offers multiple perspective of the search results?:* Very much (**48%**), Fair (**48%**), Not that Useful (**4%**), Not Useful (**0%**)
- E7 *Mark the perspective(s) that you think are redundant:* Triples Tab (**0%**), Entities Tab (**8%**), Graph Tab (**8%**), Schema Tab (**40%**) QA Tab (**16%**) All tabs are useful, none is redundant (**44%**)
- E8 *Have you used DBpedia before:* Never (**40%**), Only a few times (without using SPARQL) (**16%**), Quite a lot (I have used SPARQL to query it) (**44%**).
- E9 *How would you rate the entire system?* Very Useful (**32%**), Useful (**60%**), Little Useful (**8%**), Not Useful (**0%**)
- E10 *You can report here errors, problems, or recommendations.* (free text of unlimited length)

#### 5.1.4.3 Results Analysis and Discussion

**User Ratings.** As regards *ratings*, most users appreciated the multi-perspective approach (the positive options of E6, Very Much and Fair, sum to 96%). Moreover, all tabs received positive results by some users. By adding the percentages of Very Useful and Useful, the ranked list of *more preferred* tabs is:

$\langle \{ \text{GraphTab (84\%)}, \text{TriplesTab (84\%)} \}, \text{EntitiesTab (72\%)}, \text{SchemaTab (56\%)}, \text{QATab(52\%)} \rangle$ .

The *less preferred* tabs, according to the sum of Little Useful and Not Useful percentages, is:

$\langle \text{QATab (48\%), SchemaTab (44\%), EntitiesTab (28\%), \{GraphTab (16\%), TriplesTab (16\%\}} \rangle$ .

Note that these numbers correspond to the percentages of users that *would not be satisfied* if only the corresponding perspective were provided to them.

It is also clear that different users have different preferences for perspectives: there are persons that rated the Schema Tab as Very Useful, while others marked it as Redundant. Probably this depends on the background of the participants: a person with no knowledge of RDF would not be able to understand (and exploit) the notion of schema, and we have seen that 20% of the participants were undergraduate and 40% have never used DBpedia. This is also evident from Figure 5.2 that depicts the sum of Very Useful and Useful percentages per tab; the black bars correspond to the users that had never used DBpedia, while the white bars correspond to the users that had used DBpedia before.

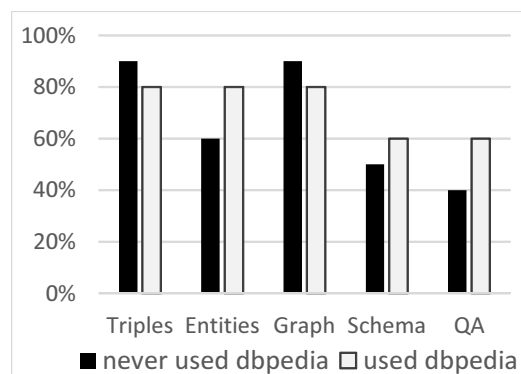


Figure 5.2: ‘Very Useful’ and ‘Useful’ preference percentages per tab and category of users.

By looking at the responses of the questionnaire, we can see that the group of users that had *never used DBpedia*, preferred the Triples Tab and the Graph Tab (40% found them Very Useful, 50% Useful, and 10% Little Useful, for both tabs), and the least useful tab for them was the Schema Tab (10% Very Useful, 40% Useful, and 50% Little Useful), because a basic understanding of the RDF data model is required to use it. Regarding this user group’s opinion of the multi-perspective approach, 30% found it to be Very Useful, and 60% found it Fair. Only one user did not find the approach useful. Also, 50% of these users responded that None of the perspectives are redundant.

**Statistical Significance.** As regards *statistical significance*, by assuming as *positive* the options Very Useful and Useful, and as *negative* the options Little Useful and Not Useful, the lower bound of *Wilson score confidence interval* shows that



with 95% confidence, the percentage of users (of the entire community) that would upvote each perspective would be:

⟨ TriplesTab (65%), GraphTab (65%), EntitiesTab (52%), SchemaTab (37%), QATab (33%) ⟩

Now by considering *all* 4 options quantified as: Very Useful (4), Useful (3), Little Useful (2), Not Useful (1), we can use *Bayesian Approximation* to compute the *expected average rating* for each perspective, in the scale 1 (Worst) - 4 (Best), in the entire community of users. These expected ratings are:

⟨ TriplesTab (2.84), GraphTab (2.73), EntitiesTab (2.69), SchemaTab (2.30), QATab (2.27) ⟩

where a perspective with score  $X$  means that it will have an average rating greater than  $X$ , with 95% confidence.

**Task Performance.** As regards *task performance*, i.e., the responses to the 11 tasks, from the  $11 \times 25 = 275$  responses, 46 (16.7%) reported failure to find the requested information. The failure rate was 20.9% in the (10) users that had never used DBpedia, and 13.9% in the rest (15) users. As shown in Figure 5.3, the participants faced problems, mainly in T2, T4, T5: T2 is tricky (there is such a space-engineer not astronaut), while T4 and T5 are hard to answer, due to dataset issues (non existing information, `wikiPageWikiLink` with no explanation) therefore these cannot be considered as failures of the system. Another interesting observation is that for most tasks *inexperienced users were almost as successful as experienced ones*.

