NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS
POSTGRADUATE PROGRAM IN
DATA SCIENCE AND INFORMATION TECHNOLOGIES

MSc THESIS

A Greek Question Answering System over Knowledge Graphs

Kriton A. Pittos

Supervisor: **Manolis Koubarakis,** Professor

ATHENS

JULY 2024

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ
ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΕΠΙΣΤΗΜΗ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΕΣ ΠΛΗΡΟΦΟΡΙΑΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Σύστημα Ερωταπαντήσεων για την Ελληνική Γλώσσα πάνω σε Γράφους Γνώσης

Κρίτων Α. Πίττος

Επιβλέπων: **Μανόλης Κουμπαράκης,** Καθηγητής

ΑΘΗΝΑ

ΙΟΥΛΙΟΣ 2024

MSc THESIS

A Greek Question Answering System over Knowledge Graphs

Kriton A. Pittos
S.N.: 7115152100014

SUPERVISOR: **Manolis Koubarakis,** Professor

EXAMINATION COMMITTEE: **Manolis Koubarakis**, Professor NKUA
Mema Roussopoulos, Professor NKUA
Alexandros Ntoulas, Assistant Professor NKUA

Examination Date: 9 July 2024

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Σύστημα Ερωταπαντήσεων για την Ελληνική Γλώσσα πάνω σε Γράφους Γνώσης

Κρίτων Α. Πίττος
Α.Μ.: 7115152100014

ΕΠΙΒΛΕΠΩΝ: **Μανόλης Κουμπαράκης,** Καθηγητής

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ: **Μανόλης Κουμπαράκης**, Καθηγητής ΕΚΠΑ
Μέμα Ρουσσοπούλου, Καθηγήτρια ΕΚΠΑ
Αλέξανδρος Ντούλας, Επίκουρος Καθηγητής ΕΚΠΑ

Ημερομηνία Εξέτασης: 9 Ιουλίου 2024

ABSTRACT

In the rapidly evolving fields of information retrieval and artificial intelligence, question answering systems have become increasingly popular and have seen significant advancements over the years. These systems provide an intuitive method for querying structured data sources allowing users to ask questions in natural language and receive accurate and relevant answers. Utilizing the power of Knowledge Graphs users can get the direct answer rather than a list of potential results like popular search engines used to do.

While Knowledge Graph Question Answering systems have seen an increase in English and other major languages, there is a lack of support for the Greek language. Through this thesis, we aim to develop a KGQA system specifically designed for the Greek language. Our focus for our pipeline is able to answer generic but simple questions over Wikidata Knowledge Graph. Our proposed method is designed to be modular in order to support additional open-source Knowledge Graphs.

SUBJECT AREA: Natural Language Processing

KEYWORDS: Question Answering, Knowledge Graphs, Wikidata

ΠΕΡΙΛΗΨΗ

Στους ταχέως εξελισσόμενους τομείς της ανάκτησης πληροφοριών και της τεχνητής νοημοσύνης, τα συστήματα απάντησης ερωτήσεων έχουν γίνει όλο και πιο δημοφιλή και έχουν σημειώσει σημαντικές προόδους με τα χρόνια. Αυτά τα συστήματα παρέχουν έναν κατανοήτο και εύκολο τρόπο για ερωτήσεις σε δομημένα δεδομένα, επιτρέποντας στους χρήστες να κάνουν ερωτήσεις σε φυσική γλώσσα και να λαμβάνουν ακριβείς και σχετικές απαντήσεις. Χρησιμοποιώντας ιδιαιτέρως τα πλεονεκτήματα των γράφων γνώσης, οι χρήστες μπορούν να λαμβάνουν άμεσες απαντήσεις αντί για μια λίστα πιθανών αποτελεσμάτων όπως κάνουν οι δημοφιλείς μηχανές αναζήτησης.

Παρόλο που τα σύστημα απαντήσεων ερωτήσεων με χρήση γράφων γνώσης έχουν σημειώσει αύξηση για την Αγγλική και άλλες διαδεδομένες γλώσσες, υπάρχει έλλειψη υποστήριξης για την Ελληνική γλώσσα. Μέσω αυτής της διατριβής, στοχεύουμε στην ανάπτυξη ενός συστήματος KGQA ειδικά σχεδιασμένου για την Ελληνική γλώσσα. Στοχέουμε στην δημιουργία ένος συστήματος ώστε να μπορεί να απαντά σε γενικές αλλά απλές ερωτήσεις πάνω στον γράφο Wikidata. Η σχεδίαση της μεθόδου μας έχει γίνει με γνώμονα ώστε να είναι εύκολο να υποστηρίξει επιπλέον ανοιχτού κώδικα γράφους γνώσης.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Επεξεργασία Φυσικής Γλώσσας

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Σύστημα Ερωταπαντήσεων, Γράφοι Γνώσης, Wikidata

To my beloved family.

ACKNOWLEDGEMENTS

I am deeply grateful to my advisor, Prof. Manolis Koubarakis, for the opportunity to collaborate with him and for his ongoing guidance and support throughout the development of this thesis.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Objectives and Scope

In the rapidly evolving domain of information retrieval and artificial intelligence, the development of Question Answering (QA) systems is a significant milestone to minimize the gap for experienced or not, users, to quickly and accurate retrieve data. These systems, especially the ones that are build upon Knowledge Graphs (KG) have transformed the way we interact with data. By the intuitive way of writing a query in natural language we can retrieve accurate and context-aware responses to the most complex queries. These systems can build upon any KG to a wide variety of domains, from healthcare to domain business specific.

Despite the increase of Knowledge Graph Question Answering (KGQA) systems in English and other widely spoken languages, there is a notably lack of support for the Greek language. This gap not only limits access to information for Greek speakers but also represents a missed opportunity to leverage the Natural Language Processing (NLP) techniques for the Greek language. Recognizing this, the primary goal of this research is to bridge this gap by developing a KGQA system tailored to the Greek language.

1.2 Challenges

Designing and implementing a Question Answering system over Linked Data that supports Greek presented several challenges. The biggest challenge we faced is the lack of datasets, like QALD-9, suitable for training an AI model for KGQA systems. Another significant challenge is the lack of pre-trained models for entity linking that support Greek and this limitation has a huge impact of the overall performance in any QA system since it is the most important component. Furthermore, the absence of previous studies and attempts to develop systems similar to ours significantly increased the complexity of the project, as we had to develop a pipeline from scratch.

