NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

PROGRAM OF UNDERGRADUATE STUDIES

BACHELOR'S THESIS

Improvements to GeoQA, a Question Answering system for Geospatial Questions

Markos K. Iliakis

SUPERVISORS: **Manolis Koubarakis,** Professor
Dharmen Punjani, PhD Candidate

ATHENS

APRIL 2019

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΓΡΑΜΜΑ ΠΡΟΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Βελτιώσεις στο GeoQA, ένα Σύστημα Απάντησης Γεωχωρικών Ερωτήσεων

Μάρκος Κ. Ηλιάκης

ΕΠΙΒΛΕΠΟΝΤΕΣ: **Μανώλης Κουμπαράκης,** Καθηγητής
Dharmen Punjani, Υποψήφιος Διδάκτορας

ATHENS

APRIL 2019

BACHELOR'S THESIS

Improvements to GeoQA, a Question Answering System for Geospatial Questions

Markos K. Iliakis
S.N.:1115201500185

SUPERVISORS: **Manolis Koubarakis,** Professor
Dharmen Punjani, PhD Candidate

EXAMINING COMMITTEE: Manolis Koubarakis, Professor

Examination Date: July,2019

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Βελτιώσεις στο GeoQA, ένα Σύστημα Απάντησης Γεωχωρικών Ερωτήσεων

Μάρκος Κ. Ηλιάκης
Α.Μ.:1115201500185

ΕΠΙΒΛΕΠΟΝΤΕΣ: **Μανώλης Κουμπαράκης,** Καθηγητής
Dharmen Punjani, Υποψήφιος Διδάκτορας

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ: **Μανώλης Κουμπαράκης,** Καθηγητής

Ημερομηνία εξέτασης: Ιούλιος,2019

ABSTRACT

We study the question answering GeoQA which was proposed recently. GeoQA is the first template-based question answering system for linked geospatial data. We improve this system by exploiting the data schema information of the kb's it's using, adding more templates for more complex queries and by improving the natural language processing module in order to recognize the patterns. The current work is also an attempt to concentrate, study and compare some other question-answering systems like QUINT, Qanary methodology and Frankenstein framework for question answering systems.

SUBJECT AREA: Question Answering, Semantic Web, Artificial Intelligence

KEYWORDS: Template-Based, Linked Open Data, Geospatial

ΠΕΡΙΛΗΨΗ

Η παρούσα εργασία αποτελεί μια προσπάθεια για συγκέντρωση, μελέτη και σύγκριση συστημάτων απάντησης ερωτήσεων όπως τα QUINT, TEMPO και NEQA και του σκελετού συστημάτων απάντησης ερωτήσεων Frankenstein. Η μελέτη επικεντρώνεται στην απάντηση ερωτήσεων σε γεωχωρικά δεδομένα και πιο στο σύστημα GeoQA. Το σύστημα αυτό έχει προταθεί πρόσφατα και ειναι το πρώτο σύστημα απάντησης ερωτήσεων πάνω σε συνδεδεμένα γεωχωρικά δεδομένα βασιζόμενο σε πρότυπα. Βελτιώνουμε το παραπάνω σύστημα χρησιμοποιώντας τα δεδομένα για το σχήμα των βάσεων γνώσης του, προσθέτοντας πρότυπα για πιο σύνθετες ερωτήσεις και αναπτύσσοντας το υποσύστημα για την επεξεργασία φυσικής γλώσσας.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Απάντηση Ερωτήσεων, Σημασιολογικός Ιστός, Τεχνητή Νοημοσύνη

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Βασισμένο σε προτυπα, Συνδεδεμένα ανοιχτα δεδομενα, Γεωχωρικα

ACKNOWLEDGMENTS

I would like to thank my parents for being so supportive all along these years and for giving me the opportunity to stay focused on my studies. Also I would like to thank professor and supervisor Manolis Koubarakis for his invaluable guidance throughout the thesis and for offering me the chance to work with his diverse and cross-functional team on this very interesting topic of question answering.

I would also like to thank my supervisor, PhD student Dharmen Punjani, for his patience, help and support throughout the entire thesis.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

PREFACE

The present thesis is part of the requirements for the acquisition of a Bachelor's degree in the Department of Informatics and Telecommunications of the National and Kapodistrian University of Athens. Working on this subject was the most interesting experience as I managed to gain a wealth of knowledge about things I had never heard about, while also expanding my knowledge on various scientific topics.

1. INTRODUCTION

Question answering[1] (QA) is a computer science discipline within the fields of information retrieval[2] and natural language processing[3] (NLP), which is concerned with building systems that automatically answer questions posed by humans in a natural language.

Nowadays Question Answering Systems play a significant role to retrieve exact answers for user's specific questions. A typical Question Answering System takes the user's question in some natural language as an input. This question is then optionally modified using some query modification technique (also called query expansion) and output of this modification process is a set of queries similar in meaning to the original question. These modified questions are fed into a knowledge repository while answers of the user query or modified queries are retrieved and re-ranked, based on their relevance to the user's query. Finally the most relevant of them is presented as an answer to the user's question.

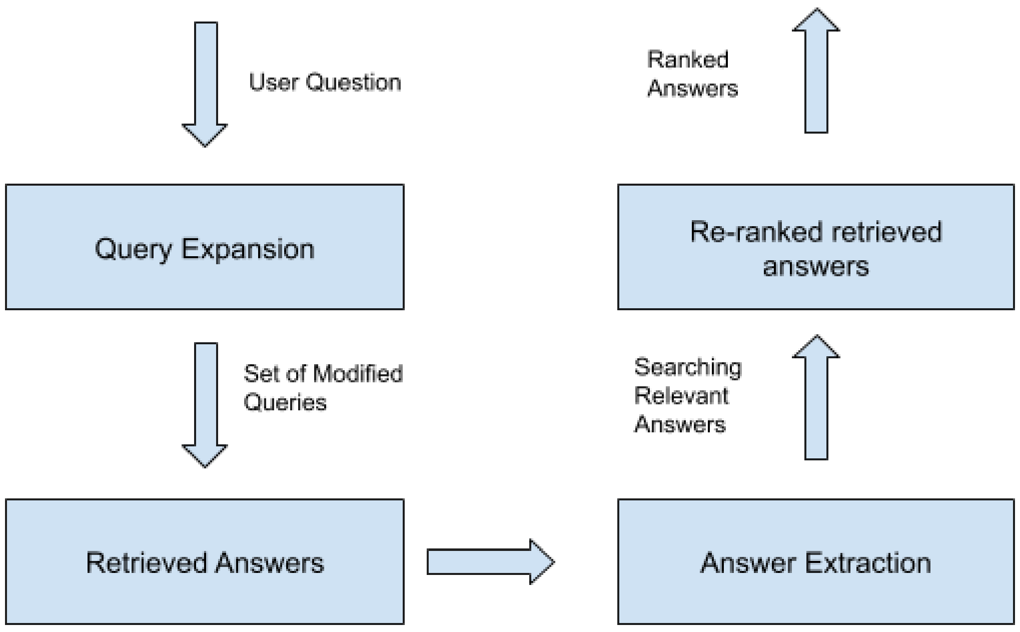


Fig. 1. A typical Question Answering System architecture

The most popular usage of Question Answering Systems today is the worldwide and everyday use of personal (or virtual) assistants. Software systems like these use voice queries and a natural-language user interface to answer questions, make recommendations, manipulate applications (especially in mobile devices), and perform actions by delegating requests to a set of internet services. The software adapts to users' individual language usages, searches, and preferences, with continuing use and return individualized results. Some major virtual assistants are Siri[4], developed by Apple, Google Assistant[5], developed by Google, Alexa[6], developed by Amazon and Cortana[7] which is developed by Microsoft.

[1] https://en.wikipedia.org/wiki/Question_answering
[2] https://en.wikipedia.org/wiki/Information_retrieval
[3] https://en.wikipedia.org/wiki/Natural_language_processing
[4] https://en.wikipedia.org/wiki/Siri
[5] https://en.wikipedia.org/wiki/Google_Assistant
[6] https://en.wikipedia.org/wiki/Amazon_Alexa
[7] https://en.wikipedia.org/wiki/Cortana

In this thesis, we are studying a Question Answering System that focuses on geospatial queries utilizing DBpedia, OSM, and GADM. We analyzed and improved its components, comparing the final results with the previous.

This thesis is organized as follows: section 2 provides details about how most Question Answering Systems and their components work as well as information about key ideas and related work. In section 3, we explain the modules, the metrics, the results and the problems of the existing GeoQA System. In section 4, we explain the improvements that were made in this system and we show the final results. In section 5, we have stated our conclusion and future directions to further improve GeoQA.

2. RELATED WORK

In this chapter we will present research works related to this dissertation. We cover the research areas of knowledge bases, query languages for both general and geospatial data along with some question answering systems. Further on we provide a historical background on question answering, its origin and evolution.

2.1 Domains of Question Answering

A QA implementation, usually a computer program, may construct its answers by querying a structured database of knowledge or information, usually a knowledge base. QA research attempts to deal with a wide range of question types including fact, list, definition, How, Why, hypothetical, semantically constrained, and cross-lingual questions. However, we can categorize QA in two major categories, Closed-domain, and Open-domain QA.

Closed-domain question answering deals with questions under a specific domain (for example, medicine or automotive maintenance), and can exploit domain-specific knowledge frequently formalized in ontologies. Alternatively, closed-domain might refer to a situation where only a limited type of questions are accepted, such as questions asking for descriptive rather than procedural information. QA systems in the context of machine reading applications have also been constructed in the medical domain, for instance, related to Alzheimer's disease.

On the other hand, Open-domain question answering deals with questions about nearly anything, and can only rely on general ontologies and world knowledge. These systems usually have much more data available from which to extract the answer.

Another categorization we can make is that of Text-based QA and Knowledge-based QA. The first one relies on the enormous amounts of information available as text on the Web or in specialized collections such as PubMed[8]. Given a user question, information retrieval techniques extract passages directly from these documents, guided by the text of the question.

The method processes the question to determine the likely answer type (often a named entity like a person, location, or time) and formulates queries to send to a search engine. The search engine returns ranked documents which are broken up into suitable passages and reranked. Finally, candidate answer strings are extracted from the passages and ranked.

In the second category (Knowledge-based QA) we instead build a semantic representation of the query. The meaning of a query can be a full predicate calculus statement. So the question “What states border Texas?” might have the representation:

$$state(x) \wedge borders(x, Texas)$$

Alternatively, the meaning of a question could be a single relation between a known and an unknown entity. Thus the representation of the question “When was Ada Lovelace born?” could be

birth-year (Ada Lovelace, ?x).

[8] https://www.ncbi.nlm.nih.gov/pubmed/

Whatever meaning representation we choose, we'll be using it to query databases of facts. These might be complex databases, perhaps of scientific facts or geospatial information, that need powerful logical or SQL queries. Or these might be databases of simple relations, triple stores like Freebase[9] or DBpedia[10].

2.2 Knowledge Bases

A knowledge base (KB) is a technology used to store complex structured and unstructured information used by a computer system. The term "knowledge-base" was coined to distinguish this form of knowledge store from the more common and widely used term *database* as they have so many different properties.

The main difference is that instead of the classically used tabular format with strings and numbers in relational databases, KBs maintain more structured data with pointers to other objects that in turn have additional pointers. The ideal representation for a KB is an object model (Ontology) with classes, subclasses, and instances. The most commonly used template for representing these classes, subclasses, and instances is that of the RDF Triples.

