Learning Multilingual Semantic Parsers for Question Answering over Linked Data

A comparison of neural and probabilistic graphical model architectures

Sherzod Hakimov

Bielefeld University

Supervisor: Prof. Dr. Philipp Cimiano

Submitted in partial fulfillment of the requirements for the degree of
Doctor rerum naturalium (Dr. rer. nat.)

July 2018

Reviewer:
Prof. Dr. Philipp Cimiano
Prof. Dr. Axel-Cyrille Ngonga Ngomo

Abstract

The task of answering natural language questions over structured data has received wide interest in recent years. Structured data in the form of knowledge bases has been available for public usage with coverage on multiple domains. DBpedia and Freebase are such knowledge bases that include encyclopedic data about multiple domains. However, querying such knowledge bases requires an understanding of a query language and the underlying ontology, which requires domain expertise. Querying structured data via question answering systems that understand natural language has gained popularity to bridge the gap between the data and the end user.

In order to understand a natural language question, a question answering system needs to map the question into query representation that can be evaluated given a knowledge base. An important aspect that we focus in this thesis is the multilinguality. While most research focused on building monolingual solutions, mainly English, this thesis focuses on building multilingual question answering systems. The main challenge for processing language input is interpreting the meaning of questions in multiple languages.

In this thesis, we present three different semantic parsing approaches that learn models to map questions into meaning representations, into a query in particular, in a supervised fashion. Each approach differs in the way the model is learned, the features of the model, the way of representing the meaning and how the meaning of questions is composed. The first approach learns a joint probabilistic model for syntax and semantics simultaneously from the labeled data. The second method learns a factorized probabilistic graphical model that builds on a dependency parse of the input question and predicts the meaning representation that is converted into a query. The last approach presents a number of different neural architectures that tackle the task of question answering in end-to-end fashion. We evaluate each approach using publicly available datasets and compare them with state-of-the-art QA systems.

Gedruckt auf alterungsbeständigem Papier nach DIN-ISO 9706
Printed on non-aging paper according to DIN-ISO 9706

Acknowledgements

The work presented in this thesis was carried out in the Semantic Computing Group, headed by Prof. Dr. Philipp Cimiano, at the Faculty of Technology, Bielefeld University. This work was supported by the Cluster of Excellence Cognitive Interaction Technology CITEC (EXC 277) at Bielefeld University, which is funded by the German Research Foundation (DFG).

First of all, I am grateful to Philipp Cimiano. He gave me the opportunity to work with him on this thesis and he provided his guidance and experiences for me reaching this point in my academic career. I would also like to thank Axel-Cyrille Ngonga Ngomo for reviewing this thesis.

I would like to express my gratitude to all current and past members of the Semantic Computing Group. They helped me in getting familiar with the life in the city, the academic life, the language and with my PhD work. In particular, I am grateful to Hendrik ter Horst, Soufian Jebbara, Maximilian Panzner, Sebastian Walter, Christina Unger, Basil Ell, Pejman Sajjadi, Roman Klinger, John McCrae and Matthias Hartung with whom I worked on multiple projects or exchanged ideas.

My special thanks go to my colleagues: Christina Unger, Sebastian Walter, Basil Ell and Matthias Hartung for reviewing my thesis and sharing their experiences.

I would also like to thank my girlfriend Stefania for providing her love and support throughout my PhD studies and also helping me with the review of this thesis.

Finally, my biggest gratitude goes to my parents Talip and Dilber, my sister Shahnoza and my brothers Ismoil and Isroil. They made it possible for me to reach this important point in my life. They have always supported me, believed in me throughout the years.

Contents

List of Tables

List of Figures

List of Abbreviations

BiLSTM	Bidirectional Long Short-term Memory
BiRNN	Bidirectional Recurrent Neural Network
CCG	Combinatory Categorial Grammar
CG	Categorial Grammar
CRF	Conditional Random Fields
CNN	Convolutional Neural Network
DG	Dependency Grammar
DUDES	Dependency-based Underspecified Discourse Representation Structures
EL	Entity Linking
LD	Linked Data
LOD	Linked Open Data
LSTM	Long Short-term Memory
NED	Named Entity Disambiguation
NEL	Named Entity Linker
NER	Named Entity Recognizer
NLI	Natural Language Interface
NLP	Natural Language Processing
NLU	Natural Language Understanding
NN	Neural Network
POS	Part-Of-Speech
RNN	Recurrent Neural Network
QA	Question Answering
QALD	Question Answering over Linked Data
UD	Universal Dependencies
URI	Uniform Resource Identifier

Chapter 1

Introduction

In this chapter, we provide content about the addressed challenges and introduction into the field of semantic parsing approaches for question answering. We also present the research questions along with the contributions of this thesis.

1.1 Natural Language Interfaces

Natural language interfaces (NLI) are type of user interfaces where the manipulation of components is done based on a language input. The popularity of such interfaces increased because of smart assistants available in mobile operating systems. Apple's Siri, Google Now and Microsoft Cortana can perform various tasks where a user types the command in natural language or sends an audio input. Voice-enabled devices such as Amazon Alexa, Google Home, Apple HomePod enable the interaction based on voice input. This type of user interaction is preferred for its speed and convenience for people from various backgrounds or age groups in comparison to traditional user interfaces with different UI components e.g. buttons etc. Users can interact with them as if "chatting" with a friend, sending audio recording or using gestures.

Humans are able to interpret a written text by taking into account the domain knowledge, the context of the sentence, the grammatical structure, pragmatics and the personal understanding of the domain. NLI that interpret written input need to incorporate syntax, semantics and inference mechanism in order to imitate the human understanding of language. Natural Language Understanding (NLU) is a subfield of Natural Language Processing (NLP) that focuses on understanding the meaning of a written text and act accordingly. NLI are able to apply some sort of NLU models to interpret the input and respond accordingly.

NLI systems have been a focus for building human-computer interaction tools. NLU models enable such interaction on a written language level. NLU models that use semantic parsers enable easier ways of interaction where users can provide the input in free form e.g. *natural language*. There have been many systems developed over the years such as ELIZA (Weizenbaum, 1966) for the purpose of aiding people with psychological issues. Users could engage with ELIZA to talk about their problems and the system imitates a psychotherapist. SHRDLU (Winograd, 1971) could move objects in a "blocks world" using a robotic arm where the performed action was triggered by a natural language command. Even though these systems performed well at the time, they had limited understanding of the language where they try to imitate an expert in a certain domain. The underlying vocabulary is restricted and the adaptation of such systems to another domain is not straightforward. More advanced systems have been developed over the years. For example, IBM developed Watson as an assistant that interprets the spoken or written language. The system was tailored for factoid questions and won the *Jeopardy!*[1] quiz show in 2011 against two human champions.

[1] http://www.jeopardy.com/

Increasing popularity of messaging applications such as WhatsApp, Facebook Messenger, LINE, WeChat etc. opened new use cases for building NLI systems on one of such messaging platforms. Commercial applications of NLI systems by means of messaging evolved into building conversational agents. Conversational agents are systems that conduct a conversation by means of text or speech where serve as a new interface for consuming services or provide information. These interfaces are sometimes called *chatbots*. Messaging applications have developed frameworks where developers can build and host their chatbots. For instance, Facebook has developed such platform and started hosting chatbots since 2015. There were estimated to be 100.000 chatbots deployed on Facebook in 2017.[2] These chatbots provide services for various domains in many languages. For instance, chatbots started to answer questions posted on a regular channels posted by customers on various topics. In cases where the questions are complicated, the chatbots can redirect cases to regular human agents, e.g. call-center employees. Such use cases could potentially reduce the workload on human agents where chatbots are trained to perform simple and repetitive tasks. Some of these chatbots use a scripted approach to handle user input. Scripted approach here means to predefine user inputs and respond only to those that are covered. Some chatbots use NLU methods to interpret the intent of user messages, which increases the coverage by not relying on scripted inputs only.

WeChat, the messaging application from the Chinese company Tencent, has more than 500 million users. This company provides a platform for developers to build dialog systems that offer various services such as ordering a taxi, paying for a restaurant bill, shopping, booking a flight ticket and many others. Xiaoice[3] is a chatbot developed by Microsoft that engages with users via chatting in Chinese language. It became very popular among young people because it can converse about various topics. Specifically users like it for chit-chatting purposes.[4]

Next, we explain how NLU methods can be applied to build factoid question answering systems.

1.2 Semantic Parsing

Most NLI systems have a common way of operating: they take an input from a user and map it to their internal meaning representation of knowledge, execute it and return the result. Semantic parsing addresses the method for capturing relevant parts of the linguistic input into a structured, machine-readable meaning representation. This representation captures the meaning of the given input with respect to the domain knowledge. Depending on the use case it can be an executable query, a nested object, a structured document in XML, JSON formats, etc.

There are various ways of representing meaning. One common way of representation is using lambda calculus (explained in more detail in Section 2.3.3). It is a formalism in mathematical logic for expressing computation based on functional abstraction and application. Such a formalism is language independent and can be adapted to other domains. Any selected domain needs to define functions, constants, variables, quantifiers and connectors.

For example, consider some personal assistant that can set appointments or notifications in a smart phone for a user. The following natural language input “Set an alarm for 7 in the morning” must be interpreted by the system in order to execute it. The meaning representation for this can be defined using a function called “set_alarm(time)” that takes a single

[2] https://www.theregister.co.uk/2017/04/19/chatbots_facebook/

[3] http://www.msxiaoice.com/

[4] https://techxplore.com/news/2018-04-xiaoice-chatbot-chat-human-sounding.html

argument as an input and creates an event in the user's calendar with the provided time, such that the following meaning representation can be obtained:

1. set_alarm(07:00)

In the provided example above, the hours of the day are predefined set of constants and the expression "set_alarm(time)" is a function. NLI systems can use a semantic parsing component to convert the input into a meaning representation of the respective domain. Of course it is only useful if there is some other component that makes use of the meaning representation. This component, sometimes called "executor", takes the meaning representation and executes it. In the case of setting an alarm, the executor runs the command "set_alarm(07:00)" that leads into the user's calendar having a new entry for alarms.

Another possible example for semantic parsing can be the natural language question "What is Michael Jordan's height?". Let's say the question can be answered using Wikipedia data and we have access to the database with factual data. The executor needs a query to retrieve the answer from that database. A possible style of the meaning representation can be as follows:

2. λx. height (Michael_Jordan, x)

The given expression is in lambda calculus format where "height" represents a function with a defined constant "Michael_Jordan". The function "height" takes two arguments, a defined constant such as "Michael_Jordan" and a numerical value. In the given meaning representation, the numerical value is missing and represented by the variable "x". The executor can take as input such meaning representation, convert it into any executable query-like language and retrieve results from a database, e.g. "x" equals to "1.98 meters".

Early NLI approaches tried to build semantic parsing components either with defined patterns or using syntactic and semantic analyzers. Winograd (1971, 1972) developed one of the earliest system that understands English sentences, answers questions and executes commands via dialog with a user. The application domain was interacting with a system that could manipulate different geometric shapes like moving objects, asking questions about certain conditions that occur within the manipulation. The user interacts with the system via dialogs. The approach used syntactic and semantic interpreters.

Another approach called LUNAR (Woods et al., 1972) had a semantic parsing unit that was based on rules. The system could answer questions about rocks that were collected during the Apollo 11 mission for non-experts that needed to query the database. However, writing such rules manually does not scale to other domains or languages as it requires a domain expert to define the input patterns.

The coverage of the system would also be limited by the rules. Following the advancements in NLP, semantic parsing approaches based on statistical models evolved. Instead of writing rules manually, systems are trained with sample data and learn a statistical model.

Tang and Mooney (2001) defined the first semantic parsing approach that was based on a statistical model. They proposed to use a shift-reduce parser together with Logic Programming. They created a training data based on a database of geographical knowledge of the U.S., consisting of a question and a meaning representation pair. The dataset is called Geoquery. Meaning representation is expressed using lambda calculus. We give detailed explanation on meaning representations in Section 2.3. An example is given in Figure 1.1 with a question and lambda calculus pair.

Question: *Which states border Texas?*

Lambda calculus expression: $\lambda x.\texttt{state}(x) \wedge \texttt{next\_to}(x, \texttt{state:Texas})$

FIGURE 1.1: An instance from Geoquery dataset.

Later on Zettlemoyer and Collins (2005) defined a method for learning a statistical semantic parser using CCG (Steedman, 2000) for syntax and lambda calculus for semantics. They evaluated their approach on the Geoquery dataset. They showed that syntax and semantics mapping can be learned together. Later on, another semantic parsing approach (Berant et al., 2013) focused on training semantic parsers from question and answer pairs. Labeling questions with meaning representations is a harder problem than pairing them with expected answers. The availability of large knowledge bases such as DBpedia (Auer et al., 2007), Wikidata (Vrandečić and Krötzsch, 2014), YAGO (Suchanek et al., 2007) and Freebase (Bollacker et al., 2008) made it possible to label training data easier. Instead of writing a query for a question, users match each question with answers from knowledge bases.

1.3 Natural Language Question Answering System

In this thesis, we focus on building different semantic parsing approaches for Question Answering (QA) using the syntax of natural language questions and RDF data from knowledge bases. The goal of a semantic parsing system is to map natural language questions to meaning representations that are composed of knowledge base entries. Meaning representations can be transformed into executable queries where the answer can be found.

We focus on building QA systems that enable to query RDF data in particular DBpedia (Auer et al., 2007) and Freebase (Bollacker et al., 2008) using natural language expressions. Both knowledge bases are available in RDF format and can be queried using SPARQL (see Section 2.1.3). However, querying the data requires some understanding of the query language and the underlying ontology. For this purpose, we build a QA system that users can use to query the data using natural language in multiple languages without having to deal with the ontology and query language specifications. In this thesis, we focus on building and evaluating QA systems on QALD (Cimiano et al., 2013; Unger et al., 2014, 2015, 2016) and SimpleQuestions (Bordes et al., 2015) datasets (see Chapter 3 for more details on datasets).

Consider the following questions shown in Figure 1.2, asking the same information in English, German and Spanish respectively:

1. How tall is Amazon Eve?
2. Wie groß ist Amazon Eve?
3. ¿Cómo de alta es Amazon Eve?

FIGURE 1.2: Questions in English, German and Spanish asking the height of Amazon Eve

All questions refer to the same fact in different languages. The answer for these questions can be found in DBpedia by executing the following SPARQL query.

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>
SELECT DISTINCT ?o
WHERE {
      dbr:Amazon_Eve dbo:height ?o.
}
```

FIGURE 1.3: An example SPARQL query from DBpedia.

The query contains a single statement also called an RDF *triple*. RDF triples are based on the idea of making statements about resources in expressions of the form subject-predicate-object. The subject denotes the resource, the object denotes another resource or aspects of the subject and the predicate expresses a relationship between the subject and the object. It can be seen that the resource *dbr:Amazon_Eve* is the subject and *dbo:height* is the predicate of the triple. The object is expressed as a variable *?o* where it stands for the answer to the questions above.

1.3.1 Task Definition

Considering the questions above in Figure 1.2, the task of a QA system is to construct the query in Figure 1.3 for the supported languages. The system should map expressions in the question to the URIs in the expected query. QA system needs to incorporate external resources to make such mappings since the underlying knowledge base may not contain the needed data for mapping. The task of Questions Answering over Linked Data can be defined as:

Task: Question Answering over Linked Data
Return an executable query that answers the natural questions given an RDF knowledge base.

1.4 Motivation

DBpedia (Auer et al., 2007), YAGO (Suchanek et al., 2007), Wikidata (Vrandečić and Krötzsch, 2014) and Freebase (Bollacker et al., 2008) are open-access knowledge bases that contain information about many domains. Each knowledge base contains RDF data collected from various sources with an underlying ontology definition.

DBpedia has become a central hub in Linked Data (LD)[5] containing structured encyclopedic data extracted from Wikipedia over the last 10 years. DBpedia has an ontology that defines more than 1000 properties, 800 classes, etc., with labels in many languages. The latest release of DBpedia provides data in RDF format for more than 18 million entities in 127 languages (see Section 2.1.5). DBpedia is being updated in a yearly fashion by adding more data and improving data quality. The data is open-access with availability to download and accessible via endpoint for querying.[6] This makes us consider DBpedia as one of the main source of knowledge for building a multilingual QA system since it supports multiple languages. Freebase was also considered as another large knowledge base with public access until it was closed in 2015[7] where data has been merged with Wikidata (Vrandečić and Krötzsch, 2014).

[5] http://linkeddata.org/
[6] http://dbpedia.org/sparql
[7] https://developers.google.com/freebase/

With a strong focus on *linking* RDF data initiative proposed by Linked Data[8] project, these knowledge bases also include interconnected links among each other making a huge graph of knowledge about real-world entities.

Making such knowledge base systems open access does not necessarily mean that users can use them easily. Even though SPARQL is the only query language, it is still difficult for every user to learn the query language. Moreover, each knowledge base has its own ontology definition and users need to get familiar with the data and ontology in order to build queries. In thesis, we provide the solutions for building Question Answering (QA) systems so that users can query the knowledge bases using natural language without any knowledge of the query language and the ontology.

Consider the questions in Figure 1.4 in English, German and Spanish and the expected SPARQL query (in Figure 1.5) to answer those questions.

1. Who created Wikipedia?
2. Wer hat Wikipedia gegründet?
3. ¿Quién creó Wikipedia?

FIGURE 1.4: Questions in English, German and Spanish asking about the creator of Wikipedia

All questions refer to the same fact in different languages. The answer for these questions can be found in DBpedia by executing the following SPARQL query.

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>
SELECT DISTINCT ?o
WHERE {
      dbr:Wikipedia dbo:author ?o .
}
```

FIGURE 1.5: An example SPARQL query from DBpedia that returns the creator of Wikipedia.

A multilingual QA system should map the given questions in different languages into a meaning representation that can be transformed into a SPARQL query. However, the mapping from question to query is not straightforward. In the English question, the word *created* should be mapped to DBpedia Ontology property *dbo:author* as shown in Figure 1.5. However, the DBpedia Ontology does not contain such lexicon that allows for mapping the word *created* into the property *dbo:author*. The labels for the property *dbo:author* that the ontology contains are shown below, the tags *en, de, es* represent the languages English, German and Spanish respectively.

```
"author"@en
"autor"@de
"autor"@es
```

In order to answer the English question, the QA system needs to map the verb "created" to the property *dbo:author*. It applies to German and Spanish questions as well where the

[8] http://linkeddata.org

words *gegründet* and *creó* should be mapped to the same property. The missing lexicon is known as the "lexical gap" challenge for a QA system to overcome to be able to map natural language phrases to knowledge base entries.

Another challenge here lies in the structure of the knowledge base. The challenge is to consider the positions of resources in triples. The property *dbo:author* defines a relation between entities of class type *dbo:Work* and *dbo:Person*. The resource *dbr:Wikipedia* has a class type *dbo:Work*. The position of the resource *dbr:Wikipedia* has to be in the subject of the triple. These types of relations and ontology restrictions are shown below with RDF triples from the DBpedia.

```
dbo:author rdfs:domain dbo:Work .
dbo:author rdfs:range dbo:Person .

dbr:Wikipedia rdf:type dbo:Work.
```

Let's consider the following questions in three languages given in Figure 1.6 and the QA system expected to construct the query given in Figure 1.7.