These challenges underscore the difficulties of working with less-resourced languages, like Greek, in the field of natural language processing. It shows that there's a big need for more focused research and development in this area.

1.3 Thesis Structure

This thesis consists of 5 more chapters, which are as follows:

Chapter 2: provides an overview of related work in the field of question answering over knowledge graphs.

Chapter 3: covers the background knowledge necessary to understand concept that will be discussed in the rest of the thesis. It elaborates on, knowledge graphs, question answering over knowledge graphs and query language to help retrieve data over KG.

Chapter 4: provides an in-depth analysis of the architecture of our system and it's components.

Chapter 5: describes the methodology to evaluate our system and our benchmark dataset.

Chapter 6: summarizes the thesis. Challenges and limitations we faced as well as future work is discussed.

2. BASIC CONCEPTS

In this section, we will outline and explain all the basic concepts required for this thesis, including: Knowledge Graphs, Wikidata and SPARQL.

2.1 Knowledge Graphs

The term Knowledge Graph has been publicly used when Google introduced it in 2012. Since then it has a variety of interpretations. Knowledge graphs are large networks of entities, their semantic types, properties, and relationships between entities. There are many publicly available or open source Knowledge Graphs with the most popular being the Wikidata [17], DBpedia [1], YAGO [16] and BabelNet [11]. These published KGs are cover a wide variety of domains, they support multiple language and they either supported by volunteers or they automatically extract useful information from world wide web.

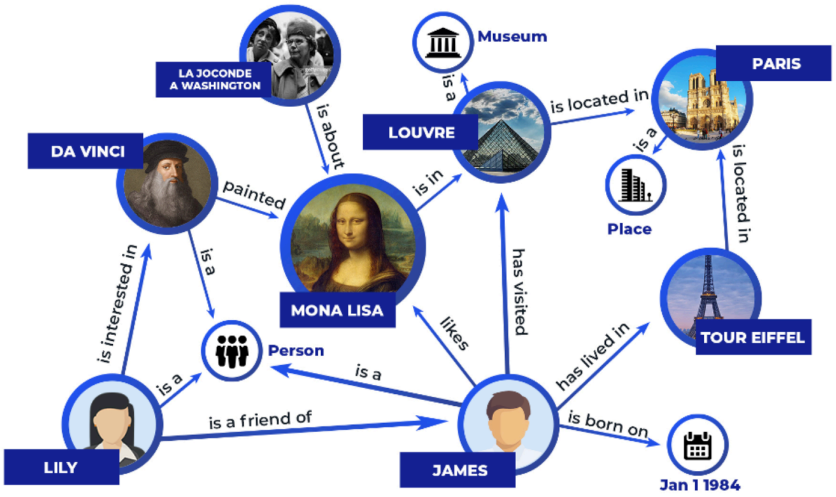

Figure 2.1: Visual Representation of a Knowledge Graph

2.2 Wikidata

Our focus will be on Wikidata, hence it's worth expanding on its structure, functionalities, and applications in greater detail. Wikidata is an open source multilingual knowledge base collecting structured data based on Wikipedia. Wikidata belongs to the *Wikimedia Foundation*[1] and it was launched in October 2012. It contains more than 500 million users and more than 2 billion edits have been made since the project's launch. It contains more than 109 million unique items which represent any kind of topic or object. This offers a high quality of data covers a wide variety of domains.

The best features of Wikidata:

- **Open Source**, anyone can access, edit or even add any kind of topic in the KG, this helps the system to stay up to date while the graph is always growing with newly added information.

[1] https://www.wikimedia.org/

- **Multilingual data**, the philosophy of the project is to be used and be useful for everyone, currently supports more than 300 human languages.
- **Accessibility**, Wikidata offers an endpoint[2] through a powerful Web interface to query and retrieve any information from the graph on the spot. Also, this endpoint is accessible via an API to help developers fetch results in multiple data formats like JSON, TSV or CSV.

The Wikidata KG consists mainly of items, each one having a label, a description and properties. Each item or entity has a unique identifier that starts with Q following by n number of integers. For example Douglas Adams has the unique identifier Q42[3]. Each item has it is own properties, each one starts with P, and are used to describe items between properties.

