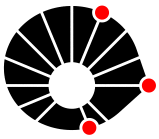

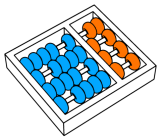

# Spatio-temporal question answering based on RDF knowledge bases

*Pedro Henrique Ferreira Stringhini, Julio Cesar dos Reis*

Relatório Técnico - IC-PFG-19-08
Projeto Final de Graduação
2019 - Julho

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

The contents of this report are the sole responsibility of the authors.
O conteúdo deste relatório é de única responsabilidade dos autores.

# Spatio-temporal question answering based on RDF knowledge bases

Pedro Henrique Ferreira Stringhini, Julio Cesar dos Reis*

July 2019

**Abstract**

Question Answering techniques are part of an ongoing effort to allow more seamless human-computer interactions. These techniques promote accessibility and deliver a potentially better user experience overall, which are desired assets of today's technology. In this context, several challenges remain open such as how to handle complex natural language questions concerning spatial and temporal attributes. Further investigations are required to process these questions and to fully explore existing structured knowledge bases to obtain answers. In this work, we study a spatio-temporal question answering system based on the use of RDF knowledge graphs. Our proposal develops an extension to the Temporal Question Answering system *TEQUILA* consisting of a template-based solution for Spatial Question Answering. Our results effectively create a hybrid Question Answering system that retrieves answers from multiple RDF datasets.

## 1 Introduction

Users need more ways to easily describe their information needs. The use of keywords in search engines, for instance, facilitates automatic query processing but limits the ability of querying direct information from a system. It poses difficulties to non-literate computer users who are usually more inclined to create questions expressing their information needs than to provide a set of keywords. On the other hand, the exploration of natural language questions requires adequate techniques to interpret the inner concepts of the phrase and particularities of the language. Several challenges are present in ways of handling the ambiguity of the language and properly processing questions into structured, adequate queries.

*Instituto de Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP

Question Answering (QA) is a process that interfaces Natural Language Processing and Information Retrieval [19]. It consists in automatically answering natural language questions posed by users using a software system, which receives them as an input and queries a knowledge base to obtain a correct answer [19]. The main objective of this field of study is allowing seamless human-computer interactions, in which the user is spared of having to learn how to build a query with the arbitrary database language in favor of communicating with the system in a natural and direct way. This in turn potentially delivers a better experience overall by promoting accessibility and inclusion.

Question Answering challenges can be split in different aspects [19]. The key tasks in Question Answering include Named Entity Recognition and Disambiguation, Relation Extraction and Query Building [20]. Currently, there are open issues related to how to handle complex questions such as those involving temporal aspects, nested subjects, particular constraints, *etc.* These types of questions involve additional decisions in their treatment regarding pre and post processing, which demands correctly detecting the constraints from natural language and filtering the necessary information.

In this work, we report on a study conducted related to Question Answering with the use of RDF knowledge bases [14]. We aim to handle complex questions related to spatio-temporal constraints. Our proposed system focuses on techniques to parse questions into an intermediary representation with an entity tagger and relation linker, which is then classified accordingly and converted to template-defined queries in SPARQL, a RDF query language [9].

The development of this work relies and extends a pre-existing Temporal Question Answering System named *TEQUILA* [26]. *TEQUILA* implements several different temporal models to address questions, such as "*Who was the President of USA when Maathaad Maathaadu Mallige was released?*" or "*Who is the first husband of Julia Roberts?*". In this investigation, *TEQUILA* was extended with an internally developed solution for Spatial Question Answering to properly answer queries with spatial constraints, which enable us to answer questions such as "*Which is the second closest University from the Eiffel Tower*" or "*Which cities are near Brasília*". Our proposal provides flexibility in the type of inserted natural language questions and enables the combination of queries to explore spatio-temporal requirements.

In our extended system, the input query is processed in order to retrieve constraints such as a date, in case of temporal questions, or coordinates, in case of spatial ones. Then, with the constraint set, a subquery is used to retrieve the main information requested from RDF graphs available in the Linked Open Data (LOD) [22]. In particular, our solution defines SPARQL templates based the DBPedia knowledge graph [1]. At the final stage, the obtained results are filtered depending of the needs of the question.

The remaining of this report is organized in the following structure: Section 2

presents fundamental concepts related to Semantic Web techniques and languages in addition to Question Answering. This section also discusses related work. Section 3 describes the development of our Spatial Question Answering as an extension of *TEQUILA* system [26]. Section 4 discusses the obtained results. Finally, Section 5 describes the conclusions and future work.

# 2 Theoretical Background and Related Work

In this section, we present the concept of Structured Knowledge Bases with the use of the RDF model (Subsection 2.1). We discuss key parts of question answering systems in Subsection 2.2. Subsection 2.3 describes a synthesis of related work.

## 2.1 RDF-based Structured Knowledge Bases

This work explores the use of graph-based knowledge bases for question answering. In particular, we rely on datasets described with the *Resource Description Framework* (RDF). RDF is an data modeling standard that facilitates information interoperability across different underlying schemas [6]. A RDF graph is composed by a set of triples formed by a subject, object and predicate. The triples aim at defining a relationship between two resources (the subject and the object) [6]. These resources are defined by an *Uniform Resource Identifier* (URI) [6] as an address that unambiguously identifies an element of the triple. The set of triples generates an oriented graph, which can be queried.

Figure 1 presents an example of a RDF graph. In this graph, the green element at the start of the arrows is the subject, "me", which the URI identifies as a contact (address before the # sign). At the end of the arrows there are objects, which contain information regarding contacts. In this example, "me" is characterized as a 'Person'. The arrows themselves are the predicates, that indicate the kind of relationship between the subject and the object.

RDF is one of the key aspects of the Semantic Web, which is an effort towards organizing information with machine-readable semantics, and thus allowing connecting and processing distributed knowledge [8]. It revolves around the concept of Linked Open Data (LOD) [22], free, open-source, reusable data composed by URIs and connected to other data sets. Currently, the amount of data that follows this standard is increasing and can be openly used by software applications. One meaningful example is *DBPedia* [1], a community effort to extract structured data from Wikipedia.

This data is accessed through SPARQL, a RDF querying language. SPARQL allows querying information in triples that satisfy a specified set of constraints. The information queried can be related to object, subject, predicate or any combination of those three, and a range of combinations is available. In the following, we present

Figure 1: Example of RDF graph [7]. The start of the arrows are the subject and their end, their respective objects. The arrows themselves are the predicate. All elements are identified by URIs

an example of a SPARQL Query.

The following query selects the band members of Punk Rock bands, outputting the name of the person and the corresponding band name. It also, for illustrative purposes, outputs the predicate used to indicate which genre a band belongs to. In This query, it is possible to notice the definition of prefixes for URIs (accessing *DBPedia* and *xmlns*), the variables queried (all of which are identified by the *?* sign), and several constraints of different types for triples.