A Semantic Triple, or simply Triple, is the atomic data entity in the Resource Description Framework (RDF) data model. As its name indicates, a triple is a set of three entities that codifies a statement about semantic data in the form of subject–predicate–object expressions. For example

"Bob is 35"   "Bob knows John"

This format enables knowledge to be represented in a machine-readable way. Particularly, every part of an RDF triple is individually addressable via unique URIs (Unified Resource Identifiers) — for example, the second statement above might be represented in RDF as

Instance	Relation	Instance
http://example.name#BobSmith12	*http://xmlns.com/foaf/0.1/knows*	*http://example.name#JohnDoe34*

Given this precise representation, semantic data can be unambiguously queried and reasoned about.

The components of a triple could be an Instance (specific object eg. JohnDoe34), a Relation (eg. knows), a Concept (eg. the word "city" in the question "Which city is near Athens), or another triple

Mike → said → (triples → can be → objects)

[9] https://en.wikipedia.org/wiki/Freebase
[10] https://en.wikipedia.org/wiki/DBpedia

In the end, gathering all the formal naming and definition of the categories, properties, and relations between the concepts, data, and entities that we need in a specific domain, we create an Ontology (or Knowledge Graph).

Some of the most popular open source knowledge bases are the DBpedia, Yago[11], LinkedGeoData from OpenStreetMap project (OSM)[12], Database of Global Administrative Areas (GADM)[13], Data.gov[14], Wikidata[15], Freebase, and UMBEL. While DBpedia, Yago, and Wikidata are projects aiming to extract structured content from the information created in the Wikipedia project, GADM is a high-resolution knowledge base of country administrative areas with a goal of “all countries, at all levels, at any time period” and LinkedGeoData is an effort to add a spatial dimension to the Web of Data / Semantic Web. LinkedGeoData uses the information collected by the OpenStreetMap project and makes it available as an RDF knowledge base according to the Linked Data principles.

2.3 Query Languages

To query RDF data there have been proposed many query languages (RDQL, ICS-FORTH RQL, etc) as we need a specialized language other than SQL. The current W3C recommendation is SPARQL (Protocol and RDF Query Language)[16] which is a semantic query language for databases, able to retrieve and manipulate data stored in RDF format. For example:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?email
WHERE{
        ?person a             foaf:Person.
        ?person foaf:name  ?name.
        ?person foaf:mbox  ?email.
}
```

Unlike relational databases, the object column is heterogeneous: the per-cell data type is usually implied (or specified in the ontology) by the predicate value. Also unlike SQL, RDF can have multiple entries per predicate; for instance, one could have multiple "child" entries for a single "person", and can return collections of such objects, like "children".

SPARQL contains capabilities for querying required and optional graph patterns along with their conjunctions and disjunction. Also supports aggregation, subqueries, negation, creating values by expressions, extensible value testing, and constraining queries by source RDF graph. The results of SPARQL queries can be result sets or RDF graphs.

In extension to SPARQL, there are some more specialized languages such as GeoSPARQL and stSPARQL. GeoSPARQL is a standard for representation and querying of geospatial linked data for the Semantic Web from the Open Geospatial Consortium (OGC)[17]. The definition of a small ontology based on well-understood OGC standards is

[11] https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago
[12] http://linkedgeodata.org/About
[13] https://gadm.org/
[14] https://www.data.gov/
[15] https://www.wikidata.org/wiki/Wikidata:Main_Page
[16] https://www.w3.org/TR/sparql11-query/
[17] http://www.opengeospatial.org/

intended to provide a standardized exchange basis for geospatial RDF data which can support both qualitative and quantitative spatial reasoning and querying.

Moreover, GeoSPARQL offers components like geometry extension and topology vocabulary extension. The first one provides the vocabulary for asserting and querying data about the geometric attributes of a feature (ex. geo:SpatialObject, geo:Geometry, geo:Feature, geo:asWKT, etc). Also provides functions through the “OpenGIS Simple Feature Access” standard, like geof:distance, geof:intersection, geof:boundary, geof:intersection, etc. The second one is used for representing topological information about features. Topological information is inherently qualitative and it is expressed in terms of topological relations like containment (geo:sfContains), adjacency(geo:sfTouches), overlap(geo:sfCrosses, geo:sfIntersects, geo:sfOverlaps), etc. Topological information can be derived from geometric information or it might be captured by asserting explicitly the topological relations between features.

An example of GeoSPARQL usage is

```
Select ?m
Where {
        ?m rdf:type gadm:Region.
        ?m geo:sfContains gadm:London.
}
```

On the other hand, stSPARQL offers the same functions as in the geometry extension and geometry topology extension of GeoSPARQL, with the addition of spatial aggregate functions like strdf:geometry, strdf:intersection, strdf:extent. Apart from these, it also offers some features for the temporal dimension.

An example of stSPARQL usage is