1. Give me all movies with Tom Cruise.
2. Gib mir alle Filme mit Tom Cruise.
3. Dame todas las películas con Tom Cruise.

FIGURE 1.6: Questions in English, German and Spanish asking the movies with Tom Cruise

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT DISTINCT ?uri
WHERE {
      ?uri rdf:type dbo:Film.
      ?uri dbo:starring dbr:Tom_Cruise.
}
```

FIGURE 1.7: An example SPARQL query from DBpedia that returns the list of movies with Tom Cruise.

By looking at the English question and the query, we can observe that the word *movies* should be mapped to *?uri rdf:type dbo:Film*. For German and Spanish questions, the words that express the *rdf:type dbo:Film* part of the query are *Filme* and *películas* respectively. These words are considered to be direct translations of the English word "film". The system needs a domain-specific lexicon to perform the matching of keywords to the expected URIs, which is possible using external dictionaries with synonyms, e.g. WordNet (Miller, 1995a).

However, the property *dbo:starring* can not be expressed with any word in the given questions. The preposition *with* can have different meanings depending on the context. It is another challenge for a QA system to infer the property without any explicit words in questions to map. The same challenge exists for German and Spanish questions given above with prepositions *mit* for German and *con* in Spanish. QA system needs to cope with under-specified properties in respect to the given question where the URI that can not be directly retrieved from question text but must be inferred based on the context.

As shown above, QA systems need a mechanism to map keywords in questions to URIs in queries. Especially, mapping properties in queries poses a bigger challenge than mapping entities (*dbr:Tom_Cruise, dbr:Wikipedia*) or classes (*dbo:Film*). It is caused by the variability in natural language for expressing relations between entities. For example, *starred in, appeared, co-starred, co-star, leading role, played, acted* all refer to one property in DBpedia *dbo:starring*. Welty et al. (2010) mentioned over 50 variations of natural language expressions to express the relation between an actor and a movie.

A QA system could use indexes for different types of URIs for retrieval as well as a prepocessor step that includes Named Entity Recognizer (NER) and then Named Entity Linker (NEL). Even retrieving the expected URIs does not imply a successful interpretation of the question. A QA system still needs to put those URIs in the right place in order to construct a valid query. A wrong placement of URIs in the query does not return the same answers as expected.

Consider the questions in Figure 1.8, the questions are about founding members of the company Intel as expressed in the query given in Figure 1.9. The answers returned from DBpedia are: Robert Noyce, Andrew Grove, Gordon Moore. Swapping the position of *dbr:Intel* to the object position of the triple leads to the query given in Figure 1.10. This query returns companies that Intel founded: Dossia, Trusted Computing Group, which would a wrong interpretation for the given questions. By considering the syntax of sentences, a QA system should deal with the specifications of the knowledge base, as it leads to a different query than expected. The syntax of the question above can determine the correct slot for the resource *dbr:Intel*.

1. Who founded Intel?
2. Wer hat Intel gegründet?
3. ¿Quién fundó Intel?

FIGURE 1.8: Questions in English, German and Spanish asking the founder of Intel

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT DISTINCT ?uri
WHERE {
      dbr:Intel dbo:foundedBy ?uri.
}
```

FIGURE 1.9: SPARQL query for founding members of Intel from DBpedia.

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT DISTINCT ?uri
WHERE {
      ?uri dbo:foundedBy dbr:Intel.
}
```

FIGURE 1.10: SPARQL query for companies that Intel founded from DBpedia.

The ambiguity of natural language expressions can be considered as another challenge since the mapping phase will generate multiple candidates. For instance, the word *created* can map to the property *dbo:creator* or *dbo:author*. We are assuming that the *lexical gap* is handled using an external resource and that we can map the word *created* to the given properties. QA system still needs to prefer one property over another. Additionally, personal names are also quite ambiguous. A QA system needs a mechanism to disambiguate where the mapping results in multiple matches.

We can summarize the challenges explained above in four main categories:

Challenge 1: Mapping natural language phrases to knowledge base entries. (Questions in Figure 1.4 and SPARQL query in Figure 1.5)

Challenge 2: Inferring knowledge base entries based on the context where words for direct mapping in the question are missing. (Questions in Figure 1.6 and SPARQL query in Figure 1.7)

Challenge 3: Considering the structure of the knowledge base and the syntax of sentences. (Questions in Figure 1.8 and SPARQL query in Figure 1.9)

Challenge 4: Handling the ambiguity in mapping natural language entries to knowledge base entries where words map to multiple candidates.

Our motivation in this thesis is to combine a syntactic and a semantic analyzer for the task of building a QA system. In particular build and evaluate different semantic parsing approaches that use syntactic information of a sentence in combination with semantics of underlying RDF data. The approaches that focus on developing solutions for the challenges described above have a strong focus on a multilingual QA system. The syntax tells us how sentences are constructed grammatically and the semantics tells us how the meaning of sentences can be constructed from smaller units in sentences.

In this thesis, we focus on training supervised models that learn to incorporate syntax together with semantics. The models learn to map to translate the structure of the syntax to the structure of the semantics.

Recent efforts in developing cross-lingual treebank resulted in Universal Dependencies (UD) (Nivre et al., 2016; Nivre, 2017). UD aims to capture syntactic similarities between languages for developing multilingual NLP approaches. UD has defined a universal set of morphological and syntactic specifications e.g. POS Tags, dependency relations with current version including 70 treebanks for 50 languages.[9] A more detailed description about the syntax of sentences is given in Section 2.2.2. In Figure 1.11, the dependency parse trees are given for questions in Figure 1.4 based on Universal Dependencies syntactic specifications.

[9] http://universaldependencies.org/v2, 70 treebanks, 50 languages

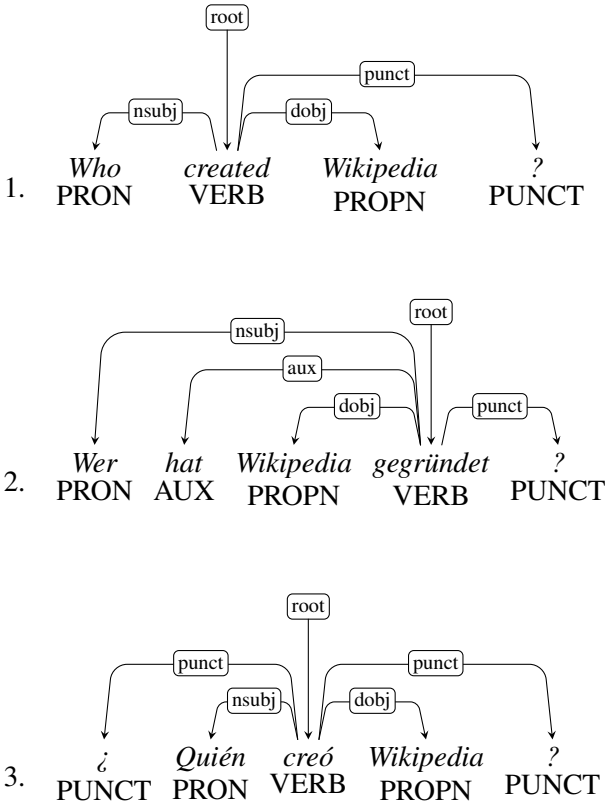


FIGURE 1.11: Dependency parse trees with part-of-speech (POS) tags for questions in English, German and Spanish languages based on Universal Dependencies syntactic specifications.

A dependency parse tree differs from a constituency parse tree in a way that the edges that connect nodes are specified. The dependency relation *dobj* can be seen between the *root* nodes and the node *Wikipedia* in all 3 languages, indicating that the node *Wikipedia* is the *direct object* of the main verb. Additionally, pronouns *Who, Wer, Quién* have the *nsubj* relation with the *root* nodes, which tells us that the node is the *nominal subject* of the main verb. These dependency relations are better suited for the task of QA on a structured knowledge base because the triples in DBpedia are constructed in a similar way. Each triple has a predicate (it is expressed by verbs in sentences), subject and object. The SPARQL query that answers the questions above is given in Figure 1.12. The structure of the query and the dependency relations between nodes in the dependency parse trees show significant resemblance. The root node *created* in English question is represented the predicate *dbo:author*. The node *Wikipedia* is connected to the main verb with a dependency relation similar to the resource *dbr:Wikipedia* being on the subject position of the triple with the predicate *dbo:author*. Note here that the resource *dbr:Wikipedia* is on the subject position while the dependency relation between the node *Wikipedia* and the parent node *created* indicates a *dobj* (direct object) relation. Relying on the dependency relation position for the node *Wikipedia* would result in the resource *dbr:Wikipedia* being inserted into object position of the triple as the dependency parse tree suggests an object relation between its parent node *created*, which would be an incorrect query. A QA system still needs to learn when to take the dependency relation into account and what sort of information from it in order to build the expected query.

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>
SELECT DISTINCT ?o
WHERE {
      dbr:Wikipedia dbo:author ?o .
}
```

FIGURE 1.12: An example SPARQL query from DBpedia that returns the creator of Wikipedia.

A QA system can be trained to learn the mapping between the underlying dependency parse tree and the structure of knowledge represented in triples. The main contribution of using dependency parse trees based on UD is that it provides a universal set of syntactic specifications for many languages and the learning mechanism can be applied to any supported language. Additionally, the learned models can also be transferred into another language since the same syntax is used for all languages in UD. As shown in Figure 2.11, the dependency relations between the verb (the root node) and the node *Wikipedia* are the same as well as the dependency relation between pronouns and the main verb. The POS tags are also the same for those nodes, e.g. VERB for nodes *created*, *gegründet* and *creó*.

Open-domain knowledge bases such as DBpedia contain language-independent content with labels in many languages. Syntactic analysis tools such as dependency parse tree generators provide syntax of a given question. Considering the language-independent structured data together with syntactic information of multiple languages motivates us to build multilingual semantic parsing systems for the application of QA systems while most of previously published work focused on a solution for a single language, mainly English.

1.5 Research Questions

In this thesis, we focus on building QA systems that generalize better and easily adaptable to other languages without changing the architecture and provide means of incorporating linguistic data. The solution proposed for this purpose should handle the challenges described above in terms of lexical ambiguity, knowledge base structure and language-specific linguistic data retrieval.

This leads to the following research questions:

Question 1: How to map natural language phrases into knowledge base entries for multiple languages? Which linguistic resources can be used?

Question 2: How to disambiguate URIs when multiple candidates are retrieved from mapping natural language tokens into knowledge base entries?

Question 3: How to use syntactic information of a natural language question together with semantic representations of entries in a knowledge base?

Question 4: What are the advantages and the disadvantages of a multilingual QA system vs. a monolingual system built for each language?

Question 5: What effort is required to adapt our QA pipelines to another language?

1.6 Contributions

The contributions of this thesis can be grouped under the three main chapters. These chapters define the proposed approaches in them with contributions and answers to the research questions raised above:

1. CCG-based Semantic Parsing Approach (Chapter 6)
2. Dependency parse tree-based Semantic Parsing Approach (Chapter 7)
3. Neural Network-based Semantic Parsing Approach (Chapter 8)

The CCG-based Semantic Parsing Approach is based on the CCG formalism for syntax and lambda calculus for semantics (see Section 2.2.1). It is an implementation of the semantic parsing method proposed by Zettlemoyer and Collins (2005) for the Geoquery (Tang and Mooney, 2001) dataset. We adapted it into the QALD dataset and added additional features. The approach learns a probabilistic model for syntax and semantics simultaneously from the labeled data. The approach was adapted only to English since each language needs a definition of CCG rules separately. We use the M-ATOLL (Walter et al., 2014) ontology lexicalisations as an external source for mapping natural language phrases into knowledge base entries, specifically properties. The M-ATOLL provides ontology lexicalisations in English, German and Spanish languages. The main contribution of this method is to show the **lexical gap** that occurs when the tokens in questions can not be mapped directly to a knowledge base data and the solution for this problem can be use external linguistic resources such as ontology lexicalisations from M-ATOLL to bridge the gap.

The Dependency parse tree-based Semantic Parsing Approach uses Universal Pependency (UD) parse trees for syntax and learns the mapping from syntax in the parse trees into knowledge base entries. The approach is adapted to English, German and Spanish languages. We combine multiple external resources such as M-ATOLL, WordNet and dictionary computed from distributional semantics hypothesis using cosine similarity between words in natural language and labels of knowledge base properties. These external resources serve as a lexicon to map natural language phrases. We give more details on combined lexicon in Section 5.2. The main contribution of this approach is the **multilingual** architecture for building QA systems. Since it is based on UD dependency parse trees, the pipeline can be adapted to all languages supported by UD. Another important contribution of this approach is the solution for **mapping** multilingual natural language questions into knowledge base entries by combining multiple external resources.

The approach uses DUDES (see Section 2.3.3) for expressing semantics. It has a compositional feature (see Section 2.3.4) that can be used in tandem with the syntax (dependency parse tree). DUDES enable to create semantics where the URI is not specified but a composition of meaning representation is carried on without an explicit URI. It allows to infer URIs that can not be mapped from a question. It has advantages over monolingual pipelines in a way that the same pipeline can be used to train QA system regardless of a language. The approach can be easily extended to other languages.

Finally, the Neural Network-based Semantic Parsing Approach uses neural networks that learn mappings between natural language expressions to knowledge base entries using the latent embeddings of words and characters. The chapter in fact compares four different model architectures that are similar to recent state-of-the-art systems. The contribution of this chapter is to compare different model architectures under the same environment which allows to understand the pipelines better, compare them fairly under the same hood and highlight the strengths of each architecture. Throughout the thesis, we select BiLSTM-Softmax as an architecture to compare with the other approaches described in Chapter 6 and Chapter 7. The

pipeline uses only words and characters as features by embedding them into the Bidirectional LSTM (Graves et al., 2013) that learns contextual and syntactic dependencies between words in a sentence. The approach is not necessarily built for a single language and can be adapted to others. However, it is trained on the SimpleQuestions (Bordes et al., 2015) dataset that contains questions only in English.

The main contribution of this thesis is to present three different model architectures for building question answering systems and compare them. We focus specifically on the point of building a multilingual approach that can be extended easily to other languages and domains. We present a detailed evaluation and analysis of each proposed model architecture by highlighting strengths and weaknesses. Other contributions can also be listed as making the implemented approaches and the used linguistic data open for the research community.

Chapter 6: `https://github.com/ag-sc/CCGParsing`

Chapter 7: `https://github.com/ag-sc/AMUSE`

Chapter 8: `https://github.com/ag-sc/SimpleQA`

Published Work

The thesis is based on the following papers published earlier:

- Hakimov S, Unger C, Walter S, Cimiano P. (2015). Applying semantic parsing to question answering over linked data: Addressing the lexical gap. In *Proceedings of International Conference on Applications of Natural Language to Information Systems (NLDB)*
- Hakimov S, ter Horst H, Jebbara S, Hartung M, Cimiano P. (2016). Combining textual and graph-based features for named entity disambiguation using undirected probabilistic graphical models. In *Proceedings of 20th International Knowledge Engineering and Knowledge Management Conference (EKAW)*
- Ell B, Hakimov S, Cimiano P. (2016). Statistical Induction of Coupled Domain/Range Restrictions from RDF Knowledge Bases. In *Proceedings of 4th NLP and DBpedia Workshop, co-located with the 15th International Semantic Web Conference (ISWC)*
- Hakimov S, Jebbara S, Cimiano P. (2017). AMUSE: Multilingual Semantic Parsing for Question Answering over Linked Data. In *Proceedings of the 16th International Semantic Web Conference (ISWC)*
- Ell B, Hakimov S, Braukmann P, Cazzoli L, Kaupmann F, Mancino A, Altaf Memon J, Rother K, Saini A, Cimiano P. (2017). Towards a Large Corpus of Richly Annotated Web Tables for Knowledge Base Population. In *Proceedings of 5th International Workshop on Linked Data for Information Extraction, co-located with the 16th International Semantic Web Conference (ISWC)*
- Hakimov S, Jebbara S, Cimiano P. (2019). Evaluating Architectural Choices for Deep Learning Approaches for Question Answering over Knowledge Bases. In *Proceedings of the 13th International Semantic Computing Conference (ICSC)*

1.7 Chapter Structure

The remaining chapters are structured as follows:

- Chapter 2 presents the foundations the thesis is based on. It covers preliminary knowledge about the topics that are essential in understanding this work. It starts by describing the foundations of RDF & SPARQL along with publicly open knowledge bases. Next, it describes the notion of syntax and semantics used in this thesis along with application in building Question Answering systems.
- Chapter 3 presents publicly available datasets from the research community to evaluate Question Answering systems. We give detailed overview of each dataset along with comparisons.
- Chapter 4 describes previously published work on building Question Answering systems. We provide detailed overview on research done in semantic parsing along with comparisons to the state-of-the-systems evaluated on the QALD and the SimpleQuestions datasets.
- Chapter 5 presents methods for mapping natural language phrases into knowledge base entries by introducing an inverted index for retrieval of URIs. We combine several external resources and compare their performance.
- Chapter 6 presents a semantic parsing that uses Combinatory Categorial Grammars (CCG) for syntax and lambda calculus for semantics to build a QA system. The approach is applied to the QALD dataset on English. We give detailed information on the approach along with comparisons to other systems.
- Chapter 7 presents another semantic parsing approach that abstracts from the underlying language by using dependency parse trees from the Universal Dependencies project. The approach is based on building a multilingual pipeline for the English, German and Spanish languages. We compare our approach with other published systems and evaluate the performance.
- Chapter 8 presents four different semantic parsing approaches for building QA systems on the SimpleQuestions dataset that are evaluated under the same environment where each chosen architecture is based on previously published systems.
- Chapter 9 presents discussions on proposed approaches and highlights differences in terms of role of syntax & semantics, multilinguality, manual effort for adapting to another domain or language and effects of datasets on these approaches.
- Chapter 10 sums up the thesis with ideas for future work. We also provide answers for research questions addressed in the thesis.

Chapter 2

Preliminaries

In this chapter, we present the foundations this thesis is based on. It covers preliminary knowledge about the topics that are essential for understanding this work. It starts by describing the foundations of RDF & SPARQL along with publicly open knowledge bases. Next, it describes the notion of syntax, semantics and how the semantics of natural language questions is composed.

2.1 RDF & Semantic Web

In this section we introduce the fundamental technologies about Semantic Web, RDF, SPARQL and Linked Data.

2.1.1 Semantic Web

The Semantic Web was proposed as an extension for the Web of Documents by Berners-Lee et al. (2001) with a new name "Web of Data". This extension addressed the change of information representation that makes data machine-readable and interpretable. The term "semantic" was intended to add **meaning** to the Web while many web pages contain unstructured data that is only human-readable. The Semantic Web provides a way for publishing data in structured formats using standards such as: Resource Description Framework (RDF), Web Ontology Language (OWL). The machine-readable aspect of web documents can be achieved by adding additional tags to the current Hypertext Markup Language (HTML) based documents. For instance, consider the first HTML statement given below without semantic web tags. The same information can be encoded by adding "semantic" tags that machines could use in interpreting the HTML document. The tag "rdf:about" defines a relation between the given text "Semantic Web" and the link. The relation is not just another hyperlink but it also defines the relation name as shown below.

```
<item>Semantic Web</item>

<item rdf:about="https://www.w3.org/standards/semanticweb/">Semantic Web</item>
```

2.1.2 RDF

Resource Description Framework (RDF) is a data model for the Semantic Web used for conceptual modeling of data. RDF is based on allowing to make *statements* about resources in *subject-predicate-object* form also known as *triples*. The subject denotes a resource, the object denotes a resource or some literal value and the predicate denotes a relation between subject and object. RDF is a W3C standard[1] with the latest version 1.1 published in 2004.

[1] https://www.w3.org/RDF/

A URI (Uniform Resource Identifier) is a string used to identify resources. A resource can be anything from a person, to abstract objects. For example, the URI **http://dbpedia.org/resource/Bielefeld** identifies the city of Bielefeld located in Germany. The URI has to be unique within its domain (per definition). It is also possible to use literals for object position instead of resources. A literal can be a string or a data type value. RDF data consists of a number of triples and can be viewed as graph where the resources represent vertices and predicates represent edges. An example of RDF graph is given below.

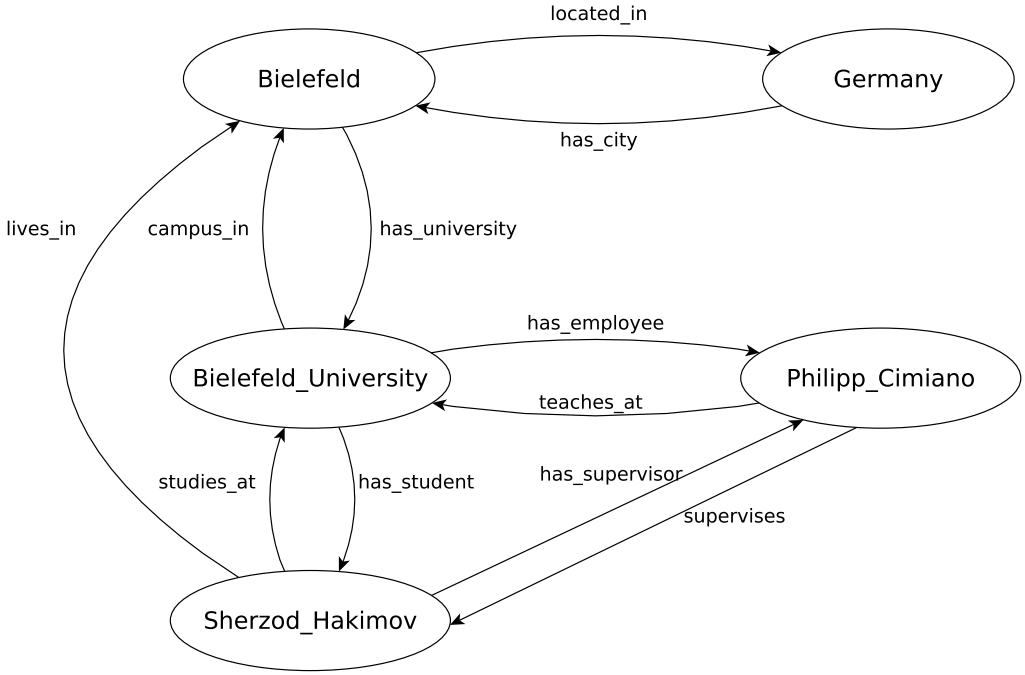


FIGURE 2.1: RDF Graph showing triples about Bielefeld city, University

RDF data in the graph can be serialized using common formats such as: N-Triples, N-Quads,Turtle, RDF/XML, JSON-LD formats. An example of serialization of triples in N-Triple and JSON-LD format is given below. The N-Triple format is a more human-friendly serialization format. JSON-LD is designed to transform existing JSON data into RDF data quicker.