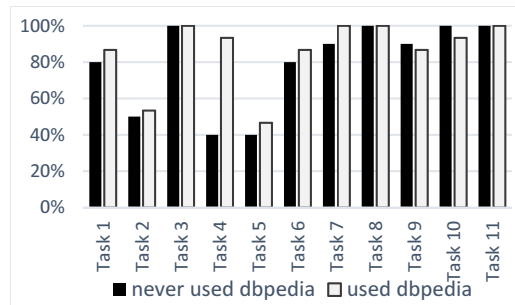


Figure 5.3: Success rates for experienced and inexperienced users.

**Free form Feedback.** With respect to the *free form feedback*, 18 of the 25 users provided very interesting and lengthy comments. For reasons of space, here we only summarize the main ones. In general, they (a) spotted problems related to the DBpedia dataset (missing relationships, unexplained `wikiPageWikiLink` relationships, duplicates), and (b) they made suggestions for improving the tabs:

Triples Tab (not score with 1.0 a triple if not all query terms are included in that triple, addition of property filters), Schema Tab (add the more frequent labels in the edges of the schema graph, highlight the query words in the hits), Graph Tab (set the size so that all related entities are shown).

**General Remarks.** Overall, the rating and the feedback that users provided was very positive. Of course, it is not hard to understand that the results depend on the quality of each individual perspective (which in turn depends also on the effectiveness of the underlying search service). Moreover the *order of tabs* affects the results that concern *user preferences*: in information needs that the first tab(s) provide a satisfying answer, the user will not visit the subsequent tabs (or just a few for verification purposes). That means the harder an information need is, the higher the probability the user visits all tabs. However, our main research hypothesis is not related to the comparison of the individuals tabs, but on the usefulness of the multi-perspective approach, and the results of the evaluation provide positive evidence about the value of the multi-perspective approach. Overall, they key finding is that users not familiar with RDF (a) managed to complete the information-seeking tasks (with performance very close to that of the experienced users), and (b) they rated positively the approach.

#### 5.1.4.4 Log Analysis

Since the system became public and was disseminated in social media on April 27, 2020, below we report some points related to the total traffic of the system; not only from the task-based evaluation with users. More than half of the users (102, in total) have interacted with at least 3 different tabs. The most visited tab is the Triples Tab (35.7% of requests for a tab) which is expected since it is the first tab presented to the user, followed by the Entities Tab (19.1%), the Schema Tab (18.7%), the Graph Tab (16.8%), and the QA tab (9.7%). On average, a user issued 4.6 requests per query (where a request involves: clicking a tab, changing page, adjusting the number of shown triples, or clicking a class or property in the schema tab). Also, a user in average performed 6.7 interactions per query in the schema tab. This is expected since the Schema Tab allows for interactive exploration of the data by clicking on classes and predicates, and adjusting the number of retrieved triples.

#### 5.1.4.5 Discussion: Related Systems

To our knowledge, the only system that is currently available and offers unrestricted free-text search (which is the focus of our work) is DBpedia Search & Find (<http://dbpedia.org/fct/>). This system offers a single visualisation of the results, in particular it returns *entities*, so it is like using only the Entities Tab provided by our system. The objective of our evaluation is to investigate if a single visualisation method is enough, what is answered by the user study; if the

Entities Tab were enough, this would be evident in the evaluation results, e.g., in the answers of the questions E1-E7.

## 5.2 Answer Type Prediction Evaluation

### 5.2.1 Evaluation Metrics

We report results for the following metrics:

- *Accuracy*, for category prediction (the percentage of questions classified in the correct category).
- *Precision*, for type prediction (the percentage of the questions for which the top *type* found by the system was one of the types provided in the test dataset, without considering type specificity).
- *Lenient NDCG@k* (with a Linear decay) [3], for resource type prediction.

Lenient NDCG@k, which has been introduced in [3], measures the distance between the predicted type and the most specific type of the answer  $d(t, t_q)$ . Then it converts this distance into a Gain measure, with a linear decay function. The gain is calculated as:  $G(t) = 1 - d(t, t_q)/6$ , where 6 is the maximum depth of the hierarchy. For example, for the question “Which company founded by Fusajiro Yamauchi gives service as Nintendo Network?”, the top 5 classes found as the answer type by our system are: ‘dbo:Company’, ‘dbo:Organisation’, ‘dbo:University’, ‘dbo:Agent’, ‘dbo:RecordLabel’ (in this order). The true types specified on the dataset are: ‘dbo:Company’, ‘dbo:Organisation’, ‘dbo:Agent’. The most specific of these 3 classes is ‘dbo:Company’, so we calculate the gain for each type found by our system using the distance from the class ‘dbo:Company’. Then we compute DCG as:  $DCG_p = gain_1 + \sum_{i=2}^p \frac{gain_i}{\log_2 i}$ . We also compute the ideal DCG ( $iDCG$ ) using the gains of the correct types provided in the dataset, and normalized DCG ( $nDCG$ ) as  $\frac{DCG}{iDCG}$ . Finally we compute and report the average  $nDCG$  over all questions in the test dataset.

### 5.2.2 Results on split of the DBpedia training set

Initially, we had no access to the final test dataset of the SMART challenge, so we used 90% of the DBpedia training set<sup>1</sup> as our training dataset and the remaining 10% as our test dataset. For category prediction and literal type prediction we also use the questions from the training dataset for Wikidata for training the classifiers. Our approach achieved the results shown in Table 5.4. We notice a superior performance of category prediction (96.4% accuracy) and a very high performance of type prediction (83% precision and 79% lenient NDCG@5).