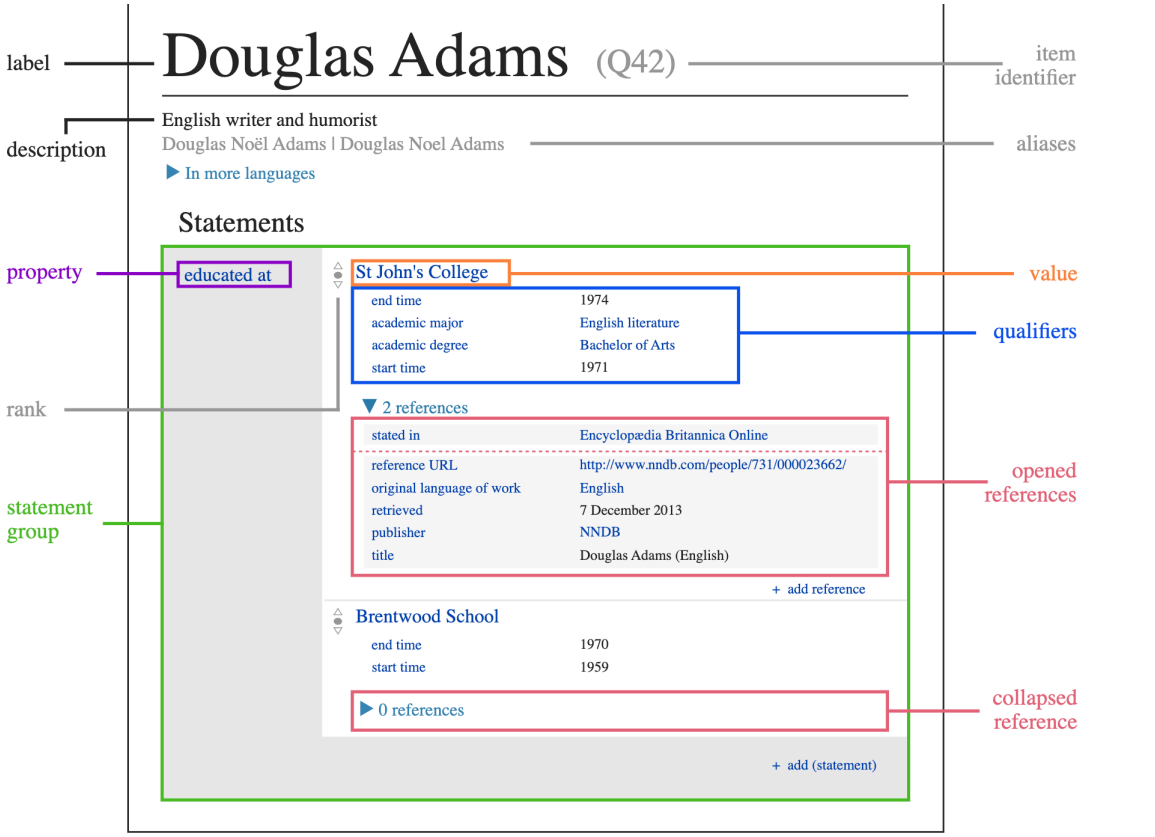

Figure 2.2: Diagram of a Wikidata item

2.3 SPARQL

To effectively interact with and retrieve information from Knowledge Graphs like Wikidata, we utilize SPARQL or "SPARQL Protocol and RDF Query Language" the query language specifically designed for querying Resource Description Framework (RDF) data. SPARQL has SQL-like structure which makes it accessible to those familiar with relational databases. It has similar query and aggregation forms to SQL for relational databases and allows users to modify and retrieve complex information from any KG. Each open-source

[2] https://query.wikidata.org/
[3] https://www.wikidata.org/wiki/Q42

KG provides its own endpoints, which allow users to retrieve information through a UI. SPARQL supports four types of queries, each with a specific purpose:

1. **ASK:** Determines whether there is at least one match for the query pattern in the RDF graph data.
2. **SELECT:** Retrieves all or some of the matches in a tabular form
3. **CONSTRUCT:** Builds an RDF graph by substituting the variables in the matches into a set of triple templates.
4. **DESCRIBE:** Generates an RDF graph that provides a relevant description of the matches found.

However, for the purposes of a Question Answering system, the focus is primarily on the first two query types: *ASK* and *SELECT*.

3. RELATED WORK AND PRIOR RESEARCH

3.1 Introduction

In recent years we notice a spike in the most popular search engines the use of information boxes as part of search results to address a quick information retrieval rather than a list of websites. Most web users expect a search engine to deliver a direct, structured answer rather than a list of relevant web pages or documents on which they will be redirected and they have to search for the answer within them. This is possible due to semantic web [2] which made the Web data machine-readable rather than human readable only. The Semantic Web aligns with the formal definition of a Knowledge Graph. Querying knowledge bases using natural language has become a major research focus in both the database (DB), Information Retrieval (IR) and Natural Language Processing (NLP) communities, as it offers an intuitive method for exploring knowledge bases even for non technical users. Most of the existing methods follow a framework that involves generating a structured query, like SPARQL, from the input question and then executing it over a Knowledge Graph to match the sub graph and retrieve the desired answer or fact.

3.2 Related Work and Literature

As already discussed in 1.2, there isn't currently a general purpose KGQA system that supports Greek. However, in this chapter, we will review existing work on KGQA systems and explore the frameworks and methodologies used in these systems. This review will help us implement an existing framework rather than re-invent the wheel, ensuring we build on proven approaches and best practices. The latest bibliography focuses on solving and improving a KGQA for single factoid questions.

Many earlier works [5, 10] that solve the single factoid question answering problem decompose the task into multiple sub-tasks: entity detection, entity linking, and relation prediction. In the latest research by D. Lukovnikov [8] an approach of transfer learning for question answering over knowledge graphs is explored using for the first time the pre-trained transformer BERT [7]. The well known SimpleQuestions [4] dataset was used to fine tune the model. Again, D. Lukovnikov follows the common practice of splitting the problem into two main sub-tasks: (1) entity span detection and (2) relation prediction. The most common architecture to identify the entity and the relation within an input query is to train a model for each task. However, in this approach, a single network was trained for both tasks simultaneously. The sequence classifier is trained with annotations that are automatically generated from the training data and entity labels in Freebase [3]. The next step is Entity Candidate Generation, which retrieves from Freebase all similar entities based on a string matching algorithm and selects the one with the highest number of outgoing relations. Having found the entity and the relation, the final step is to generate the triple on which the query will be posed over the KG to retrieve the answer.

In Chapter 4 we provide the detailed architecture of how we addressed the problem of Knowledge Graph Question Answering for the Greek language, following the same sub-tasks as outlined in previous researches. Despite the absence of a dataset for training purposes, we successfully found alternative for these sub-tasks and developed a working solution.

4. IMPLEMENTATION

Any KGQA system, regardless of the implementation approach, requires certain fixed sub-tasks related to its structure. We briefly summarize these steps as mentioned in the bibliography [6, 15, 14]:

1. *Entity Linking* - the task to determine which KG entity a specific phrase in the NL query refers to.
2. *Relation Detection* - identifying the connections or associations between entities in a text.
3. *Property Identifier* - the task of mapping the entity to corresponding property of a KG.
4. *Determining Logical Form Structure* - this is the final of task of which we incorporate all the data assembled by preceding sub tasks into formatting a query in machine language, usually it's SPARQL.

Most recent KGQAs are build upon transformers [9] leveraging the strength of transfer learning with models like BERT. However, as already discussed in 1.2, we were unable to follow that certain path due to lack of resources that support Greek. Therefore, we designed an approach to solve the aforementioned sub-tasks based on classic NLP tools.

In the figure 4.1 we have the high level overview of our system design and it's components which we will explain it in depth in the following sections. The pipeline has been built using Python 3.9 and consists of modular components, allowing it to be adapted for use with various Knowledge Graphs, not limited to Wikidata.