N-Triple

```
Subject : http://example.org/resource/Bielefeld
Predicate :  http://example.org/predicate/located_in
Object :  http://example.org/resource/Germany
```

JSON-LD

```
{
  "@graph" : [ {
    "@id" : "res:Bielefeld",
    "located_in" : "res:Germany"
  } ],
  "@id" : "urn:x-arq:DefaultGraphNode",
  "@context" : {
    "located_in" : {
      "@id" : "http://www.example.org/predicate/located_in",
```

```
      "@type" : "@id"
    },
    "res" : "http://www.example.org/resource/",
    "pred" : "http://www.example.org/predicate/"
  }
}
```

RDF Vocabulary, RDF Schema and OWL

The RDF Vocabulary consists of RDF terms such as *rdf:type* and *rdf:Property* [2]. RDF Schema[3] extends the RDF Vocabulary with additional terms such as *rdfs:subClassOf, rdfs:subPropertyOf, rdfs:domain, rdfs:range*. The Web Ontology Language (OWL)[4] is a markup language for defining and publishing ontologies. It is an extension on top of RDF Vocabulary with additional terms for fine-grained definitions of properties, classes, individuals and data types.

2.1.3 SPARQL

SPARQL is a W3C recommendation for querying RDF data[5]. The latest version SPARQL 1.1 was published in March, 2013[6]. SPARQL has four different query variations:

- SELECT Query: to extract data from a SPARQL endpoint, the results are returned in a structured format
- ASK Query: to provide True/False results for a query from an endpoint
- DESCRIBE Query: to extract RDF descriptions about an RDF term
- CONSTRUCT Query: to extract data from an endpoint and transform the results into a defined RDF template

Each query type above is followed by a WHERE clause to add restrictions. Each restriction has to be in triple format. Additional filtering options can also be added. Aggregation clauses for ordering the query results are possible via ORDER BY clauses. Consider the following SELECT SPARQL query in Figure 2.2 for retrieving top-10 cities ranked by population in Germany using DBpedia endpoint.

The prefixes are defined in the first four lines in Figure 2.2. The query is of type SELECT with projection variables: city, population and label. The WHERE clause defines the four query statements and FILTER clause for selection of only English labels. The four query statements define a city that is located in Germany and has a certain population and label. The FILTER clause filters out all labels except English as given with the parameter “en”. The ORDER BY is an aggregation statement that ranks cities in descending order with OFFSET and LIMIT clauses to select the top-10 results from the query.

An ASK Query example is provided in Figure 1.7. The query is about checking whether there exists a river in Germany that has length of more than 500.000 km. The FILTER clause filters rivers by length with bigger than ”>” operation. The query returns true because such data exists in DBpedia.

[2] rdf : http://www.w3.org/1999/02/22-rdf-syntax-ns#
[3] rdfs: https://www.w3.org/TR/rdf-schema/
[4] owl:https://www.w3.org/TR/owl-ref/
[5] https://www.w3.org/TR/sparql11-overview/
[6] https://www.w3.org/TR/2013/REC-sparql11-overview-20130321/

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT DISTINCT ?city ?population ?label WHERE {

?city dbo:country dbr:Germany .
?city rdf:type dbo:City.
?city dbo:populationTotal ?population.
?city rdfs:label ?label.

FILTER (lang(?label) = "en")

}
ORDER BY DESC(?population)
OFFSET 0 LIMIT 10
```

FIGURE 2.2: SPARQL SELECT query for retrieving top-10 cities in Germany by population

2.1.4 Linked Data

Linked Data enables everyone to contribute to the Semantic Web by publishing their data in RDF format. The most important aspect of this project is providing links to other published datasets, thus the name comes from the term "linked". The project was proposed by Berners-Lee (2006) to publish RDF datasets and provide links between datasets. The **owl:sameAs** predicates is used to link resources that refer to the same resource in different datasets. Linked Open Data (LOD) depicts the open-access RDF datasets that provide links among them. Everyone can contribute to LOD with their datasets to be published if the following requirements are met:

- URIs must be resolvable (*http:// or https://*)
- The data must be in RDF format.
- The dataset should contain at least 50 links to the published datasets in the LOD Cloud.
- Access to the data must be possible via RDF dumps and SPARQL endpoints.

The current LOD Cloud (Abele et al., 2017) is visualized in Figure 2.4 where datasets are grouped by domains. Cross-domain datasets such as DBpedia and Freebase are visualized as main hubs with many links coming from other datasets.

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

ASK WHERE {

?river dbo:sourceCountry dbr:Germany .
?river rdf:type dbo:River.
?river dbo:length ?length.

FILTER (?length > 500000)
}
```

FIGURE 2.3: SPARQL ASK query to check if there are rivers in Germany with length more than 500.000

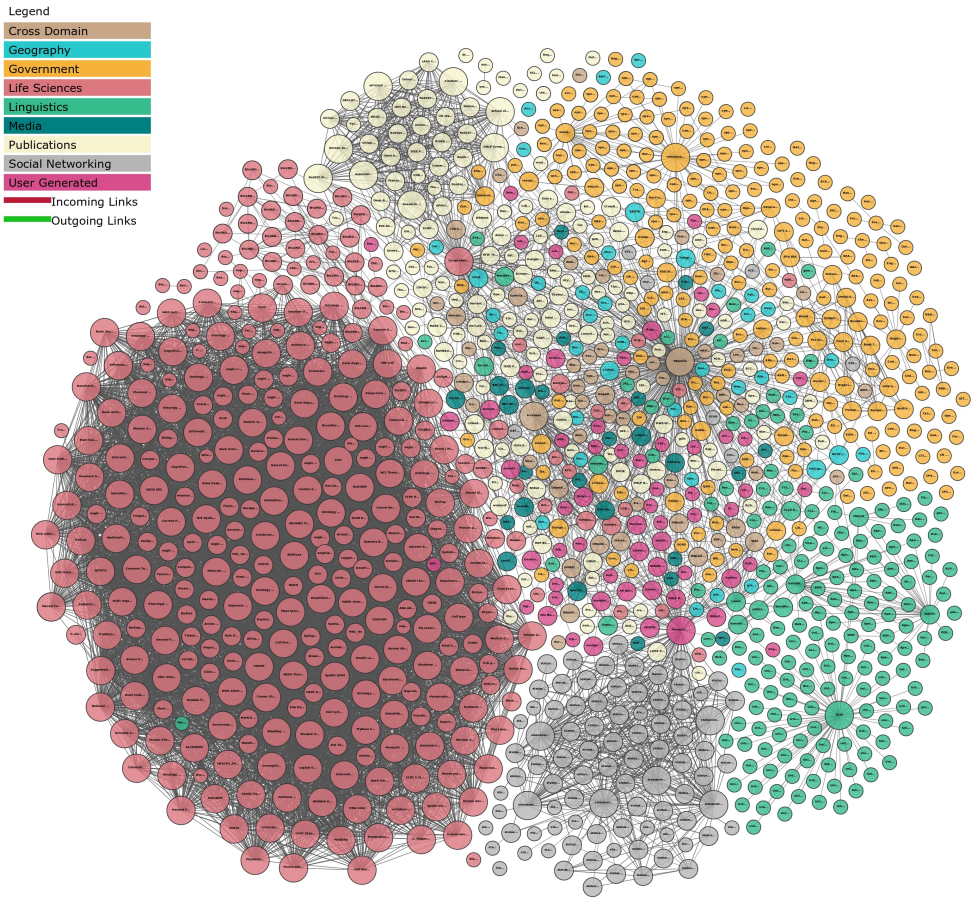


FIGURE 2.4: Linked Open Data Cloud showing links between datasets, February 2017, http://lod-cloud.net.

2.1.5 Knowledge Bases

A knowledge base is a system to store structured and unstructured data. With the popularity of publishing RDF data as Linked Open Data, many domain specific and cross-domain

knowledge bases emerged. For instance, DBpedia (Auer et al., 2007), YAGO (Suchanek et al., 2007), Wikidata (Vrandečić and Krötzsch, 2014) and Freebase (Bollacker et al., 2008) are open-access knowledge bases that have factual information about entities and their relationships. The BabelNet (Navigli and Ponzetto, 2012) is another example of open-access knowledge base that contains multilingual linguistic data by combining data from other knowledge bases.

Wikipedia has become a central source of knowledge for daily life, being maintained by thousands of contributors in many languages. DBpedia is a crowd-sourced community effort to extract structured information from Wikipedia. As shown in Figure 2.5, the structured content of a Wikipedia article is visualized on the right side. This content is extracted from all articles. The data is transformed into RDF by matching the concepts on structured content to the DBpedia Ontology. DBpedia has an ontology with 1105 object properties, 1622 data type properties and 760 classes. The ontology and the data is being updated yearly since 10 years from its initial release in 2007. The latest release of DBpedia has localized editions of data in more 130 languages.[7] For example, the DBpedia Ontology has a property *dbo:capital* defined for relationships between countries and their capital cities. As highlighted in the Figure 2.5, the triple *dbr:Turkmenistan dbo:capital dbr:Ashgabat* is added to DBpedia.

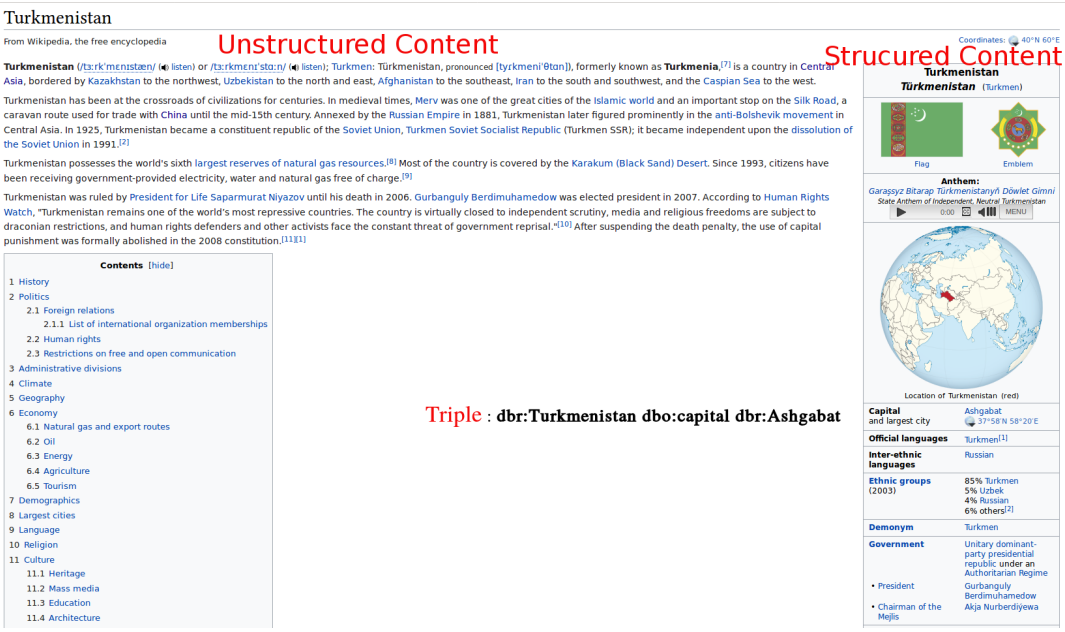


FIGURE 2.5: Structured and unstructured content in Wikipedia article, taken from https://en.wikipedia.org/wiki/Turkmenistan

2.2 Syntax

Syntax is a set of rules that govern the structure of sentences in a language. These rules govern word order, conjugation, etc. in sentences. Basic syntax of many languages can be expressed with subject (S), verb (V) and object (O) notations. Sentences can be built using different combinations of them e.g. SVO, SOV, VSO etc. For instance, English follows the SVO notation "John (S) likes (V) Mary (O)". Grammar is a set of such rules that govern syntax in a given language. There are many approaches for defining grammar. In this thesis, we focus on two grammar formalisms: Categorial Grammar and Dependency Grammar. In the following sections we explain each approach in detail.

[7] https://wiki.dbpedia.org/downloads-2016-10

2.2.1 Categorial Grammar

Categorial Grammar (CG) is an approach for defining the syntax of sentences with syntactic categories. Phrases can be built either by atomic or complex syntactic categories. These atomic categories are given below:

- S: sentence
- N: noun
- NP: noun phrase

Combinatory Categorial Grammar (CCG) (Steedman, 1996, 2000) is a linguistic grammar formalism for describing constituency-based structures using the categorial grammar. It uses the atomic syntactic categories given above and the operators / (slash), \ (back slash), where A/B (or $A\backslash B$) denotes a function that takes an argument of type B to its right (or left) and returns an object of type A. The operator / (slash) expects the intended argument on the right, the operator \ (back slash) expects the argument to be on the left.

More complex categories such as transitive verbs can be built with combination of atomic categories, operators and parentheses. For instance, in English transitive verbs are assigned the category (S\NP)/NP, where it first expects a noun phrase (NP) on the right then another noun phrase (NP) to its left then the phrase is a sentence. The main intuition of CCG is that words in natural language act like functions. In the example of transitive verbs, the function consumes two noun phrases, given the right order of noun phrases it results in a sentence.

CCG consists of combination rules and a set of lexical items where tokens are paired with syntactic categories. These combination rules can be specified as follows:

- *Forward application:* $\dfrac{A/B \quad B}{A}$
- *Backward application:* $\dfrac{B \quad A\backslash B}{A}$
- *Function composition:* $\dfrac{A/C \quad C/B}{A/B}$

FIGURE 2.6: CCG combination rules

A forward application is based on the / operator where A/B expects another B on the right. The combination of these syntactic categories results in A. Backward application is similar with difference in the direction of application. The syntactic category $A\backslash B$ expects another syntactic category B on its left and the combination of these results in A. Function composition combines two complex categories. The category C being common in both and the operator / results in A/B after combination.

The combination rules shown above are the same for all languages that use CCG. However, complex categories must be defined for each language individually. A simple example of a CCG lexicon for the sentence *Barack Obama is married to Michelle Obama* is given in Table 2.1. Each entry is paired with a phrase and syntactic category.

TABLE 2.1: Lexicon based on CCG

Phrase	Syntactic category
Barack Obama	NP
is	(S\NP) / (S\NP)
married to	(S\NP) / NP
Michelle Obama	NP

In Figure 2.7, we illustrate how the rules above are applied to the lexicon in Table 2.1 in order to construct the syntactic structure of the sentence. Similar to CCG combination rules, lambda calculus expressions can also be combined in order to obtain meaning representations for bigger phrases. More on applying CCG together with lambda calculus is explained in Section 2.3.

As shown in Figure 2.7, the lexical item *married to* of category (S\NP)/NP is combined with the lexical item *Michelle Obama* of category NP using forward application. The result is an expression of category S\NP.

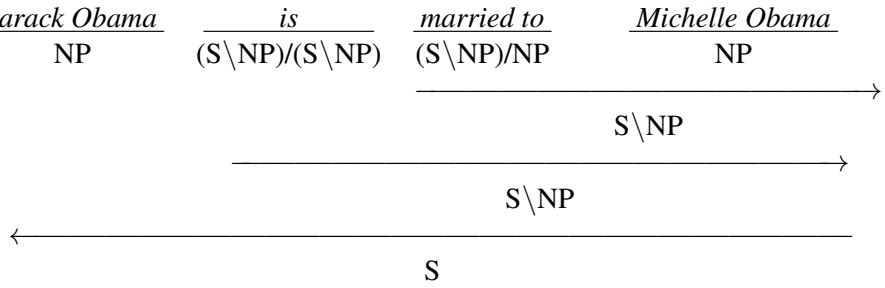


FIGURE 2.7: CCG parse tree for the sentence *Barack Obama is married to Michelle Obama.*

The syntactic category of *is* can be combined with the category S\NP, the syntactic category for combination of *married to* and *Michelle Obama*, using forward application. This combination results in S\NP. Finally, the syntactic category NP of *Barack Obama* can be combined with S\NP. This application process is continued until all items are combined. If the result of the last combination results in the syntactic category is S, as in our example, then the derivation is considered a valid parse tree for the given sentence.

Different parse trees can be obtained using the same lexicon and combination rules. For example, the category of *is* and *married to* can be combined via function composition as a first combination that results in (S\NP)/NP. The remaining items *Barack Obama* and *Michelle Obama* are two NP categories with one being on the left and the other on the right. It is shown in Figure 2.8. All possible valid parse trees can be obtained by bottom-up parsing approach using dynamic programming algorithms such as CYK (Cocke, 1970; Kasami, 1965; Younger, 1967).

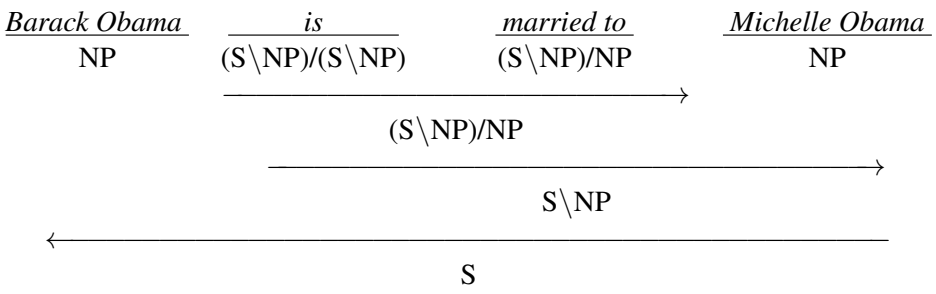


FIGURE 2.8: Alternative CCG parse tree for the sentence *Barack Obama is married to Michelle Obama.*

2.2.2 Dependency Grammar

Dependency Grammar (DG) is a formal way of representing the syntax of a sentence using dependency relations between words. Lucien Tesnière's work called "Éléments de syntaxe structurale (Elements of Structural Syntax)" defined the basis of this approach. Dependencies between words are expressed with direct links. These dependencies define syntactic properties of words in a sentence. A sentence can be represented with DG as a tree with nodes, words in a sentence, and edges, representing the dependency relations between words. Such tree using DG is referred as a *dependency parse tree*. An example parse tree is given in Figure 2.9 for the sentence *Albert Einstein invented Relativity Theory*.

The node *invented* is the root of the sentence indicating the main action of the sentence. The node *Albert Einstein* is the nominal subject of the node *invented* as depicted by the dependency relation *nsubj*. The node *Relativity Theory* is the direct object of the node *invented* as shown via the edge *dobj*. Finally, the dot (.) has a punctuation dependency relation (*punct*) to the root node.

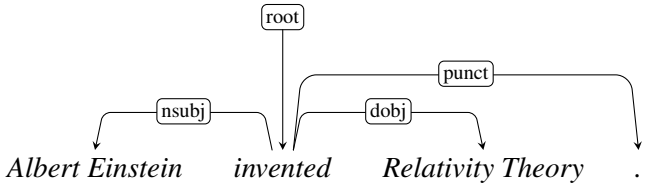


FIGURE 2.9: Dependency parse tree for the sentence *Albert Einstein invented Relativity Theory* in Universal Dependencies syntax

Recent efforts in developing a cross-lingual treebank resulted in Universal Dependencies (UD) (Nivre et al., 2016; Nivre, 2017). UD aims to capture syntactic similarities between languages for developing multilingual NLP approaches. UD has a universal set of morphological and syntactic specifications for part-of-speech (POS) tags and dependency relations. Currently, UD includes 70 treebanks for 50 languages. Having a universal set of POS tags will enable NLP systems to learn models based on particular language and transfer learned models to other languages easier. The current set of POS Tags are specified as follows[8]:

- ADJ: adjective
- ADV: adverb
- AUX: auxiliary verb
- CONJ: coordinating conjunction
- DET: determiner
- INTJ: interjection

[8] http://universaldependencies.org/v2, 70 treebanks, 50 languages

- NOUN: noun
- NUM: number
- PART: particle
- PRON: pronoun
- PROPN: proper noun
- PUNCT: punctuation
- SCONJ: subordinating conjunction
- SYM: symbol
- VERB: verb
- X: other

UD has a set of universal dependency relations based on different syntactic properties of words in different languages (De Marneffe et al., 2014). The revised version of dependency relations in UD are grouped under the categories given in Figure 2.10 [9]:

Using the multilingual corpora of treebanks annotated by the community UD builds a model with the available languages. Then, given a sentence, UD can generate a dependency parse tree for the available languages. In Figure 2.11 dependency parse trees are given for questions in English, German and Spanish languages. The dependency parse trees also include POS tags for each node. The questions are asking the same fact in each language. It can be seen that pronoun words *Who, Wer, Quién* have the same dependency relation *nsubj* to the root and the root nodes have the same POS tag *VERB*. Furthermore, the node *Wikipedia* is the direct object of the root as depicted by *dobj* relation in all languages.