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbp: <http://dbpedia.org/resource/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?name ?bandName ?genrePredicate where {
    ?band dbo:bandMember ?person ;
          foaf:name ?bandName ;
          ?genrePredicate dbp:Punk_rock .
    ?person foaf:name ?name .
}
```

For instance, the first constraint has the subject and object as variables, connecting the meaning of the variable "person" to a "bandMember" of "band". Note the semicolon at the end of the line indicating that the subject of the next constraint is the same as the current one. Similarly to the first query, the next line associates the variable "bandname" to a "name" of the "band" variable. The next constraint indicates a "Punk_rock" object; this means that only subjects "band" who have a property named "Punk_Rock" will be outputted, effectively filtering for punk rock bands even if the predicate itself is not defined and set as a variable. The last constraint associates the same "person" which is a "bandMember" to their name, which will be also outputted as can be seen in the "SELECT" line.

## 2.2 Question Answering with RDF databases

Figure 2 presents a Question Answering system flow. First, processing steps represented by the Question Parsing and Query Construction element convert a Natural Language Question into a intermediary representation. This intermediary representation, in turn, should go through additional processing steps to generate a query format chosen, used to access the knowledge base. These processing steps range from disambiguation techniques to systems to circumvent the lexical gap existing in the data [19]. Then a SPARQL query [9] is used to reach the RDF database. Its output is then parsed into an answer, commonly presented also in natural language, but not restricted to it.

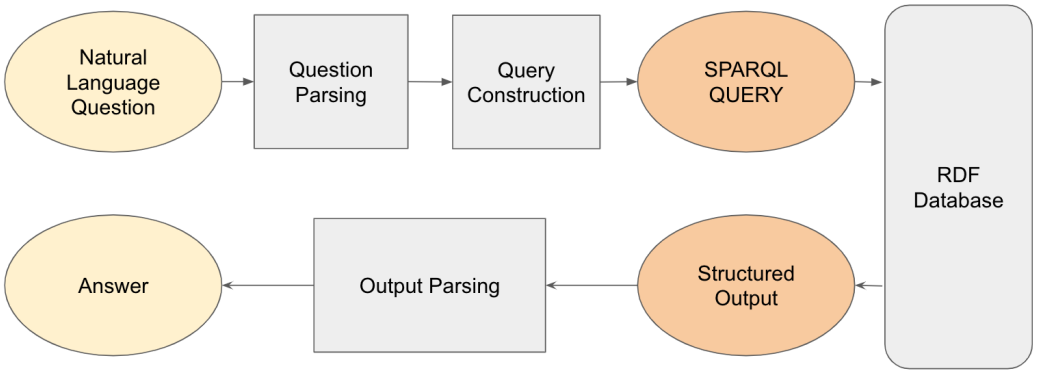


Figure 2: Basic QA workflow with the use of RDF datasets. [19]

Figure 3 presents a process that converts Natural Language Questions into SPARQL queries. Question Parsing is the first step of the Question Answering task and a research problem in the field of Natural Language Processing. This process, in turn, can be separated into a few substeps, which are, commonly, tokenization, tagging, lemmatization and regular expression matching. Tokenization splits sections (commonly words) from a raw string, tagging attributes values for the tokens if necessary and effectively converting this raw string into a list of tokens with these tags; lemmatization links inflected forms of the words in the token list (*i.e.*: the word "good" is the lemma of the word "better"); and regular expression matching identifies whether the phrase (in this case, the question in the tokenized and normalized representation) is in a form accepted by the system, by associating it with a predefined pattern [20].

Named Entity Disambiguation is the task of extracting important elements of the input question, such as the the tokens that relate with RDF entities, and processing them in terms of adding proper tags and applying techniques to prevent ambiguity. It effectively fine-tunes the Natural Language Parsing, clearing whether, for instance, "*leaves*" should be classified as a form of the verb "*leave*" or the plural of the noun "*leaf*" instead. Relation Linking connects the elements of the question derived from the token list by using the predicates and formatting of the question. Query Building uses the intermediary representation given by the other components to build a SPARQL query [20].

## 2.3 Related Work

In the study "*Why Reinvent the Wheel – Let's Build Question Answering Systems Together*", Singh *et al.* [20] proposed the devise of a system called *Frankenstein*.

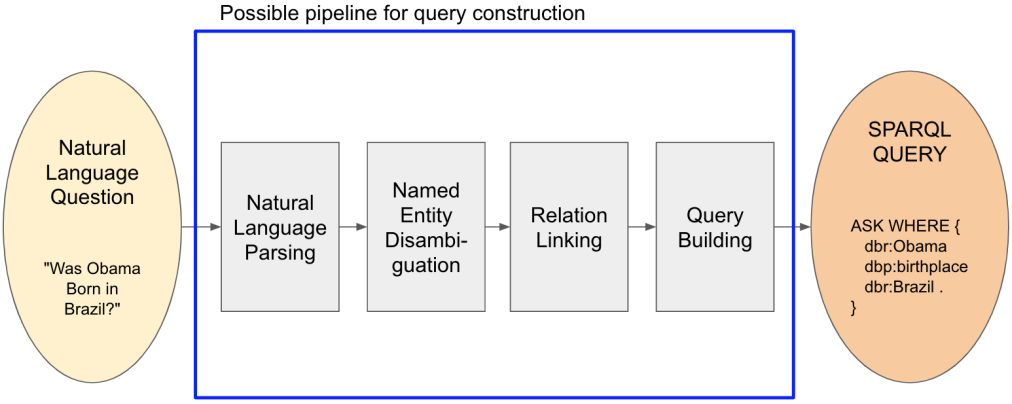


Figure 3: Conversion of Natural Language Questions into SPARQL queries. This model describes the Question parsing and Query Construction steps of figure 2 in more detail.

This system selects, from a group of provided Question Answering components, which combination of them are better suited for a particular task, using trained classifiers. In the study, several components were tested that perform the tasks of Named Entity Disambiguation, Relation Linking and Query Building. *Frankenstein*'s classifiers, then, select a combination of these components, in order to build an optimal Question Answering Pipeline. On the other hand, Unger *et al.* [14] proposed a system that uses template matching to create SPARQL queries to directly mirror the natural language questions received as input.

Höffner *et al.* [19] conducted a recent survey on challenges of question answering in the Semantic Web. They analyzed several Semantic Question Answering Systems and discussed the addressed challenges, proposing solutions and recommendations for future systems.

**Lexical gap.** This includes: string normalization/similarity functions, which aims to address typos and lexical variations of the same word (verb tenses for example); automatic query expansion to address synonyms and hyper-hyponym-pairs; pattern libraries, which address for example variations on how the same question can be formulated; and question entailment to get information of other questions that imply the answer of the desired one.

**Ambiguity.** This includes both homonomy (same string with different concepts) and polysemy (string with different but related concepts). Höffner *et al.* [19] discussed disambiguation methods, both Corpus-Based, which is interested in the context of the information with statistical approaches; and Resource-based, which exploit the fact

that the information is stored in RDF structures to analyze its connections and evaluate candidates. In this context, Lou *et al.* [21] proposed a system as an approach to ambiguity that creates a semantic query graph, which looks for subgraphs that match the requested structure. The system provides disambiguation of natural language questions in the subgraph matching phase, instead of in the query creating phase.

**Complex queries.** These relate to usually bigger questions that contain nested or composite information. *LC-QuAD* is a corpus for complex question answering over knowledge graphs [23], as an effort made to create an extensive dataset with 5000 questions ensuring a high variety of types, including complex ones, alongside their respective SPARQL queries over the DBPedia dataset. The following questions and respective SPARQL queries are examples of the contents of the *LC-QuAD* dataset [4]. These complex questions pose open challenges for the development of question answering systems. This is due to the fact that creating templates for each one of arbitrarily, context-free complex questions can potentially be very inefficient, which demands more robust and varied strategies.

**Complex question example 1.** "*Which kind of conventions are held in Rosemont, Illinois?*"