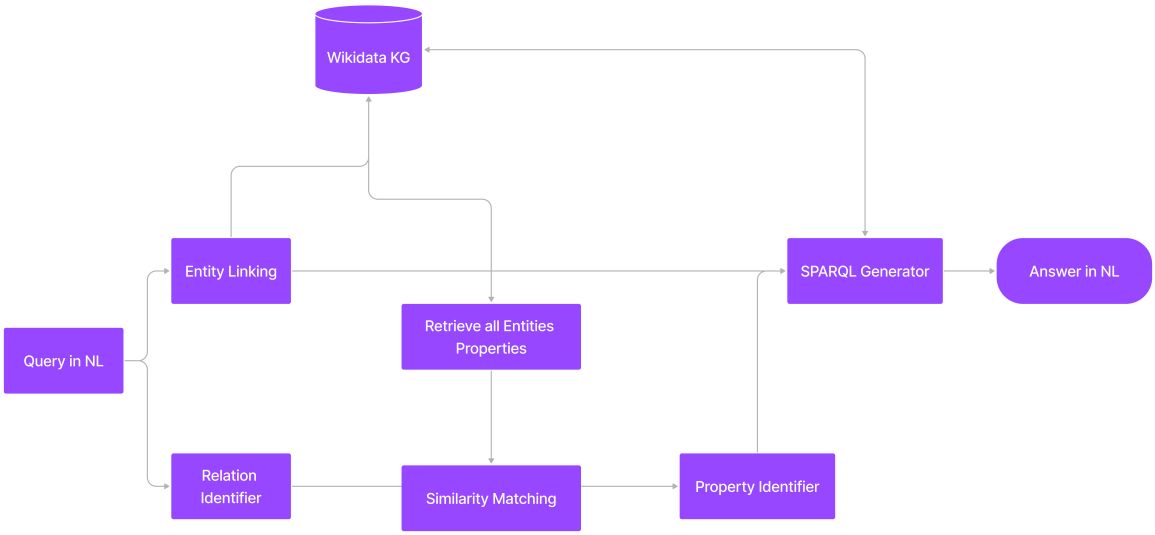

Figure 4.1: Pipeline Overview

4.1 Name Entity Linking Component

Named Entity Linking (NEL) for the Greek language presented a challenging task that we needed to address. To tackle this challenge, we utilized two tools. The first one is the well-known open-source tool called Babelfy[1], while the second one, the DebateLab-NEL[2], is a Language Model trained at the University of Crete, which they provided to us via an API.

4.1.1 Babelfy

Babelfy is based on the BabelNet[3] multilingual semantic network and performs disambiguation and entity linking in three steps:

1. It associates with each vertex of the BabelNet semantic network, i.e., either concept or named entity, a semantic signature, that is, a set of related vertices. This is a preliminary step which needs to be performed only once.
2. Given an input text, it extracts all the linkable fragments from this text and, for each of them, lists the possible meanings according to the semantic network.
3. It creates a graph-based semantic interpretation of the whole text by linking the candidate meanings of the extracted fragments using the previously-computed semantic signatures. It then extracts a dense subgraph of this representation and selects the best candidate meaning for each fragment.

In this thesis, our focus is on the Wikidata Knowledge Graph. Babelfy has the capability to associate Named Entities from a text query with entities in various Knowledge Graphs, including DBpedia and BabelNet. To align with our focus on Wikidata, we utilize a SPARQL query over an HTTP API targeting the DBpedia Endpoint. This approach enables us to retrieve the equivalent Wikidata URIs by obtaining all entities linked through the *owl:sameAs* property, thereby ensuring seamless integration with the Wikidata KG.

```
SELECT *
WHERE {
    <http://dbpedia.org/resource/Entity> owl:sameAs ?wikidataURI
    FILTER(regex(str(?wikidataURI), "wikidata" ))
}
```

This approach is not a robust method and it might perform poorly if the DBpedia entity is lacking an owl:sameAs Wikidata URI.

4.1.2 DebateLab-NEL

The tool, named DebateLab-NEL [12], is a component of the DebateLab project focusing on Named Entity Linking (NEL) for Greek news articles. It semantically annotates entities mentioned in text with entities described in knowledge bases, employing a pipeline that

[1] http://babelfy.org/
[2] https://gitlab.isl.ics.forth.gr/papanton/debatelab-nel
[3] https://babelnet.org/

integrates state-of-the-art tools such as GreekBERT, wikipedia2vec, and fastText for entity recognition, candidate generation, and disambiguation. This modular approach facilitates the validation of arguments and evaluation of their credibility by linking textual mentions to factual resources. It leveraging multiple wiki-based KBs such as Wikidata, DBpedia, Wikipedia and YAGO. By establishing direct links to Wikidata URIs for entities, we avoid the potential information loss — a challenge previously encountered when using Babelfy. In contrast to Babelfy's indirect method of retrieving Wikidata URIs through DBpedia, our approach ensures a more reliable and seamless integration with the Wikidata knowledge base.

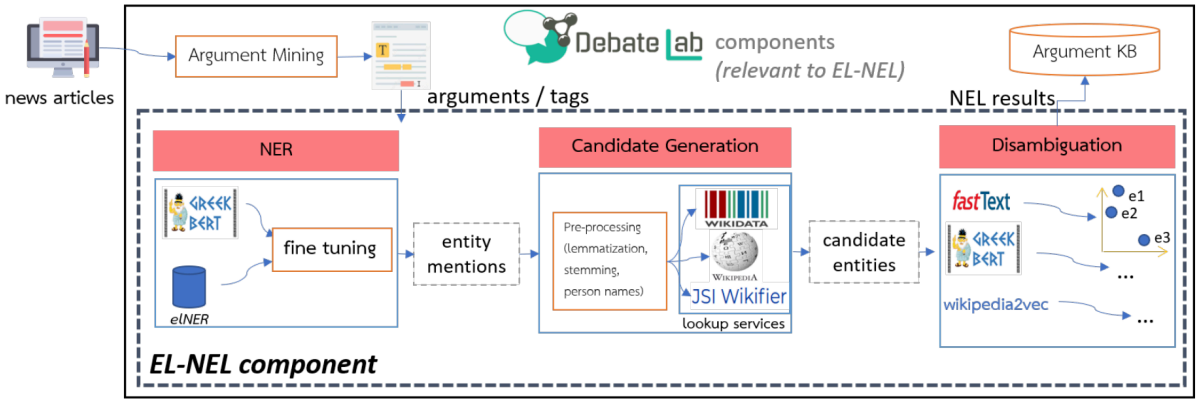

Figure 4.2: The architecture of DebateLab-NEL

Hence, given an input in NL the component identifies the entity and it links it to the unique wikidata identifier as follows:

Πότε χτίστηκε το Ελευθέριος Βενιζέλος? → Ελευθέριος Βενιζέλος → Q211734[4]

4.1.3 A Hybrid Approach

Testing both methods of Entity Linking across various scenarios with different entities, we observed that DebateLab-NEL outperforms Babelfy for Greek input. However, there have been cases where DebateLab-NEL could not retrieve an entity, but Babelfy succeeded. To leverage the strengths of both tools, we implemented a hybrid approach; if DebateLab-NEL fails to link an entity, Babelfy is deployed as a fallback mechanism. This strategy not only increased the coverage but also enhanced the overall performance of our system.