<sup>1</sup><https://github.com/smart-task/smart-dataset/tree/master/datasets/DBpedia>

Running the same experiments without the rewarding mechanism, we notice an around 2% drop in the performance (Lenient NDCG) of literal/resource type prediction.

Table 5.4: Evaluation results

Accuracy (category prediction)	0.964
Precision (literal/resource/boolean type prediction)	0.826
Lenient NDCG@5 with linear decay (literal/resource type prediction)	0.786
Lenient NDCG@10 with linear decay (literal/resource type prediction)	0.778

**Tuning of the k parameter** To find the optimal value for the parameter k, which is the minimum sample size required to include a class in the subset of classes included in the classifier, we evaluated our system using 4 different values: 5, 10, 30 and 50. Table 5.5 shows the number of classes included in the classifier for each different value of k and the corresponding performance. We notice that the best results are obtained using k=10, while the results for all other cases are slightly worse.

Table 5.5: Results for different values of k

Value	Classes	NDCG@5	NDCG@10
5	180	0.775	0.765
10	151	0.786	0.778
30	79	0.785	0.772
50	55	0.785	0.748

**Error analysis.** To better understand the classification performance of category prediction, literal type prediction, and resource type prediction, we inspected their confusion matrices. The results are shown in Table 5.6. As regards category prediction, we see that our system classifies in the correct category 99% of the boolean questions, 92% of the literal questions, and 98% of the resource questions. For literal type classification, our system classifies in the correct type 98.4% of date questions, 99.5% of number questions, and 99.5% of string questions. We notice that, for category prediction, most errors occur between the classes literal and resource. For instance, 41 questions of literal type are misclassified as of type resource. As regards resource type prediction, the table shows the confusion matrix for the top-5 (most frequent) resource classes. We notice that there is significant confusion between the classes City and Country, as well as between the class Person and other classes.

By manually inspecting several of the misclassification cases, we noticed that some of these errors occur on questions where the correct category is very ambiguous, such as the question *“In what area is Fernandel buried at the Passy Cemetery?”* (labeled as a literal question with type ‘string’, while our system classifies it as a resource question of type ‘dbo:Place’), or the type provided in the dataset is wrong, e.g. the question *“What did the pupil of Mencius die of?”* is

Table 5.6: Confusion matrices for *category* (top left), *literal* (top right), and *resource* (bottom) type prediction.

		Actual			Sum
		Boolean	Literal	Resource	
Predicted	Boolean	287	2	5	294
	Literal	1	497	13	511
	Resource	2	41	905	948
Sum		290	540	923	1753

		Actual			Sum
		Date	Number	String	
Predicted	Date	120	0	0	120
	Number	2	182	1	185
	String	0	1	191	192
Sum		122	183	192	497

		Actual					Sum
		Person	City	Country	Award	Organisation	
Predicted	Person	148	4	3	3	0	86
	City	3	67	16	0	0	23
	Country	4	2	42	0	0	17
	Award	1	0	0	37	0	0
	Organization	1	2	5	1	32	42
	Other	15	1	8	3	6	351

labeled as a literal question with type ‘date’, while our system predicts that the question category is resource and ‘dbo:Disease’ is one of the predicted classes.

### 5.2.3 Results over the final DBpedia test set

After the final test dataset was released, we evaluated our system again, using the script provided by the challenge organizers. We obtain the results shown in Table 5.7 (using  $k=30$ ). We notice that the results are very close to those reported for the split on the training dataset (cf. Table 5.4).

Table 5.7: Evaluation results over the final test set

Accuracy (category prediction)	0.962
Lenient NDCG@5 with linear decay (literal/resource type prediction)	0.777
Lenient NDCG@10 with linear decay (literal/resource type prediction)	0.762

### 5.2.4 Efficiency

**Fine-tuning.** We fine-tuned the models on Google Colab<sup>2</sup>, a Jupyter notebook environment that runs in the cloud and offers access to GPUs. With a batch

<sup>2</sup><https://colab.research.google.com/>

size of 32, number of epochs set to 3 and using an Nvidia Tesla K80 GPU, the time required for fine-tuning each classifier is: 49 mins and 25 secs for the resource question type classifier, using 26,259 questions, 27 mins and 51 secs for the question category classifier, using 14,814 questions, and 15 mins and 3 secs for the literal question type classifier, using 8,025 questions.

**Execution.** To classify a question into a category and predict its answer type, we execute the system locally on a machine with 2 cores and 8 GB of RAM, without using a GPU. While the system is running, it requires approximately 2.3 GB of RAM to load the 3 classifiers in memory.

This means that the proposed approach has low main memory requirements.

Moreover, this memory footprint can be further reduced if we use a smaller and lighter language model, such as DistilBERT [55], while sacrificing a small percentage of accuracy. The time required to classify a single question is less than a second (0.17 seconds on average), which is important for the application context that we have in mind (more below). To obtain the system output required to evaluate our system for the SMART challenge, we classified each one of the 4,381 questions provided in the test set sequentially. The process took 12 minutes and 24 seconds.

## 5.3 Question Answering Evaluation

In Section 5.3.1 we evaluate our approach over WebQuestions, a benchmark collection obtained from popular questions asked on the web that are answerable by Freebase, a different knowledge base than DBpedia, which our system retrieves information from, so essentially we evaluate how good our approach for open domain question answering is, while retrieving information from a *different source* and without having been previously trained over this specific dataset.

In Section 5.3.2 we investigate how that task of Answer Type Prediction affects the effectiveness of QA.