```
SELECT DISTINCT ?uri WHERE {
?x dbo:location dbr:Rosemont,_Illinois .
?x dbo:recordLabel ?uri .
?uri rdf:type dbo:Convention .
}
```

**Complex question example 2.** "*Which labels sign up progressive rock bands?*"

```
SELECT DISTINCT ?uri WHERE {
?x dbp:genre dbr:Progressive_rock .
?x dbo:recordLabel ?uri .
?uri rdf:type dbo:MusicalArtist .
}
```

**Complex question example 3.** "*Name the scientist whose supervisor was Ernest Rutherford and had a doctoral students named Charles Drummond Ellis?*"

```
SELECT DISTINCT ?uri WHERE {
?uri dbo:doctoralAdvisor dbr:Ernest_Rutherford .
?uri dbp:pastMembers dbr:Charles_Drummond_Ellis .
?uri rdf:type dbo:Scientist .
}
```

**Procedural, Temporal and Spatial questions**. These types of questions were identified as key challenges. They relate respectively to questions that ask for procedures (*i.e.* "how to do something"), and that order events based on time (*i.e.*,

"which of the European capitals is the oldest") and proximity (*i.e.*, "list mountain ranges close to São Paulo"). These types of questions demand possibly more processing layers to successfully extract and parse the information requested to the end user. This means, for instance, handling multiple queries and post-processing of the answers obtained from the RDF dataset. Subsections 2.3.1 and 2.3.2 describe further details of the challenges presented in these types of questions. Temporal and spatial question answering are the focus of this investigation.

### 2.3.1 Temporal Question Answering

A key open challenge in Question Answering is related to complex questions which are connected to temporal information, potentially demanding additional layers of processing to achieve a correct answer. Zhen *et al.* [26] argued that question decomposition is necessary to correlate temporal information between events in a question, when there is not an explicit, single-event temporal tag.

Saquete *et al.* [16] proposed a taxonomy for temporal questions separating them in four types:

1. *Single temporal questions without temporal expression*: questions formed by a single event able to be solved by a standard question answering system. For instance, "*When Was Albert Einstein born?*".

2. *Single event temporal questions with temporal expression*: the event in the question is accompanied by an explicit temporal annotation that needs to be analyzed. For instance, "*Who won the Nobel Prize in 1922?*".

3. *Multiple events temporal questions with temporal expression*: there are multiple events related by a temporal signal plus temporal expressions. For instance, "*Which paper did Einstein publish after the 1924 Summer Olympics?*".

4. *Multiple events temporal questions without temporal expression*: There are multiple events with implicit temporal signals, indicating ordering of events. For instance, "*Which papers did Einstein publish after the end of the Second World War?*".

Saquete *et al.* [16] defined a system split into a Question Decomposition Unit, a general purpose Question Answering System, and an Answer Recomposition Unit. Their investigation focused on the decomposition unit, which was separated in 3 steps. First, it classifies the question into one of the four types described above. Then it recognizes both explicit and implicit temporal expressions to correctly create an interval in which to filter the answers queried by the Question Answering system. Then the unit finds temporal signals (such as *before*, *during* and *since*) to both be

used in the recomposition unit and to split the question into two ones to be queried, if they are classified as types 3 or 4.

Zhen *et al.* [26] proposed the *TEQUILA* system, which is similar with the study conducted by Saquete *et al.* [16] in terms of the pipeline converting the original natural language question into sub-questions.

Zhen *et al.* defined a classification based on these time constraints. It has four classes:

1. *Constraint has both a named entity and a relation.* For instance: "*Where did Albert Einstein live before winning a Nobel Prize?*". Here the named Entity is "*Nobel Prize* and the relation is "*winning*".

2. *Constraint has no entity but a relation.* For instance: "*Where did Albert Einstein live before starting secondary school?*". Here, the relation is "*starting secondary school*"

3. *Constraint has no relation but an named entity.* For instance: "*Who won the Nobel Prize before Einstein?*". Here, the named entity is "*Einstein*" and the relation is not present, but inferred as "*...Won the Nobel prize*", in the end of the sentence.

4. *Constraint is an event name.* For instance: "*Which papers did Einstein publish after the 1924 Summer Olympics?*". Here, the event is the "*1924 Summer Olympics*", which has an intrinsic date value.

The *TEQUILA* system queries temporal questions for answers, decomposing the question into a main question and temporal sub-questions, and using these sub-questions as time constraints to filter the main question.

The Decomposition system in *TEQUILA* searches for a temporal signal (such as the token "when") to split the question into a temporal sub-question and a non-temporal main question. After the spitting, SPARQL queries are built. The sub-question is used to retrieve the time constraints. These time constraints retrieved in the temporal sub-questions are used to filter the answers of the non-temporal ones to build a final answer. Subsection 3.1 provides further details of the *TEQUILA* framework as the basis for this work.

For evaluation purposes, the *TEQUILA* was attached to the Question Answering Systems AQQU [17] and QUINT [13]. The system was then evaluated with the *TempQuestions* [25] corpus and the 341 temporal questions in the *ComplexQuestions* corpus.

### 2.3.2 Spatial Question Answering

Spatial Question Answering is a key challenge in the field. It aims at using geospatial data to retrieve questions with spatial constraints [15]. This task, like in Temporal Question Answering, demands extra pre and post processing steps such as, for instance, question splitting, constraint retrieving and answer filtering.