4.2 Relation Detection Component

Relation Extraction is the task of identifying and categorizing the connections or associations between entities in text, typically involving predefined relationship types. In the triple <Athens, :birthplaceOf, :Socrates>, :birthplaceOf can be a relation between the two denoting that Socrates was born in Athens. Instead of linking two entities, a relation can also link an entity to a literal, e.g. <Mount Everest, :elevation, 8849m>.

[4] https://www.wikidata.org/wiki/Q211734

Due to the lack of tools that support Greek language and datasets that can be used in order to train a relation prediction as a sequence classification task, we had to work with classical NLP techniques.

4.2.1 Dependency Parse Tree

Dependency parsing is a technique used in computational linguistics and NLP to analyze the grammatical structure of a sentence. It aims to identify the dependencies between words in a sentence. These dependencies are form a tree that satisfies the following properties:

- There is a single root node.
- Each vertex has exactly one incoming arc.
- There is a unique path from the root node to each vertex.

The analogy between the root of a dependency parse tree in NLP and the predicate in a semantic triple is that both serve as central nodes connecting entities (subject and object) through a specific relation. In the dependency tree, the root is crucial for understanding the sentence's overall structure and meaning. Similarly, in semantic triples, the predicate (relation) is key to understanding the type of link between the subject and the object.

We deploy Spacy in order to generate the Dependency Parse Tree. The root represents the main word of the sentence, it encapsulates the information we need as it is always related to the Named Entity. In the following figure we can see visually how a dependency parse tree forms for a given sentence.

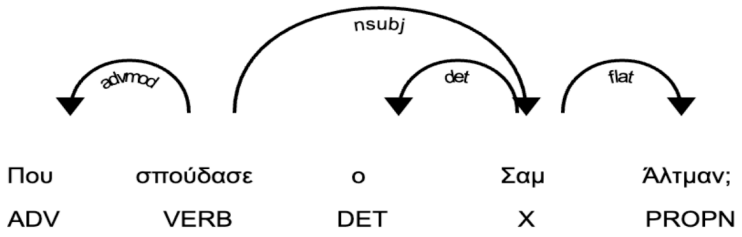

Figure 4.3: Dependency Parse Tree

Examples:

- Που σπούδασε ο Σαμ Άλτμαν? → σπούδασε
- Πότε χτίστηκε το Ελευθέριος Βενιζέλος? → χτίστηκε

Identifying the relation in a question is crucial for effectively linking it to the right property in a Knowledge Graph such as Wikidata because it directly influences the accuracy and relevance of the next component which is the Property Identifier. When a question is asked, the extracted relation it serves as a guide to navigate this graph, pinpointing the exact property that connects the extracted entity with its answer. For instance, if the question is "What is the birthplace of Einstein?", identifying the relation "birthplace" allows us to link it to the corresponding property in Wikidata that holds the birthplace information for the entity "Einstein". Without correctly identifying and mapping this relation to the KG's properties, the query might return irrelevant data or fail to retrieve any information at all.

4.3 Property Identifier Component

A property describes the data value of a statement and can be thought of as a category of data, for example ”color” for the data value ”blue”. Properties, when paired with values, form a statement in Wikidata. Properties are also used in qualifiers. Wikidata currently has 11,651 properties. Properties have their own pages on Wikidata and are connected to items, resulting in a linked data structure.[5]

Figure 4.4: Wikidata Property

In this component of our research, we concentrate on the critical task of identifying the closest related property to the extracted relation. Once the relation and the entity are extracted, the next step is to fetch all the properties from the retrieved entity from the KG. Retrieving properties from a specific entity in Wikidata involves querying the Wikidata Query Service with SPARQL. For example, once we have identified a specific entity in Wikidata, such as Q913 (representing Socrates), we do an API call on Wikidata Query Service with a SPARQL query, as shown below, to retrieve all properties in Greek associated with this entity. This query would look for all statements where Q913 is the subject, and then it would return the property (predicate) and its value (object).

```
SELECT DISTINCT ?a ?propertyLabel WHERE {
  wd:Q12508 ?a ?b.
  SERVICE wikibase:label { bd:serviceParam wikibase:language "el". }
  ?property wikibase:directClaim ?a.
}
```

To further optimize our workflow and reduce processing time, we incorporate an initial cleaning step aimed at refining the set of properties under consideration. This step involves the removal of unwanted properties by filtering out properties that are irrelevant like identifiers.