In Section 5.3.3 we evaluate how answers from our QA pipeline can contribute to the entity retrieval task over DBpedia Entity dataset [23], and entity ranking in general. In section 5.3.6 we discuss the efficiency of the system and in section 5.3.5 we provide a summary of the evaluation results.

In Section 5.3.4 we evaluate how our QA pipeline could affect entity search.

### 5.3.1 Experiment 1: Webquestions

WebQuestions [5] is a popular dataset for benchmarking QA engines, especially ones that work on structured knowledge bases. It is a dataset of question-answer pairs obtained from non-experts.

This dataset contains 6,642 questions collected using the Google Suggest API to obtain questions that begin with a wh-word and contain exactly one entity. Answers were generated using Amazon Mechanical Turk. The AMT task requested

that workers answer the question using only the Freebase page of the question's entity.

Question: "What countries are part of the UK?"

Answers: "Scotland", "England", "Wales", "Northern Ireland"

To evaluate our approach over this benchmark, we obtained answers from our system for all 2032 questions in the test collection. Then, we compute the following metrics:

- Precision: The percentage of terms in the ground truth that are also terms in the system output, averaged over all questions
- Recall: The percentage of terms in the ground truth that are also terms in the system output, averaged over all questions
- F1: The harmonic mean of precision and recall
- Accuracy: The percentage of questions that received at least one correct answer

For reasons of performance, we limit the number of facts returned by the SPARQL endpoint to 20 (see section 5.3.6). We compute the evaluation scores for different sets of answers of varying confidence by considering only answers that have a score above a specific threshold  $t$  and trying different values for  $t$ . The results are displayed in Table 5.8.

Threshold	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Precision	7.007	16.170	18.607	21.290	23.710	25.261	28.101	31.185	37.543	<i>43.363</i>
Recall	31.263	27.712	29.016	27.894	29.078	30.882	31.279	33.465	34.506	<i>40.477</i>
F1	9.710	16.957	19.074	19.695	21.664	23.224	25.039	28.356	31.443	<i>39.200</i>
Accuracy	<i>53.759</i>	47.597	47.570	46.893	47.697	48.031	47.867	48.765	52.380	52.174

Table 5.8: F1 and Exact scores over WebQuestions

We can see that a threshold value of 0.9 yields the best rewards for Precision, Recall and F1 Accuracy is higher for a threshold value of 0 because including all answers (score  $\geq 0$ ) leads to a higher probability that at least one correct answer will be included. Our system performs well, similar to the highest scores published in [codalab](https://worksheets.codalab.org/worksheets/0xba659fe363cb46e7a505c5b6a774dc8a)<sup>3</sup>, even though our system has not been previously trained on this specific dataset. The highest is 0.557 average F1 score obtained by [28]. However that system was based on freebase and it was trained on this specific dataset, while our approach relies on a different knowledge base (DBpedia) and does not require training on this dataset.

<sup>3</sup><https://worksheets.codalab.org/worksheets/0xba659fe363cb46e7a505c5b6a774dc8a>

### 5.3.2 Without Answer Type Prediction

To examine the value that is added to this Question Answering pipeline by the answer type prediction component, we evaluate our system over the same dataset and metrics as section 5.3.1, but without using the answer type prediction component.

Therefore, in this case, the text provided to the extractive Question Answering model is the textual description of each entity retrieved by the entity search system, without being expanded with facts matching the answer type, as described in sections 4.3 and 4.4.2. We report the following results using the best value for the answer score threshold (0.9) determined in Experiment 1. Precision: 37.356, Recall: 32.966, F1: 32.181 and Accuracy: 48.122. We can see that in this case results are 4-8% lower.

### 5.3.3 Experiment 2: DBpedia Entity: QA

DBpedia-Entity is a standard test collection for entity search over the DBpedia knowledge base. It is meant for evaluating retrieval systems that return a ranked list of entities (DBpedia URIs) in response to a free text user query. This dataset contains named entity queries, keyword queries, list queries and question answering queries. We focus on the subset of question answering queries, which contains 140 queries from QALD-2 (Question Answering over Linked Data) challenge [39] These are natural language questions that can be answered by DBpedia entities, for example, "Who is the mayor of Berlin?" Each entity/answer is accompanied by a score in 3-point relevance scale:

- Highly relevant (2): The entity is a direct answer to the query
- Relevant (1): The entity can be shown as an answer to the query, but not among the top results
- Irrelevant (0)

For our experiments we only use the relevant entities, i.e. those with score of 1 or 2.

Other systems that report results over this benchmark use the NDCG@10 and NDCG@100 metrics because they focus on entity search. In our case, since we use this benchmark for question answering, we also compute Precision scores, in order to find out whether the top answers returned by our system are relevant to the query.

We evaluate the performance of our approach as a standalone QA system for the task of entity search. To do this we compute the Precision scores at the values 1, 3 and 5. The results obtained for varying values of answer score threshold are given in Table 5.9.

The results are good in the sense that more than 69% of answers are relevant to their corresponding questions.



Table 5.9: Precision @1, @3, @5 for varying answer score threshold over DBpedia Entity

Threshold	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
P@1	33.573	49.331	55.432	55.641	55.938	56.190	58.857	57.241	57.273	<b>69.444</b>
P@3	27.840	42.006	48.265	47.951	46.595	50.813	52.905	51.207	52.348	<b>69.444</b>
P@5	24.543	41.008	47.147	46.836	45.768	49.716	51.905	51.207	52.348	<b>69.444</b>

Below (in §5.3.4) we shall also explore how this component can improve the performance of a dedicated Entity Search system by adding the set of answers to the set of entities retrieved by the search system.