```
Select ?burntArea (strdf:intersection(?baGeom, strdf:union(?fGeom)) as ?burntForest)
Where{
        ?burntArea   rdf:type             noa:BurntArea.
        ?burntArea   strdf:hasGeometry    ?baGeom.
        ?forest             rdf:type              clc:Region.
        ?forest             clc:hasLandCover      clc:ConeferousForest.
        ?forest             strdf:hasGeometry     ?fGeom.
        Filter (strdf:intersects(?baGeom, ?fGeom))
}
Group By ?burntArea ?baGeom
```

2.4 Question Answering Systems Problems and Performance

The need to query information content available in various formats including structured and unstructured data (text in natural language, semi-structured Web documents, structured RDF data in the Semantic Web, etc.) has become increasingly important. Thus, Question Answering Systems (QAS) are essential to satisfy this need.

While most of the time Question Answering Systems have good performance, their weakness is the quality of the data that they are getting. A very disappointing example, that we will also analyze later, is the huge amount of noise that exists in DBpedia data, one of the largest knowledge bases worldwide. There are too many false inserted triples that even the best Question Answering Systems techniques can't do much about it.

Another critical point in the procedure of question answering is that of final query evaluation and extraction. As most of the modern systems generate many possible queries from the starting question, they need a strong technique in order to choose one from these queries. Newer systems, that utilize machine learning and especially supervised learning in order to train the query evaluation components to behave as they want, seems very promising and capable of solving the problem.

Last but not least, Natural Language Processing systems also have to make major improvements, with the main problems being the reasoning about large or multiple documents, dealing with the low-resource scenarios and the actual datasets and evaluation procedures that are appropriate to measure the progress towards concrete goals.

As we can see, there is much room for improvement in every basic component of question answering systems and that is what we have tried to do with the template-based question answering over linked geospatial data system in the last part of this thesis.

2.5 History of Question Answering Systems

Question answering was one of the earliest NLP tasks, and early versions of the text-based and knowledge-based paradigms were developed by the very early 1960s. The text-based algorithms generally relied on simple parsing of the question and of the sentences in the document and then looking for matches. This approach was used very early on (Phillips, 1960) but perhaps the most complete early system, and one that strikingly prefigures modern relation-based systems, was the Protosynthex system of Simmons et al. (1964).

The alternative knowledge-based paradigm was implemented in the BASEBALL[1] system (Green et al., 1961). This system answered questions about baseball games like “Where did the Red Sox play on July 7” by querying a structured database of game information. The database was stored as a kind of attribute-value matrix with values for attributes of each game.

Another important progenitor of the knowledge-based paradigm for question answering is work that used predicate calculus as the meaning representation language. The LUNAR[2] system (Woods et al. 1972, Woods 1978) was designed to be a natural language interface to a database of chemical facts about lunar geology. It could answer questions like “Do any samples have greater than 13 percent aluminum?” by parsing them into a logical form.

The rise of the web brought the information-retrieval paradigm for question answering to the forefront with the TREC QA[3] track beginning in 1999, leading to a wide variety of factoid and non-factoid systems competing in annual evaluations. At the same time, Hirschman et al. (1999) introduced the idea of using children's reading comprehension tests to evaluate machine text comprehension algorithm. They acquired a corpus of 120 passages with 5 questions each designed for 3rd-6th-grade children, built an answer extraction system, and measured how well the answers given by their system corresponded to the answer key from the test's publisher. Their algorithm focused on word overlap as a feature; later algorithms added named entity features and more complex similarity between the question and the answer span (Riloff and Thelen 2000, Ng et al. 2000).

Other major question-answering systems that gained a lot of attention are Watson, EAGLi and Wolfram Alpha. Started in 2005 and developed in IBM's DeepQA project, Watson was initially developed to answer questions on the quiz show Jeopardy and, in 2011, competed, winning the first place prize. EAGLi, on the other hand, is a modern question answering system with use in health and life sciences. Wolfram Alpha is also a widespread computational knowledge engine, developed by Wolfram Alpha LLC. It is used as an online service that answers factual queries directly by computing the answer from externally sourced "curated data", rather than providing a list of documents or web pages that might contain the answer.

Other question-answering tasks include Quiz Bowl, which has timing considerations since the question can be interrupted (Boyd-Graber et al., 2018). Question answering is also an important function of modern personal assistant dialog systems.

2.6 Major Question Answering Systems and Performance

2.6.1 QAS for Latin Languages

Due to the popularity, importance, and features of the English language, tens of QA Systems are available since 1960 with this specialization. The current trend is moving towards Linked Data. Next, we highlight some of the most prominent work in this area.

Table 1 : Major QA Systems

QAS	Features	Techniques
PRECISE[4]	Natural Language Interfaces to Databases	Identifying classes of questions
The formal semantic approach[5]	Natural Language Interfaces to Databases	Intermediate representation language
MASQUE[6]	Natural Language Interfaces to Databases	Portable NL front end to SQL databases
BASEBALL[1]	Natural Language Interfaces to Databases	Specific domain Systems
LASSO[7]	Document Based Question Answering	Deep linguistic analysis and iterative strategy
FALCON[8]	Document Based Question Answering	Hierarchies of question types based on the types of answers sought
DIMAP[9]	Document Based Question Answering	Semantic categories of answers are mapped into categories

		covered by a NE Recognizer. When the answer type is identified, it is mapped into answer taxonomy, where the top categories are connected to several word classes from WordNet
Mulder[10]	Question Answering On the Web	Extracting “semantic relation triples” after the document is parsed, converting the document into triples.
FAQ Finder[11]	Question Answering On the Web	QA System for factual questions over the Web
QALC[12]	Question Answering On the Web	Statistical or semantic similarities
QRISTAL[13]	Question Answering On the Web	Provides answers to English factoid questions based on syntactic and semantic analysis
WebQA[14]	Question Answering On the Web	Based on named entities’ recognition, and conceptual and thematic analysis
Ask.com[15]	Question Answering On the Web	Using the template-mapping technique to define the question and the type clustering technique to extract multiple answer blocks

2.6.2 Ontology-based Question Answering Systems

Ontology-based QA Systems take queries expressed in NL and a given ontology as input and return answers drawn from one or more KBs that subscribe to the ontology. Therefore, they do not require the user to learn the vocabulary or the structure of the ontology. Ontology-based QA Systems vary on two main aspects: (i) the degree of domain customization they require, which correlates with their retrieval performance, and (ii) the subset of NL they are able to understand (full grammar-based NL, controlled or guided NL, pattern-based).

Table 2 : Major Ontology based QA Systems

QAS	Techniques
AquaLog[16]	Allows the user to choose an ontology and then ask NL queries with respect to the universe of discourse covered by the ontology
PowerAqua[17]	QAS focusing on querying multiple semantic Web

	resources
QACID[18]	Relies on an ontology, a collection of user queries, and an entailment engine that associates new queries to a cluster of existing queries.
ORAKEL[19]	Translates factual wh-queries into Flogic or SPARQL and evaluates them with respect to a given KB
GINSENG[20]	Controls user's input via a fixed vocabulary and predefined sentence structures through menu-based options
PANTO[21]	Portable NLI that takes an NL question as input and executes a corresponding SPARQL query on a given ontology model
FREYA[22]	Providing improvements with respect to a deeper understanding of a question's semantic meaning
QAKIS[23]	A technique for matching NL fragments and textual patterns, auto-collected from Wikipedia
SPARQL2NL[24]	In the side of converting a SPARQL query into natural language.
SWIP[25]	The processing of the NL query is based on the use of the pivot query: from the NL user query into a pivot query, and the formalization of this pivot query.