In this context, Pujani *et al.* proposed a system which uses *Frankenstein* [20] to develop a multi-component question answering to retrieve answers from multiple sources, including: 1) *DBPedia*; 2) *LinkedGeoData* [5], an Open Linked Data modeling of the Open Street Maps data; and 3) General Administrative Divisions data set (GADM) [3], which contains info regarding the administrative division and boundaries of several countries.

Pujani *et al.* created a gold-standard set of geospatial questions by interlinking data from *DBPedia*, *LinkedGeoData* and *GADM*. They created 201 questions based on these three databases. The authors used SPARQL and the extension *GeoSPARQL* to query the three sources and rank the answers.

In *Enabling the geospatial Semantic Web with Parliament and GeoSPARQL*, [24] Battle *et al.* described the motivation behind *GeoSPARQL*, which is unifying access to the geospatial Semantic Web, due to the rising amount of data stored with an inherent spatial subtext. *GeoSPARQL* became an Open Geospatial Consortium (OGC)-standard for accessing geospatial data stored as a RDF triple.

# 3 Spatio-temporal query answering with RDF graphs

We describe the way the *TEQUILA* system operates (Subsection 3.1) followed by our method for developing a *TEQUILA*'s extension for Question Answering with spatial constraints (Subsection 3.2). Then, we report the implementation aspects of our solution (Subsection 3.3).

## 3.1 *TEQUILA* Framework

*TEQUILA*'s architecture is structured in a sequential pipeline of operations over a natural language question as input. These operations incrementally convert the input question into the desired model, which is a set composed of one or more *SPARQL* queries. Then, additional operations are applied to the data retrieved from the defined queries to provide an answer to the original question. In *TEQUILA*, these operations are called processes. Figure 4 presents the main original *TEQUILA* pipeline, focusing in relevant features of the system. In this pipeline, the question is parsed and split into a main question and a sub-question, both of which are queried to an external QA system. The sub-question answers are used as filters for the main question answer.

The order of the processes applied for answering temporal questions are as follow.

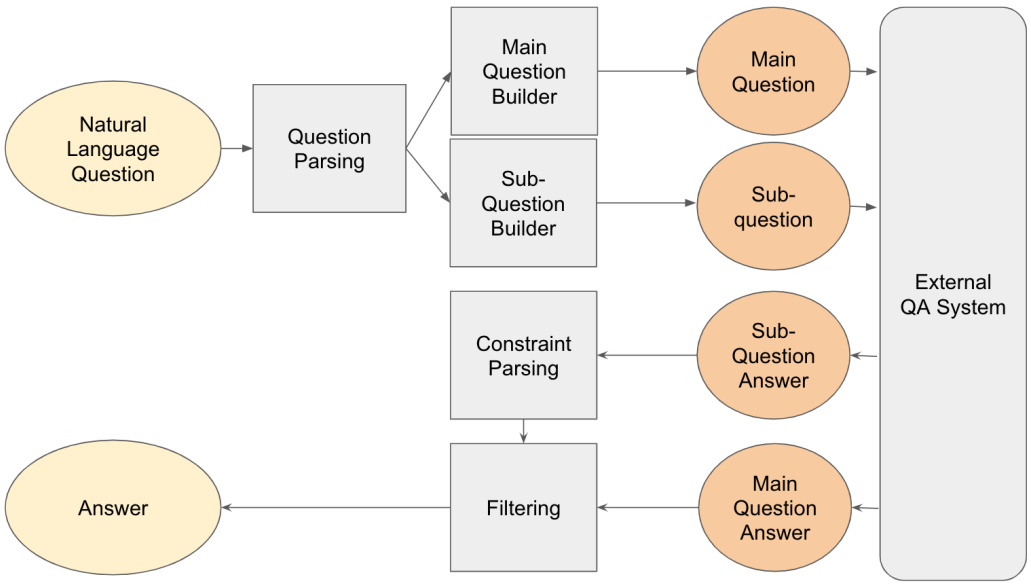


Figure 4: *TEQUILA* Temporal system overview. Sub-question handling and filtering processes are key challenges of spatio-temporal question answering systems, which differentiate this pipeline from the standard QA workflow presented in Figure 2

1. NLP start-up process
2. Named Entity Recognition process
3. Event tagging process
4. Time tagging by Heidel Time process
5. Sentence Decomposition process
6. Marking "*When*" tag process
7. Marking Date Ordinal rule process
8. Fixing invalid tags process
9. Rewriting simple question process
10. Rewriting sub-question process
11. QA Policy Selection process
12. Question Answering process

13. QA Service process

14. Filtering process

Figure 5 represents the flow of the *TEQUILA*'s pipeline for formatting a question. Processes 1 to 4 are indicated by the first 4 steps of the workflow. Next to them, processes 5 and 6 are encapsulated by the Question Decomposition box, as well as processes 7 and 8 in the Additional Tagging Processes box. Processes 9 and 10 are indicated by the Question Builder Boxes, and process 11 is indicated by the Question Classification box. The last 3 processes happen after the questions are formatted and classified, completing the flow represented by figure 4.

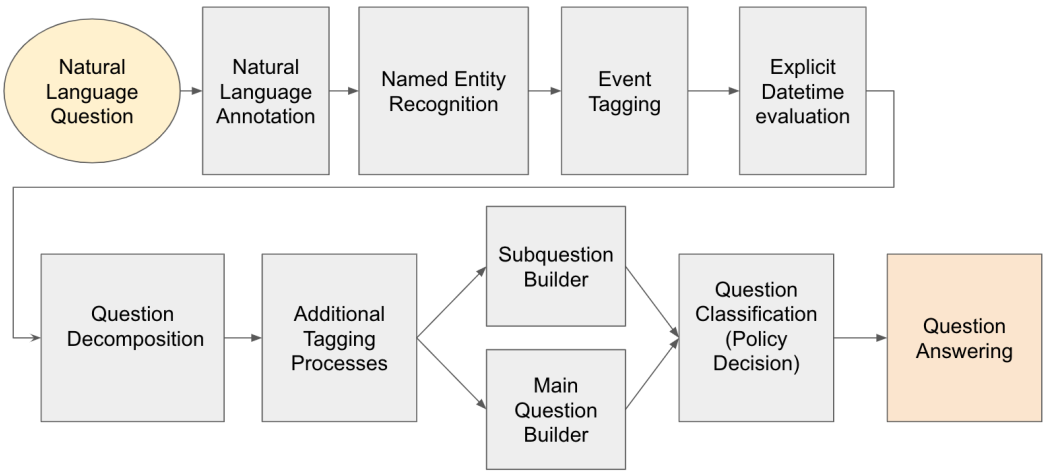


Figure 5: *TEQUILA*'s Question processing workflow.

The NLP start-up process uses Stanford Natural Language Processing tools [11] [10] to annotate the input question, which means adding metadata with additional, potentially useful information from the sentence. It uses log-linear Part-of-Speech tagging [11] to describe the representation of which role elements take in the sentence. These elements can be classified in categories such as noun, verb, adjective, and even more specifically, like "verb-past" or "noun-plural". This preprocessing step is used to tokenize and annotate the sentence and facilitates further analysis.

Named Entity Recognition is the process of identifying words or sequences of words that represent named objects [10], such as person or location names. For instance, in the sentence "*Dorothy is not in Texas anymore, but in New York*", there are 3 named entities: "Dorothy", "Texas" and "New York". In *TEQUILA*, this is specifically useful for query building.

Event tagging, in this context, searches for actions or events in the sentence, such as "run" or "created". Afterwards, *TEQUILA* considers *HeidelTime* [18] as a

system for extraction and normalization of temporal expressions. It can effectively convert different types of expressions representing dates (such as "October of 1995" or "21st of December in 2007") to the standardized format "*YEAR-MONTH-DAY*", for example: "2007-08-17". This data is used when the question has an explicit date constraint for filtering purposes.

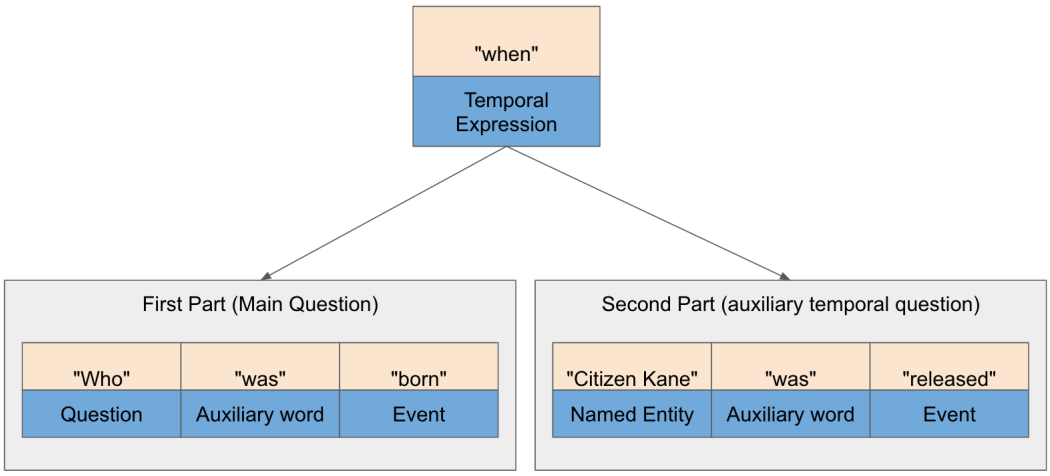


Figure 6: The resulting structure of the question decomposition process for the question "*Who was born when Citizen Kane was released?*". This process uses the tagging information to separate the question into tokens, represented in yellow, and register its type, represented in blue by the figure. The question is also divided into a main question and an auxiliary question, based on the temporal tag. The sub-question is represented by the rightmost part, and can be blank for simple questions where there is no demand for splitting.