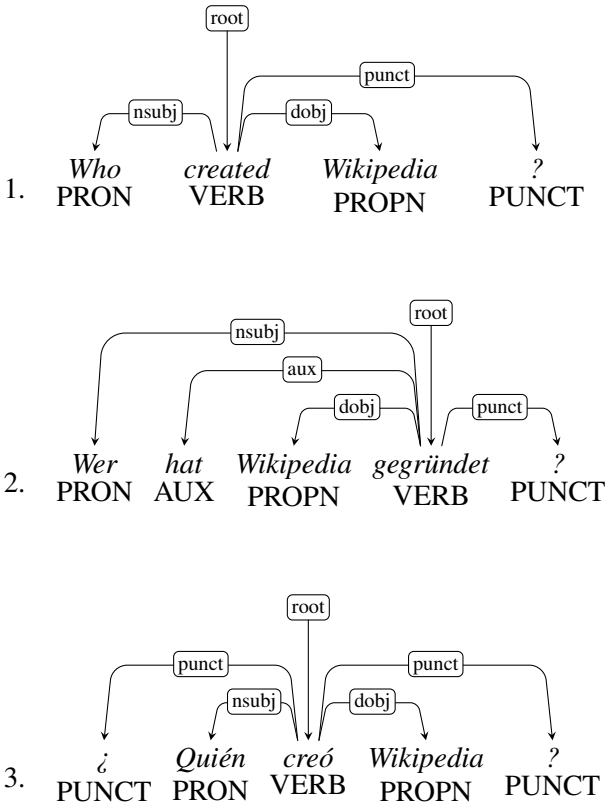


FIGURE 2.11: Universal Dependency parse trees with part-of-speech (POS) tags for questions in English, German and Spanish languages

[9] http://universaldependencies.org/u/dep/index.html

- **Core dependents of clausal predicates**
  - nsubj: nominal subject
  - nsubjpass: passive nominal subject
  - dobj: direct object
  - iobj: indirect object
  - csubj: clausal subject
  - csubjpass: clausal passive subject
  - ccomp: clausal complement
  - xcomp: open clausal complement
- **Noun dependents**
  - nummod: numeric modifier
  - appos: appositional modifier
  - nmod: nominal modifier
  - acl: clausal modifier of noun
  - acl:relcl: relative clause modifier
  - amod: adjective modifier
  - det: determiner
  - neg: negative modifier
- **Case-marking, prepositions, possessive**
  - case: case marking
- **Non-core dependents of clausal predicates**
  - advcl: adverbial clause modifier
  - advmod: adverbial modifier
  - nmod: nominal modifier
  - neg: negative modifier
- **Compounding and unanalyzed**
  - compound: compound
  - compound:prt : separable verb particle
  - mwe: multi-word expression
  - goeswith: goes with
  - name: name
  - foreign: foreign words
  - flat: flat multi-word expression
- **Loose joining relations**
  - list: list
  - dislocated: dislocated elements
  - parataxis: parataxis
  - remnant: remnant in ellipsis
  - reparandum: overridden disfluency
- **Special clausal dependencies**
  - vocative: vocative
  - aux: auxiliary
  - auxpass: passive auxiliary
  - discourse: discourse element
  - cop: copula
- **Coordination**
  - conj: conjunct
  - cc: coordinating conjunction
- **Other**
  - root: root
  - dep: unspecified dependency

FIGURE 2.10: Dependency relations available in Universal Dependencies

2.3 Semantics

In this section we introduce information about semantics. Semantics is the study of meaning in language. The term semantics derives from Ancient Greek term "seme" that means "sign". Meaning can be put as function of signs in a language. The notion of meaning caught the attention of many scholars throughout the history of humankind. Greek philosophers such as Plato and Aristotle had different views on the notion of semantics: whether the meaning of words depended on the sound it produced or on the context they appear in. Ludwig Wittgenstein defined the meaning of a word as "the role a word plays in a sentence". In linguistics, semantics is a field of study devoted to understanding the meaning of words, phrases and sentences with respect to the representation of meaning. The study is oriented around understanding relationships between linguistic units. In the following sections, we describe lexical, distributional, formal and compositional semantics.

2.3.1 Lexical Semantics

Lexical semantics is a subfield in linguistics that focuses on understanding meanings of individual phrases. WordNet (Miller, 1995b) is an example for defining semantics of individual words. It was developed for English and later the approach was adapted to other languages as well (Bond and Foster, 2013; Vossen, 1998). Essentially, WordNet is a graph of word senses. Words are nodes in the graph where an edge links two words based on a defined relationship. The most common relation among words is *hypernym* relation. It links more general words like *bird* into more specific ones like *parrot*. This relation is sometimes called ISA relation. *Meronym* is another relation that is being used to define the part-whole relation. For instance, a *car* has a *wheel* is an example for a meronym relation. Besides these two relations, WordNet also includes more specific relations based on the verbs. For instance, *fish lives in water*. All of the relation types explained earlier are illustrated with examples in Figure 2.12.

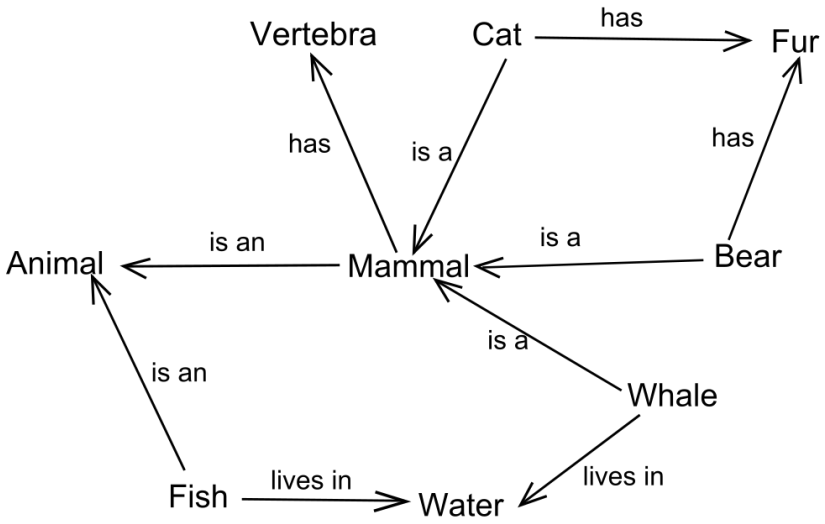


FIGURE 2.12: Lexical semantics, WordNet semantic network of words, taken from https://en.wikipedia.org/wiki/Lexical_semantics

In Computer Science, specifically in Ontology Engineering, the term semantics refers to the meaning of concepts, properties and their relationships that represent real-world entities. The DBpedia Ontology represents factual information about the world, which is defined

using RDF, RDFS and OWL vocabulary. The concept such as *Person* is defined with the following triples:[10]

```
dbo:Person rdf:type owl:Class.
dbo:Person subClassOf dbo:Agent.
rdfs:label "person"@en.
rdfs:label "Person"@de.
```

The concept *dbo:Person* has a hypernym relation with the general concept called *owl:Class*. This relation is represented by *rdf:type* property in DBpedia. The property *dbo:author*[11] is defined with the following triples:

```
dbo:author rdf:type owl:ObjectProperty.
dbo:author rdfs:range  dbo:Person.
dbo:author rdfs:domain dbo:Work.
dbo:author rdfs:label "author"@en.
dbo:author rdfs:label "Autor"@de.
```

Based on the definitions given above, the property *author* defines a relation between individuals that are instances of concepts such as *Work* and *Person*.

In this thesis, we focus on semantic parsing approaches that map natural language questions into ontology concepts, properties and resources in a given knowledge base.

2.3.2 Distributional Semantics

Another way of expressing semantics can be done using distribution of words in a large corpus. Distributional semantics focuses on developing theories and methods for categorizing semantic similarities of linguistic items based on samples of language data. It differs from lexical semantics in a way that the meaning of words is based on the context they appear in. The semantics of words are categorized based on a large sample of the corresponding language data. This became popular with Firth's saying "You shall know a word by the company it keeps" (Firth, 1957). This field of study builds on the *distributional hypothesis*. The hypothesis is based on the idea that linguistic items with similar distributions have close meanings.

In recent years, researchers focused on calculating distributional semantics of words from large corpora such Wikipedia articles. The well-known method called word2vec was proposed by Mikolov et al., 2013 that transforms words into dense vector representations by embedding the context of words. It is based on the idea that words that have similar context tend to have similar vector representations. Based on this vector representations, the similarity can be calculated using the cosine similarity metric. An illustration of distributional semantics of words and their semantic similarities are shown in Figure 2.13.

[10] http://dbpedia.org/ontology/Person
[11] http://dbpedia.org/ontology/author

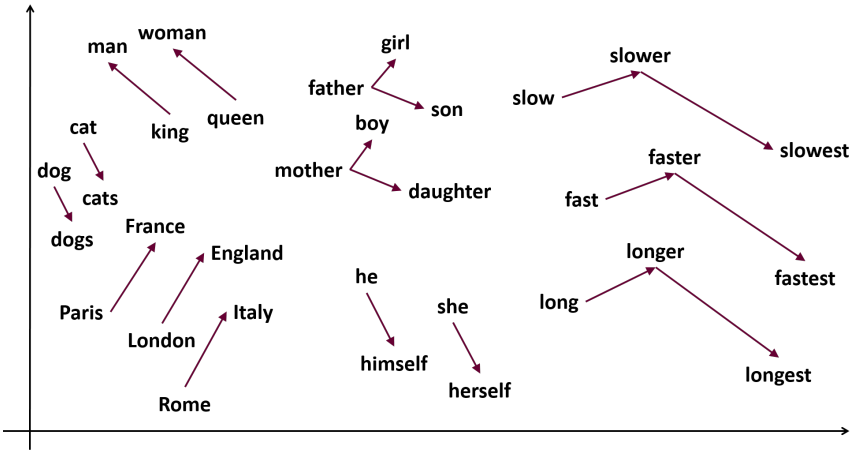


FIGURE 2.13: Distributional semantics, word2vec visualization of words in 2-dimensional vector space, taken from http://www.samyzaf.com/ML/nlp/nlp.html

The dimensions of vector representations can be set to any dimension, they are usually set to 300 dimensions. The illustration in Figure 2.13 is shown as 2-dimensional representation of word vectors, which is only for illustration purposes.

Words that appear in similar contexts such as *father, girl, son* are clustered closer to each other. It can be seen that semantic relations such as plural forms of words, capital cities to countries, adjectives to superlative forms can be captured using the distributional semantics of words. For instance, the following vector operations can be performed since the relation between two capital cities and the countries looks similar: W("London") - W("England") $\cong$ W("Rome") - W("Italy"). Another famous example that follows the same pattern is W("woman") - W("man") $\cong$ W("queen") - W("king").

2.3.3 Formal Semantics

In linguistics, formal semantics focuses on understanding the meaning of words based on formal language. A formal language is a set of expressions defined for a specific domain. The idea became popular with Richard Montague's work known as Montague Grammar (Montague, 1970a,b). He showed how a natural language could be treated as any formal language such as a programming language. The most important part of the theory is how the semantics of a sentence is computed. The semantics of a sentence is the functional application of the semantics of single words. It is an influential work for considering syntax and semantics in tandem because the syntax of the sentence is the combination of grammar rules based on single words.

Lambda Calculus

The formal semantics of a language can be expressed using mathematical logic expressions. Lambda calculus (Church, 1936) is a such formalism for expressing semantics based on function abstraction and application. The name comes from the Greek letter λ used in lambda expressions for denoting functions.

Lambda expressions consist of lambda terms. Lambda terms are valid strings for defining lambda calculus expressions.

Lambda expressions are composed of :

- Variables : $x_1 \ldots x_n$

- Abstraction symbols lambda 'λ' and dot '.'
- Parentheses ()

Formally, the set of lambda expressions, Λ, can be defined inductively:

- Variable: If x is a variable, then $x \in \Lambda$
- Abstraction : If x is a variable and $M \in \Lambda$, then $(\lambda x.M) \in \Lambda$
- Application: If $M, N \in \Lambda$, then $(M\,N) \in \Lambda$

An abstraction $\lambda x.M$ is a function that takes an input x and replaces it in the expression M. For example, λx. λy. *(x+y)* is a lambda abstraction where *(x+y)* stands for the expression M and it takes 2 inputs defined by the variables *x, y*. It represents the function $f(x,y) = x + y$. The variables x and y are bound to the term *(x+y)* by abstraction.

An application $(M\,N)$ means the application of function M to an input N. For instance, if we apply the function λx. λy. *(x+y)* to the input "2" then the variable x is replaced with "2" using Beta reduction rule. The rule reduces the functional application (λx. λy. *(x+y)*) *2* into λy. *(x+y)*[*x*:=*2*]. The resulting expression will be λy. *(2+y)*. The order of application in the function depends on the order the variables. The left-most variable is replaced first in this case. Bracketing can also be used to disambiguate the lambda expression.

Let's consider the application domain for a QA system. Geoquery (Tang and Mooney, 2001)[12] is a dataset consisting of pairs of question and lambda calculus expression. The questions are about US geography and the lambda calculus expression is the semantics of questions based on the database of geographic data. The database contains concepts such as river, state, city. It also contains properties to express relationships. For example, the property *next_to* is a binary relation between two states that stands for bordering states. The property *state* is a unary relation that expresses an individual belonging to a concept. There are also individuals that represent entities existing in the world, e.g. city:Boston, state:Texas, river:Mississippi. An example pair of a sentence and the lambda expression is given below.

Sentence: *Give me all states that border Texas.*

Lambda expression: $\lambda x.(\texttt{state}(x) \wedge \texttt{next\_to}(x, \texttt{state:Texas}))$

The lambda expression is composed of the following two expressions:

some individual is a state : $\lambda x.\texttt{state}(x)$
some individual borders Texas : $\lambda x.\texttt{next\_to}(x, \texttt{state:Texas})$

DUDES

DUDES (*Dependency-based Underspecified Discourse Representation Structures*) (Cimiano, 2009) is a formalism for specifying meaning representations and their composition. They are based on *Underspecified Discourse Representation Theory* (UDRT) (Cimiano et al., 2007; Reyle, 1993), and the resulting meaning representations. Formally, a DUDE is defined as follows:

Definition 1 *A* DUDE *is a 5-tuple* $(v, vs, l, drs, slots)$ *where*

[12] http://www.cs.utexas.edu/users/ml/nldata/geoquery.html

- *v is the* main variable *of the DUDES*
- *vs is a (possibly empty) set of variables, the* projection variables
- *l is the label of the* main DRS
- *drs is a DRS (the main semantic content of the DUDE)*
- *slots is a (possibly empty) set of semantic dependencies*

The core of a DUDES is thus a *Discourse Representation Structure* (DRS) (Kamp and Reyle, 1993). The main variable represents the variable to be unified with variables in slots of other DUDES that the DUDE in question is inserted into. Each DUDE captures information about which semantic arguments are required for a DUDE to be complete in the sense that all slots have been filled. These required arguments are modeled as sets of slots that are filled via (functional) application of other DUDES.

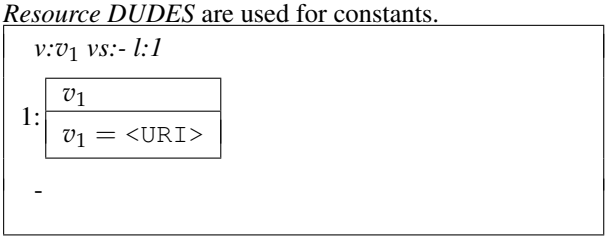
Definition 2 *A* slot *is a 3-tuple* (v, a, l) *where*

- *v is the argument entity in the DUDES*
- *a is an* anchor *that connects this entity to the syntactic element that provides its semantic content; this could, e.g., be the label of a syntactic dependency*
- *l is the label of the DRS to which the semantic content of v is to be added*

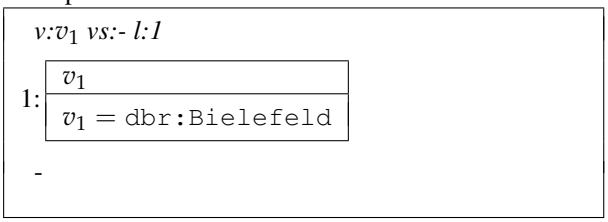
The projection variables are relevant in meaning representations of questions; they specify which entity is asked for. When converting DUDES into SPARQL queries, they will directly correspond to the variables in the `SELECT` clause of the query. Finally, slots capture information about which syntactic elements map to which semantic arguments in the DUDE.

As basic units of composition, we consider 6 pre-defined DUDES types that correspond to data elements in RDF datasets. Below we give the definition of each DUDE type and a corresponding instantiated example of a DUDE.

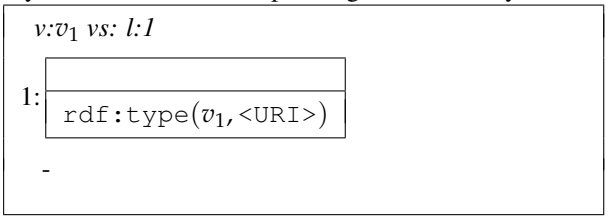
1. *Resource DUDES* are used for constants.



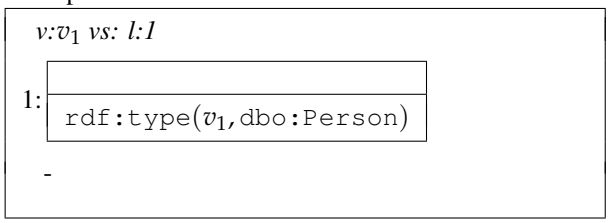
Example:



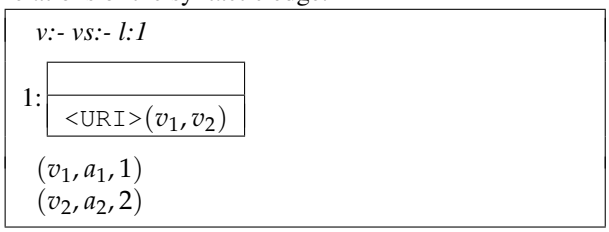
2. *Class DUDES* contain one condition: a binary predicate `rdf:type` between a discourse entity and the class URI, expressing that this entity is of the specified type.



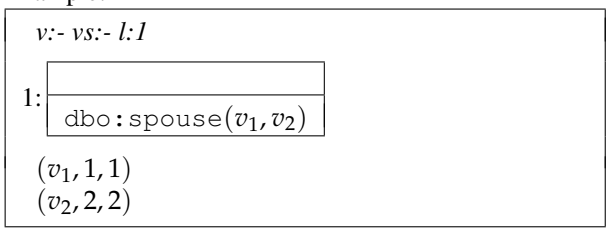
Example :



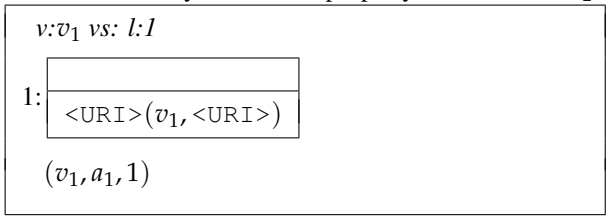
3. *Property DUDES* also contain one condition: a binary predicate specified by the property URI between two discourse entities. The entities are contributed by the meaning representations of those syntactic arguments related to the property expression by means of dependency relations a_1 and a_2. The specific relations depend on the syntactic type of the property expression (transitive verbs, relational nouns, etc.) and the dependency relation set used by the parser. We use slot labels 1,2 to *nsubj, dobj, amod, nmod, etc.*, which are flexible compared to dependency relations of the syntactic edge.

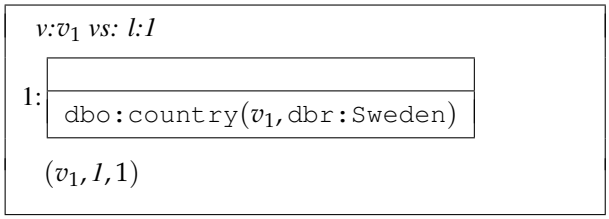


Example:

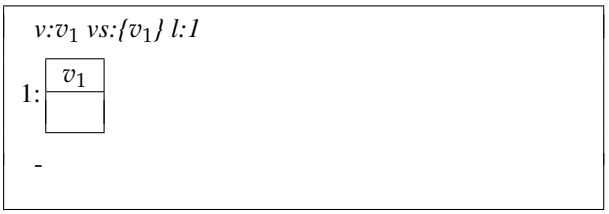


4. *Restriction class DUDES* are like class DUDES, but use another property than `rdf:type` and contain one slot for the discourse entity. In addition to classes typically expressed by nouns such as *mountain* referring to the DBpedia class `dbo:Mountain`, we define class DUDES for restriction classes, typically expressed by adjectives. For example, the expression *Swedish* can be represented as a restriction class of all resources that are related to the resource `dbr:Sweden` by means of a property `dbo:country` or `dbo:birthPlace`.