Pythia[26]	Using ontology in the process of interpretation of user query.
SQUALL[27]	Using a controlled natural language for translation to SPARQL query.
TBSL[28] LODQA[29]	The user question is transformed into a template query in order to generate the SPARQL query from the NL query using the template model.
DeepQA[30]	Using unstructured and structured data (RDF format) to extract and score evidence.
CASIA[31]	A Markov Logic Networks algorithm is used for learning a joint model, for detecting phrases and for mapping semantic Items. For these phases, the semantic items are grouping into a graph.

2.6.3 Performance

Here we can see the performance of some of the previous systems measured as correct answered questions/ total questions

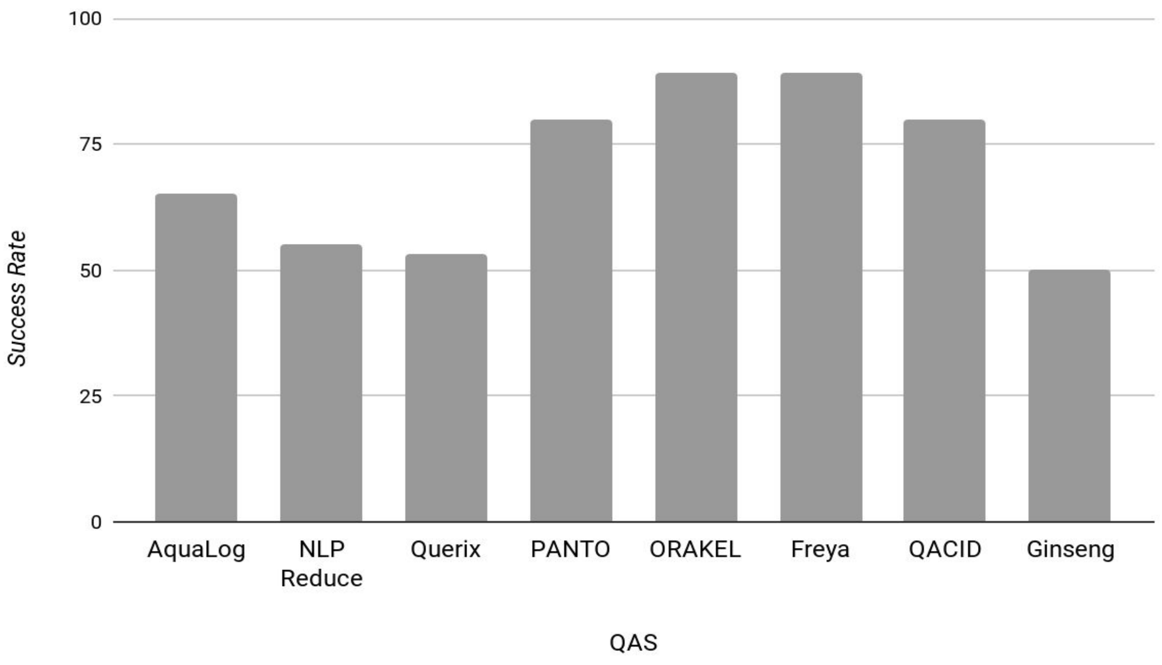


Fig. 2. Question Answering Systems Performance

2.7 Automated Template Generation for QA Over KBs

Most of the Question Answering Systems over KBs usually handle questions by parsing them, translating the utterance to a formal query language like SPARQL and executing this query over a KB. In order to make this translation possible, many of the systems utilize a set of manually defined rules or templates. The main drawback of these approaches is the limited coverage of templates, making them brittle when it comes to unconventional question formulations.

To solve this problem, a relatively new approach has emerged. It has been proposed by the Question Answering System QUINT[32] and uses a technique that automatically learns templates from question-answer pairs. To do this, it has two separate phases, the first one being the training phase and the second one the answering phase. In the training phase, a set of natural language utterances along with the corresponding gold answer set are given as input. An example would be :

"Where was Obama educated"
paired with
{ColumbianUniversity, HarvardUniversity, PunahouSchool}

Another critical feature this system introduced in the answering phase was the visualization of the derivation steps for generating an answer. This step is critical as it is helping users gain confidence when correct answers are returned, and make sense of the limitations of the system by looking at explanations for the wrong answers[18]. Furthermore, for expert users, explanations also contribute to identifying the exact point of failure in the knowledge base - question answering system pipeline, resulting in easier debugging and an easier way to find workarounds.

[18] https://gate.d5.mpi-inf.mpg.de/quint/quint.

The outline of the whole system is shown here:

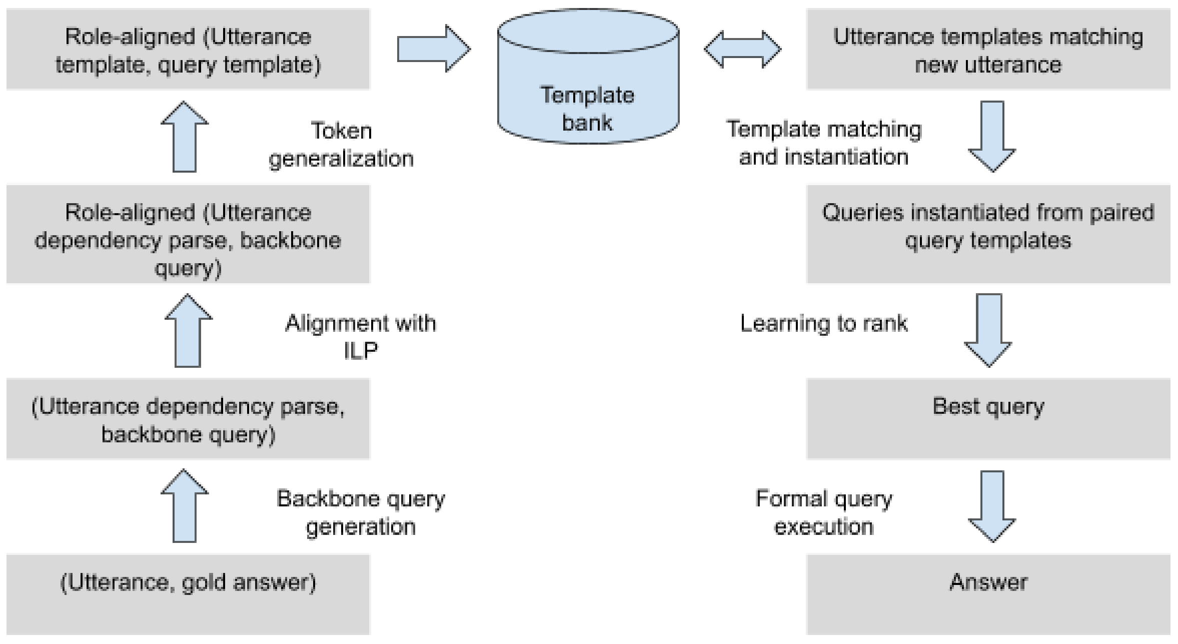

Role-aligned (Utterance template, query template)
Template bank
Utterance templates matching new utterance
Token generalization
Template matching and instantiation
Role-aligned (Utterance dependency parse, backbone query)
Queries instantiated from paired query templates
Alignment with ILP
Learning to rank
(Utterance dependency parse, backbone query)
Best query
Backbone query generation
Formal query execution
(Utterance, gold answer)
Answer

Fig. 3. Overview of QUINT

2.8 Modular Approach - Qanary - Frankenstein

One of the key features of the system that we will later examine, GeoQA, is that it was built based on the Frankenstein - Qanary Platform/Methodology.

2.8.1 Overview

So far, Question Answering community has released a considerable body of research as well as valuable running components accomplishing various QA tasks. The main problem though is that these state-of-the-art components can't be integrated or evaluated and run together. The result of this state is that researchers have to develop full QA pipelines even if their intentions were to create and optimize a specific component.

To solve this kind of problems researchers have created modular approaches such as openQA[34][19], Qanary[35][20], OKBQA[36][21], QALL-ME[37][22] and Frankenstein[38][23]. These approaches manage to overcome the above problems and add reusability and flexibility to Question Answering systems. For example, Qanary, where Frankenstein is based on, is a methodology for integrating components of QA systems which utilizes qa vocabulary for annotation, is independent of programming languages, is agnostic to domains and datasets and integrates components on different granularity levels.

Frankenstein, the latest of these platforms, is a smart solution on top of Qanary to the limitations observed in the prior approaches. For example, openQA expects Java

[19] http://openqa.aksw.org/
[20] https://github.com/WDAqua/Qanary
[21] http://www.okbqa.org/
[22] http://qallme.fbk.eu/
[23] http://frankenstein.qanary-qa.com/

implementations of the components which are not possible in most of the cases. Also, openQA and QALL-ME have configuration difficulties and its components are not directly reusable in other approaches. More importantly, these frameworks do not support a dynamic pipeline methodology. Thus, Frankenstein is concerned with a prediction mechanism to predict the performance of a component given a question and a required task and an approach for composing performance-optimized pipelines by integrating the most accurate components for the current QA tasks.

Furthermore, the distinguished features united within Frankenstein makes it scalable, user-friendly and fully automatic which are rare in the prior approaches. Moreover, the characteristics of this system allow the researchers to focus on improving individual stages of QA pipelines while reusing other components to complete their pipeline.

2.8.2 Entity Linking

One of the basic functions that most Question Answering Systems need, is that of Entity Linking[24] (EL). EL is the task of determining the identity of entities mentioned in text by linking them with a Knowledge Base’s entities. For example if we used DBpedia KB, the entities of the sentence:

“Paris is the capital of France”

should be recognised and linked as

“http://dbpedia.org/resource/Paris”
“http://dbpedia.org/ontology/Capital”
“http://dbpedia.org/resource/France”

In order to achieve this goal, Entity Linking task is often divided into two subtasks, Named Entity Recognition (NER)[25] and Named Entity Disambiguation (NED)[26]. The first one seeks to locate and classify named entity mentions in the unstructured text into predefined categories such as the person names, organizations, locations, etc. For example to produce from :

“Jim bought 300 shares of Acme Corp. in 2006”

the annotated and classified

“[Jim]$_{Person}$ bought 300 shares of [Acme Corp.]$_{Organization}$ in [2006]$_{Time}$.”

On the other hand, NED is different from named entity recognition in that NED identifies the specific entity from the Knowledge Base, given as input the annotated one. e.x:
“*http://dbpedia.org/resource/Jim*”

[24] https://en.wikipedia.org/wiki/Entity_linking
[25] https://en.wikipedia.org/wiki/Named-entity_recognition
[26] https://en.wikipedia.org/wiki/Entity_linking

2.8.3 Qanary Reusable Components

In order to better understand the components of Frankenstein, we can check these of Qanary. The following components integrated into the Qanary ecosystem solve the tasks of NER and NED.

Table 3 : Qanary Components

Component	Function
Stanford NER[39]	Natural Language Processing tool that can be used to spot entities for any ontology, but only for languages where a model is available
FOX NER[40]	Integrates four different NER tools using ensemble learning
DBpedia Spotlight spotter NER[41]	Uses lexicalizations i.e. ways to express named entities, that are available directly in DBpedia
DBpedia Spotlight disambiguator NED[41]	Disambiguates entities by using statistics extracted from Wikipedia texts
AGDISTIS NED[42][27]	Tool that uses the graph structure of an ontology to disambiguate entities
ALCHEMY[28] NER + NED	Commercial service
Lucene Linker NER + NED	Component that follows the idea of SINA QA system[43], which employs information retrieval methods

The conceptual architecture of Qanary would be:

[27] https://github.com/dice-group/AGDISTIS
[28] http://alchemyapi.com

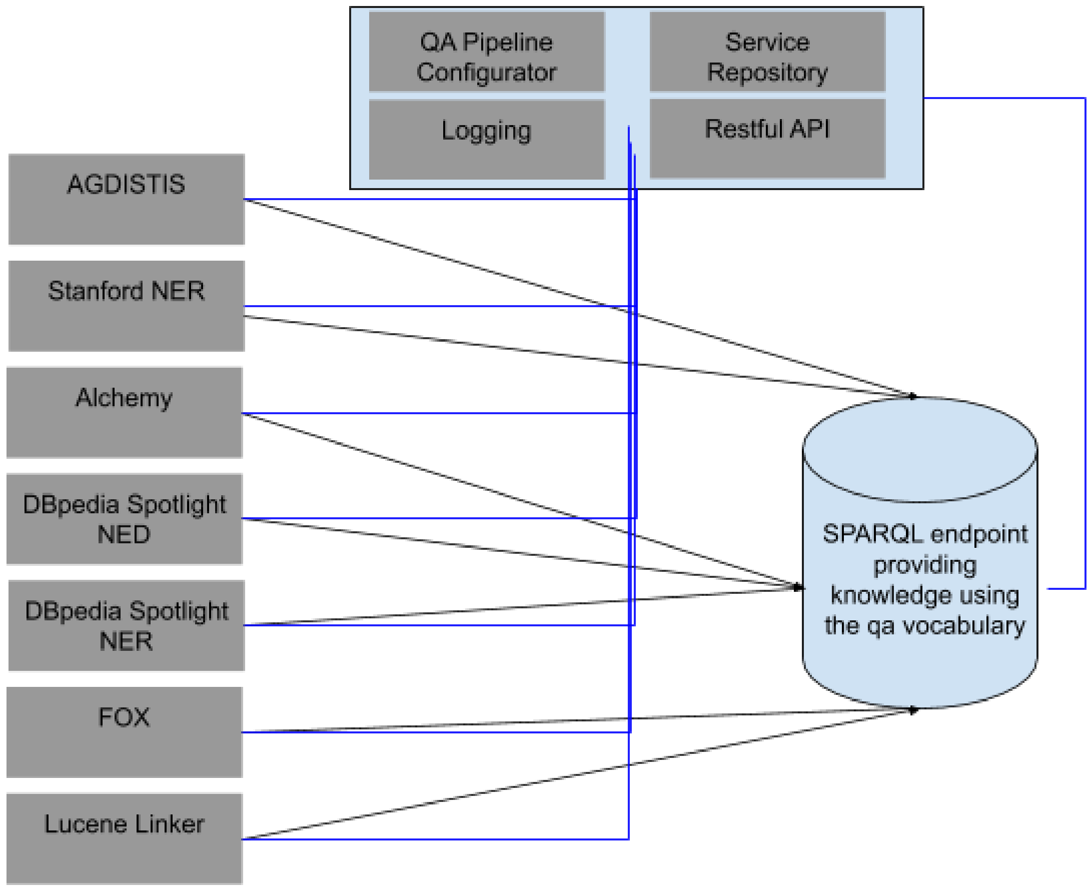


Fig. 4. Overview of Qanary

2.8.4 Frankenstein Reusable Components

As Frankenstein was made on top of Qanary, they have many similar Named Entity Recognition and Disambiguation modules. Apart from them, Frankenstein also provides some Relation Linking components in order to disambiguate natural language relations present in a question to its corresponding mention in a knowledge base e.g.

“Who is the mayor of Berlin”
↓
“mayor of” → “http://dbpedia.org/ontology/leader”

Also, Class Linking is provided in order to disambiguate the classes against the ontology. E.g.

“Which river flows through Seoul”
↓
“river” → “http://dbpedia.org/ontology/River”

Some of these components along with their functionality are:

Table 4 : Frankenstein Components

Component	Function
Entity Classifier	Uses rule base grammar to extract entities in a text[44]
Stanford NLP Tool	Same as Qanary, Stanford tool uses Gibbs sampling for information extraction to spot entities

Babelfy[45][29]	Multilingual, graph based approach that uses random walks and the densest subgraph algorithm to identify and disambiguate entities