After the annotation processes (event and time tagging), the sentence decomposition process gets both the original question and the metadata generated to create a structured representation of the provided information, composed by nodes. These nodes store the tokens retrieved for the question alongside their metadata. In figure 6, for instance, one of the nodes is formed by the token "Citizen Kane" alongside its classification as a Named Entity.

The following Marking "*When*" tag process looks for tokens that represent temporal signals, which are: "before", "prior to", "after", "during", "while", "when", "since", "until", "in" and "at the same time as".

If a temporal signal is found, then the question is classified as complex and the structure is split in two parts. One part has the tokens before the signal and the other has the tokens after the signal. This effectively creates a tree-like structure with an

empty root level and the a level with these two parts. Otherwise, if it is a simple sentence, this structure has only one non-empty part composed of all nodes of the sentence. The second part is empty and subsequently not used.

Based on this structure set, an additional tagging process occurs to the structured information tree generated. It searches for **Date** ordinal tags based on specific matches of a rule set. Ordinal tags are tokens that represent order (such as "first", "second" and "last"). This aids in question classification. Then, the fixing invalid tags process removes all duplicated or wrongly analyzed tags, if they exist.

In the sequence, the rewriting simple question process trims the first part of the structure created until it reaches its simple form, which removes date components and focuses on the main information queried. For instance, if a question is "*Which is the first husband of Julia Roberts?*", this process provides the simple question "*Who is the Husband of Julia Roberts?*" in string form, which in later stages would output every person in a knowledge base who is or has been Julia Robert's husband.

If the second part of the structured information tree created exists (cf. Figure 6), then it is analyzed by the rewriting sub-question process. This process generates a string that represents a sub-question. This sub-question queries for time constraints, which are then used to filter the main question in later stages. For instance, the sub-question extracted from the original question "*Who was the president of Brazil when Marie Curie won the Nobel prize for Chemistry?*" is "*When did Marie Curie win the Nobel prize for Chemistry?*". The answer of this sub-question provides the year 1911, which is later used to filter the main question generated, "*Who was the president of Brazil?*" for the correct date constraint.

At this stage, the process of QA policy selection takes place. QA policy selection analyzes the structured information tree provided from the previous processes to obtain a decision on what kind of Question Answering and filtering Processes to use. This decision takes into consideration: 1) if sub-questions were created from the original question; 2) if there are ordinal tags present; and 3) what is the format of the syntactic tree-like structure created.

The QA policy selection process defines both the Question Answering process and the filtering process. The actions taken for their respective question types are the follows:

- **Simple questions with no temporal data.** A basic Question Answering process is called, retrieving an answer from the service of choice and returning it.

- **Questions that have an explicit date constraint**. The main question is queried from the service of choice, returning a list of candidate answers which are then filtered using date constraint to eliminate answers outside the desired temporal limits provided. This filter can be a specific date (with day, month and year), or an interval consisting of days, months or years.

- **Questions that have a sub-question created in the previous processes**. Firstly, the sub-question is queried from the service of choice, whose answer is then parsed to a standard format, which is the same as the formatting for explicit date types. Then, the main question is queried and filtered with a process that is equivalent to the previous one.

- **"*when*"-type questions**. The result is in itself a date. These questions receive the same treatment of the aforementioned sub-questions, outputting a formatted date.

- **Question presenting an ordinal tag**. This is an additional case that happens when there is an ordinal tag attached to the other types of questions. This leads to an additional processing that sorts the answers according to their date values, to then select the desired answer. For instance, when there is a "second" ordinal tag, the answers are ordered and the one with the second oldest date information is selected as the correct answer.

With the QA Policy selected, the respective question answering and filtering systems take place, taking the question and sub-question strings as input.

One point worth noting is that the question answering process uses the QA service process to retrieve the answers from the questions generated. The QA service is an interface process that connects with an external question answering system selected by the client in the application start-up. That way, the user selects which service *TEQUILA* must forward to to properly create a query for the simple, trimmed down questions. By default, *TEQUILA* offers back-end capabilities and front-end support for *AQQU* [17] and *QUINT* [13]. *AQQU* provides a learning-to-rank methodology and addresses with considerably effectiveness the entity recognition problem, efficiently building queries for *Freebase* [2], a large collaborative RDF database. *QUINT* is a Question Answering system that automatically learns role-aligned utterance-query templates [13].

## 3.2 Extended *TEQUILA* for Spatial Question Answering

We extended the *TEQUILA* system with another category of Question Answering, addressing the challenge of answering questions with spatial constraints. These type of questions, in parallel to temporal question answering, demand multiple processing steps and potentially more than one query to be created and submitted to structured knowledge bases [15]. For this purpose, we defined an amount of changes in the original architecture of *TEQUILA*.