### 5.3.4 Experiment 3: DBpedia Entity: QA+RANKING

We use the DBpedia Entity dataset [23] to evaluate the performance of Elas4RDF as an entity retrieval system for Question Answering. Our goal for this experiment is to find out how the answers retrieved using this work affect the performance of Elas4RDF for Natural Question Answering tasks. Therefore, we use the group of queries from the DBpedia Entity collection that are Natural Language Questions (e.g. "Who is the mayor of Berlin?"). This group contains 140 of the 467 total queries in the benchmark. Over this group of queries, we compute the NDCG scores (@10 and @100) for:

- Entities retrieved by the Elas4RDF search service [31]
- Entities retrieved by the Elas4RDF search service *combined* with high scoring answers from the QA tab

To fuse the set of answers from the QA tab with the set of entities from the search service, we retrieve all entities from the search service, then we select a number ( $a$ ) of answers from the QA tab and add them to the list of entities. Each entity has a score computed by the search service and each answer a score computed by the QA component. All scores are in the range scale of 0 to 1. We try two approaches to compute these scores:

- I Keep the score from each entity and answer as computed by the entity search system and question answering component.
- II Sum scores for entities in both rankings.

Finally, we sort the list of combined entities and answers by these scores, and we keep the top 10 or 100 results, depending on the NDCG metric that we wish to compute.

The results are displayed on Tables 5.10,5.11. The row baseline corresponds to results for the entities returned by the QA component when no additional answers have been added. The other rows correspond to results for varying number of top answers from the QA component added to the baseline. We can see that including

answers from the QA tab to the list of entities improves the NDCG score in all cases. The highest improvement occurs for the NDCG@100 metric when the 5 top answers are added to the list of entities.

Answers added	NDCG@100		NDCG@10	
	Score	Difference	Score	Difference
0 (baseline)	0.325	0	0.325	0
1	0.352	0.027	0.352	0.027
3	0.372	0.047	0.353	0.028
5	<b>0.384</b>	<b>0.059</b>	0.354	0.029
10	0.382	0.057	0.353	0.028

Table 5.10: NDCG scores over Natural Language Questions of the DBpedia Entity collection for approach I: Keep Initial Scores

Answers added	NDCG@100		NDCG@10	
	Score	Difference	Score	Difference
0 (baseline)	0.325	0	0.325	0
1	0.355	0.03	0.355	0.03
3	0.375	0.05	0.358	0.033
5	<b>0.387</b>	<b>0.062</b>	0.357	0.032
10	0.386	0.061	0.356	0.031

Table 5.11: NDCG scores over Natural Language Questions of the DBpedia Entity collection for approach II: Sum Scores

As regards the comparison of approaches I and II, we can see that approach II obtains better results with a small difference (0.003 improvement of NDCG@100 using 5 answers). The reason for this is that approach II handles cases where an answer is returned by both the entity search system and the QA component.

Overall we can say that our QA pipeline could be considered as a method for ranking entities in the context of entity search. In comparison to a “plain” entity search, our pipeline is computationally more expensive because of the memory and time requirements added by the answer type prediction, entity expansion and answer extraction components (see section 5.3.6), but it can give better results in certain cases. Specifically, it improves NDCG@100 by 6.2%

### 5.3.5 Executive Summary

We can summarize the evaluation results as follows:

We have shown that our approach for open domain Question Answering can obtain satisfactory results 54% accuracy, 39% F1 over popular question answering benchmarks, similar to other state of the art approaches, something that is very interesting since it has not been trained on specific datasets and it has used different information sources than the ones intended by the benchmarks. We have also showed how the answer type prediction and entity expansion stages improve

Precision by 6%, Recall by 7% and F1 score by 7% (over WebQuestions). In addition we have shown that our approach can be used in combination with an entity search system to improve entity search tasks by 6% NDCG@100 (over DBpedia Entity dataset).

### 5.3.6 Efficiency

While running, the system’s memory footprint is approximately 1.4 GB, and it takes up 511 MB of space to store all required models. To evaluate the time required to answer a question, we record times for each step of the pipeline as well as the overall time required to provide the final answers for all (2032) questions in the webquestions dataset (section 5.3.1) and compute their average. This experiment was performed on a machine with 6 physical cores running Debian Linux. We found that the average time for the Answer Type Prediction stage is 0.1 seconds, for the Entity Expansion stage 3.9 seconds, for the Answer Extraction stage 4.3 seconds and the overall average time required to provide the final answers is 8.3 seconds. We can see that answer type prediction is the fastest stage, because it uses a lighter language model (DistilBERT) while the other 2 stages are quite slower, because of the response time of the SPARQL queries for Entity Expansion and the larger language model used for Answer Extraction.

Table 5.12: Average time cost for each stage of the pipeline

Answer Type Prediction	Entity Expansion	Answer Extraction
0.1 sec (1.2%)	3.9 sec (47%)	4.3 sec (51.8%)

To improve efficiency, one could use a locally hosted triple store that would provide a faster response time. Moreover one could speed up the answer extraction stage by using the RoBERTa model on a GPU.

The number of returned facts could also be limited by setting a maximum response size, or using more strict SPARQL queries (e.g. by ignoring the equivalent classes), or using equivalence-aware indexes like those described in [45].