5. *QueryVar DUDES* introduces a discourse entity that is added to the set of projection variables (i.e. will end up in the `SELECT` clause of the resulting SPARQL query), it used for specific, domain-independent expressions relevant for wh-words, such as *which* and *what*.



6. *Null DUDES* introduce no information.

2.3.4 Compositional Semantics

Compositional semantics is a field of study focused on studying how the compound meaning expressions can be composed using semantic rules and syntax. The smaller semantical units build up the compound expressions, building larger blocks of semantics by merging smaller units. The units are combined by defined composition rules. Next we explain two compositional semantics approaches.

Semantic Composition with CCG and Lambda Calculus

This approach is based on CCG application rules and the compositionality property of the lambda calculus. CCG application rules for syntax are given in Section 2.2.1. The categorial grammar for a sentence is composed of lexical entries that are paired with CCG category and a corresponding phrase. The semantics of each phrase can be expressed using lambda calculus expressions (Section 2.3.3). For instance, proper names such as *Texas* stand for individuals in a specific domain. In the Geoquery (Tang and Mooney, 2001) dataset each question is paired with a corresponding lambda expression. The CCG combination rules can be adapted to compose the syntax and semantics of entries by combination rules given below.

- *Forward application:* $\dfrac{A/B : f \qquad B : x}{A : f(x)}$
- *Backward application:* $\dfrac{B : x \qquad A\backslash B : f}{A : f(x)}$
- *Function composition:* $\dfrac{A/C : f \qquad C/B : g}{A/B : \lambda x.f(g(x))}$

FIGURE 2.14: CCG combination rules for syntax and semantics

As given in Figure 2.14, each combination rule includes combination of syntax and semantics together, e.g. A/B is a CCG category and f is a lambda calculus expression. For instance, when *Forward application* is applied then the CCG categories are combined into resulting category A. The semantics of this expression is the functional application of the lambda expressions *f(x)* to *x*. In Table 2.2 we provide for each phrase a corresponding CCG category and lambda calculus expression for the sentence *Barack Obama is married to Michelle Obama*. This sentence is expressed with the following triple in DBpedia :

`dbo:spouse(dbr:Barack_Obama,dbr:Michelle_Obama)`

Phrase	Syntax	Semantics
Barack Obama	NP	dbr:Barack_Obama
is	(S\NP)/(S\NP)	$\lambda f.\lambda x.f(x)$
married to	(S\NP)/NP	$\lambda y.\lambda x.$dbo:spouse(x,y)
Michelle Obama	NP	dbr:Michelle_Obama

TABLE 2.2: Lexical entries with syntactic and semantic representations.

Each lambda calculus expression is adapted to the DBpedia Ontology where individuals represent resources and transitive verbs represent predicates. The first entry has a string *Barack Obama* that stands for an individual with CCG category NP and lambda expression dbr:Barack_Obama. The string *married to* is paired with CCG category for a transitive verb (S\NP)/NP. The semantics for this entry is a DBpedia Ontology predicate *dbo:spouse* where the relation is expressed using two arguments, subject and object from the definition of triples. Thus, the two-ary lambda expression $\lambda y.\lambda x.$dbo:spouse(x,y) expects two inputs to replace variables x and y. The string *is* is a auxiliary verb that does not hold any DBpedia Ontology information with a CCG category (S\NP)/(S\NP).

The semantics of the sentence is composed by applying the combination rules in Figure 2.14 above to the entries in Table 2.2. This application process is continued until all items are combined. If the resulting syntactic category is S, as in our example, then the derivation is considered a valid parse tree for the given sentence, and its meaning is expressed by the corresponding logical expression, in our case:

$$\texttt{dbo:spouse}(\texttt{dbr:Barack\_Obama},\texttt{dbr:Michelle\_Obama})$$

The parse tree for obtaining the lambda expression given above and the CCG category S is given below in Figure 2.15.

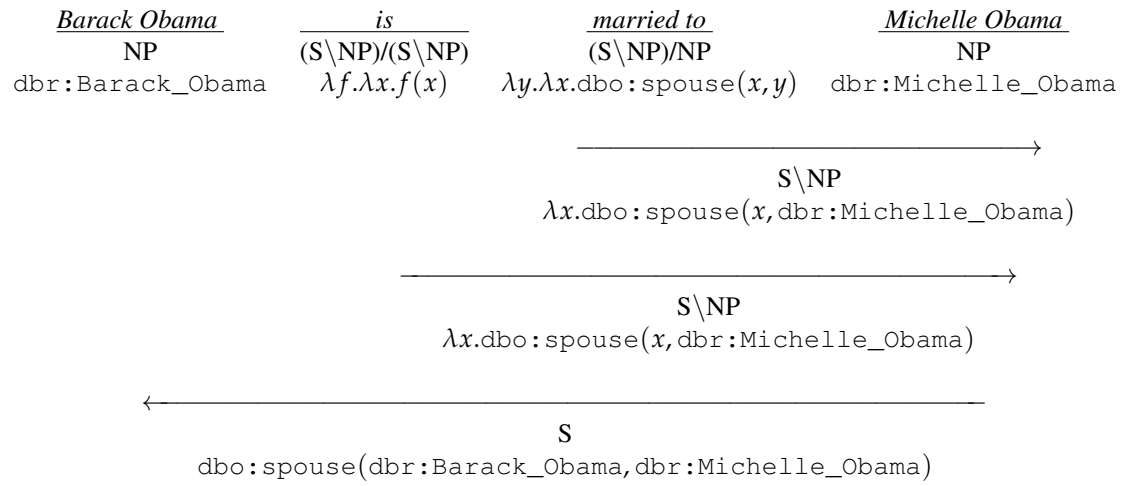


FIGURE 2.15: CCG parse tree with syntax and semantics for the sentence *Barack Obama is married to Michelle Obama*.

The first composition of entries for *Michelle Obama* and *married to* are obtained by applying the combination rule *Forward application*. It results in CCG category S\NP. The lambda calculus expression of both entries are combined based on the combination rules. The entry $\lambda y.\lambda x.$dbo:spouse(x,y) is applied to the entry dbr:Michelle_Obama. This application results in the replacement of the variable y with the entry dbr:Michelle_Obama,

since the variable y is the first variable for in the function $\texttt{dbo:spouse}(x, y)$. The resulting lambda expression is:

$$\lambda x.\texttt{dbo:spouse}(x, \texttt{dbr:Michelle\_Obama})$$

The same procedure is applied to all entries as shown below to combine both syntax and semantics. The second combination with the entry *is* results in S\NP CCG category based on *Forward application* and the same semantics. Since, the lambda calculus $\lambda f.\lambda x.f(x)$ for the entry *is* is a placeholder semantics that results in the input after the application.

The last combination replaces the variable x with $\texttt{dbr:Barack\_Obama}$ in the semantics (*Backward application*). The syntax of this combination results in the CCG category S. The composition of entries stops at this point because no entries are left. If the final CCG category is S then the parse tree is considered valid. The validity of the semantics of the parse tree is measured by comparing it to the expected lambda calculus. In this example, the expected lambda expression is obtained.

Semantic Composition with Dependency Parse Trees and DUDES

The composition of DUDES proceeds in parallel to the syntactic structure and in a standard bottom-up fashion. Different DUDES can be combined to express more complex terms. The composition of DUDES differs from lambda calculus in a way that DUDES composition is not based on order of application. For instance, consider the lambda expression below:

$$\lambda y.\lambda x.\texttt{dbo:spouse}(x, y)$$

The variable y needs to be replaced first before the variable x. DUDES, on the other hand, do not depend on the order of variables.

Definition 3 *Given a DUDES* $(v_1, vs_1, l_1, drs_1, slots_1)$ *with* $(v, a, l) \in slots_1$ *and a DUDES* $(v_2, vs_2, l_2, drs_2, slots_2)$ *that is syntactically marked as a (e.g. in form of a dependency relation*[13]*), they can be composed into the DUDES* $(v_1, vs_1 \cup vs_2, l_1, drs_1 \cup drs_2[v_2 \mapsto v]$, $slots_1 \backslash (v, a, l) \cup slots_2)$*, where*

- $\cup$ *is set union,*
- $\backslash$ *is set subtraction,*
- $drs_2[v_2 \mapsto v]$ *is like* drs_2 *but* v_2 *is replaced by* v*, and*
- $drs_1 \cup drs_2$ *is the union of two DRSs* (l_1, D_1, C_1) *and* (l_2, D_2, C_2) *defined as* $(l_1, D_1 \cup D_2, C_1 \cup C_2)$.[14]

As an example consider the sentence : *Walt Disney created Goofy*. The sentence has subject *Walt Disney*, an object *Goofy* and a main verb *created*. The syntax of this sentence can be expressed with the following dependency parse tree.

[13] For the composition of DUDES on the basis of a Lexicalized Tree Adjoining Grammar structure, see Cimiano et al. (2014).

[14] Note that this union is directed in the sense that the label of the resulting DRS is l_1, not l_2. Which label is picked does not make a difference semantically, but it needs to be in accordance with the main DRS label specified in the DUDES.

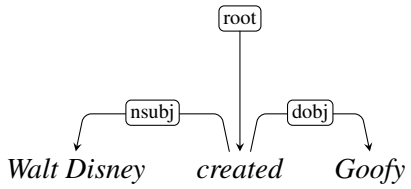


FIGURE 2.16: Dependency parse tree for the sentence *Walt Disney created Goofy* in Universal Dependencies syntax.

Each node in this dependency parse tree can be expressed with a DUDES. The root node is the main verb of this sentence. Thus, it is expressed as a Property DUDES. Its DUDES can be depicted as a DRS together with a main variable v, a set *vs* of projection variables, a URI *dbo:creator*, a DRS label l, and a set of two slots (nsubj, dobj):

1.
v:- vs:- l:1

1:	
$dbo{:}creator(v_2, v_1)$

$(v_1, nsubj, 1)$
$(v_2, dobj, 2)$

Here the main variable is not set, there are no projection variables, and only one DRS labeled 1. This DRS contains the semantic contribution of the verb *to create*: the statement $dbo{:}creator(v_2, v_1)$, where v_2 is going to be contributed by the syntactic subject of the verb depicted by the relation *nsubj*, and v_1 is going to be contributed by the direct object, *dobj*. The nodes *Walt Disney* and *Goofy* are individuals in this sentence and they are specified as Resource DUDES as shown below.

2.
v:v_3 vs:- l:1

1:	v_3
$v_3 = dbr{:}Goofy$

-

3.
v:v_4 vs:- l:1

1:	v_4
$v_4 = dbr{:}Walt\_Disney$

-

Its DRS introduces an entity, v_3, that is set equal to *dbr:Goofy* and v_4 in the other Resource DUDES for *dbr:Walt_Disney*.

Small units of the sentence are expressed with the instantiated DUDES explained above. The semantics of the whole sentence needs to be obtained by merging the smaller units with each other. DUDES are designed for the composition along a dependency parse tree where the composition is independent of the order of application.

The semantic composition of the whole sentence is obtained by bottom-up parsing where the child nodes are merged with their parent nodes. The final DUDES at the *root* node

represents the semantics of a given sentence. For instance, if we apply the child node *Goofy* with its parent node *created* it yields the following DUDES:

4.
$$\begin{array}{|l} v\text{:- } vs\text{:}\{\}\ l\text{:}1 \\ 1\text{:}\ \begin{array}{|l} \hline \\ \hline dbo\text{:}creator(dbr\text{:}Goofy, v_1) \\ \hline \end{array} \\ (v_1, nsubj, 1) \end{array}$$

The Resource DUDES is unified with the corresponding argument variable v_2 since the node *Goofy* is syntactically connected by dependency relation *dobj* to the node *created*. The resulting DUDES is unified with the remaining Resource DUDES 3 with the URI *dbr:Walt_Disney*. It is applied to the remaining argument v_1 since the node *Walt Disney* is connected via dependency relation *nsubj* as given in the Figure 2.16. This result in the following DUDES that represents the whole sentence since all edges in the parse tree have been explored:

5.
$$\begin{array}{|l} v\text{:- } vs\text{:}\{\}\ l\text{:}1 \\ 1\text{:}\ \begin{array}{|l} \hline \\ \hline dbo\text{:}creator(dbr\text{:}Goofy, dbr\text{:}Walt\_Disney) \\ \hline \end{array} \\ - \end{array}$$

This DUDES does not contain any variable to be replaced and it does not contain any projection variables (*vs is empty*). Note that the resulting DUDES 5 can be obtained no matter which DUDES (2 or 3) is applied first, which is different than lambda calculus based semantic composition.

Translating DUDES into SPARQL query

DUDES are translated into SPARQL queries by translating their DRS into the query body. The projection variables, if any, will be used for the `SELECT` clause. If the set of projection variables is empty, the query is either constructed as an `ASK` or `SELECT *` query. In the following we use super script T as translation operator; the translation of DUDES can then be defined as follows.

$$(v, vs, l, drs, slots)^T = \begin{cases} \texttt{ASK WHERE } \{drs^T\} & \text{if } vs = \emptyset \text{ and not } b \\ \texttt{SELECT * WHERE } \{drs^T\} & \text{if } vs = \emptyset \text{ and } b \\ \texttt{SELECT } vs \texttt{ WHERE } \{drs^T\} & \text{otherwise} \end{cases}$$

The translation of a DRS boils down to translating all of its conditions, i.e. $(l, D, C)^T = C^T$, where the translation of conditions is defined as follows:

$$
\begin{aligned}
P(t_1, t_2)^T &= t_1^T \; P \; t_2^T \; . \\
(t_1 = t_2)^T &= t_1^T \; \texttt{owl:sameAs} \; t_2^T \; . \\
(t_1 \circ t_2)^T &= \texttt{FILTER} \; (\; t_1^T \circ t_2^T \;) \\
&\qquad \text{for all } \circ \in \{<, \leq, >, \geq\} \\
(\texttt{not}\; d)^T &= \texttt{FILTER} \; (\; \texttt{!EXISTS} \; \{ \; d^T \; \} \;) \\
(d_1 \; \texttt{or} \; d_2)^T &= \{ \; d_1^T \; \} \; \texttt{UNION} \; \{ \; d_2^T \; \}
\end{aligned}
$$

The DUDES 5 yields the following ASK query since there are no projection variables:

ASK WHERE { dbr:Goofy dbo:creator dbr:Walt_Disney . }

Note that only two-place predicates are translated into triples; predicates of other arities are currently ignored.

Terms are translated straightforwardly. Constants in our case are URIs, i.e. they are used as they are: $c^T = c$. Variables are implemented as integers i, thus they need to be converted into variable strings valid in SPARQL: $i^T = \texttt{?v}i$ (e.g. $1^T = \texttt{?v1}$).

2.4 Factor Graphs

In this section, we introduce the concept of Conditional Random Fields (CRF) (Lafferty et al., 2001) and factor graphs (Sutton and McCallum, 2012). The rest of this section is based on the content provided by ter Horst et al. (2018). An important aspect in modelling an NLP problem with sequences is to consider the multiple variables with dependencies to predict an output. Part-of-speech (POS) tagging or named entity recognition (NER) are NLP problems that can be modelled as a sequence prediction task. They are also called sequence-to-sequence problems since such tasks require predicting an output vector $\vec{y}$ on the basis of the input vector $\vec{x}$. In NLP, the input vector **x** can be tokens in a given document. The output vector **y** can be thought of a POS tag $y_s \in y$ for a word in the position **s**. Such tasks are sometimes called *multivariate* prediction. These problems can be modelled via a conditional distribution of the following form:

$$p(\vec{y}|\vec{x};\theta),$$

where the probability of the output is conditioned on the input vector $\vec{x}$ and parametrised by some vector θ. This can be defined as follows:

$$\vec{y^*} = \underset{\vec{y}}{\operatorname{argmax}}\, p(\vec{y}|\vec{x};\theta) \tag{2.1}$$

The complexity of a model varies in regard to the observed variables and the output variables. The output variables could represent complex structures such as parse trees or graphs where there might be complex dependencies.

Graphical models are frameworks for representing multivariate probability distributions for problems such as those described above. An important insight of using graphical models is that a distribution over many variables can be represented as product of some **local functions**. These local functions have a scope on a subset of variables. Representing joint probabilities over many variables can be computationally intractable. Such factorization of a problem makes the computation manageable and efficient. These local functions are called

factors. Thus, the name **Factor Graphs** essentially stands for probabilistic graphical model with factors. These factors are parameterised with subsets of $\vec{y}_i \subseteq \vec{y}$ and $\vec{x}_i \subseteq \vec{x}$.

Conditional Random Fields (CRF) are widely used for sequence-to-sequence problems because they can model such conditional probabilities via factors. A factor graph $\mathcal{G} = (V, E, \mathcal{F})$ is a bipartite graph composed of random variables V, edges E and a set of factors $\mathcal{F}$. Each factor $\Psi_j \in F$ represents a function: $\Psi_j : V_j \rightarrow \mathbb{R}_{\geq 0}$ that is parameterised with v_j and returns a non-negative scalar score indicating the compatibility of variables in v_j. Further, an edge $e_j \in E$ is defined as a tuple: $e_j = \langle V_j, \Psi_j \rangle$. An important aspect is that CRFs assume $\vec{x}$ as fully observed and thus do not model statistical interdependence in $\vec{x}$.

We illustrate this definition with an example provided in Figure 2.17. The factor graph contains a set of random variables $V = \{A, B, C\}$ and a set of factors $\mathcal{F} = \{\Psi_1, \Psi_2, \Psi_3\}$, denoted by black boxes. In this example each factor evaluates assignment between two random variables. The probability distribution p over these random variables and factors can be written as

$$p(A, B, C) = \frac{1}{Z}\Psi_1(A, B) \cdot \Psi_2(B, C) \cdot \Psi_3(C, A) \tag{2.2}$$

where Z is a partition function that normalises each distribution over all possible assignments.

$$Z = \sum_{a \in A, b \in B, c \in C} \Psi_1(a, b) \cdot \Psi_2(b, c) \cdot \Psi_3(c, a) \tag{2.3}$$

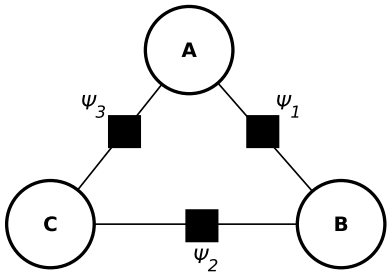


FIGURE 2.17: An example factor graph over 3 random variables $V = \{A, B, C\}$ and factors $\mathcal{F} = \{\Psi_1, \Psi_2, \Psi_3\}$ (black boxes)

Let's assume that each random variable takes two binary values where $A = \{a_1, a_2\}$, $B = \{b_1, b_2\}$ and $C = \{c_1, c_2\}$, and each factor Ψ_i computes a score that reflects the compatibility of two variables. Sample values for these assignments are given in Table 2.3.

TABLE 2.3: Factor values between with two variables. Values for the specific assignment $A = a_2, B = b_1, C = c_1$ are highlighted.

A	B	Ψ_1	B	C	Ψ_2	C	A	Ψ_3
a_1	b_1	4	b_1	c_1	**6**	c_1	a_1	2
a_1	b_2	1	b_1	c_2	2	c_1	a_2	**3**
a_2	b_1	**12**	b_2	c_1	0	c_2	a_1	4
a_2	b_2	1	b_2	c_2	3	c_2	a_2	7

To compute the probability of an assignment $y = \{A = a_2, B = b_1, C = c_1\}$ we apply Equation 2.2 as follows:

$$p(A = a_2, B = b_1, C = c_1) = \frac{1}{Z}\Psi_1(a_2, b_1) \cdot \Psi_2(b_1, c_1) \cdot \Psi_3(c_1, a_2) = \frac{1}{Z}(12 * 6 * 3) = \frac{1}{Z}(216)$$

where

$$\begin{aligned} Z &= \Psi_1(a_1, b_1) \cdot \Psi_2(b_1, c_1) \cdot \Psi_3(c_1, a_1) \\ &+ \Psi_1(a_1, b_1) \cdot \Psi_2(b_1, c_1) \cdot \Psi_3(c_1, a_2) \\ &+ \ldots \\ &+ \Psi_1(a_2, b_2) \cdot \Psi_2(b_2, c_2) \cdot \Psi_3(c_2, a_2) \\ &= 3168 \end{aligned} \tag{2.4}$$

So that the probability is calculated as: $p(A = a_2, B = b_1, C = c_1) = \frac{216}{3168} = 0.068$.