Figure 7 presents a high-level representation of the workflow of the new Spatial Question Answering-extended TEQUILA created in this study. In this approach, all the main question processing flow of *TEQUILA* happens normally, with the addition

of spatial tagging for the spatial question classification. Then, in contrast with the original *TEQUILA*, the query construction step runs internally to the main application. This step builds the spatial sub-question query and reaches the database for results. These results, in turn, are used as in-query constraints when building the main query to enable a better spatial filtering of the main question answers. The main result is then parsed and outputted for the client.

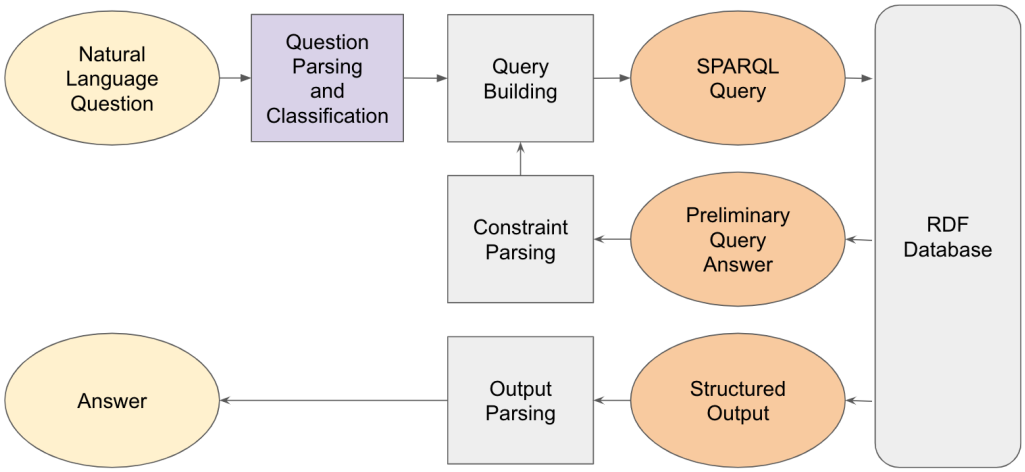


Figure 7: Extended *TEQUILA* Spatial module overview. The Purple box "Question Parsing and Classification" combines the *TEQUILA* processes 1 to 11 with the added tagging of spatial elements and the classification for spatial questions. It can be observed that, in opposition to the original *TEQUILA*, Query Building (with the new in-query filtering feature) happens inside the system and not from an external source. Also, the main query building depends on the sub-query answer, as the cyclic flow suggests.

Our contributions and key modifications in the development of the extended *TEQUILA* Spatial Question Answering system were in three main categories as follows.

- Question Classification
- *SPARQL* Query Building for DBPedia
- In-query Answer filtering

**Question Classification.** *TEQUILA* classifies the input questions in categories, selecting the Question Answering and filtering modules according to the different

types of questions. Based on this pattern, we defined a new category for spatial questions.

In *TEQUILA*, classifying questions is made primarily through comparison and fitting of templates represented by regular expressions. Consider the following question as an example for illustrating our technique throughout the defined steps: "*Which university is closest to Eiffel Tower?*".

In our example, regular expression matching is used to identify the question as spatial question with relative location information, which means the spatial information is inferred from the "*Eiffel Tower*" location. This is checked mainly by finding specific spatial tokens (in this case, "*closest*"), which is a process similar to what the original *TEQUILA* flow does with temporal data.

These tokens can indicate a location request, such as the token "where", or distance constraints, indicated by tokens such as "*close/closest/nearest*" preceded or not by an ordinal token. If a distance constraint token is found, a process splits the sentence, using the token location, in two parts. Then, these parts are formatted into a main question and spatial-constraint retrieving sub-question. For a question such as "*Which beach is near to São Paulo*", "*near*" is a distance constraint token, which means "*Which beach*" forms our main question, and "*São Paulo*" forms the spatial sub-question.

The type of question classified as a temporal question corresponds to the following format:

[WH*] [*.] [ELEM] [*.] [*ORD] [DIST_CONST] [NAMED_ENT]
[where] [is/are] [ELEM/NAMED_ENT]

In this format, we have the following elements:

- the [WH*] is a subset of question words ("wh-like" elements): "which", "what" or "who";
- the [ELEM] represents a type of element, such as "person", "hospital" or "beach house";
- the [NAMED_ENT] represents a named entity, such as "Paris" or "Grand Canyon";
- the [*ORD] represents an optional ordinal token, such as "first", "second" or "last";
- the [*.] represents generic element(s), such as "is/are" or other unimportant tokens;

In our running example, after recognizing the question as of a spatial type, "*closest*" is identified as a distance constraint. This splits the question into the main question and a sub-question, which is "*Which university*", and "*to Eiffel Tower?*".

The next step is formatting the questions. Formatting the main question means identifying the [WH*] token "*Which*" as a question word and changing it for a "*List*" token. This turns the sentence into "*List Universities*", which asks for, as the sentence itself suggests, a list of universities. For the sub-question, the Named Entity Recognition process output is retrieved. The output, in our example is the "*Eiffel Tower*" name, which is then added to a "where"-type question template, "*Where is Eiffel Tower*".

**SPARQL Query Building for DBPedia.** The QA service selection of the original version of *TEQUILA* supported the external question answering systems *AQQU* [17] and *QUINT* [13]. In this case, the main and sub-questions generated are sent in natural language for an external service, which then converts the questions appropriately and sends the generated *SPARQL* queries to the RDF dataset.

Our spatial question answering system developed a built-in template-based query generation process to access *DBPedia*. This means that, when the question is classified as a spatial question, the query building happens inside *TEQUILA* and not in an external source. This choice was made mainly due to the *DBPedia* attribute of storing geo-coordinate information for most of the applicable entities. With this strategy, the queries generated can fully explore these *DPBedia* feature.

our template of the *SPARQL* query building model obeys the following rule set:

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbp: <http://dbpedia.org/resource/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
[ADDITIONAL PREFIXES]

SELECT * WHERE {
    dbr:[RESOURCE]
        [GEO URI]:[latitude info] ?lat ;
        [GEO URI]:[longitude info] ?long ;
        [ADDITIONAL FILTERING DATA]
}
```