AGDISTIS	As in Qanary, uses HITS algorithm with label expansion strategies and string similarity measures to disambiguate entities
DBpedia Spotlight	Same as in Qanary, uses a vector-space representation of entities and the cosine similarity to recognise and disambiguate the entities
Tag Me[46][30]	Matches terms in a given text with Wikipedia and uses the in-link graph and the page dataset to disambiguate recognised entities to its Wikipedia URIs
ReMatch[47][31]	Maps natural language relations to knowledge graph properties by using dependency parsing characteristics with adjustment rules, and then carries out a match against knowledge base properties
RelationMatcher[48]	Devise semantic-index based representation of PATTY [49] and a search mechanism over this index with the purpose of enhancing relation linking task
RelationMatcher	The disambiguation module (DM) of OKBQA framework provides disambiguation of entities, classes, and relations present in a natural language question.
RNLIWOD	Relation Linker from Natural Language Interfaces for the Web of Data ((NLIWOD) community group[32] which provides reusable components for enhancing the performance of QA systems.
Spot Property	This component is the combination of RNLIWOD and OKBQA disambiguation module [49] for relation linking task.
NLIWOD CLS	Class Identifier
OKBQA Class Identifier	Part of OKBQA disambiguation module
NLIWOD QB	Template-based query builder
SINA Query Builder	Query Builder of SINA[50] natural language query search engine that is based on Hidden Markov Models for choosing the correct dataset to query

[29] http://babelfy.org
[30] https://services .d4science.org/web/TagMe
[31] https://github.com/mulangonando/ReMatch
[32] https://www.w3.org/community/nli/

Concluding, in this chapter we analyzed the basic technologies a question answering system needs in order to function. Also, we discussed many different question answering approaches along with their performances, their benefits, and drawbacks. In the end, we saw some modular approaches to question answering systems in order to make an introduction to how GeoQA works, the system we improved.

3. TEMPLATE-BASED QUESTION ANSWERING OVER LINKED GEOSPATIAL DATA

As we keep generating, collecting and using more and more geospatial data (e.g. OpenStreetMap data, administrative geographies of various countries or land cover / land use data sets), there has been a rise in interest and need of question answering systems that use as input these geospatial data in order to infer and answer simple and complex geospatial questions. GeoQA[33] is one of them and moreover the first one to implement a template-based question answering system over linked geospatial data.

GeoQA has been implemented using reusable components as part of the component-oriented Qanary question answering methodology and its most recent implementation Frankenstein. It has also been evaluated using a set of 201 natural language questions developed as a gold standard for question answering over linked geospatial data.

3.1 Geospatial QA and Knowledge Bases

As we previously discussed, geospatial questions can be answered by Question Answering Systems that utilize the query language SPARQL and some extensions of it, GeoSPARQL and stSPARQL, as well as Knowledge Bases that have the required data.

3.1.1 Interlinking KBs

GeoQA uses all SPARQL, GeoSPARQL, and stSPARQL as query languages and DBpedia, OpenStreetMap, and GADM as knowledge bases using data in RDF form. For GeoQA to be able to use the three of them, an interlinking was done using two methods. The first one created owl:sameAs relations between the data with the same meaning in GADM and DBpedia, while the second one used the tool Silk[34] to link the entities of OSM and DBpedia.

3.1.2 Usage of each KB

The purpose of using the three of them was GeoQA to be able to answer different kinds of geospatial questions and enrich DBpedia's dataset with quantitative geospatial information (i.e., geometries). GADM, for example, contains information about administrative divisions of various countries and their boundaries, thus a question to this data set would be:

"Is Liverpool east of Ireland ?"

OSM, on the other hand, is a collaborative project to create a free editable map of the world and contains information about various features like rivers, lakes, cities, roads, points of interest in geometries like points, lines or polygons. Similarly, a question to this data set would be:

"Which rivers cross London ?"

Finally, DBpedia as we mentioned earlier, is one of the most popular knowledge graphs derived from Wikipedia and its ontology. Interlinking DBpedia with the other two was

[33] http://geoqa.di.uoa.gr/
[34] http://silkframework.org/

mandatory. Apart from answering questions from its own data, this interlinking also leads into answering more complex questions like

“Which of the English counties that border Greater Manchester has the highest percentage of ethnic Asians ?”

that need GADM to find the counties that border Greater Manchester, and then DBpedia to find the percentage of various ethnic groups in these counties.

3.2 GeoQA Components

As we earlier mentioned, GeoQA consists of fully integrated and reusable Qanary/Frankenstein Components, where each component is implemented as an independent micro-service implementing the same RESTful interface. Each component communicates with the others through a central mediator and a process - independent knowledge base where the knowledge associated with the current question is stored.

3.2.1 Components Usage

The Frankenstein framework components/modules that have been created in order to implement the GeoQA pipeline are : dependency parse tree generator, concept identifier, instance identifier, geospatial relation identifier, SPARQL/GeoSPARQL query generator and SPARQL/GeoSPARQL query executor.

The task of Question Answering is performed by translating the input question to a set of SPARQL or GeoSPARQL queries, ranking these queries and executing the top-ranked query.

The conceptual architecture of the GeoQA system would be:

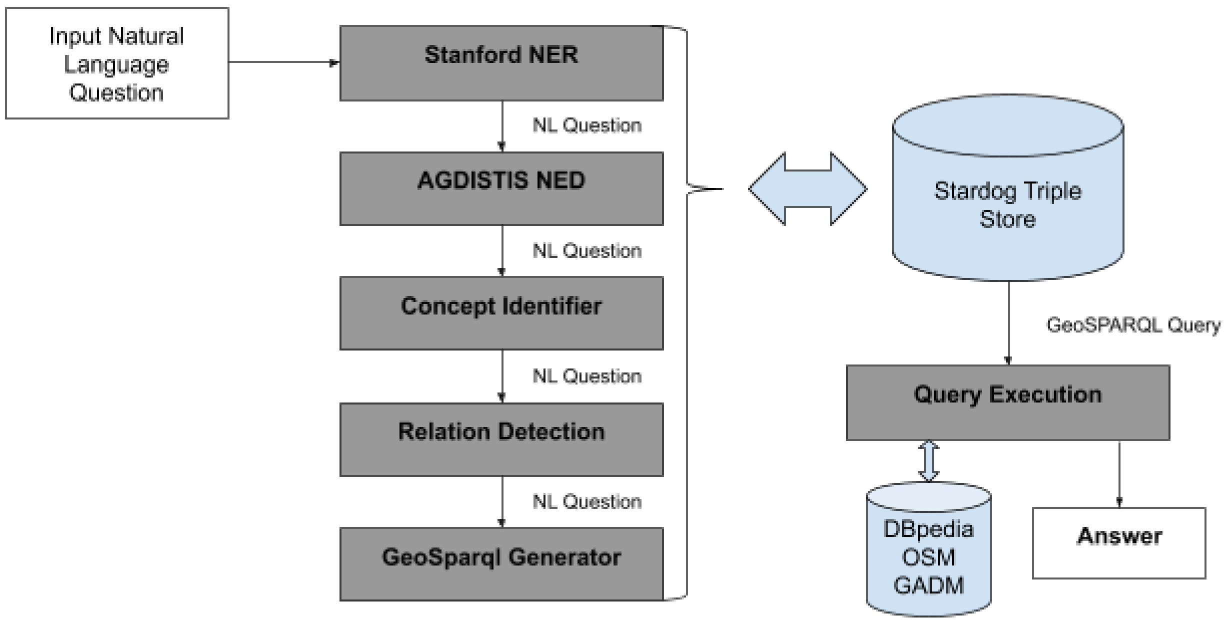


Fig. 5. Architecture of the GeoQA system

The tasks of each module are the following:

Table 5 : GeoQA Components

Component	Function
Dependency Parse Tree Generator	This component carries out part-of-speech tagging and generates a dependency parse tree for the input question using the Stanford CoreNLP software.
Concept Identifier	This module identifies the types of features specified by the user in the input question and maps them to the corresponding classes in the DBpedia, GADM and OSM ontologies (e.g. "*restaurant*"→"*http://dbpedia.org/resource/Restaurant*") using string matching, lemmatization, and synonyms.
Instance Identifier	This module identifies the features mentioned in the question and maps them to DBpedia, OSM and GADM resources (e.g. "*London*" → "*http://dbpedia.org/resource/London*") using Stanford NER and AGDISTIS.
Geospatial Relation Identifier	Identifies and maps qualitative and quantitative relations in question, such as "*borders*" and "*at most 2 km from*" to a spatial function of the GeoSPARQL or stSPARQL vocabulary, or a data property with a spatial semantics in the DBpedia ontology.
Query Generator	Creates SPARQL and GeoSPARQL queries using handcrafted query templates that are shown in table 7. Also ranks the generated queries using a simple heuristic, based on the component of GeoData201 used for obtaining the geospatial knowledge (DBpedia > GADM > OSM).
Query Executor	Executes the top-ranked SPARQL or GeoSPARQL queries using the corresponding KB endpoints.

The acceptable spatial relations along with their acceptable synonyms are :

Table 6 : Acceptable Geospatial Relations

Geospatial Relation	Synonyms in the dictionary
within	in, inside, is located in, is included in
crosses	cross, intersect
near	nearby, close to, around
borders	is/are at the border of, is/are at the outskirts of, at the boundary of
north of	above of
south of	below
east of	to the right

west of	to the left

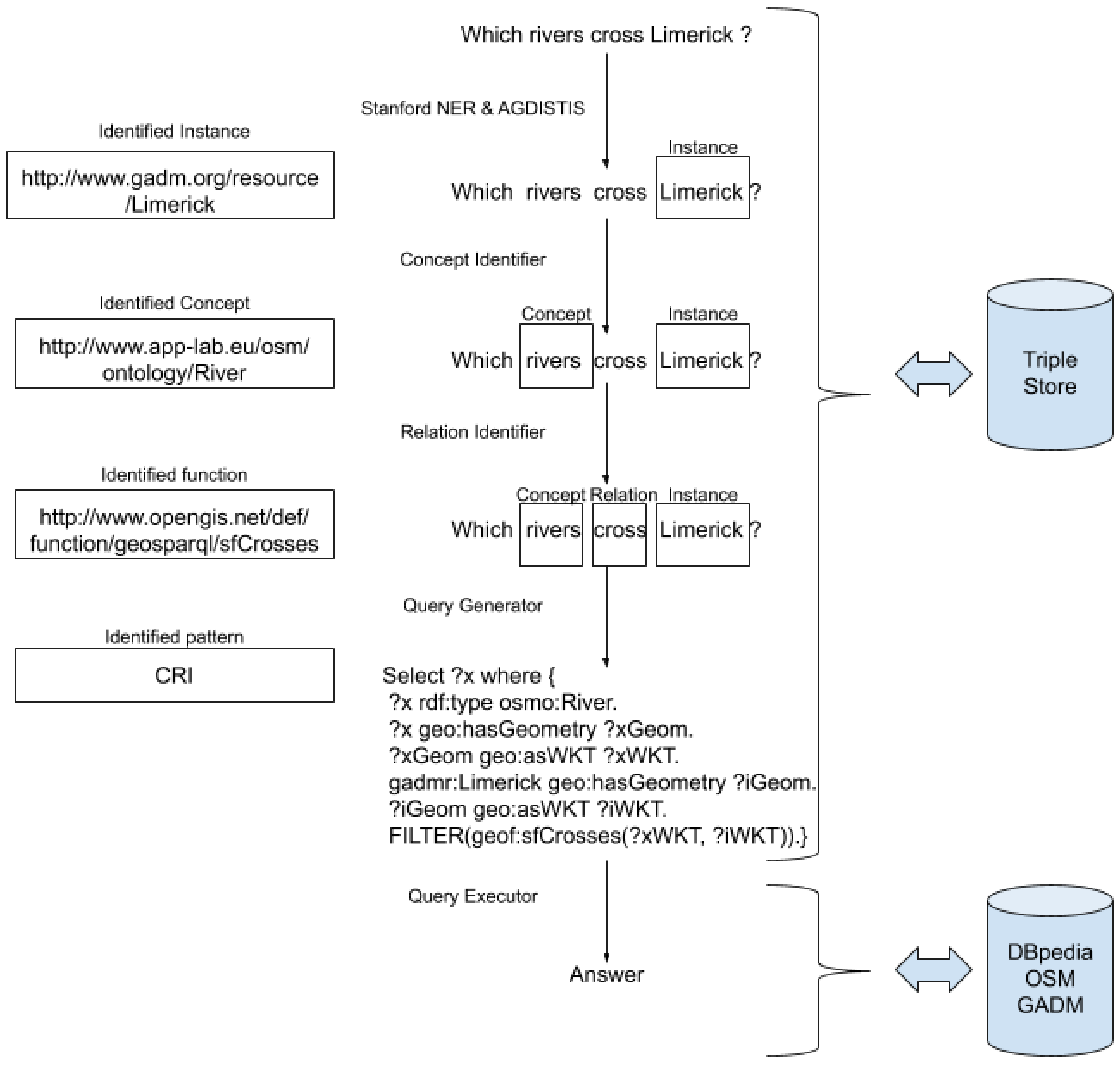
3.2.2 Templates and Data Flow

The patterns where the templates are based on are made of Concepts (C), Relations (R) and Instances (I).

Table 7 : GeoQA Patterns

Pattern	Example
CRI	“Which rivers cross Limerick ?”
CRIRI	“Which churches are close to the Shannon in Limerick ?”
CRC	“Which restaurants are near hotels ?”
CRCRI	“Which restaurants are near hotels in Limerick ?”
IRI	“Is Hampshire north of Berkshire ?”

An example that would represent the status of the system from the input of the question until the answer would be :


Which rivers cross Limerick ?
Stanford NER & AGDISTIS
Identified Instance
http://www.gadm.org/resource/Limerick
Instance
Which rivers cross Limerick ?
Concept Identifier
Identified Concept
http://www.app-lab.eu/osm/ontology/River
Concept
Instance
Which rivers cross Limerick ?
Triple Store
Relation Identifier
Identified function
http://www.opengis.net/def/function/geosparql/sfCrosses
Concept Relation Instance
Which rivers cross Limerick ?
Query Generator
Identified pattern
CRI
Query Executor
Answer
DBpedia OSM GADM