Throughout the years, researchers working on NLP have focused on building two types of graphical models: *generative* and *discriminative*. Generative models focus on computing a joint probability distribution $p(X = x, Y = y)$ whereas discriminative models focus on a conditional probability distribution $p(\vec{y}|\vec{x};\theta)$ over inputs $\vec{x}$ and outputs $\vec{y}$. Hidden Markov Models (HMM) (Rabiner, 1989) is an example of generative models where the dependency to an output variable y_t is restricted by the previous output variable y_{t-1} and the input variable x_t. Conditional Random Fields (CRF) is an example of discriminative models where the interdependencies between variables are not restricted as above. Both models have been applied to various sequence prediction problems in NLP. The main difference between two models is that generative models describe how an output label y generates input feature vector x whereas discriminative models describe in the reverse order how to a set of feature vectors x get assigned an output label y.

Both generative and discriminative models can be modeled as factor graphs for linear and non-linear architectures. Some NLP problems require a non-linear model where interdependencies between variables can be complex. In order to generalize factor graphs to linear and non-linear models the probability distribution $p(\vec{y}|\vec{x})$ is changed into

$$p(\vec{y}|\vec{x}) = \frac{1}{Z(\vec{x})} \prod_{\Psi_i \in \mathcal{G}} \Psi_i(\vec{y}_i, \vec{x}_i) \tag{2.5}$$

where the normalization function $Z(\vec{x})$ is dependent on input vector $\vec{x}$ and factors have access to all variables in $\vec{y}$. Input vectors $\vec{x}$ are considered as *observed* variables and factors assign *hidden* variables that were not observed from the input. These hidden variables are also called *output* variables where the model defines what sort of assignments to certain set of input variables would lead into the prediction of output labels $\vec{y}$.

A factor Ψ_i connects subsets of observed variables x_i and hidden variables y_i and computes a scalar score based on the exponential of the scalar product of a feature vector $f_i(x_i, y_i)$ and a set of parameters θ_i: $\Psi_i = e^{f_i(x_i, y_i) \cdot \theta_i}$. The computation of the conditional probability can now be defined as:

$$p(\vec{y}|\vec{x}) = \frac{1}{Z(\vec{x})} \prod_{\Psi_i \in \mathcal{G}} e^{f_i(\vec{x}_i, \vec{y}_i) \cdot \theta_i} \tag{2.6}$$

For a given set of observed variables, we generate a factor graph automatically making use of factor templates $\mathcal{T}$. Such templates allow for sharing parameters and grouping factors that define a common pattern. A template $T_j \in \mathcal{T}$ defines the subsets of observed and hidden variables $(\vec{x}, \vec{y})$ for which it can generate factors and a function $f_j(\vec{x}, \vec{y})$ to generate features for these variables. All factors generated by a given template T_j share the same parameters

θ_j. With this definition, we can reformulate the conditional probability as follows:

$$p(\vec{y}|\vec{x};\theta) = \frac{1}{Z(\vec{x})} \prod_{T_j \in \mathcal{T}} \prod_{(\vec{x},\vec{y}) \in T_j} e^{f_j(\vec{x},\vec{y}) \cdot \theta_j} \tag{2.7}$$

2.4.1 Inference and Learning

Computing the probability distribution over all possible variables results in exponential number of possibilities since $Z(\vec{x})$ which sums up over an exponential number of possible assignments to the variables $Y_1, ..., Y_n$. Approximative inference algorithms are used to To reduce exponential computation in inferencing for probabilistic graphical models. Markov Chain Monte Carlo (MCMC) is such an algorithm that iteratively generates stochastic samples from a joint distribution $p(\vec{y})$ to approximate the posterior distribution. Samples are selected probabilistically from a state space Y that contains (all) possible variable assignments (*state*) for $\vec{y}$. MCMC constructs a path (Markov Chain) starting from a given initial *state* to the expected *state* by walking through the state space. The method performs number of inference steps where at each step a sampled *state* is drawn from all possible states given the current state. The algorithm converges to the distribution of interest given a sufficient number of steps, which means the distribution of states within the chain approximates the marginal probability distribution of $p(y_i)$ for all $y_i \in \vec{y}$. The drawback of this method is the unknown number of steps to perform in order to converge.

Inference Metropolis–Hastings (Hastings, 1970; Metropolis et al., 1953) algorithm is a Markov Chain Monte Carlo (MCMC) method for efficiently computing a sequence of random samples from a probability distribution. In Metropolis–Hastings, new samples are drawn from a probability distribution $\mathcal{Q}$. At each sampling step the algorithm draws a new sample y' that is based on the previous sample y. If $\mathcal{P}$ is proportional to the desired distribution p, then, with sufficient samples, the Markov Chain will approximate the desired distribution by using a stochastically-based accept/reject strategy. The pseudo-code for Metropolis–Hastings is presented below.

Algorithm 1 Pseudo-code for Metropolis–Hastings Sampling

```
y_0 ← random sample
t ← 1
repeat
    y' ~ Q(y'|y_t)
    α ← acceptanceRatio(y', y_t)
    if α ≥ rand[0,1] then
        y_(t+1) ← y'
    else
        y_(t+1) ← y
    end if
    t ← t + 1
until convergence
```

The function acceptanceRatio$(\cdot,\cdot)$ calculates a score for accepting the new sampled state as the next state. This function computes the score by dividing the probability of the new sampled state to the current state.

$$\text{acceptanceRatio}(y', y) = \frac{f(y')}{f(y)}, \tag{2.8}$$

where $f(y)$ is a function that is proportional to the real density $p(y)$. Note that, if $f(y') \geq f(y)$, the new state y' will be always accepted as the resulting score is greater than 1. Otherwise, the new sampled state will be rejected.

Candidate State Generation Candidate states are generated by performing an *atomic change* to the current state. This sampling procedure generates all possible states that can be reached by making an **atomic** change. Let $\Omega(\vec{y})$ be the set of states that can be generated from $\vec{y}$ by applying one atomic change operation to $\vec{y}$, then the probability distribution $\mathcal{Q}$ can be described as:

$$\mathcal{Q}(\vec{y}', \vec{y}) = \begin{cases} q(\vec{y}') & \text{iff } \vec{y}' \in \Omega(\vec{y}) \\ 0 & \text{else} \end{cases}, \tag{2.9}$$

where

$$q(\vec{y}') = \frac{f(\vec{y}')}{\sum_{\hat{\vec{y}} \in \Omega(\vec{y})} f(\hat{\vec{y}})}. \tag{2.10}$$

Parameter Learning The learning problem consists of finding the optimal weight vector θ that maximizes the a-posteriori probability $p(\vec{y}|\vec{x};\theta)$. The parameter learning relies on a ranking objective that attempts to update the parameter vector by assigning a higher likelihood to preferred solutions.

SampleRank (Wick et al., 2009) is an online algorithm that learns to prefer sampled states to overcome the expensive computational costs that arise during inference. The parameter update is based on gradient descent on pairs of states $(\vec{y}^t, \vec{y}^{(t+1)})$ consisting of the current state $\vec{y}^t$ and the next state $\vec{y}^{(t+1)}$. Two states are compared according to the following objective preference function $\mathbb{P} : Y \times Y \rightarrow \{false, true\}$:

$$\mathbb{P}(\vec{y}, \vec{y}') = \mathbb{O}(\vec{y}') > \mathbb{O}(\vec{y}) \tag{2.11}$$

where $\mathbb{O}(\vec{y})$ is an objective function that returns a score indicating the degree of agreement with the ground truth of a certain input document. $\mathbb{Q} : Y \times Y \rightarrow [0,1]$ denotes the proposal distribution that is provided by the model, $\phi : Y \times X \rightarrow \mathcal{R}^{|\theta|}$ denotes the sufficient statistics of a specific variable assignment and:

$$\text{accept}(y, y') = p(y') > p(y) \tag{2.12}$$

if the sampled state y' has a higher probability than the current state y. The pseudo-code for the SampleRank algorithm is given below.

2.5 Neural Networks

In this section, we provide an overview on neural networks. Neural networks (NN) are computing systems inspired by the biological "neurons" which are intended to replicate the way humans learn. They are sometimes also called *artificial neural networks*. NN have gained popularity in the last decade with successful applications (Krizhevsky et al., 2012, Srivastava et al., 2014, Rastegari et al., 2016) on image recognition datasets such as ImageNet (Deng et al., 2009). Applications in NLP (Bahdanau et al., 2014, Cho et al., 2014, Sutskever et al., 2014a) has also proven the importance of NN specifically for sequence-to-sequence problems such as POS-tagging, machine translation, etc..

NN consist of input and output layers, as well as hidden layers consisting of units that transform the input into something that the output layer can use. A sample network is shown in Figure 2.18. The input data is fed into the input layer with 4 neurons that transform the

Algorithm 2 Pseudo-code for Sample Rank

Inputs: training data D
Initialization: set $\theta \leftarrow \vec{0}$, set $y \leftarrow y_0 \in Y$
Output: parameter θ
repeat
  $y' \sim \mathbb{Q}(\cdot|y)$
  $\Delta \leftarrow \phi(y', x) - \phi(y, x)$
  if $\theta \cdot \Delta > 0 \wedge \mathbb{P}(y, y')$ **then**
    $\theta \leftarrow \theta - \eta\Delta$
  else if $\theta \cdot \Delta \leq 0 \wedge \mathbb{P}(y', y)$ **then**
    $\theta \leftarrow \theta + \eta\Delta$
  end if
  if accept(y', y) **then**
    $y \leftarrow y'$
  end if
until convergence

data into vector representations and forward it into the hidden layer. The output from the hidden layer with 5 neurons is then forwarded into the output layer. The hidden layer can be composed of multiple layers stacked into each other with varying number of neurons. Each such hidden layer has a weight matrix that is optimized in supervised learning. The term "deep learning" in fact refers to the neural network with multiple layers. Generally, such architectures can learn to perform tasks based on training examples. For instance, the image recognition task involves labeling images as "cat" or "dog" given the input image. The input image is then transformed into vector representation and the output data can be modeled as probability distribution over target classes (cat, dog, etc.).

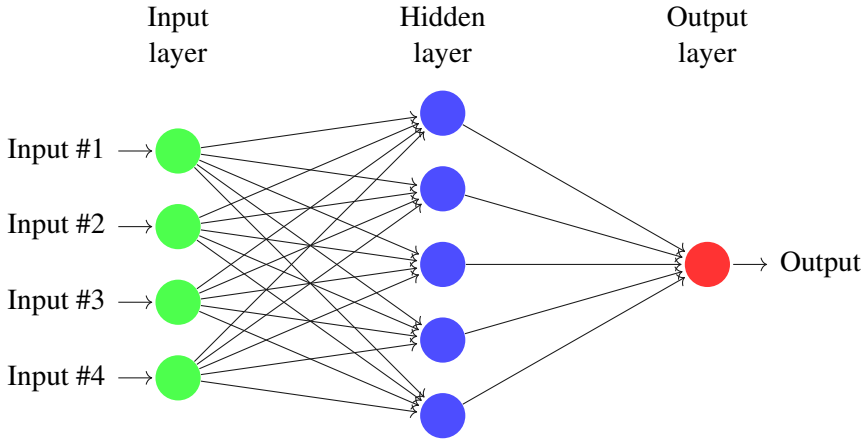


FIGURE 2.18: Neural network depicted with input, hidden and output layers

There are multiple types of neural networks, each of which has its own specific use cases and levels of complexity. The most common and basic type is called a Feed-forward Neural Network, in which the data is fed in forward fashion. The network shown in Figure 2.18 is a feed-forward network where data flow is from input to output as depicted by arrows. The output vector is computed as follows:

$$y = Wx + b \tag{2.13}$$

where W stands for weights learned by the network, x is the input vector and b is a bias term.

Recurrent Neural Networks Another common type of neural networks are called Recurrent Neural Networks (RNN). RNNs form a directed graph between connected nodes. RNNs have an internal state memory to process sequence of inputs. The idea behind RNNs is to make use of sequential information from the input. For example, in speech recognition the prediction of the next word in a sentence depends on previous words since a sentence follows a certain coherence and context. RNNs are called *recurrent* because they perform the same task for every element in a sequence where the output from the previous element is also used in computation. RNNs have an internal state "memory" that captures computed information. In Figure 2.19, we show a RNN diagram with compressed and unfolded versions. Each node (neuron) has a time-varying real-valued activation and each connection has a trainable real-valued weight. Nodes are either input nodes (x), output nodes (y), or hidden nodes (h).

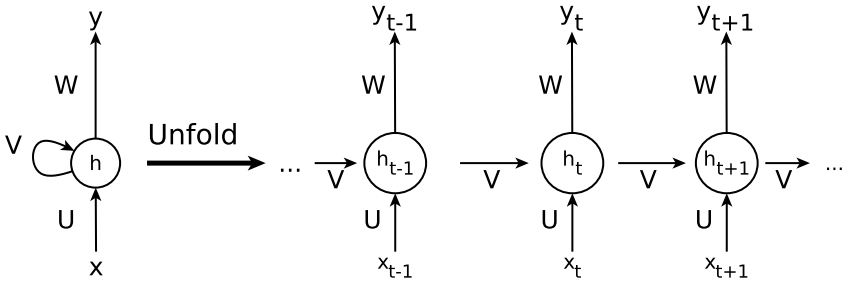


FIGURE 2.19: Recurrent Neural Network (RNN) diagram with compressed (left) and unfolded (right) versions.

Here x_{t-1}, x_t and x_{t+1} are inputs, h_{t-1}, h_t and h_{t+1} are hidden states and y_{t-1}, y_t and y_{t+1} are output states at time steps $t-1$, t and $t+1$ respectively. The computation of a hidden state h_t at time step t is calculated based on the previous hidden state h_{t-1}, the input state x_t and bias term b.

$$h_t = f(Ux_t + Vh_{t-1} + b) \tag{2.14}$$

where U and V are trainable weight vectors and f is an activation function, which can be linear or non-linear functions. Usually, non-linear functions such as *hyperbolic tangent (tanh)* or *Rectified Linear Unit (ReLU)* are widely used. The hidden states (h_{t-1}, h_t and h_{t+1}) act as internal "memory" of the network. They capture information about previous time steps. However, in practice RNNs can not capture long inner-dependencies between nodes.

The output states are computed based on only hidden states, e.g. the output state y_t at time step t is calculated based on the hidden state h_t as follows:

$$y_t = g(Wh_t) \tag{2.15}$$

The function g is another activation function, which is chosen based on the tackled problem. For instance, in classification problems for multiple labels it is chosen as *softmax* and *sigmoid* for binary labels.

RNN share the weight vectors U, W and V across all time steps. This reduces the total number of parameters for optimization. For supervised learning problems RNNs have been applied to sequence-to-sequence, e.g. POS-Tagging, NER, problems in NLP. Training such systems on RNNs requires a method called **back-propagation** to optimize the weight vectors. Back-propagation is essentially used by optimization algorithms such as stochastic

gradient descent (SGD) to adjust weights of neurons by calculating the gradient of a loss function. The errors computed by loss functions are back-propagated in each neuron and the weights are adjusted according to the occurred errors. The training process continues until the network outputs the expected results or until it reaches a certain stopping criterion, e.g. number of iterations over the training examples.

RNNs have a major drawback called **vanishing gradient problem**. It occurs during training when the same weights are used in all time steps as well as in back-propagation. The problems occurs in cases where the gradient will be very small number and it prevents the weights from changing. Activation functions such as the hyperbolic tangent have gradients in range (0,1) and the back-propagation computes the gradients by chain rule. A network with n layers results in the "front" layers converging very slowly since the gradient decreases exponentially with n. The network experiences difficulty in memorizing information from previous time steps.

Long Short-term Memory Hochreiter and Schmidhuber (1997) introduced Long Short-term Memory (LSTM) units as a special kind of RNNs that are capable of learning long-term dependencies between time steps. LSTM units can be used in an RNN that is composed of a cell, input gate, output gate and forget gate. The cell is responsible for memorizing values over different time steps. Each of three gates controls the proportions of information to forget and pass on to the next time step. A standard RNN unit is composed of a single layer. These layers apply an activation function and compute the hidden state. LSTM units have 4 layers: input, forget, output gates and cell memory. Cell memory stores a value for either long or short periods. We give an LSTM unit diagram below in Figure 2.20.

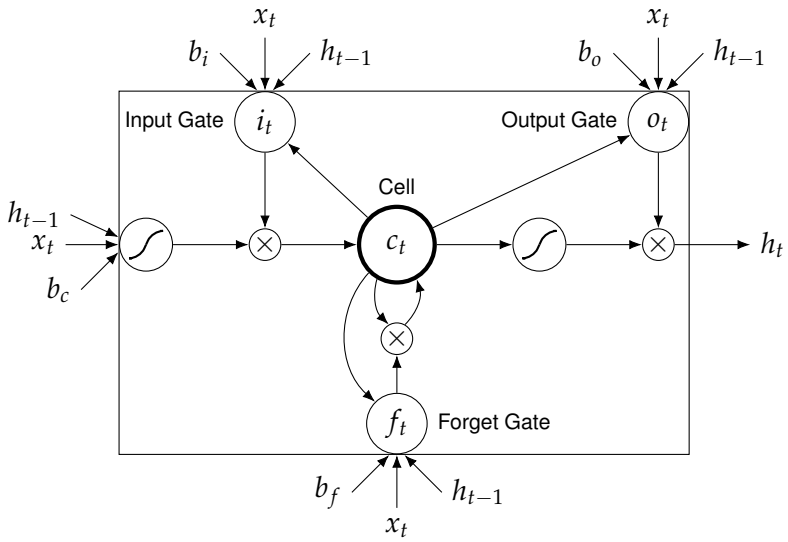


FIGURE 2.20: Long short-term memory (LSTM) diagram with input, forget, output gates and cell memory.

As shown in the diagram above in Figure 2.20, the LSTM unit takes as input the current input state x_t, the previous hidden state h_{t-1} and a bias terms b_o (output gate), b_f (forget gate), b_i (input gate) and b_c (cell memory). The unit outputs the hidden state h_t. Next, we give the equations to compute the hidden state and the cell memory.

$$
\begin{aligned}
i_t &= \sigma(W_i h_{t-1} + U_i x_t + b_i) \\
f_t &= \sigma(W_f h_{t-1} + U_f x_t + b_f) \\
\tilde{c}_t &= tanh(W_c h_{t-1} + U_c x_t + b_c) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\
o_t &= \sigma(W_o h_{t-1} + U_o x_t + b_o) \\
h_t &= o_t \odot tanh(c_t)
\end{aligned} \tag{2.16}
$$

where σ is a sigmoid activation function and $\odot$ is element wise product. Below we describe each variable used in Equation 2.16. The subscripts t or $t-1$ refer to the time steps.

- x_t: input state (vector)
- h_{t-1}: previous hidden state
- f_t: forget gate's activator vector
- i_t: input gate's activator vector
- o_t: output gate's activator vector
- c_t: cell memory
- h_t: hidden state of the LSTM unit
- U_i, U_f, U_o, U_c: weight matrices for different gates for input x_t
- b_i, b_f, b_o, b_c: bias vectors for different gates for input x_t
- W_i, W_f, W_o, W_c: weight matrices for hidden state h_t

Bidirectional RNNs Over the years researchers have developed different neural network architectures. Bidirectional RNNs (BiRNN) are based on the idea that the hidden state h_t at time step t does not only depend on the previous hidden states but also the future hidden states. The basic idea is to stack two RNN layers where one process the input as a standard RNN in forward direction and the other one processes the input in backward direction. In Figure 2.21, we show two RNN layers that process the input in forward and backward directions. Each layer computes the hidden states as any RNN layer. The hidden states of each layer (forward and backward) are used to compute the final hidden states of BiRNN. As mentioned before, LSTM units can be used in any RNN without getting affected by the vanishing gradient problem. Thus, the depicted diagram can be converted into Bidirectional LSTM (BiLSTM) (Graves et al., 2013). Such neural network architectures are capable of capturing dependencies between input sequences in both directions. For example, in NLP to predict a missing word in a sequence we can look at both the left and the right context of that word using BiLSTMs.