[5] https://www.wikidata.org/wiki/Help:Properties

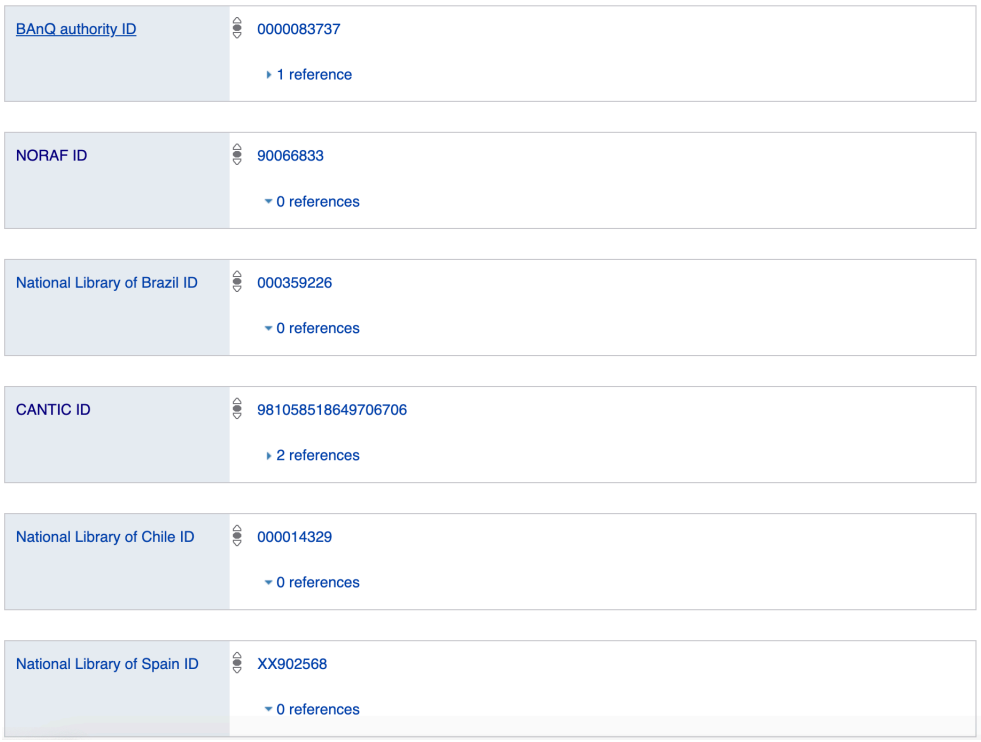

Figure 4.5: Wikidata Identifiers

4.3.1 Properties Extraction

4.3.2 Similarity Matching

4.3.2.1 String Similarity

The first approach of identifying the closest property from a list of properties with the extracted relation is to utilize a string similarity matching algorithm. We used the package called *fuzzywuzzy*[6]. This package has many useful methods to approach a variety of task. The *fuzz.ratio* calculates the Levenshtein distance where it only returns 100% if the two strings are exactly same in consideration with case sensitivity - useful if we're looking for an exact match.

However in our case it was most suitable to use *fuzz.token_sort_ratio* where it is more effective when the words are expected to be similar, but their arrangement may differ. Although this approach is performing well we noticed scenarios where the extracted relation didn't have any string similarity to the corresponding property within the list of properties.

Consider the following example, 'Που σπούδασε ο Σαμ Άλτμαν;' (ENG: Where did Sam Altman study?). Here the extracted relation is 'σπούδασε' (ENG: study) but the related property of the entity Sam Altman is 'φοίτησε σε' (ENG: educated at).

Conventional string matching algorithms are insufficient for solving this issue. To address it, we turned to more sophisticated NLP methods and employed embeddings.

[6] https://pypi.org/project/fuzzywuzzy/

4.3.2.2 Embeddings

Embeddings are numerical representations of words or phrases in a way that captures their semantic meaning and relationships. They enable machines to understand and work with language in a more context-aware manner. Similarity is determined by comparing word vectors or embeddings, multi-dimensional meaning representations of a word.

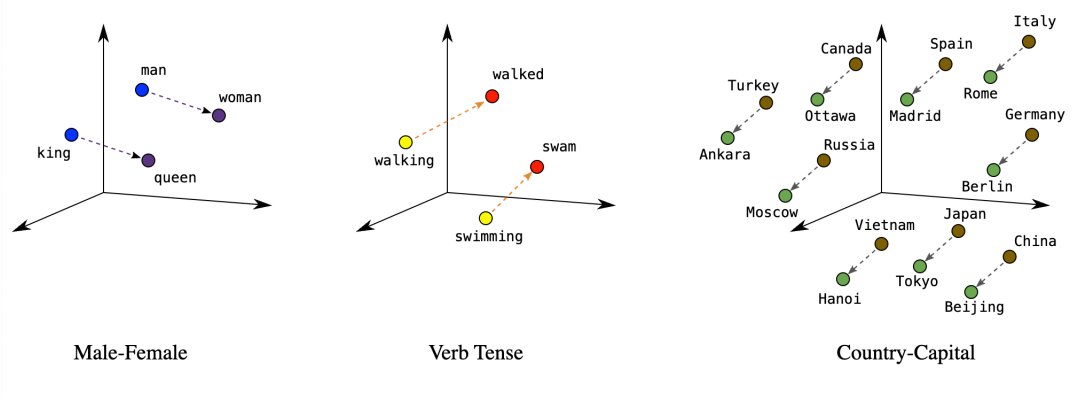

Figure 4.6: Embeddings

We used a pre-trained language model, available as an open-source provided by Spacy, to get the word-embeddings or word-vectors of 300 dimensions for every property in the properties list as well as the extracted relation. As a last step we identify the most similar properties the ones with the highest cosine similarity score with the relation.

Given the aforementioned example, ’Που σπούδασε ο Σαμ Άλτμαν;’ (ENG: Where did Sam Altman study?), Spacy returns a high similarity between ’σπούδασε’ (ENG: study) and ’φοίτησε σε’ (ENG: educated at), whereas string similarity matching failed to do so.

4.3.2.3 Hybrid Approach

Each method has its advantages and disadvantages when applied independently; thus, to enhance our workflow, we employ both techniques together. By utilizing both methods, we calculate their similarity scores and select the top 5 words. Should there be a word that matches with a 100% similarity score in the string comparison, it is identified as the extracted property. Furthermore, if a word appears in both lists, we consider this word to be the most closely related to the extracted relation. Combining string similarity for direct matches and embeddings for semantic understanding leverages the strengths of both methods: the precision of direct text comparison and the depth of semantic analysis. This hybrid approach thus provides a comprehensive strategy for finding the most similar properties, capable of handling a wide range of scenarios from simple name matching to complex semantic similarity.

4.4 SPARQL Query Generator

This component represents the last step of our process, where we convert the input question into a SPARQL query, incorporating all the data assembled by preceding components. For example, consider the query ’Που σπούδασε ο Σαμ Άλτμαν;’ (ENG: Where did Sam

Altman study?). Here, the Entity Linking component identifies 'Sam Altman', and the Property Identifier identifies the graph relation 'educated at'. Consequently, we can construct the triple pattern '<?x, :educated at, :Sam Altman>'. This process translates the question into a structured query, leveraging the assembled information.