```
Select ?x where {
 ?x rdf:type osmo:River.
 ?x geo:hasGeometry ?xGeom.
 ?xGeom geo:asWKT ?xWKT.
 gadmr:Limerick geo:hasGeometry ?iGeom.
 ?iGeom geo:asWKT ?iWKT.
 FILTER(geof:sfCrosses(?xWKT, ?iWKT)).}
```

Fig. 6. GeoQA Data Flow example

3.3 Gold Standard Dataset

The fair evaluation of any question answering system is of vital importance as it helps the detection of any weakness and enables the users to compare each QA system with the others. It remains a challenge due to the different types of answers that the Question Answering systems provide, thus a new, specific to geospatial questions, gold standard dataset is developed in order to evaluate GeoQA.

The Gold Standard consists of two parts. The first one is a linked geospatial dataset built from DBpedia, OSM and GADM data, restricted to the United Kingdom and Ireland. The second one consists of 201 geospatial questions (GeoQuestions201)[35] both simple and complex and can fall under these categories:

[35] http://geoqa.di.uoa.gr/benchmarkquestions.html

Table 8 : Gold Standard Categories

Category	Examples
Asking for the location of a feature	“Where is Loch Goil located”
Asking whether a feature is in a geospatial relation with another feature	“Is Liverpool east of Ireland?”
Asking for features of a given class that are in a geospatial relation with another feature	“Which counties border county Lincolnshire?” “Which hotels in Belfast are at most 2km from George Best Belfast City Airport?”
Asking for features of a given class that are in a geospatial relation with any features of another class	“Which churches are near castles?”
Asking for features of a given class that are in a geospatial relation with an unspecified feature of another class which, in turn, is in another geospatial relation with a feature specified explicitly	“Which churches are near a castle in Scotland?”
Asking for the previous and in addition, the thematic and/or geospatial characteristics of the features that are expected as answers	“Which mountains in Scotland have a height of more than 1000 meters?” “Which villages in Scotland have a population of less than 500 people?”
Questions with quantities and aggregates	“Which hotel is the nearest to Old Trafford Stadium in Manchester” “Which is the highest mountain in Ireland?”

3.4 Results & Problems

In order for GeoQA to be preliminary tested in the aspects of performance and reliability, it was run using the 86 out of 201 questions of the Gold Standard. Also, statistic measures were taken.

3.4.1 Evaluation Metrics

In these cases, the most useful statistic measures are Precision[36], Recall and F1 score[37]. The first one, Precision, also known as PPV (Positive Predicted Value), is the fraction of relevant instances among the retrieved instances :

$$PPV = \frac{True\ Positives}{Positives} = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

In our example, this can be translated in the fraction of correct answers among the retrieved answers which intuitively shows the ratio between correctly answered questions and wrongly answered questions :

36 https://en.wikipedia.org/wiki/Precision_and_recall
37 https://en.wikipedia.org/wiki/F1_score

$$PPV = \frac{Correct\ Answered\ Questions}{Total\ Answered\ Questions}$$

Secondly, we need to complement precision with Recall, also known as TPR (True Positive Rate). It is the fraction of relevant instances retrieved over the total amount of relevant instances :

$$TPR = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

Recall shows the ratio between correct answered questions and correct unanswered questions. In our example, Recall is the fraction of correct answers that have been retrieved over the total amount of answers (which is equal to the number of questions) :

$$TPR = \frac{Correct\ Answered\ Questions}{Total\ Questions}$$

Both precision and recall are therefore based on an understanding and measure of relevance. The reason that we need both of them is that regarding precision, finding many correct answered questions and very few wrongly answered questions is useless unless we consider also the correct unanswered questions. Respectively, regarding recall, finding many correct answered questions and leaving out very few wrongly unanswered questions is useless unless we consider also the wrongly answered questions.

On the other hand, F1 score (also called f-measure) considers both the precision and the recall and evaluates the accuracy of the test itself. It's the harmonic average of these two and reaches its best value at 1 (perfect precision and recall) and worst at 0 :

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

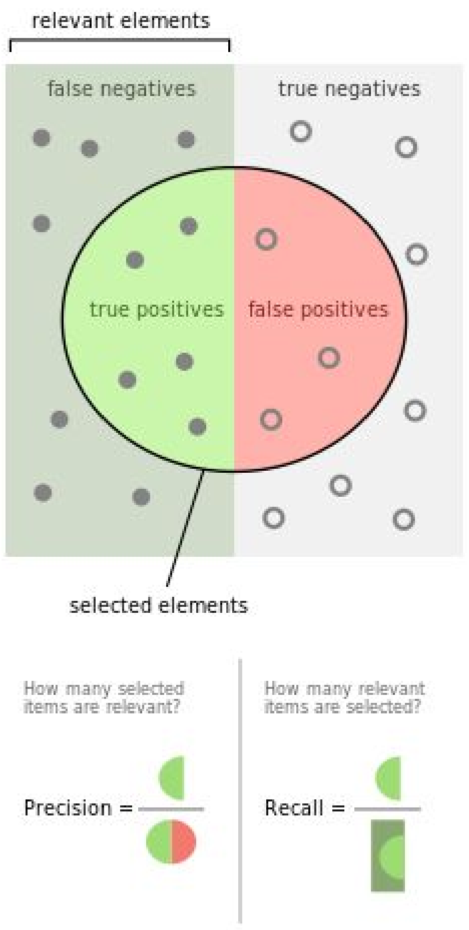


Fig. 7. Precision and Recall visually. The circle square region shows the total amount of questions, the circle is the answered questions. Precision, Recall and True/False Positive/Negatives are shown with intuitive colors.[38]

3.4.2 GeoQA Results

We can see the Precision, Recall and F-Measure of GeoQA for these 86 questions in this table:

Table 9 : GeoQA Statistics

	Precision	Recall	F-Measure
Old GeoQa	37.35%	41.43%	35.50%

The metrics are low as GeoQA fails to answer or return an empty set of answers incorrectly in 42 out of the 86 questions. Here we can see which component fails in each question from the Gold Standard:

Table 10 : GeoQA Component Failures

Module Responsible	Question Number
NER	Q56,Q269,Q333,Q17,Q46,Q92,Q98,Q283,Q312,Q320

[38] https://en.wikipedia.org/wiki/Precision_and_recall#/media/File:Precisionrecall.svg

NED	Q201,Q272,Q273,Q335,Q114,Q241,Q245
Concept Identifier	Q123,Q205,Q323,Q333,Q335,Q117
Relation Identifier	Q65,Q120,Q129,Q236,Q268,Q290,Q323
Query Generator	Q61,Q71,Q119,Q182,Q220,Q235,Q242,Q304,Q319,Q330,Q106,Q237

3.4.3 Problems to solve

The reasons for this low effectiveness vary. Undoubtedly, the component with the most failures is the Query Generator. This is due to the lack of more advanced templates for complex questions. For example questions like :

“Is there a car park at most 1km from Waterloo Bridge ?”

which is CCRI can’t be answered as there is no such pattern/template in the Query Generator module. Also, another reason is the lack of more recognized geospatial relations along with their synonyms and their matching to basic or complex relevant geospatial functions.

Aggregates like

“Which is the <u>highest</u> mountain in Ireland ?”

and conditions like

“Which rivers cross Limerick and their length is more than 300 km ?”

also, create problems as GeoQA currently does not support them and such questions fail.

Another component that fails in many questions is the Instance Identifier. The reason for these failed queries is that Instances whose name is more than one word, are often identified as two separate Instances. For example :

“Which hotels are near Big Ben ?”

In this question, Ben is recognized as a separate instance by the NER leading the whole pipeline into a failure.

Also, geospatial information taken from DBpedia needs to be augmented with geospatial information from GADM and OSM to increase the recall of GeoQA.

Finally, an inherited problem to any Question Answering system that uses Knowledge Bases such as DBpedia is that KBs of such size has too many wrong entries such as the triplet:

“http://dbpedia.org/resource/Airport | http://dbpedia.org/resource/in | http://dbpediaorg/ontology/River”

This fact leads many correctly crafted SPARQL/GeoSPARQL queries to fail.

Concluding, in this chapter we discussed the GeoQA question answering system, the modules that it uses as well as the knowledge bases that it’s connected to. Also, we saw the gold standard dataset with which we tested GeoQA. In the end, we analyzed the results and the problems that occurred in order to make an introduction to our work, improving this system.

4. IMPROVEMENTS OF GeoQA

In this chapter, we will further describe the problems of GeoQA and the fixes that were made.

4.1 Overcoming Problems

As we explained in the previous chapter, most of the failures occur in Geosparql Query Generator. Thus we have put our efforts into improving that module and especially into trying to solve three of its main problems that would have the biggest impact on results.

The first problem lies in questions like :

“Is there a river that crosses Manchester ?”

The problem here was that Geosparql Query Generator was wrongly choosing to query DBpedia as in geospatial relations like “crosses”, DBpedia has many wrong entries and thus can’t be reliable.

The second problem we tried to solve was found in many questions from the Gold Standard, like :

“Which bridges cross River Thames ?”

This type of question has unnecessary information, the word “River”, which causes the failure and raises the complexity of pattern matching in the part of template generation.

Last but not least, we managed to raise the total number of patterns/templates in order to match questions with high complexity.

4.2 Exploitation of Data Schema Information

Dealing with the faulty DBpedia data, we had to optimize the KB selection system in order for Geosparql Query Generator to not choose DBpedia if there is a problem with the data.

4.2.1 RDF Table

As checking each and every entry of DBpedia was not an option, the only way to solve the problem was to automate the process by exploiting DBpedia’s data schema information and checking the classes in order to determine if any specific information seems legit.

To do so we wrote a script to get all the available triples of Type - Geospatial Relation - Place Class by querying DBpedia and made available a table like this :

Table 11 : DBpedia Type - GR - Place triples example

Concept rdf:Type	Geospatial Relation	Instance Class
Airport	nearBy	dbo[39]:city
Building	in	dbo:location

[39] http://dbpedia.org/ontology

Canal	crosses	dbo:region
City	westOf	dbo:state

The actual table is formatted in a csv file and has all the available spatial relations with the corresponding instance's class for all the concepts of rdf:Type :

Table 12 : RDF Object Types

Subject	Predicate	Object
?x	rdf:Type	"Airport" "Bank" "Building" "Canal" "Castle" "Church" "City" "College" "Dam" "Glacier" "Hospital" "Hotel" "Island" "Library" "Lighthouse" "Monument" "Museum" "River" "Mountain" "Pharmacy" "Rapid_transit" "Restaurant" "County" "Market" "Lake" "Theatre" "Bridge" "Gym" "University" "Parking lot" "Square" "Stadium" "National park" "Pub" "Cafeteria" "Forest" "Village" "Bakery" "Town" "Bar" "Beach" "Filling station" "Tourist attraction" "train station" "Monument" "Province"

An example of a single query to get all the relations between a subject of type River and an object of type Place would be:

```
select distinct ?p
where {
        ?x rdf:type dbo:River.
        ?x ?p ?i.
        ?i rdf:type dbo:Place.
}
order by ?p
```

Fig. 8. Query Example

4.2.2 Table Checking

After filling the table with approximately 3000 entries we checked them manually and deleted any irrelevant/false content. At this point the table contains only legit rdf classes and relations that will not cause the module to fail. Instead if a specific relation from a question is not found in this table between the question's subject and object classes, DBpedia will be rejected and a search in another KBs such as OSM or GADM will be performed.

4.2.3 Checking if an Answer is Available

Before changing the architecture of GeoSPARQL Query Generator, the KB selection was done by order. If the instance and concept identifier modules could find the entities in DBpedia then the Query Generator would query only DBpedia. In other case it would try for GADM and then for OSM.
In order to engage the new table in the process we created a function to check in every question, before choosing DBpedia, if the type of the instance along with the type of the concept and the relation exists in the table as a triple. We can see this flow in the next diagram :

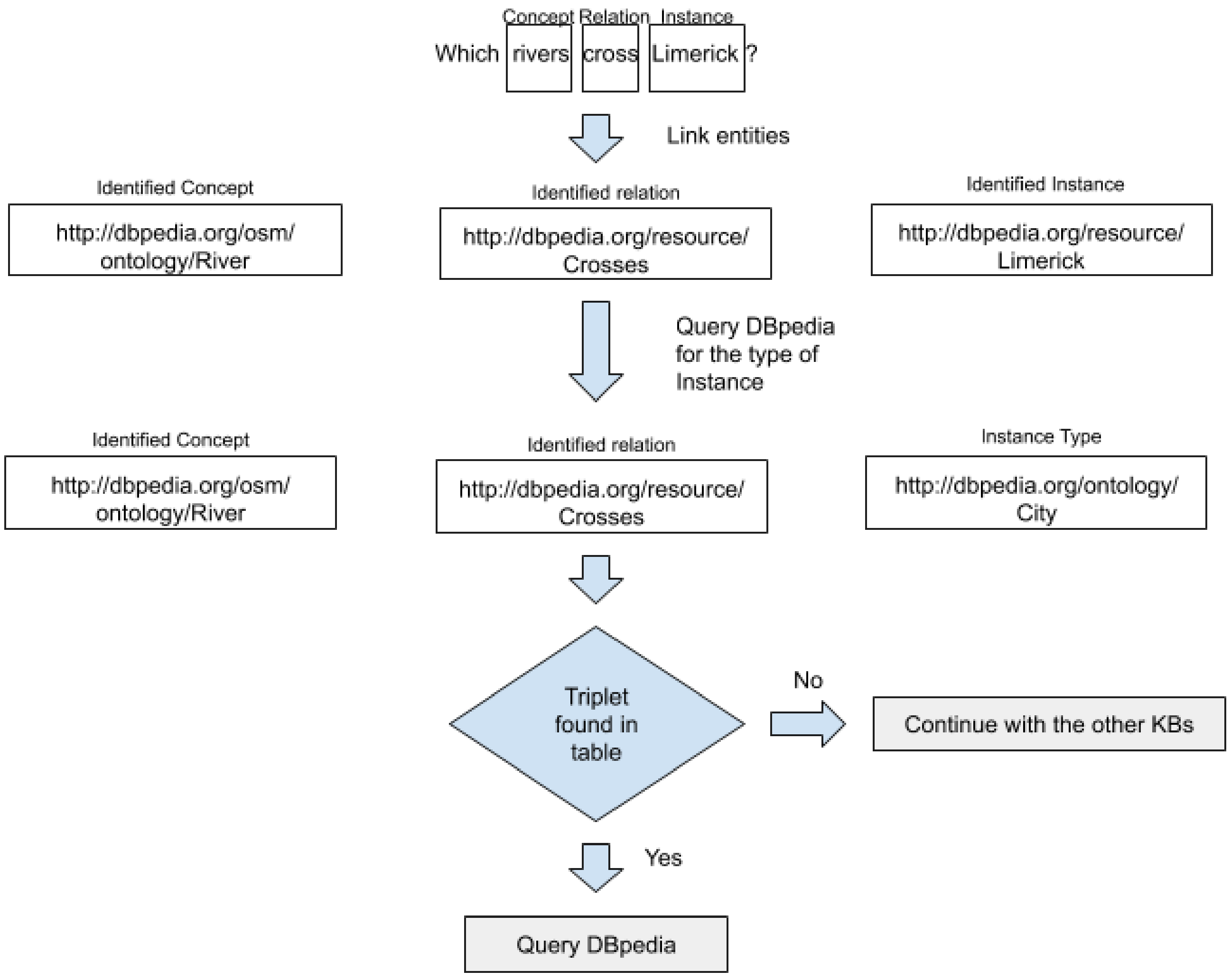

Concept Relation Instance
Which rivers cross Limerick ?
Link entities
Identified Concept
http://dbpedia.org/osm/ontology/River
Identified relation
http://dbpedia.org/resource/Crosses
Identified Instance
http://dbpedia.org/resource/Limerick
Query DBpedia for the type of Instance
Identified Concept
http://dbpedia.org/osm/ontology/River
Identified relation
http://dbpedia.org/resource/Crosses
Instance Type
http://dbpedia.org/ontology/City
Triplet found in table
No
Continue with the other KBs
Yes
Query DBpedia

Fig. 9. Exploitation of Data Schema Information

4.3 Fixing & Adding Templates

The second problem that we managed to solve and significantly improved the results is that of identifying the combined Concept - Instance as one Instance and not as separate entities.

4.3.1 Fix CIs

In order to deal with this problem we altered again the algorithm of Query Generator. In every case where we have one or more Concepts and one or more Instances we check for each of them if they are neighbours using the dependency parse tree. For example in the questions :

“Which bridge cross River Thames ?”
and
“*Which bridge cross the Thames River ?*”

“River” and “Thames” are neighbours in both cases and should both be identified as the entity with label Thames (or River Thames - Thames River depending on the KB) and rdf:type River.

In the next phase, if we have found a pair of Concept - Instance that are neighbours, we query the Knowledge Base for the type of the instance and we compare it with the concept. For example in the question “Which bridge cross River Thames ?” we would have queried the KB for the type of Thames. If the result was something like:

“http://dbpedia.org/ontology/River”

then we would have a match.

After that and as long as we have a match, we can easily combine “River” and “Thames” as one instance. This can be done in two ways. In the first one we can change the instance “Thames” to “River Thames” and delete “River” from the Concepts. The second one is to just delete “River” from Concepts and leave “Thames” as it is. The choice depends on whether the KB has the Instance as “Thames” or “River Thames”.

Here we can see the flow of this algorithm :

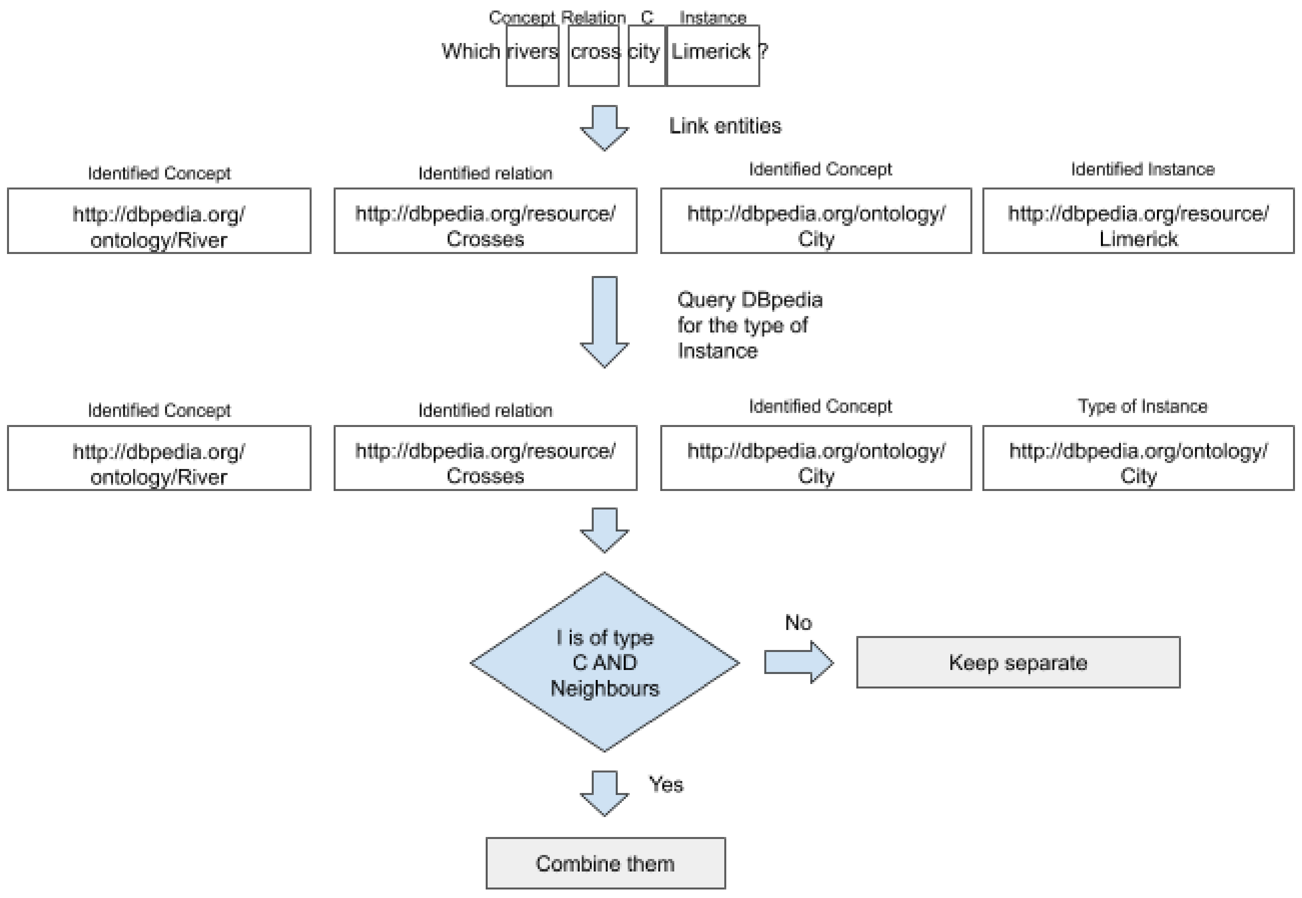

Concept
Relation
C
Instance
Which rivers cross city Limerick ?
Link entities
Identified Concept
http://dbpedia.org/ontology/River
Identified relation
http://dbpedia.org/resource/Crosses
Identified Concept
http://dbpedia.org/ontology/City
Identified Instance
http://dbpedia.org/resource/Limerick
Query DBpedia for the type of Instance
Identified Concept
http://dbpedia.org/ontology/River
Identified relation
http://dbpedia.org/resource/Crosses
Identified Concept
http://dbpedia.org/ontology/City
Type of Instance
http://dbpedia.org/ontology/City
I is of type C AND Neighbours
No
Keep separate
Yes
Combine them

Fig. 10. GeoQA CIs fix

4.3.2 Adding Patterns

As a result from the previous fix, we can now match more patterns introducing the CIs for every previous I. Here we can see the new table with the patterns/templates that GeoQA accepts :

Table 13 : GeoQA Improved Patterns

Pattern	Example
CRI	“Which rivers cross Limerick ?”
CRCI	“Which rivers cross city Limerick ?”
CRIRI	“Which churches are close to the Shannon in Limerick ?”
CRIRCI	“Which churches are close to the Shannon in city Limerick ?”
CRCIRI	“Which churches are close to the river Shannon in Limerick ?”
CRCIRCI	“Which churches are close to the river Shannon in city Limerick ?”
CRC	“Which restaurants are near hotels ?”
CRCRI	“Which restaurants are near hotels in Limerick ?”
CRCRCI	“Which restaurants are near hotels in city Limerick ?”
IRI	“Is Hampshire north of Berkshire ?”
CIRI	“Is county Hampshire north of Berkshire ?”
CIRCI	“Is county Hampshire north of county Berkshire ?”
IRCI	“Is Hampshire north of county Berkshire ?”

4.4 Improved Results

After we finished with these improvements, we rerun the same 86 questions and re-evaluated the performance of GeoQA. The new measures are :

Table 14 : New GeoQA Metrics

	Precision	Recall	F-Measure
New GeoQa	63.06%	69.73%	55.57%

As we can see there is a major rise in the values of Precision, Recall and F-Measure. This is due to the fact that now much more queries are generated, more questions are answered and even more are answered correctly. In fact, 60 questions out of 86 are now answered instead of the previous 42. We can see that in this table :

Table 15 : New GeoQA Component Failures

Module Responsible	Question Number

NER	Q56,Q269,Q333,Q17,Q46,Q92,Q283,Q312,Q320
NED	Q201,Q272,Q273,Q335,Q114,Q241,Q245
Concept Identifier	Q123,Q205,Q323,Q333,Q335,Q117
Relation Identifier	Q236,Q290,Q323
Query Generator	Q304

5. CONCLUSION AND FUTURE WORK

All in all, we developed, improved, tested and evaluated the GeoQA, a Template Based Question Answering system over Linked Geospatial data. This was a challenging task as it requires deep knowledge of how the question answering systems work, how Qanary and Frankenstein modules are put together as well as the way geospatial data are represented, queried and stored in Knowledge Bases.

Regarding the improvements, we achieved much higher results in metrics like Precision, Recall and F1-score by exploiting the data schema information of DBpedia, by fixing templates in order to process CIs as one Instance and by adding more patterns.

Our future plans include further experimentation on complex questions like “Which rivers cross Limerick and their length is more than 300 km?”, on comparing GeoQA with other non geospatial QA engines and on considering text (e.g. travel blogs etc.) as another rich source of geographic knowledge and thus making GeoQA able to discover and exploit such sources using techniques from GIR.

Apart from these, we would like to add support for aggregates like “Which is the biggest county by area in England ?” and also carry out a more detailed evaluation of GeoQA to account for cases where the geospatial information taken from DBpedia needs to be augmented with geospatial information from GADM and OSM to increase the recall of the algorithm.

Finally, other improvements to be done are to use constraints on components in order to identify correct patterns, to add even more templates and to extend the algorithm for Extended Yago dataset, a dataset emerged from WordNet and Wikipedia data, now extended with geospatial data.

REFERENCES

1. GREEN JR, Bert F; et al. (1961). "Baseball: an automatic question-answerer" (PDF). *Western Joint IRE-AIEE-ACM Computer Conference*: 219–224.
2. Woods, William A; Kaplan, R. (1977). "Lunar rocks in natural English: Explorations in natural language question answering". *Linguistic Structures Processing 5*. **5**: 521–569.
3. Hovy E. H., Gerber L., Hermjakob U., Junk M., and Lin C.Y., “Question Answering in Webclopedia.”. In Proc.of the Ninth Text Retrieval Conference (TREC-9). NIST, p.655-664. (2000)
4. Popescu A. M., Etzioni O., and Kautz H. A., “Towards a Theory of Natural Language Interfaces to Databases”. In Proc. of the International Conference on Intelligent User Interfaces, p. 149-157. ACM Press. (2003)
5. De Roeck A N., Fox C J., Lowden B G T, Turner R., Walls B., “A Natural Language System Based on Formal Semantics“. In Proc. of the International Conference on Current Issues in Computational Linguistics, Penang, Malaysia. (1991)
6. Androutsopoulos I., Ritchie G.D., and Thanisch P., “Natural Language Interfaces to Databases - An Introduction. Natural Language Engineering”. 1(1): 29-81.Cambridge University Press. (1995)
7. Moldovan D., Harabagiu S., Pasca M., Mihalcea R., Goodrum R., Girju R. and Rus V., ”LASSO: A Tool for Surfing the Answer Net”. In Proc. of the Text Retrieval Conference (TREC-8). (1999)