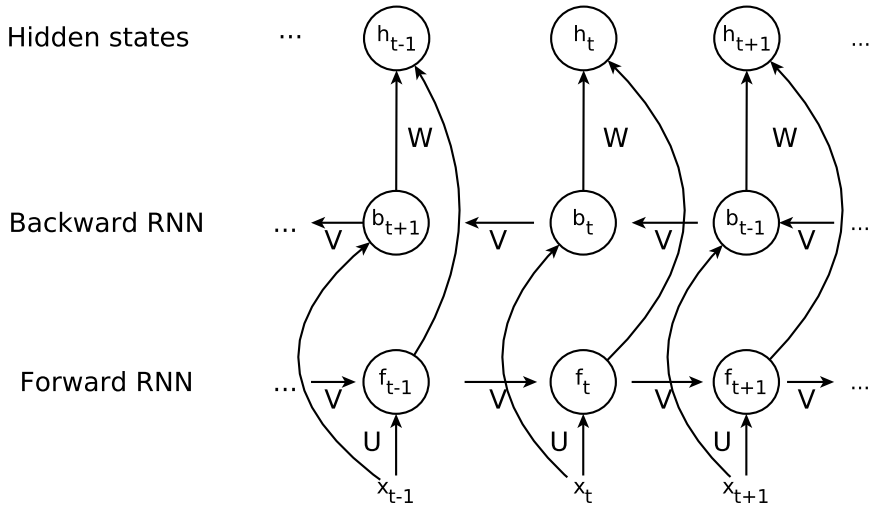


FIGURE 2.21: Bidirectional Recurrent Neural Network (BiRNN) diagram with forward and backward layers.

Convolutional Neural Networks Convolutional Neural Networks (CNN) (LeCun et al., 1998) are another type of neural networks that became popular with applications on computer vision. CNNs are feed-forward networks composed of one or multiple convolutional layers, pooling layers, fully connected layers.

Convolutional Layer: Convolutional layers apply a convolutional operation on the input, then pass the result into the next layer. These convolutional operations can be thought of different *filters*. In computer vision, these filters can extract a certain information about a region in an image such as detecting edges, etc. These layers extract local features around the window of the filter applied on a sequence. Usually, convolutional layers extract higher level features. Convolutional layers have a defined window size that is slided over the input sequences. Each application results in local features.

Pooling Layer: Pooling layers take an input from convolutional layers and applies pooling operation that selects some value and passed to the next layer. The pooling layer combines local features extract by convolutional layer. Pooling operations such as *max* or *average* are widely used. They are sometimes also called subsampling since they pick a sample from possible values.

Fully Connected Layer: Fully connected layers connect every neuron in a layer to every neuron in the next layer.

In Figure 2.22, we show a sample application of CNNs on image processing. This CNN has 2 convolutional and 2 pooling layers followed by a single fully connected layer. The input image is fed forward into convolutional layers. Next, the pooling operation is performed by selecting samples from outputs of convolutional operations. Another stack of convolutional and pooling layers are applied to get a fully connected layer. Finally, the output is computed based on the vectors in fully connected layer.

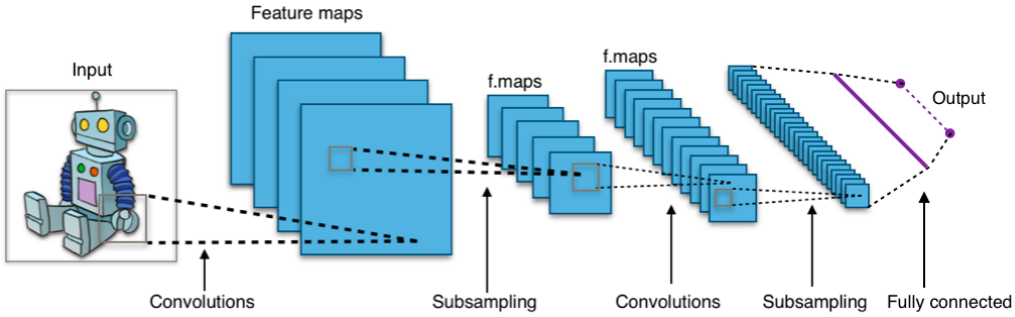


FIGURE 2.22: Convolutional Recurrent Neural Network (CNN) example applied for image processing taken from https://en.wikipedia.org/wiki/Convolutional_neural_network.

Chapter 3

Datasets

In this chapter, we introduce publicly available datasets from the research community that researchers have used to evaluate Question Answering (QA) approaches. We give an overview about each dataset along with comparisons with each other.

The datasets to evaluate semantic parsing approaches on QA can be divided into two main groups: closed-domain and open-domain. We introduce the Geoquery (Tang and Mooney, 2001) dataset from closed-domain, the SimpleQuestions (Bordes et al., 2015) and the QALD (Cimiano et al., 2013; Dragoni et al., 2017; Unger et al., 2014, 2016; Usbeck et al., 2017) from open-domain datasets. Finally, we compare the lexical overlap between train and test splits of each dataset. The overlap is calculated by taking unique tokens in train and test splits and comparing how many of the tokens in test also appear in train. It provides an overview about the lexical gap in each dataset since the higher number of tokens in the test split that do not appear in the train split points to a challenge for QA systems known as "lexical gap".

3.1 Closed-domain

Closed-domain datasets include questions about a specific domain. The vocabulary used in such datasets are limited. It means that the same vocabulary is used in the train and the test splits of a dataset. The systems trained on these datasets have less lexical gap to cope with compared to open-domain datasets. It is due to the fact that systems encounter the same vocabulary in prediction time as well as during training. Next, we describe the well-known closed-domain dataset called Geoquery.

3.1.1 Geoquery

The dataset has questions about US geography (Tang and Mooney, 2001) with information about rivers, states, mountains, etc[1]. Each instance in the dataset is comprised of a question and a meaning representation pair. The meaning representation includes unary and binary predicates as well as individuals. The meaning representation is expressed in Prolog format and converts into an executable query. Sample question and the corresponding query pairs are given below.

Question: *Give me the states that border Texas.*

Query: A,(state(A),next_to(A,B),const(B,stateid(texas)))

Question: *How long is the longest river in California?*

Query: A,(len(B,A),longest(B, river(B),loc(B,C),const(C,stateid(california)))))

[1] https://www.cs.utexas.edu/users/ml/nldata/geoquery.html

Question: *What state has the longest river?*

Query: A,(state(A),loc(B,A),longest(B,river(B)))

Zettlemoyer and Collins (2005) converted the meaning representation of questions into lambda calculus format (see Section 2.3.3). The dataset is split into 600 instances for training and 280 instances for testing. In total, the dataset contains 880 instances. An example with the question and the lambda calculus representation is shown below.

Question: *Give me all states that border Texas ?*

Lambda calculus expression: $\lambda x.\texttt{state}(x) \wedge \texttt{next\_to}(x, \texttt{state:Texas})$

3.2 Open-domain

Open-domain datasets include questions about general factual knowledge such as encyclopedia. The questions in these datasets cover many topics unlike closed-domain datasets. Below we give more details on two open-domain datasets: QALD and SimpleQuestions.

3.2.1 QALD

Question Answering over Linked Data (QALD) (Cimiano et al., 2013; Unger et al., 2014, 2016) is a series of evaluation campaigns on question answering over Linked Data.[2] These evaluation campaigns were proposed as a challenge to benchmark QA systems based on structured data such as DBpedia. The challenge has been organized eight times as of July, 2018. The challenge include datasets for each time the evaluation campaign has been organized. QALD datasets contain questions in different languages and their corresponding meaning representation is a SPARQL query. Available languages are: English (EN), German (DE), Spanish (ES), Italian (IT), French (FR), Dutch (NL), Romanian (RO), Persian (FA), Hindi (HI). QALD questions comprise of various topics with aggregation, fact checking, ASK queries, or normal WH-questions. Additionally, QALD datasets include multiple tasks that are interesting for the research community focused on QA. These tasks are Hybrid QA, Multilingual QA, Large-scale QA, Biomedical QA. In this thesis, we focus on Multilingual QA task.

A sample instance from the QALD-6 training dataset in English, German and Spanish is given below with its respective SPARQL query.

```
"id": "141",
"answertype": "resource",
"aggregation": "false",
"onlydbo": "true",
"hybrid": "false",
"question": [
       {
        "language": "en",
        "string": "Who founded Intel?",
         "keywords": "Intel, founded"
        },
        {
        "language": "de",
```

[2] http://www.sc.cit-ec.uni-bielefeld.de/qald

```
        "string": "Wer hat Intel gegründet?",
        "keywords": "Intel, gründen"
        },
        {
        "language": "es",
        "string": "¿Quién fundó Intel?",
        "keywords": "Intel, fundación"
        }
    ],
    "query": {
    "sparql": "
          PREFIX dbo: <http://dbpedia.org/ontology/>
          REFIX dbr: <http://dbpedia.org/resource/>
          SELECT DISTINCT ?uri WHERE {
              dbr:Intel dbo:foundedBy ?uri .
          } "
     }
```

In Table 3.1, we summarize all datasets in QALD benchmarks with information about the available languages, tasks, number of training and test instances and the knowledge base that the questions are based on.

TABLE 3.1: QALD 1-8 benchmark datasets with the available languages, tasks, number of training and test instances and the knowledge base that the questions are based on.

Dataset	Knowledge base	Languages	Task	Train	Test
QALD-1	DBpedia 3.6	EN	QA	100	50
QALD-1	MusicBrainz	EN	QA	50	50
QALD-2	DBpedia 3.7	EN	QA	100	100
QALD-2	MusicBrainz	EN	QA	100	55
QALD-3	DBpedia 3.8	EN	QA	100	100
QALD-3	DBpedia 3.8	ES	QA	50	50
QALD-3	MusicBrainz	EN	QA	100	55
QALD-4	DBpedia 3.9	EN, DE, ES, IT, FR, NL, RO	Multilingual QA	200	50
QALD-4	SIDER, Diseasome, Drugbank	EN	Biomedical QA	25	25
QALD-4	DBpedia 3.9	EN	Hybrid QA	25	25
QALD-5	DBpedia 2014	EN, DE, ES, IT, FR, NL, RO	Multilingual QA	300	50
QALD-5	DBpedia 2014	EN	Hybrid QA	40	10
QALD-6	DBpedia 2015	EN, DE, ES, IT, FR, NL, RO, FA	Multilingual QA	350	100
QALD-6	DBpedia 2015	EN	Hybrid QA	50	50
QALD-6	LinkedSpending	EN	Statistical QA	100	25
QALD-7	DBpedia 2016-04	EN, DE, ES, IT, FR, NL, RO, FA	Multilingual QA	215	43
QALD-7	DBpedia 2016-04	EN	Hybrid QA	105	50
QALD-7	DBpedia 2016-04	EN	Large-scale QA	100	2 million
QALD-7	Wikidata 2017-01-09	EN	QA	100	50
QALD-8	DBpedia 2016-10	EN, DE, ES, IT, FR, NL, RO, FA	Multilingual QA	219	41
QALD-8	Wikidata	EN, DE, ES, IT, FR, NL, RO, FA	Multilingual QA	100	41
QALD-8	DBpedia 2016-10	EN	Hybrid QA	100	50

3.2.2 SimpleQuestions

The SimpleQuestions (Bordes et al., 2015)[3] dataset is a collection of simple questions in English where questions are based on a single fact from Freebase. Human annotators were presented a single fact (a triple from Freebase) and were asked to write a question about it. The dataset includes over 100K instances, which makes it the largest dataset for question answering. The distribution of instances for training, validation and test are given in Table 3.2.

[3] https://research.fb.com/downloads/babi/

TABLE 3.2: SimpleQuestions dataset with the number of instances for each split

Split	Number of instances
Train	75,910
Validation	10,845
Test	21,687
Total	108,442

10,843,106 triples were extracted from Freebase and they were presented to human annotators to write questions about them. This subset of triples is referred to as the *Freebase-2M* set. 108,442 instances were created in the process where 75,910 are for training, 10,845 for validation and 21,687 instances are left for testing.

Each instance is composed of a single question and a triple. A sample instance is given below.

Question: What American cartoonist is the creator of Andy Lippincott?

Triple: Andy_Lippincott, character_created_by, Garry_Trudeau

The subject *Andy_Lippincott* and the predicate *character_created_by* are mentioned in the question implicitly or explicitly. The expected answer to this question is the entity on the object position: *Garry_Trudeau*. This example includes human-readable labels for entities. In the original dataset the entities on the subject and the object position are represented with Freebase MIDs (machine readable ids).

Questions in the dataset include wh-words or start with words like *Name*. For instance, "name a town and comune in italy". Most of the questions do not follow a proper capitalization and punctuation unlike the QALD datasets. All instances are mostly about a single fact. The dataset does not include questions that ask about aggregation of certain information or information about a certain date.

3.3 Lexical Overlap

All datasets described above have the same task: given a natural language question return an executable query. The systems that are trained on such datasets need to handle the mapping of natural language expressions appearing in a question text to query expressions. Such a mapping defines the performance of the systems. Tokens needed to do the mapping of natural language expressions to query expressions are not given in any dataset. In order to understand the complexity in mapping we did the following analysis: we compared unique tokens extracted from train and test splits of each dataset. The results are presented in Table 3.3. The overlap between the train and the test split means that a token exists in both. We chose the QALD-6 dataset from QALD benchmark datasets.

TABLE 3.3: Single token analysis for QA datasets, the number of tokens in train & test splits and the number of overlapping tokens in both splits

Dataset	**Type**	**Train**	**Test**	**Overlap**	**Percentage**
GeoQuery	Closed-domain	284	167	167	1.0
SimpleQuestions	Open-domain	50984	21142	11806	0.55
QALD-6	Open-domain	886	350	158	0.45

These results presented in Table 3.3 are based only on the single token overlap. We can see that QALD-6 has the lowest overlap of 0.45 between train and test tokens. SimpleQuestions is the second dataset with the lowest overlap of 0.55. Finally, Geoquery has full overlap

(1.0). In the next section, we evaluate these results and explain the challenges that systems face for each dataset and how the overlap affects the performance.

3.4 Dataset Complexity

The results presented in Table 3.3 suggest that the systems trained on QALD-6 would face more challenges with lexical knowledge since most of the tokens would not be seen during training. Even though the SimpleQuestions dataset introduce a simpler task compared to QALD-6, it still has the lexical gap (0.55). Geoquery, as expected, has a full overlap where the tokens in test split also appear in train. Closed-domain datasets have a limited vocabulary that is used interchangeably. Open-domain datasets cover wider vocabulary from different topics.

Open-domain datasets pose a bigger challenge than closed-domain ones in terms of vocabulary and the search space. QALD-6 and SimpleQuestions are datasets both based on large knowledge bases such as DBpedia and Freebase with millions of entities. Geoquery on the other hand, has a limited vocabulary and it is restricted to the set of entities from US geography with approximately 700 entities.

We can see that the QALD-6 dataset has a gap in lexical knowledge introduced by the **non-overlapping** tokens from test. The overlap for QALD-6 dataset is 0.45. It means that more than half (the remaining 0.55) of the single tokens in test can not be mapped to query expressions using only the vocabulary in the training data. Similar problem exists for the SimpleQuestions dataset where the lexical gap is 0.55.

Additionally, note that QALD-6 dataset includes more complex types of questions such as aggregations, ASK queries, etc. while SimpleQuestions has a single type. QALD-6 task is not limited to a single triple unlike SimpleQuestions. Considering the available number of training instances, the lexical gap, question complexities, different query templates we can conclude that the QALD-6 is the most complex dataset among these three. The next complex dataset is the SimpleQuestions with higher lexical overlap while the dataset includes over 70K training instances. The Geoquery is the dataset with the lowest complexity with limited vocabulary that appears in both train and test splits.

Chapter 4

Related Work

In this chapter, we describe previously published work on Question Answering. We provide a detailed overview on research done in semantic parsing along with comparisons to the state-of-the-systems evaluated on Geoquery, QALD and SimpleQuestions datasets.

4.1 Semantic Parsing

There is a substantial body of work on semantic parsing and its application in the task of question answering. In the next sections we compare other published work to our approach as well as discuss the strong points of each approach. We group QA systems under 2 groups based on the datasets they use for evaluation: QALD and SimpleQuestions.

Semantic parsing (SP) is the problem of transforming natural language sentences into a machine-interpretable meaning representation. It is a well studied sub-field of NLP. Early semantic parsing approaches such as Winograd (1971, 1972) and Woods et al. (1972) used patterns for understanding the content of the input sentences. These approaches were followed by statistical ones that learn the statistics from training data rather than defining rules manually. Zelle and Mooney (1996) defined the first semantic parsing approach that was based on a statistical model. The approach learns control rules from the provided input sentences paired with desired parses. The learning algorithm is Inductive Logic Programming (ILP). The output from the learning process is a shift-reduce parser that converts input sentences into parses using the learned control rules. The parses are essentially executable queries on the target domain, e.g. US geography. The approach was called CHILL. The dataset Geoquery (Tang and Mooney, 2001) was later introduced with 880 questions paired with Prolog format queries.[1] This approach was later followed by SILT (Kate et al., 2005) that uses transformation rules to associate natural language patterns with templates drawn from partial meaning representations.

SCISSOR (Ge and Mooney, 2005) is an approach based on syntactic parse trees as syntax and uses compositional semantics to obtain the meaning representation of an utterance. This approach bears similarities to our proposed approach discussed in Chapter 7 in a way that we also use syntactic trees for syntax and compositional semantics to obtain the meaning representation.

Semantic parsing approaches can be formulated as machine translation (MT) systems. MT systems take an input utterance and transform it into the target language. The target language can be formulated as a another representation such as queries. WASP (Wong and Mooney, 2006) is a MT system adapted to the semantic parsing problem using context-free grammars. It uses the word alignment technique used in MT systems to find better transformation rules introduced than SILT.

KRISP (Kate and Mooney, 2006) is an approach based on string-kernel-based classifiers. The approach learns pairs of sub-strings from the given utterance and the respective meaning

[1] http://www.cs.utexas.edu/users/ml/geo.html

representation. They use SVM string-kernels that measure the common subsequences that two strings share. The semantic parser gives as output the highest-ranking semantic parse based on probabilities.

ZC05 (Zettlemoyer and Collins, 2005) is the first semantic parser that relies on CCG for syntax and lambda calculus for semantics. The approach is based on training a probabilistic semantic parser from input training data, e.g. Geoquery. The training algorithm has a function called GENLEX that takes an input utterance and the respective lambda calculus expression as logical form. GENLEX generates lexical items by pairing n-grams from the utterance with smaller units of lambda calculus expressions, e.g. individuals such as *state:Texas*, binary predicates *state_to* with variables only. Each lexical item has a CCG category assigned to it. The algorithm applies compositional semantics to generate all possible parses that can be obtained using lexical items generated by GENLEX. Additionally, some domain-independent lexical items are needed to be defined manually for tokens such as WH-words, determiners etc. The approach learns a probabilistic model that assigns a distribution over parses under the learned grammar based on CCG.

Semantic parsing approaches such as the ones explained above are evaluated on the Geoquery dataset and the results are shown in Table 4.1 (Kate and Mooney, 2006). It can be seen that the ZC05 approach achieves higher recall than most systems at the same time keeping the precision high as well. KRISP also achieves high precision but the recall is not as high as ZC05.

Most of these systems learn patterns on how to map natural language sub-strings into smaller units of meaning representation. All systems perform relatively well given a dataset in a restricted domain such as Geoquery.

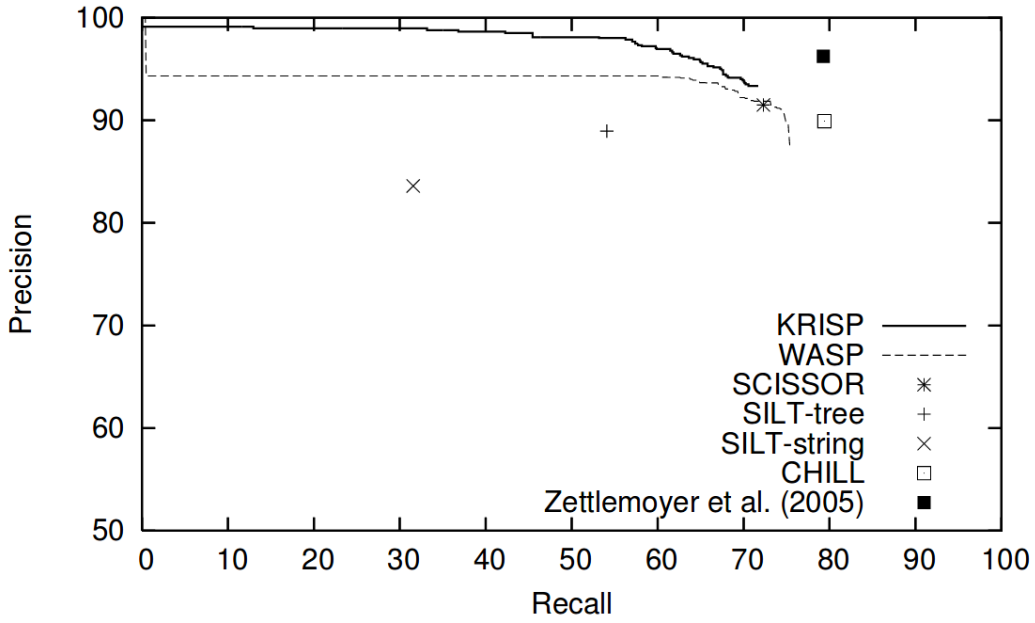


FIGURE 4.1: Evaluation of semantic parsing approaches on Geoquery dataset, taken from KRISP (Kate and Mooney, 2006)

Other statistical approaches that rely on CCG syntax have been studied as well. These approaches differ in the learning methods. UBL (Kwiatkowski et al., 2010) is another approach that uses CCG as syntax and lambda calculus where the method is tailored for multiple languages. The ZC05 is applied only to English and involves manual lexica for covering domain-independent expressions. This approach generalizes better to other languages since

it does not require any hand-engineered lexical items. The approach is based on a splitting notion unlike ZC05 where the approach learns to compose the meaning representation from smaller units.