## Chapter 6

# Conclusion & Future Work

Keyword search over RDF datasets is a challenging task. To help the user find and explore the requested information, we have investigated a *multi-perspective* approach for keyword search in which *multiple perspectives (tabs)* are used for the presentation of the search results, each tab stressing a different aspect of the hits. The user can easily inspect all tabs and get a better overview and understanding of the search results. We have focused on five fundamental (i.e. KB and task agnostic) perspectives (triples, entities, graph, schema and QA) and we have implemented this approach over a general keyword search engine over DBpedia.

With respect to related systems that provide keyword access over DBpedia, we could say that the proposed approach is probably the more complete with respect to the access methods that it offers. The task-based evaluation with users has shown that (a) 96% of the users liked the multi-perspective approach (48% Very much, 48% Fair), (b) the success rate of all users was very high (even of those not familiar with RDF), (c) users seem to have quite different preferences on perspectives.

There are several issues that are worth further work and research. We plan to improve the Graph Tab, and to add additional tabs. Moreover we would like to investigate how to exploit the equivalence (`owl:sameAs`) relationships.

Regarding Question Answering, since this task in vague or complex information needs is hard to be adequate, satisfying and pleasing for end users, in this thesis we have investigated an approach where QA complements a general purpose interactive keyword search system over RDF. We detailed a pipeline for QA in that context, that involves search services, SPARQL, and pre-trained neural networks.

We have evaluated our approach over two different datasets and showcased the value it provides for question answering and entity search tasks.

We have shown that our approach for open domain Question Answering can obtain satisfactory results 54% accuracy, 39% F1 over a popular question answering benchmark (WebQuestions), similar to other state of the art approaches, even if no training has been performed over this particular benchmark, moreover, it has used different information sources (i.e. DBpedia) than the ones intended by the

benchmark (FreeBase). We have also showed how the Answer Type Prediction and Entity Expansion stages, do improve Precision by 6%, Recall by 7% and F1 score by 7% (over WebQuestions). In addition we have shown that our approach can be used in combination with an entity search system to improve entity search tasks by 6% NDCG@100 (over DBpedia Entity dataset).

As regards future research, it is worth investigating methods for improving efficiency. In addition one could use more questions from the DBpedia Entity dataset (not only QA-related), to see whether the entity ranking is improved in all cases. The system is available to all at <https://demos.isl.ics.forth.gr/elas4rdf/>.

# Bibliography

- [1] Zahra Abbasiantaeb and Saeedeh Momtazi. Text-based question answering from information retrieval and deep neural network perspectives: A survey, 2020.
- [2] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer, 2007.
- [3] Krisztian Balog and Robert Neumayer. Hierarchical target type identification for entity-oriented queries. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 2391–2394, 2012.
- [4] Caleb Belth, Xinyi Zheng, Jilles Vreeken, and Danai Koutra. What is Normal, What is Strange, and What is Missing in a Knowledge Graph: Unified Characterization via Inductive Summarization. In *Proceedings of The Web Conference*, pages 1115–1126, 2020.
- [5] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.
- [6] Nikos Bikakis and Timos Sellis. Exploration and visualization in the web of big linked data: A survey of the state of the art. *arXiv preprint arXiv:1601.08059*, 2016.
- [7] Roi Blanco, Peter Mika, and Sebastiano Vigna. Effective and efficient entity search in rdf data. In *International Semantic Web Conference*, pages 83–97. Springer, 2011.
- [8] Gong Cheng and Yuzhong Qu. Searching linked objects with falcons: Approach, implementation and evaluation. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 5(3):49–70, 2009.
- [9] Wanyun Cui, Yanghua Xiao, Haixun Wang, Yangqiu Song, Seung-won Hwang, and Wei Wang. KBQA: Learning Question Answering over QA Corpora and Knowledge Bases. *Proc. VLDB Endow.*, 10(5):565–576, 2017.

- [10] Aba-Sah Dadzie and Emmanuel Pietriga. Visualisation of linked data—reprise. *Semantic Web*, 8(1):1–21, 2017.
- [11] Renaud Delbru, Stephane Campinas, and Giovanni Tummarello. Searching web data: An entity retrieval and high-performance indexing model. *Journal of Web Semantics*, 10:33–58, 2012. Web-Scale Semantic Information Processing.
- [12] Renaud Delbru, Nur Aini Rakhmawati, and Giovanni Tummarello. Sindice at semsearch 2010. In *WWW*. Citeseer, 2010.
- [13] Jia Deng, Jonathan Krause, Alexander C Berg, and Li Fei-Fei. Hedging your bets: Optimizing accuracy-specificity trade-offs in large scale visual recognition. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3450–3457. IEEE, 2012.
- [14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.
- [15] Eleftherios Dimitrakis, Konstantinos Sgontzos, and Yannis Tzitzikas. A survey on question answering systems over linked data and documents. *Journal of Intelligent Information Systems*, pages 1–27, 2019.
- [16] Dimitar Dimitrov, Erdal Baran, Pavlos Fafalios, Ran Yu, Xiaofei Zhu, Matthäus Zloch, and Stefan Dietze. Tweetscov19—a knowledge base of semantically annotated tweets about the covid-19 pandemic. In *29th ACM International Conference on Information and Knowledge Management (CIKM 2020)*, 2020.
- [17] Dennis Dosso and Gianmaria Silvello. A scalable virtual document-based keyword search system for RDF datasets. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 965–968, 2019.
- [18] Dennis Dosso and Gianmaria Silvello. Search Text to Retrieve Graphs: A Scalable RDF Keyword-Based Search System. *IEEE Access*, 8:14089–14111, 2020.
- [19] Shady Elbassuoni and Roi Blanco. Keyword search over rdf graphs. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 237–242. ACM, 2011.
- [20] Shady Elbassuoni, Maya Ramanath, Ralf Schenkel, and Gerhard Weikum. Searching rdf graphs with sparql and keywords. *IEEE Data Eng. Bull.*, 33(1):16–24, 2010.