Hence, 'Που σπούδασε ο Σαμ Άλτμαν;' is translated into:

```
SELECT ?x ?xLabel
WHERE {
  wd:Q7407093 wdt:P69 ?x.
  SERVICE wikibase:label
  { bd:serviceParam wikibase:language "[AUTO_LANGUAGE],el". }
}
```

4.4.1 Wikidata Endpoint

As the final step, we build a method that is responsible to query the Wikidata endpoint and retrieve the SPARQL's results in a JSON format within our app.

Wikidata URI	Wikidata Label
http://www.wikidata.org/entity/Q41506	Stanford University
http://www.wikidata.org/entity/Q6224030	John Burroughs School

Table 4.1: Wikidata results

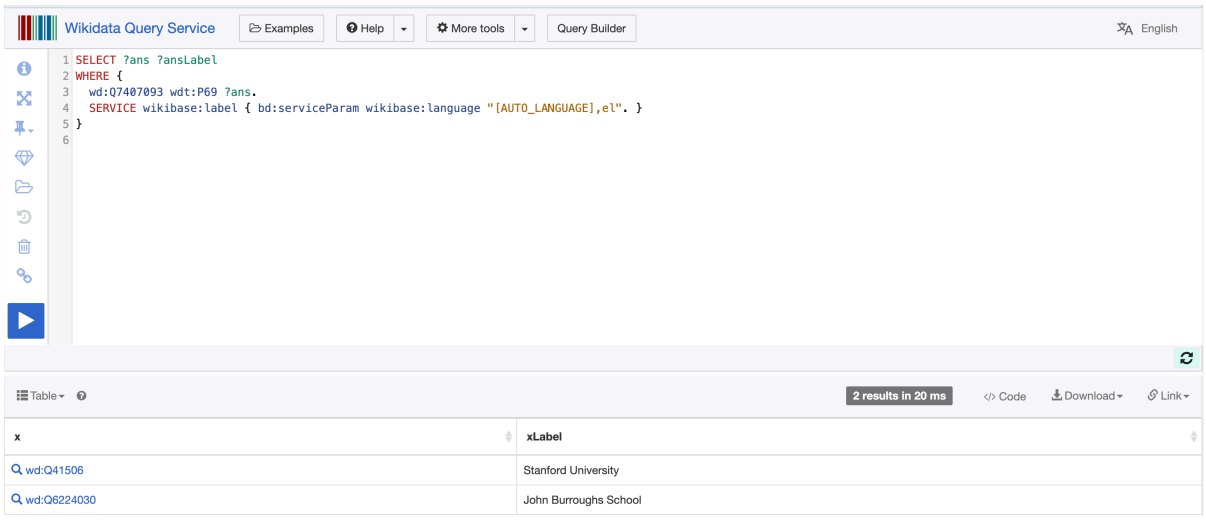

Figure 4.7: Wikidata Endpoint

5. EVALUATION

5.1 Constructing a Benchmark Dataset

Most of KGQA systems are using for their evaluation the well-known SimpleQuestions dataset [4]. SimpleQuestions is a large-scale factoid question answering dataset. It consists of 108,442 natural language questions, each paired with a corresponding fact from Freebase knowledge base. Each fact is a triple (subject, relation, object) and the answer to the question is always the object. However, this dataset is only supports English questions. Other datasets that support multi-language like QALD-9-plus [13] by introducing high-quality questions' translations to 8 languages provided by native speakers, and transferring the SPARQL queries of QALD-9 from DBpedia to Wikidata, but it doesn't support Greek either.

The lack of datasets for Greek lead us to create a benchmark dataset in a similar structure as the SimpleQuestions dataset. It consists of 100 natural language questions. Each row represents a question, its answer, and associated Wikidata entity and relation. While it has 85 unique Entities and 61 unique Relations.

Τι ύψος έχει το Μπιγκ Μπεν;	Q41225	P2048
Πόσους μαθητές έχει το Πανεπιστήμιο του Στανφορντ;	Q41506	P2196
Τι προυπολογισμό έχει το Πανεπιστήμιο του Στανφορντ;	Q41506	P2769
Ποιος είναι ο αριθμός ακολούθων στα κοινωνικά δίκτυα του Πανεπιστήμιο του Στανφορντ;	Q41506	P8687
Που φοίτησε ο Σαμ Άλτμαν;	Q7407093	P69
Τι επιφάνεια έχει η Φλόριντα;	Q812	P2046
Ποιο το προσδόκιμο ζωής στην Ελβετία;	Q39	P2250
Ποιά είναι η επίσημη γλώσσα στην Ελβετία;	Q39	P37
Ποιος ο αριθμός εδρών στο Κοινοβούλιο του Ηνωμένου Βασιλείου;	Q11010	P1342
Τι έσοδα έχει το Νετφλιξ;	Q907311	P2139
Ποια η θρησκεία του Νέλσον Μαντέλα;	Q8023	P140
Σε ποιο πολιτικό κόμμα είχε ενταχθεί η Φρίντα Κάρλο;	Q5588	P102
Τι επαγγέλεται ο Ισαάκ Νιούτον;	Q935	P106

Figure 5.1: Dataset Overview

5.2 System Evaluation

To measure the performance of our system on our benchmark dataset, we assessed three main components: Named Entity Disambiguation, Property Identifier, and the overall pipeline accuracy of retrieving the correct answer on a given natural language query. The Named Entity Disambiguation component achieved an accuracy of 75%, showing an effective handling on that task under a variety of complex queries. On the contrary, Property Identifier did not performed as well, with an accuracy of 57%, highlighting areas for potential improvement. This was expected since Property Identifier consists of two sub components the Relation Extraction and the Similarity Matching. Therefore the under performance lies on the fact that either the Relation Extraction did not identify the correct relation on the input query or the relation was identified but it did not mapped correctly on a Wikidata property. Lastly, the overall pipeline showed an accuracy of 53%, indicating moderate effectiveness as a result of the low performance of the Property Identifier. This evaluation points the strengths and weaknesses of our pipeline, providing clear indications for improvements on future iterations.