The ***ADDITIONAL PREFIXES*** variable offers the possibility of dynamically defined prefixes, generally corresponding to geo information. One widely used in the actual queries is:

```
PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
```

Other options, however, can be used for the GEO URI field, such as *DBPedia* property (*dbp*) itself.

The ***RESOURCE*** variable can refer to one of two meanings, depending on the type of the question. On one hand it can be a Named Entity, such as "*Eiffel Tower*", when querying a sub-question or a simple question (for example, "*Where is Eiffel Tower?*"). In this sense, the posed query is used to get coordinates of this particular Named Entity. On the other hand, in questions such as "*Which Universities are in Paris*" the ***RESOURCE*** field becomes a *SPARQL* variable. This occurs because this resource is actually the answer we are looking for, which is, in the example, names of universities that match the required constraints.

The ***ADDITIONAL FILTERING DATA*** piece of the request has two main uses. The first is providing additional information to reduce the amount of outcome answers. For instance, listing all Universities could generate many results, so filtering with the city of Paris (*dbr:location* or *dbo:city*) is interesting to reduce additional processing such as ordering procedures.

The second use of ADDITIONAL FILTERING DATA is to directly indicate which type of entity the question is interested in. In this sense, for the case of listing universities, the field "?res a dbo:University" is a necessary one. This information is retrieved from the simple question, extracting the token which corresponds to the entity type info.

Using this template, the query generated for the question "*Where is the Eiffel Tower*" is:

```
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>

SELECT * WHERE {
    dbr:Eiffel_Tower
        geo:lat ?lat ;
        geo:long ?long .
}
```

For the question "*Which Universities are in Paris*", the generated query is:

```
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>

SELECT * WHERE {
    ?resource
```

```
        geo:lat ?lat ;
        geo:long ?long ;
        a dbo:University ;
        dbo:city dbr:Paris .
}
```

**In-query answer filtering.** The traditional *TEQUILA* temporal question answering process delegates the filtering process to occur after the results from the queries are gathered from the RDF dataset. In our new Spatial Query System, an additional option was developed to make the correct constraint analysis as in-query answer filtering. It means that after the sub-question query is built and its information is retrieved from the database, the same information is used inside the main query for filtering purposes. For instance, if the answer for the sub-query "*Where is Eiffel Tower*" is a set of coordinates, this coordinates are added to the filtering portion of the main query. This decision was made to reduce the number of results retrieved from *DBPEdia*, already trimming at the source those which do not fit optimally in the constraints.

This feature of *SPARQL* queries supported by *DBPedia* is represented by the keyword *FILTER*. It allows basic true/false (Boolean) operations with the data queried, selecting which answers are or not representative of the desired result. In our spatial question answering, this filter, alongside the support for floating number comparison, is useful when the target coordinates are already defined from the sub-question. An example of this filtering created in the system is:

```
FILTER ( ?long > [LONGITUDE_CONSTRAINT] - [RADIUS] &&
?long < [LONGITUDE_CONSTRAINT] + [RADIUS] &&
"?lat > [LATITUDE_CONSTRAINT] - [RADIUS] &&
?lat < [LATITUDE_CONSTRAINT] [RADIUS]
```

In this filtering, the constraints are the latitude and longitude retrieved from *DBPedia*, and the radius equals the distance in which the queried resources are valid resources. For instance, if the queried information is "*Universities close to Eiffel Tower*", this filter can look for Universities with coordinates in a 10 kilometer radius of the main point. The radius itself can vary, so if no valid answer is provided for a first query request, another query can be posed with a wider area range. This means that, in our example, if no matches are found, the query can be posed again increasing the radius to 50 kilometers or more, repeating until the system finds an answer or concludes there are no close universities to *Eiffel Tower*.

This technique reduces the amount of candidate answers provided by delegating those filtering processes to the database. In the question "*Which universities are close to Eiffel Tower?*", given that the coordinates of the *Eiffel Tower* are, in degrees,

48.8582 in latitude and 2.2945 in longitude, and that 10 kilometers can be roughly approximated by $\frac{10km}{EarthRadius} = \frac{10km}{40000km} = 0.08983$, the full query is as follows:

```
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>

SELECT * WHERE {
    ?resource
        geo:lat ?lat ;
        geo:long ?long ;
        a dbo:University .
FILTER (
    ?long > 2.2945 - 0.08983 &&
    ?long < 2.2945 + 0.08983 &&
    ?lat > 48.8582 - 0.08983 &&
    ?lat < 48.8582 + 0.08983)
}
```

The previous steps are sufficient to filter questions asking which elements are within a close distance to another, such as "*Which universities are close to Eiffel Tower?*". However, for questions demanding an element in a specific position relative to others, additional processing must be made. For instance, in the running example "*Which university is closest to Eiffel Tower?*", the Universities found by the last query must be ordered based on the distance to the target coordinates. Finally, the one with the shortest distance is selected as the final answer of the question, which is then provided to the client.

## 3.3 Implementation Aspects

The *TEQUILA* Spatial Question Answering system was built in Java 8, on top of the original *TEQUILA* application. We turn public the source code of our extended version of *TEQUILA*[1]. Our version provides built-in access to *DBPedia*, which did not exist in *TEQUILA*'s original code. The implementation was aided by the Stanford Natural Language processing libraries [11] for document annotation by including the Tasks of Named Entity Recognition and Event tagging, in addition to the Tree Tagger for tagging the various syntactic elements of sentences [12].

One important aspect of the implementation process was the focus on modularization, aiming to provide a reusable, scalable and extensible set of tools. This was possible due to the architecture based on processes. This allowed the creation of

[1] https://gitlab.ic.unicamp .br/jreis/question-answering/

new processes and improvements/extensions of original ones, followed by an easy, straight-forward injection of those, generating the new pipeline.

In this sense, components such as the *SPARQL* Query Building for *DBPedia* can easily be extended to provide answers for more types of questions, and new components can be added/removed by extending the *CNodeProcessorBase* abstract class. This processor class provides the stacking/unstacking ability for the processes. These processes occur sequentially, receiving the context from the previous one as a parameter. For instance, one process builds a different representation for a sentence and stores it in a context variable, which is then passed to the next process. That way, this next process has access to the work from the previous processes and can build something new from the context given.