- [21] Laura Faulkner. Beyond the five-user assumption: Benefits of increased sample sizes in usability testing. *Behavior Research Methods, Instruments, & Computers*, 35(3):379–383, 2003.
- [22] Sébastien Ferré. Sparklis: an expressive query builder for SPARQL endpoints with guidance in natural language. *Semantic Web*, 8(3):405–418, 2017.
- [23] Faegheh Hasibi, Fedor Nikolaev, Chenyan Xiong, Krisztian Balog, Svein Erik Bratsberg, Alexander Kotov, and Jamie Callan. Dbpedia-entity v2: A test collection for entity search. In *SIGIR*, pages 1265–1268. ACM, 2017.
- [24] Philipp Heim, Sebastian Hellmann, Jens Lehmann, Steffen Lohmann, and Timo Stegemann. RelFinder: Revealing Relationships in RDF Knowledge Bases. In *Semantic Multimedia*, volume 5887, pages 182–187. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.
- [25] Xin hu, Jiangli Duan, and Depeng Dang. Natural language question answering over knowledge graph: the marriage of sparql query and keyword search. *Knowledge and Information Systems*, 01 2021.
- [26] Filip Ilievski, Wouter Beek, Marieke Van Erp, Laurens Rietveld, and Stefan Schlobach. LOTUS: Linked open text unleashed. In *COLD*, 2015.
- [27] Filip Ilievski, Wouter Beek, Marieke van Erp, Laurens Rietveld, and Stefan Schlobach. LOTUS: Adaptive text search for big linked data. In *ESWC*, pages 470–485. Springer, 2016.
- [28] Sarthak Jain. Question answering over knowledge base using factual memory networks. In *Proceedings of the NAACL Student Research Workshop*, pages 109–115, San Diego, California, June 2016. Association for Computational Linguistics.
- [29] Mohamad Yaser Jaradeh, Allard Oelen, Kheir Eddine Farfar, Manuel Prinz, Jennifer D’Souza, Gábor Kismihók, Markus Stocker, and Sören Auer. Open research knowledge graph: Next generation infrastructure for semantic scholarly knowledge. In *Proceedings of the 10th International Conference on Knowledge Capture*, pages 243–246, 2019.
- [30] Thomas Johnson. Indexing linked bibliographic data with JSON-LD, BibJSON and Elasticsearch. *Code4lib Journal*, 19:1–11, 2013.
- [31] Giorgos Kadilierakis, Pavlos Fafalios, Panagiotis Papadakos, and Yannis Tzitzikas. Keyword Search over RDF using Document-centric Information Retrieval Systems. In *Extended Semantic Web Conference (ESWC’2020)*, 2020.
- [32] Giorgos Kadilierakis, Christos Nikas, Pavlos Fafalios, Panagiotis Papadakos, and Yannis Tzitzikas. Elas4RDF: Multi-perspective triple-centered keyword search over RDF using elasticsearch. *Extended Semantic Web Conference (ESWC) – Posters & Demonstrations Track*, 2020.

- [33] Kalliopi Kontiza and Antonis Bikakis. Web search results visualization: Evaluation of two semantic search engines. In *International Conference on Web Intelligence, Mining and Semantics (WIMS'14)*, pages 1–12, 2014.
- [34] Vangelis Kritsotakis, Yannis Roussakis, Theodore Patkos, and Maria Theodoridou. Assistive query building for semantic data. In *SEMANTICS Posters&Demos*, 2018.
- [35] Petri Leskinen, Goki Miyakita, Mikko Koho, Eero Hyvönen, et al. Combining faceted search with data-analytic visualizations on top of a sparql endpoint. In *VOILA@ ISWC*, pages 53–63, 2018.
- [36] Xiaoqing Lin, Fu Zhang, and Danling Wang. Rdf keyword search using multiple indexes. *Filomat*, 32(5):1861–1873, 2018.
- [37] Xitong Liu and Hui Fang. A study of entity search in semantic search workshop. In *Proc. of the 3rd Intl. Semantic Search Workshop*, 2010.
- [38] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.
- [39] Vanessa Lopez, Christina Unger, Philipp Cimiano, and Enrico Motta. Evaluating question answering over linked data. *Web Semant.*, 21:3–13, August 2013.
- [40] Xiaolu Lu, Soumajit Pramanik, Rishiraj Saha Roy, Abdalghani Abujabal, Yafang Wang, and Gerhard Weikum. Answering Complex Questions by Joining Multi-Document Evidence with Quasi Knowledge Graphs. In *ACM SIGIR*, pages 105–114, 2019.
- [41] Nandana Mihindukulasooriya, Mohnish Dubey, Alfio Gliozzo, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, and Ricardo Usbeck. SeMantic Answer Type prediction task (SMART) at ISWC 2020 Semantic Web Challenge. *CoRR/arXiv*, abs/2012.00555, 2020.
- [42] José Moreno-Vega and Aidan Hogan. Grafa: Scalable faceted browsing for rdf graphs. In *International Semantic Web Conference*, pages 301–317. Springer, 2018.
- [43] Michalis Mountantonakis and Yannis Tzitzikas. LODsyndesis: global scale knowledge services. *Heritage*, 1(2):335–348, 2018.
- [44] Michalis Mountantonakis and Yannis Tzitzikas. Large-scale Semantic Integration of Linked Data: A Survey. *ACM Computing Surveys (CSUR)*, 52(5):103, 2019.