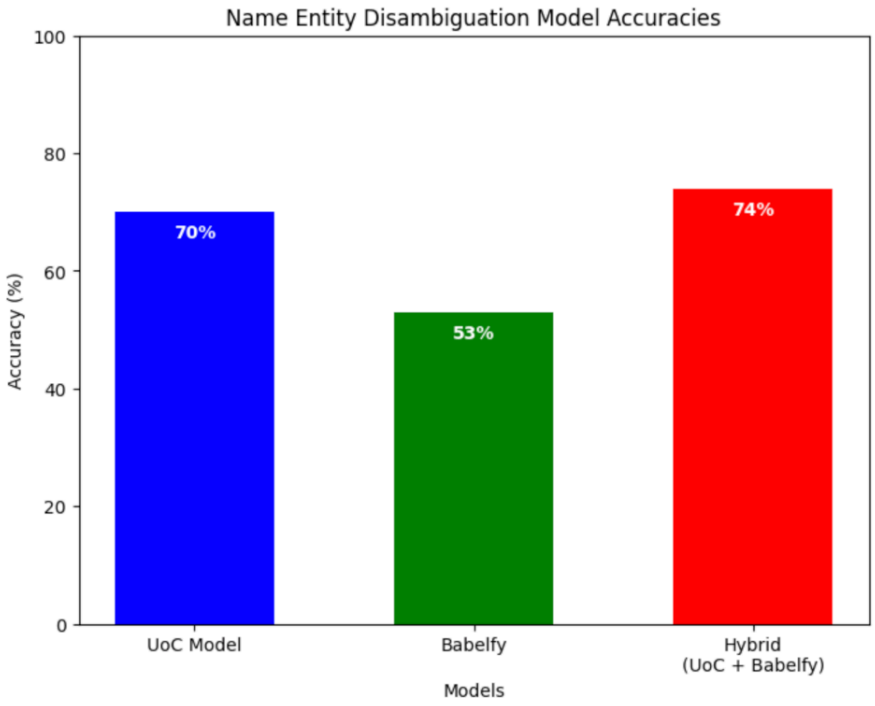

Figure 5.2: NEL Results

Table 5.1: System Performance

	Pipeline	**Named Entity Disambiguation**	**Property Identifier**
Accuracy	53%	75%	57%

6. CONCLUSIONS

6.1 Summary

In this Thesis we studied the fundamentals of a KGQA pipeline and their applications. We focused our research on a KGQA system that supports Greek and while there wasn't one for general purposes we created one end-to-end over the KG of Wikidata. We focused on identifying suitable solutions for two specific challenges: locating NLP processing tools compatible with the Greek language and incorporating them into our system's pipeline.

The current approach is based on re-usable components that make it easier to maintain and improve if new tools that support Greek will roll-out. Furthermore, the current implementation can support various KGs with Greek attributes with minor tweaks.

6.2 Future Work

Our future plans include to concentrate on improving the accuracy of our pipeline by re-defining the existing components and especially the Property Identifier component. Fine-tuning an LLM like Greek-BERT on the downstream task of similarity matching might outperform our current hybrid approach of the pre-trained Spacy model and conventional string matching algorithm.

Furthermore, we want to add the support of more complex questions containing more than one entity or more than one relation within the input NL query.

ABBREVIATIONS - ACRONYMS

KG	Knowledge Graph
KGQA	Knowledge Graph Question Answering
RDF	Resource Description Framework
NLP	Natural Language Processing
SPARQL	SPARQL Protocol and RDF Query Language
API	Application Programming Interface
NED	Named Entity Disambiguation
URI	Uniform Resource Identifier

BIBLIOGRAPHY

[1] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *international semantic web conference*, pages 722–735. Springer, 2007.

[2] Tim Berners-Lee, James Hendler, and Ora Lassila. Web semantic. *Scientific American*, 284(5):34–43, 2001.

[3] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250, 2008.

[4] Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*, 2015.

[5] Happy Buzaaba and Toshiyuki Amagasa. A modular approach for efficient simple question answering over knowledge base. In *Database and Expert Systems Applications: 30th International Conference, DEXA 2019, Linz, Austria, August 26–29, 2019, Proceedings, Part II 30*, pages 237–246. Springer, 2019.

[6] Nilesh Chakraborty, Denis Lukovnikov, Gaurav Maheshwari, Priyansh Trivedi, Jens Lehmann, and Asja Fischer. Introduction to neural network-based question answering over knowledge graphs. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 11(3):e1389, 2021.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[8] Denis Lukovnikov. *Deep learning methods for semantic parsing and question answering over knowledge graphs*. PhD thesis, Universitäts-und Landesbibliothek Bonn, 2022.

[9] Denis Lukovnikov, Asja Fischer, and Jens Lehmann. Pretrained transformers for simple question answering over knowledge graphs. In *The Semantic Web–ISWC 2019: 18th International Semantic Web Conference, Auckland, New Zealand, October 26–30, 2019, Proceedings, Part I 18*, pages 470–486. Springer, 2019.

[10] Salman Mohammed, Peng Shi, and Jimmy Lin. Strong baselines for simple question answering over knowledge graphs with and without neural networks. *arXiv preprint arXiv:1712.01969*, 2017.

[11] Roberto Navigli and Simone Paolo Ponzetto. Babelnet: Building a very large multilingual semantic network. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 216–225, 2010.

[12] Katerina Papantoniou, Vasilis Efthymiou, and Giorgos Flouris. El-nel: Entity linking for greek news articles. In *ISWC (Posters/Demos/Industry)*, 2021.

[13] Aleksandr Perevalov, Dennis Diefenbach, Ricardo Usbeck, and Andreas Both. Qald-9-plus: A multilingual dataset for question answering over dbpedia and wikidata translated by native speakers. In *2022 IEEE 16th International Conference on Semantic Computing (ICSC)*, pages 229–234. IEEE, 2022.

[14] Dharmen Punjani, Kuldeep Singh, Andreas Both, Manolis Koubarakis, Iosif Angelidis, Konstantina Bereta, Themis Beris, Dimitris Bilidas, Theofilos Ioannidis, Nikolaos Karalis, et al. Template-based question answering over linked geospatial data. In *Proceedings of the 12th workshop on geographic information retrieval*, pages 1–10, 2018.

[15] Saeedeh Shekarpour, Kemele M Endris, Ashwini Jaya Kumar, Denis Lukovnikov, Kuldeep Singh, Harsh Thakkar, and Christoph Lange. Question answering on linked data: Challenges and future directions. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 693–698, 2016.

[16] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706, 2007.

[17] Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.