# 4 Discussion

Answering complex natural language questions based on knowledge graphs remains an open research challenge. Although temporal question answering systems have been proposed, there is a lack of systems designed to be flexible in the types of complex questions they can handle. This work created a module in *TEQUILA* for Question Answering with spatial constraints, effectively extending the range of a temporal Question Answering system to handle spatial questions. To this end, new query building and filtering aspects were implemented. This study showed the application of temporal Question Answering concepts to the context of spatial question answering, with the aid of spatial Question Answering techniques and geospatial information from Linked Open Data datasets.

Our study advanced the way of building SPARQL Query relying on DBPedia and the in-query filtering for addressing spatial constraints. Open challenges and next steps involve incorporating new types of questions to be handled by the system, and improving the current *TEQUILA* functionality, such as parsing and classifying questions. This will be simplified by the modular nature of *TEQUILA*.

One interesting new functionality would be integrating the spatial and temporal modules in the same question. Currently, the system splits the question into either temporal or spatial, having the next modules of the pipeline after the classification behaving independently. One possible course of action is to use the existing sub-question querying and filtering data processes in the same pipeline, for the same question. However, there are a few problems that this integration would face, for example: questions that have both complex spatial and temporal constraints are rare and not easy to create; current databases should support the kind of temporally-enabled geo-information required for these questions' particular types, which limits the functionality of the system.

Other open challenges involve multilingualism for allowing flexibility in both

queries and knowledge bases in terms of language. Further studies may involve the use of distributed knowledge for accessing information spread in multiple, possibly unconnected knowledge bases.

# 5 Conclusion

Question Answering refers to a rich field of study that presents a lot of research opportunities and challenges. This study addressed how to handle complex natural language questions concerning aspects of spatial and temporal constraints. Our goal was the development of a spatio-temporal question answering system over RDF knowledge graphs. To this end, we extended the temporal Question Answering system *TEQUILA* tacking the open challenge of spatial question handling. Our work obtained a executable system suited to answer complex natural language questions that include spatial constraints. The system relies on processing sub-questions and filtering results from DBPedia. Future work involves increasing the range of questions supported by the system, adding multilingualism support and integrating complex spatial and temporal constraints.

# References

[1] Dbpedia (https://wiki.dbpedia.org/), accessed in 06/12/2018.

[2] Freebase (https://en.wikipedia.org/wiki/freebase), accessed in 18/06/2018.

[3] General administrative divisions dataset (https://gadm.org/about.html), accessed in 06/12/2018.

[4] Lc-quad organization (http://lc-quad.sda.tech/), accessed in 06/12/2018.

[5] Linkedgeodata (http://linkedgeodata.org/), accessed in 06/12/2018.

[6] Rdf - w3 (https://www.w3.org/rdf/), accessed in 06/12/2018.

[7] Rdf schema (https://en.wikipedia.org/wiki/resource_description_framework /media/file:rdf_graph_for_eric_miller.png), accessed in 06/12/2018.

[8] Semantic web - w3 (https://www.w3.org/standards/semanticweb/), accessed in 06/12/2018.

[9] Sparql - w3 (https://www.w3.org/tr/rdf-sparql-query/), accessed in 06/12/2018.

[10] Stanford named entity recognition (https://nlp.stanford.edu/software/crf-ner.html), accessed in 18/06/2019.

[11] Stanford part-of-speech tagger (https://nlp.stanford.edu/software/tagger.shtml), accessed in 18/06/2019.

[12] Treetagger web site (https://www.cis.uni-muenchen.de/ schmid/tools/treetagger/), accessed in 20/06/2019.

[13] Mohamed Yahya Gerhard Weikum Abdalghani Abujabal, Rishiraj Saha Roy. Quint: Interpretable question answering over knowledge bases. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, page 61–66, 2017.

[14] Jens Lehmann Axel-Cyrille Ngonga Ngomo Daniel Gerber Philipp Cimiano Christina Unger, Lorenz Bühmann. Template-based question answering over rdf data. *Proceedings of the 21st international conference on World Wide Web*, pages 639–648, 2012.

[15] A. Both M. Koubarakis I. Angelidis K. Bereta T. Beris D. Bilidas T. Ioannidis N. Karalis C. Lange D. Pantazi C. Papaloukas G. Stamoulis D. Punjani, K. Singh. Template-based question answering over linked geospatial data. *Proceedings of the 12th Workshop on Geographic Information Retrieval*, page Article number 7, 2018.

[16] J.L. Vicedo E. Saquete, P. Martínez-Barco. Splitting complex temporal questions for question answering systems. *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page article number 566, 2004.

[17] Elmar Haussmann Hannah Bast. More accurate question answering on freebase. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, pages 1431–1440, 2015.

[18] Michael Gertz Jannik Strotgen. Heideltime: High quality rule-based extraction and normalization of temporal expressions. *SemEval '10 Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 321–324, 2010.

[19] Edgard Marx Ricardo Usbeck Jens Lehmann Axel-Cyrille Ngonga Ngomo Konrad Höffner, Sebastian Walter. Survey on challenges of question answering in the semantic web. *Semantic Web*, 8:895–920, 2017.

[20] Andreas Both Saeedeh Shekarpour Ioanna Lytra-Ricardo Usbeck Akhilesh Vyas Akmal Khikmatullaev Dharmen Punjani Christoph Lange Maria Esther Vidal Jens Lehmann Sören Auer Kuldeep Singh, Arun Sethupat Radhakrishna. Why reinvent the wheel – let's build question answering systems together. *Proceedings of the 2018 World Wide Web Conference*, pages 1247–1256, 2018.

[21] Haixun Wang Jeffrey Xu Yu Wenqiang He-Dongyan Zhao Lei Zou, Ruizhe Huang. Natural language question answering over rdf — a graph data driven approach. *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, pages 313–324, 2014.

[22] Amit P. Sheth Kunal Verma Prateek Jain, Pascal Hitzler and Peter Z. Yeh. Ontology alignment for linked open data. In *International Semantic Web Conference (ISWC)*, pages 402–417, 2010.

[23] Mohnish Dubey Jens Lehmann Priyansh Trivedi, Gaurav Maheshwari. Lc-quad: A corpus for complex question answering over knowledge graphs. In *Proceedings of the International Semantic Web Conference (ISWC)*, pages 210–218, 2017.

[24] Dave Kolas Robert Battle. Enabling the geospatial semantic web with parliament and geosparql. *Semantic Web*, 3:355–370, 2012.

[25] Rishiraj Saha Roy Jannik Strotgen Gerhard Weikumo Zhen Jia, Abdalghani Abujabal. Tempquestions: A benchmark for temporal question answering. *Companion Proceedings of the The Web Conference 2018*, pages 1057–1062, 2018.

[26] Rishiraj Saha Roy Jannik Strotgen Gerhard Weikumo Zhen Jia, Abdalghani Abujabal. Tequila: Temporal question answering over knowledge bases. *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1807–1810, 2018.