- [45] Michalis Mountantonakis and Yannis Tzitzikas. Content-based union and complement metrics for dataset search over rdf knowledge graphs. *Journal of Data and Information Quality (JDIQ)*, 12(2):1–31, 2020.
- [46] Christos Nikas, Pavlos Fafalios, and Yannis Tzitzikas. Two-stage semantic answer type prediction for question answering using BERT and class-specificity rewarding. In Nandana Mihindukulasooriya, Mohnish Dubey, Alfio Gliozzo, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, and Ricardo Usbeck, editors, *Proceedings of the SeMantic Answer Type prediction task (SMART) at ISWC 2020 Semantic Web Challenge co-located with the 19th International Semantic Web Conference (ISWC 2020), Virtual Conference, November 5th, 2020*, volume 2774 of *CEUR Workshop Proceedings*, pages 19–28. CEUR-WS.org, 2020.
- [47] Christos Nikas, Giorgos Kadilierakis, Pavlos Fafalios, and Yannis Tzitzikas. Keyword Search over RDF: Is a Single Perspective Enough? *Big Data and Cognitive Computing*, 4(3):22, August 2020.
- [48] Natasha Noy, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, and Jamie Taylor. Industry-scale knowledge graphs: lessons and challenges. *Queue*, 17(2):48–75, 2019.
- [49] Dominic Oldman and Diana Tanase. Reshaping the knowledge graph by connecting researchers, data and practices in researchspace. In *International Semantic Web Conference*, pages 325–340. Springer, 2018.
- [50] Hanane Ouksili, Zoubida Kedad, Stéphane Lopes, and Sylvaine Nugier. Using patterns for keyword search in rdf graphs. In *EDBT/ICDT Workshops*, 2017.
- [51] Maria-Evangelia Papadaki, Yannis Tzitzikas, and Nicolas Spyrtatos. Analytics over rdf graphs. In *International Workshop on Information Search, Integration, and Personalization*, pages 37–52. Springer, 2019.
- [52] José R Pérez-Agüera, Javier Arroyo, Jane Greenberg, Joaquin Perez Iglesias, and Victor Fresno. Using bm25f for semantic search. In *Proceedings of the 3rd international semantic search workshop*, page 2. ACM, 2010.
- [53] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don’t know: Unanswerable questions for SQuAD, 2018.
- [54] Mohamad Rihany, Zoubida Kedad, and Stéphane Lopes. Keyword Search Over RDF Graphs Using WordNet. In *BDCSIntell*, pages 75–82, 2018.
- [55] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2019.
- [56] Vinay Setty and Krisztian Balog. Semantic answer type prediction using BERT IAI at the ISWC SMART task 2020. In Nandana Mihindukulasooriya,

- Mohnish Dubey, Alfio Gliozzo, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, and Ricardo Usbeck, editors, *Proceedings of the SeMantic AnswER Type prediction task (SMART) at ISWC 2020 Semantic Web Challenge co-located with the 19th International Semantic Web Conference (ISWC 2020), Virtual Conference, November 5th, 2020*, volume 2774 of *CEUR Workshop Proceedings*, pages 10–18. CEUR-WS.org, 2020.
- [57] Saeedeh Shekarpour, Edgard Marx, Axel-Cyrille Ngonga Ngomo, and Sören Auer. Sina: Semantic interpretation of user queries for question answering on interlinked data. *Journal of Web Semantics*, 30:39–51, 2015. Semantic Search.
- [58] Kuldeep Singh, Ioanna Lytra, Arun Sethupat Radhakrishna, Saeedeh Shekarpour, Maria-Esther Vidal, and Jens Lehmann. No one is perfect: Analysing the performance of question answering components over the dbpedia knowledge graph, 2020.
- [59] Martin G Skjæveland. Sgvizler: A javascript wrapper for easy visualization of sparql result sets. In *Extended Semantic Web Conference*, pages 361–365. Springer, 2012.
- [60] Christian Stab, Kawa Nazemi, Matthias Breyer, Dirk Burkhardt, and Jörn Kohlhammer. Semantics visualization for fostering search result comprehension. In *Extended Semantic Web Conference*, pages 633–646. Springer, 2012.
- [61] Yannis Tzitzikas, Nikos Manolis, and Panagiotis Papadakos. Faceted exploration of RDF/S datasets: a survey. *Journal of Intelligent Information Systems*, 48(2):329–364, 2017.
- [62] Hernán Vargas, Carlos Buil-Aranda, Aidan Hogan, and Claudia López. RDF Explorer: A Visual SPARQL Query Builder. In *ISWC*, pages 647–663. Springer, 2019.
- [63] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.
- [64] David S Wishart, Yannick D Feunang, An C Guo, Elvis J Lo, Ana Marcu, Jason R Grant, Tanvir Sajed, Daniel Johnson, Carin Li, Zinat Sayeeda, et al. Drugbank 5.0: a major update to the drugbank database for 2018. *Nucleic acids research*, 46(D1):D1074–D1082, 2018.
- [65] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame,

Quentin Lhoest, and Alexander M. Rush. Huggingface’s transformers: State-of-the-art natural language processing, 2020.

- [66] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding, 2019.