8. Harabagiu S., Moldovan D., Pasca M., Mihalcea R., Surdeanu M., Bunescu R, Girju R., Rus V., and Morarescu P., “Falcon - Boosting Knowledge for Answer Engines”. In Proc. of the 9th Text Retrieval Conference (Trec-9), p.479-488. (2000)
9. Litkowski, K. C., “Syntactic Clues and Lexical Resources in Question-Answering”. The Ninth Text REtrieval Conference (TREC-9), NIST Special Publication 500-249. (2001)
10. Kwok C., Etzioni O., and Weld D., “Scaling question answering to the Web”. In Proc. of the 10th International Conference on World Wide Web, p.150-161, Hong Kong, China. ACM (2001)
11. Burke R. D., Hammond K. J.and Kulyukin V., “Question Answering from Frequently-Asked Question Files: Experiences with the FAQ Finder system”. In Proc. of the World Wide Web Internet and Web Information Systems, 18(TR-97-05): 57-66. Department of Computer Science, University of Chicago. (1997)
12. Ferret O., Grau B., Huraults-Plantet M., “Finding an answer based on the recognition of the issue focus”. In Proceedings of TREC-10. (2000).
13. Laurent D., Séguéla P., and NègreCross S., “Lingual Question Answering using QRISTAL for CLEF 2006“. Lecture Notes in Computer Science, Vol. 4730, 2007, pp. 339-350.
14. Parthasarathy S. and Chen J., “A Web-based Question Answering System for Effective e-Learning”. In Proceedings of IEEE International Conference on Advanced Learning Technologies, 2007, pp. 142-146. (2007)
15. Ask.com, checked Aug. 2nd, (2013).
16. Lopez V., Uren V., Motta E., and Pasin M., ”AquaLog: An ontology-driven question answering system for organizational semantic intranets”. Journal of Web Semantics: Science Service and Agents on the World Wide Web, 5(2): 72-105. (2007)
17. Lopez V., Fernández M., Motta E., and Stieler N., “PowerAqua: supporting users in querying and exploring the Semantic Web”. Semantic Web 3(3), 249–265 (2012)
18. Fernandez O., Izquierdo R., Fernandez S., Viciedo J. L., “Addressing Ontology-based question answering with collections of user queries“. Information Processing and Management, 45 (2): 175-188. Elsevier.(2009)
19. Cimiano P., Haase P. and Heizmann J., “Porting Natural Language Interfaces between Domains An Experimental User Study with the ORAKEL System“. In Chin, D. N., Zhou, M. X., Lau, T. S., and Puerta A. R., editors. In Proc.of the International Conference on Intelligent User Interfaces, p. 180-189, Gran Canaria, Spain. ACM. (2007)
20. Bernstein A., Kauffmann E., Kaiser C., and Kiefer C., ”Ginseng: A Guided Input Natural Language Search Engine”. In Proc. of the 15th workshop on Information Technologies and Systems (WITS 2005), p. 45-50. MVWissenschaft, Münster. (2006)
21. Wang C., Xiong M., Zhou Q. and Yu Y., “PANTO: A portable Natural Language Interface to Ontologies”. In Franconia, E., Kifer, M., May, W., editors. In Proc. of the 4th European Semantic Web Conference, p.473-487, Innsbruck, Austria. Springer Verlag. (2007)
22. Damljanovic D., Agatonovic M., and Cunningham H,” Natural Language interface to ontologies: Combining syntactic analysis and ontology-based lookup through the user interaction”. In Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L. and Tudorache, T., editors. In Proc. of the European Semantic Web Conference, Heraklion, Greece. Springer Verlag. (2010)

23. Cabrio E., Cojan J., Aprosio A. P., Magnini B., Lavelli A., and Gandon F., “QAKiS: an open domain QA system based on relational patterns”. In Proceedings of the ISWC 2012 Posters & Demonstrations Track. CEUR Workshop Proceedings, vol. 914 (2012)
24. Axel-Cyrille N. N., Bühmann L., and Unger C., “Sorry, I don’t speak SPARQL – Translating SPARQL Queries into Natural Language”. International World Wide Web Conference Committee (IW3C2).WWW 2013, May 13–17, 2013, Rio de Janeiro, Brazil. ACM 978-1-4503-2035-1/13/05. (2013)
25. Pradel C., Haemmerlé O., and Hernandez N., “Swip: A Natural Language to SPARQL Interface Implemented with SPARQL”, ICCS 2014, LNAI 8577, pp. 260–274, 2014.Springer International Publishing Switzerland (2014)
26. Unger C. and Cimiano P., “Pythia: Compositional meaning construction for ontology based question answering on the semantic web”. In: Muñoz, R., Montoyo, A., Métais, E. (eds.) NLDB 2011. LNCS, vol. 6716, pp. 153– 160. Springer, Heidelberg (2011)
27. Ferré S., “SQUALL: A Controlled Natural Language as Expressive as SPARQL 1.1”. NLDB 2013, LNCS 7934, pp. 114–125, 2013. Springer-Verlag Berlin Heidelberg (2013)
28. Unger C., Bühmann L., Lehmann J., Ngomo A. C. N., Gerber D. and Cimiano P., ”Template-based question answering over RDF data”. In Proceedings of the 21st International Conference on World Wide Web, pp. 639– 648. ACM (2012)
29. Kim J. D. and Cohen K. B., ”Natural language query processing for SPARQL generation: A prototype system for SNOMEDCT”. In Proceedings of BioLINK SIG (2013)
30. Kalyanpur A., et al., “Structured data and inference in DeepQA”. IBM Journal of Research & Development 56(3/4) (2012)
31. Shizhu H., Yuanzhe Z., Liu K., and Zhao J., “CASIA@V2: A MLN-based question answering system over linked data”. In CLEF 2014 Working Notes Papers, (2014).
32. Abdalghani Abujabal, Rishiraj Saha Roy, Mohamed Yahya, and Gerhard Weikum, “QUINT: Interpretable Question Answering over Knowledge Bases”. International World Wide Web Conference Steering Committee (2017)
33. Marx, E., Usbeck, R., Ngonga Ngomo, A.-C., H ¨offner, K., Lehmann, J., Auer,
34. S.: Towards an open question answering architecture. In: Proceedings of the 10th International Conference on Semantic Systems, SEMANTICS 2014, Leipzig, Germany, 4–5 September 2014, pp. 57–60. ACM (2014)
35. Both, A., Diefenbach, D., Singh, K., Shekarpour, S., Cherix, D., Lange, C.: Qanary – a methodology for vocabulary-driven open question answering systems. In: Sack, H., Blomqvist, E., d’Aquin, M., Ghidini, C., Ponzetto, S.P., Lange, C. (eds.) ESWC 2016. LNCS, vol. 9678, pp. 625–641. Springer, Cham (2016). https://doi.org/10. 1007/978-3-319-34129-3 38
36. Kim, J.-D., Unger, C., Ngonga Ngomo, A.-C., Freitas, A., Hahm, Y.-G., Kim, J., Nam, S., Choi, G.-H., Kim, J.-U., Usbeck, R., et al.: OKBQA framework for collaboration on developing natural language question answering systems (2017)
37. Ferrandez, O., Spurk, C., Kouylekov, M., Dornescu, I., Ferrandez, S., Negri, M., Izquierdo, R., Tomas, D., Orasan, C., Neumann, G., Magnini, B., Gonz´alez, J.L.V.: The QALL-ME framework: a specifiable-domain multilingual question answering architecture. J. Web Sem. 9(2), 137–145 (2011)
38. Singh, K., Radhakrishna, A.S., Both, A., Shekarpour, S., Lytra, I., Usbeck, R., Vyas, A., Khikmatullaev, A., Punjani, D., Lange, C., Vidal, M.E., Lehmann, J., Auer, S.: Why reinvent the wheel let's build question answering systems together. In: The Web Conference (WWW 2018)
39. J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by Gibbs sampling. In Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, ACL ’05, 2005.
40. R. Speck and A. Ngonga Ngomo. Ensemble learning for named entity recognition. In The Semantic Web – ISWC 2014: 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I. Springer International Publishing, 2014.
41. P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer. DBpedia spotlight: Shedding light on the web of documents. In Proceedings of the 7th International Conference on Semantic Systems, I-Semantics ’11, 2011.
42. R. Usbeck, A. Ngonga Ngomo, M. Röder, D. Gerber, S. Coelho, S. Auer, and A. Both. AGDISTIS - graph-based disambiguation of named entities using linked data. In The Semantic Web - ISWC 2014 - 13th Int. Semantic Web Conference, Riva del Garda, Italy. Proceedings, Part I. Springer, 2014.
43. S. Shekarpour, E. Marx, A.-C. Ngonga Ngomo, and S. Auer. SINA: Semantic interpretation of user queries for question answering on interlinked data. Web Semantics: Science, Services and Agents on the WWW, 30, 2015.

44. Dojchinovski, M., Kliegr, T.: Entityclassifier.eu: real-time classification of entities in text with Wikipedia. In: Blockeel, H., Kersting, K., Nijssen, S., Zelezn´ ˇ y, F. (eds.) ECML PKDD 2013. LNCS (LNAI), vol. 8190, pp. 654–658. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40994-3_48
45. Moro, A., Raganato, A., Navigli, R.: Entity linking meets word sense disambiguation: a unified approach. TACL 2, 231–244 (2014)
46. Ferragina, P., Scaiella, U.: Fast and accurate annotation of short texts with Wikipedia pages. IEEE Softw. 29(1), 70–75 (2012)
47. Mulang, I.O., Singh, K., Orlandi, F.: Matching natural language relations to knowledge graph properties for question answering. In: Semantics 2017 (2017)
48. Singh, K., Mulang, I.O., Lytra, I., Jaradeh, M.Y., Sakor, A., Vidal, M.-E., Lange, C., Auer, S.: Capturing knowledge in semantically-typed relational patterns to enhance relation linking. In: Proceedings of the Knowledge Capture Conference, K-CAP 2017, Austin, TX, USA (2017)
49. Nakashole, N., Weikum, G., Suchanek, F.M.: PATTY: a taxonomy of relational patterns with semantic types. In: EMNLP-CoNLL (2012)
50. Shekarpour, S., Marx, E., Ngonga Ngomo, A.-C., Auer, S.: SINA: semantic interpretation of user queries for question answering on interlinked data. J. Web Sem. 30, 39–51 (2015)