Given the utterance and the meaning representation, the splitting algorithm extracts some substrings from the utterance, small units of meaning representation and pairs them. These paired lexical items get assigned a CCG category and the logical form. The remaining part of the utterance and the meaning representation is kept as it is. Having these two lexical items, the original training instance can be obtained by applying compositional semantics based on CCG and lambda calculus as in the ZC05 approach. An example is given below. A training instance consists of an utterance paired with CCG category S (sentence) and the semantic representation as shown below.

Training instance : New York borders Vermont - S : *next_to(NY, VT)*
Split part: Vermont - NP : *VT*
Remaining part : New York borders - S/NP : $\lambda x.$*next_to(NY, x)*

From the original sentence, some substrings such as *Vermont* and some small units of semantic representation *VT* is removed and formed into new lexical item. The CCG category of NP is assigned since *VT* stands for an individual. The remaining part gets assigned a CCG category where NP is missing on the right to form a sentence. The reason for missing a CCG category NP on the right is that the word *Vermont* is on the right side of the given utterance. As a result, the remaining part of the sentence *New York borders* gets assigned S/NP. The semantics of this phrase is obtained by replacing *VT* in *next_to(NY, VT)*. As a result, the following lambda expression $\lambda x.$***next_to(NY, x)*** is assigned to the remaining part of the sentence.

Following such splits over the training corpus, the algorithm learns which split pairs fit better using the probabilistic model. Different splits are also possible for the given example, which leads to many different semantic parses obtained from the learned probabilistic model. For efficiency reasons, most systems incorporate inference methods such as beam-search to reduce the search space during interpretation of the natural language utterance. This approach differs from ZC05 in obtaining lexical items. ZC05 generates all possible lexical items and then evaluates each of them based on whether the combination of those lexical items leads to a valid semantic parse. The ZC05 also relies on hand-engineered lexical items for domain-independent expressions, which have to be defined for each language. However, UBL learns how to split utterances and semantic representations in order to obtain the expected semantic parse. It does not need any manually hand-engineered lexica.

It has been shown that statistical approaches (Kwiatkowski et al., 2013; Zettlemoyer and Collins, 2007, 2009) can learn from pairs of natural language utterance and meaning representation in restricted domains. However, generating such datasets requires experts that understand formal languages such as lambda calculus. DCS (Liang et al., 2011) introduced the notion of building semantic parsers without any pairs of natural language question and the meaning representation.

Instead, the approach learns from pairs of natural language question and the answers for that question as an expected output from the semantic parser. The meaning representation of questions is not provided and it is inferred during the parsing process as a hidden variable. The performance of such systems is evaluated with respect to the answers retrieved from the executable query, generated by the semantic parser, rather than comparing meaning representations. Other similar methods (Artzi and Zettlemoyer, 2011; Clarke et al., 2010; Goldwasser and Roth, 2011; Krishnamurthy and Mitchell, 2012) have been proposed as well because it is easier to create datasets by letting annotators write answers for questions by

extracting from some knowledge bases such as Freebase, DBpedia, etc rather than asking them to write queries. These methods have weak supervision on the model where the query to extract answers is a hidden variable.

DCS stands for dependency-based compositional semantics, which is a formal language to define semantic representations. It provides a formalism for defining semantics as tree structures called DCS trees where the semantics is defined between different semantic units (nodes) in terms of relations (edges) between them. An illustration of a DCS tree is shown in Figure 4.2.

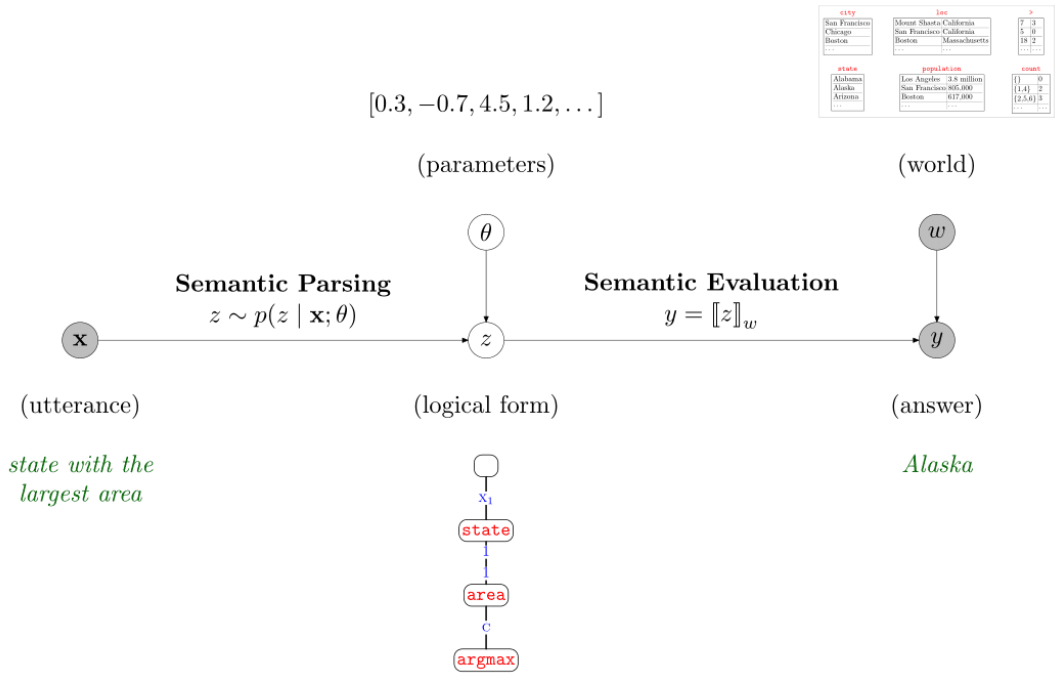


FIGURE 4.2: DCS formalism along with probabilistic model for the given example, taken from Liang et al. (2011)

The utterance and its respective answer (Alaska based on the grounded world with cities, states, etc.) is shown at each end as input and output. The inferred semantic representation (also referred to logical form) of the utterance is shown in the middle of the pipeline with z. The approach learns how to map an utterance to answers by inferring a logical form based on the probabilistic model where parameters are estimated by training on some dataset.

Liang et al. (2011) evaluated the semantic parsing systems on the Geoquery dataset. The results are presented in Table 4.1. Some approaches are evaluated on the basis of returned semantic representation and some evaluated on the basis of answers retrieved from the database after executing the query. It can be seen that the DCS-based semantic parsing approach achieves the highest results in terms of recall compared to other proposed approaches.

TABLE 4.1: Recall values for systems evaluated on Geoquery dataset, taken from Liang et al. (2011)

System	Semantic Representation	Answer
CHILL	79,4	-
SCISSOR	77,5	77,5
SILT	54,1	-
KRISP	71,7	-
WASP	74,8	-
λ-WASP	86,6	-
ZC05	79,3	-
UBL	88,2	88,9
FUBL	**88,6**	88,9
DCS-based (with augmented triggers)	-	**91,4**

A recent focus in the research community shifted towards to learning language-independent meaning representations and building multilingual QA systems. This allows to build approaches that can potentially scale to multiple languages or domains. Reddy et al. (2017) developed such a system called UDEPLAMBDA that maps natural language questions to lambda calculus expressions. The multilinguality aspect of this system is based on using cross-lingual dependency parse trees from Universal Dependencies (UD) (Nivre et al., 2016; Nivre, 2017). The semantic representation of a natural language is based on lambda calculus. This approach bears similarity with our proposed Dependency parse tree-based Semantic Parsing Approach (see Chapter 7). Our method differs from theirs in the way of applying the semantic composition and the way we express the semantics with DUDES. Our semantic composition method does not depend on the order of functional applications, which is a feature of DUDES, whereas their method requires an ordered application since it is based on lambda calculus. Their approach improves upon the previously published system called DEPLAMBDA (Reddy et al., 2016), which was built on dependency parse trees for English only. UDEPLAMBDA uses the syntactic dependencies of words in UD form and introduces enhancements over the dependency parse trees. These enhancements are adding long-dependency relation between words and language-specific set of POS tags that are not part of UD. Our approach Dependency parse tree-based Semantic Parsing Approach does not depend on any additional syntax other than the one provided by UD such as long-dependency relations and additional POS tags.

Additionally, other work has focused on building multilingual semantic representations such as (Evang and Bos, 2016; Vanderwende et al., 2015; White et al., 2016). Vanderwende et al. (2015) developed a system that converts natural language into Abstract Meaning Representation (AMR) (Banarescu et al., 2013) where the aim is to assign the meaning representations to sentences that are similar. They showed the application of their method on German, French, Spanish and Japanese. White et al. (2016) proposed a framework for defining *universal decompositional semantics* with an aim to provide semantic annotations on word senses, semantic roles and event properties across all languages. These semantic annotations provide an understanding of a sentence given the syntax in form of the dependency parse tree. Evang and Bos (2016) proposed to learn semantic parsers based on CCG using the pre-trained parser on a specific language and transforming the parser into another target language using a parallel corpus. They applied a CCG semantic parser trained on English into Dutch. Such approaches can be extended to multiple languages using the parallel data or machine translation systems directly.

Other weak-supervision approaches that learn from question-answer pairs followed the trend by switching from restricted (closed) domains to open-domains. Berant et al. (2013) introduced a dataset called WebQuestions. The dataset is more challenging than Geoquery because the questions are based on various topics that appear in Freebase. SimpleQuestions (Bordes et al., 2015) is another dataset that is based on Freebase as explained earlier in

Section 3.2.2. Other approaches that use semantic web data have gained popularity in recent years. QALD datasets in particular have been used to evaluate systems that use DBpedia as a knowledge base. We discuss the published systems on SimpleQuestions and QALD datasets in the following sections.

4.2 QALD Systems

Höffner et al. (2017) surveyed QA systems that have been evaluated on QALD datasets till July 2015. The survey analyses the challenges these systems face and discusses the strengths and weaknesses of the proposed methods. They identified seven challenges that state-of-the-art systems need to handle in order to reach high performance in question answering task. Next, we explain these seven challenges and then methods that have proposed as possible solutions to each challenge and describe them in detail.

Challenge 1 - Lexical Gap: natural language questions can be formulated in different ways while asking for the same fact. However, knowledge bases contain limited label information (with *rdfs:label* property) about RDF resources. The gap occurs if the vocabulary used in a question is missing in a knowledge base. Systems need such lexical information to map natural language words and phrases to knowledge base items (resources, predicates or classes).

For instance, to answer the question "Who created Wikipedia?" the system needs to map the word "created" to the predicate "dbo:author", which is not provided by DBpedia. The predicate "dbo:author" has a label "author"@en, which does not help to do the mapping in this case. External dictionaries are required in order to fill the gap in natural language question and the knowledge base.

Challenge 2 - Ambiguity: this challenge occurs usually when systems try to solve the lexical gap and generate more candidates for a given search term. For instance, the same phrase can map to multiple predicates and it causes a challenge for QA systems because the number of possible queries for a given question increases. A QA system needs to disambiguate and select the best candidate query given the question where there might be multiple candidate queries.

For instance, the name "Barcelona" can be mapped to the city of Barcelona or the football club F.C. Barcelona.

Challenge 3 - Multilingualism: knowledge bases contain factual information in many languages (DBpedia in more than 120 languages) but QA systems still lack the notion of multilinguality. The multilinguality for QA systems means that questions in multiple languages can be answered simultaneously. It is an important aspect to consider for building a QA system since users would like to interact in their native languages. The previous challenge **lexical gap** needs to solved in multiple languages as well.

It becomes a challenge together with the lexical gap since the language specific part of a QA system lies in mapping natural language phrases to knowledge base entries. If a QA system uses some syntactic analyzer, then this syntactic analyzer should be able to produce similar outputs for other languages as well.

Challenge 4 - Complex Queries: the complexity of a question increases by the number of facts mentioned in the question. For instance, the question "Who is the mayor of the city that is the capital of Germany?" requires access to the syntax (relative clauses, coreference resolution, etc.) of the question to correctly answer it as shown in the SPARQL query below.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>
SELECT DISTINCT ?uri WHERE {
    ?uri dbo:leaderName ?city .
    ?city rdf:type dbo:City .
    ?city dbo:capital dbr:Germany .
}
```

Challenge 5 - Distributed Knowledge: The Linked Open Data Cloud contains interlinked knowledge bases. A question may involve a fact that requires combination of data from multiple sources.

For instance, to answer the question given below, a QA system requires YAGO classes in DBpedia.

Which Greek goddesses dwelt on Mount Olympus?

```
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX yago: <http://dbpedia.org/class/yago/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbp: <http://dbpedia.org/property/>
SELECT DISTINCT ?uri WHERE {
    ?uri rdf:type yago:GreekGoddesses .
    ?uri dbp:abode dbr:Mount_Olympus .
}
```

Challenge 6 - Procedural, Temporal and Spatial Questions: Procedural QA expects an explanation why a certain fact is returned as an answer. It is not tackled by any systems that use RDF data because all triples in a knowledge base are considered to be true. Temporal questions introduce a challenge for QA systems because detection, mapping and reasoning of temporal information in a question requires a thorough analysis of the underlying knowledge base ontology and the natural language question. Spatial questions pose a similar challenge as procedural and temporal questions where the focus is on integration of spatial knowledge bases with natural language questions.

Challenge 7 - Templates: QA systems that answer natural language questions using RDF data generate SPARQL queries where the answers can be found by executing the query. Simple questions usually require a single triple in a SPARQL query to answer the given question. Complex questions that require aggregation, or even more facts in a question require different templates of SPARQL queries.

The complexity of a question shown below lies in combining two triples, ordering the results and picking the second result.

What is the second highest mountain on Earth?

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT DISTINCT ?uri WHERE {
    ?uri rdf:type dbo:Mountain .
    ?uri dbo:elevation ?elevation .
}
    ORDER BY DESC(?elevation)
    OFFSET 1 LIMIT 1
```

Recently, Diefenbach et al. (2017a) published a survey on core techniques of QA systems that have been developed over the years to tackle challenges such as the ones defined above. These core techniques are:

Question Analysis: the technique involves analyzing natural language questions using syntactic information e.g. tokenization, part-of-speech (POS) tagging, named entity recognition, dependency parsing etc. This part usually requires systems to identify the question type (e.g. whether it has an aggregation), detecting phrases that stand for named entities, predicates, etc. For instance, SINA (Shekarpour et al., 2015) uses n-grams to detect named entities. PowerAqua (Lopez et al., 2012) instead depends on manually defined rules based on POS tags to detect question type, and group phrases as candidates for phrase mapping. Xser (Xu et al., 2014a) introduces a method for identifying phrases in a question that stand for a knowledge base item as a sequence labeling problem. One of our proposed systems (Hakimov et al., 2015) (see Chapter 6) extracts n-grams from the question and the query pair and learns how to map certain URIs to natural language expressions. Systems such as Intui2 (Dima, 2013) or FREyA (Damljanovic et al., 2010) use constituency parse tree generators to analyze phrasal dependencies between words. Systems such as gAnswer (Zou et al., 2014a), CASIA (He et al., 2014), DEANNA (Yahya et al., 2012) use dependency parse trees to analyze dependencies between words, which is similar to our proposed approach AMUSE (Hakimov et al., 2017) (see Chapter 7).

Phrase Mapping: this technique involves mapping phrases that occur in a question to knowledge base entries, e.g. named entity linking by mapping named entities in a question, mapping predicates etc. For instance, consider the question and the query given below. The natural language words that map to certain URIs in the query are highlighted with the same color.

Who **created** *Goofy* ?

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>
SELECT DISTINCT ?uri WHERE {
dbr:Goofy dbo:creator ?uri .
}
```

QA systems need an index to retrieve URIs for a given query term (words in questions). Most QA systems have created a specific indexing structure for each type of URI: predicates, resources and classes. For instance, the question above needs some mechanism that maps the word "created" to predicate *dbo:creator* and the word "Goofy" to the resource *dbr:Goofy*. Mapping resources in the query is straightforward in this question since the URI and the word in the question share the same label. However, the index may return many URIs that share the same label, e.g. Barcelona mapping to both the city and the football club. The predicate *dbo:creator* has the same lemma with the verb in the question "created".

QA systems need a lemmatizer or a stemmer in order to convert the words into the root form, which makes the matching easier because the index can cover only the root forms. An indexing solution such as Apache Lucene[2] can map subsequences of phrases efficiently but it may result in too many false negative matches. String similarity measures such as Levenshtein can be used to filter some matches that have

[2] http://lucene.apache.org/

lower similarity (predefined threshold) with the query term. Usually, the words that stand for predicates are more ambiguous compared to resources and classes. In natural language there are many variations for words that express certain relations between entities. For instance, "married to, spouse of, partner of, husband, wife, life partner of" are different natural language variations for the property *dbo:spouse* in DBpedia. Personal names can also be in different formats such as given name, full name, first name and last name, etc. QA systems need to consider such variations in natural language while retrieving query terms. It corresponds to the **Challenge 1 - Lexical Gap** explained above. The same problem exists not only for relations and entities but also for classes (also called concepts). For instance, the question "Give me all movies with Tom Cruise" requires an index that retrieves *dbo:Film* for the phrase "movies".

External dictionaries can be used on top of ontology labels to increase the coverage for mapping because in some cases the string similarity or subsequence matching methods are not adequate. WordNet provides *synsets* that act as synonyms for each word in the graph, e.g. the word "film" and "movie" are synonyms in WordNet. TBSL (Unger et al., 2012), SemSek (Aggarwal and Buitelaar, 2012) and PowerAqua (Lopez et al., 2012) use WordNet synsets as synonyms and use all as candidate terms while querying a certain index. Additionally, there are other approaches that provide lexicalisations for certain ontologies such as in the BOA Framework (Gerber and Ngomo, 2012), M-ATOLL (Walter et al., 2014) and PATTY (Nakashole et al., 2012). The BOA and M-ATOLL extract multilingual DBpedia Ontology lexicalisations from large text corpora, e.g. Wikipedia. TBSL uses the BOA Framework lexicalisations for mapping, PATTY is used by Xser (Xu et al., 2014a) and Hakimov et al. (2013) and our approach AMUSE uses M-ATOLL lexicalisations. Such approaches bridge the gap in phrase mapping in a multilingual setting as explained above by the **Challenge 2 - Multilingualism**.

Entity Linking (EL) approaches such as DBpedia Spotlight (Mendes et al., 2011) or NERFGUN (Hakimov et al., 2016) provide an inverted index of resources with alternative surface forms including the frequencies of each surface form occurring with a certain URI. Such inverted indexes can be used to map named entities in questions to resources. NERFGUN extracts anchor links in Wikipedia articles along with labels from DBpedia properties that express alternative names and aggregates the frequencies of how often each surface form co-occurs with a URI. This sort of surface form extraction approach can be applied to any language supported by DBpedia (over 120). Our approach AMUSE uses the index from NERFGUN for phrase mapping in English, Spanish and German questions.

Distributional semantics approaches such as word2Vec (Mikolov et al., 2013) can be used as another alternative to an external dictionaries. The word vector representation of a query term is compared with word vector representation of all candidate URIs. The cosine similarity is used to rank candidate URIs and top-k ranking ones are returned as a result. More details about usage of distributional semantics in mapping natural language phrases is described in Section 5.2. CASIA (He et al., 2014) uses this technique to map phrases to classes. Our approach AMUSE uses word vectors trained on multiple languages to map phrases to predicates.

Disambiguation: this technique involves picking the most likely candidate out of many options since certain words cant have different meanings based on the context and they map to multiple URIs, as explained before in the Phrase Mapping.

Different approaches have been proposed to disambiguate possible interpretations of a question. Some systems use local features such as string similarity to disambiguate

candidates while some systems train machine-learning models. Local feature disambiguation is used to compare string or context similarity to compare the word and the mapped URI. Additionally, knowledge base domain/range restrictions are used for fact checking. Predicates detected in the Phrase Mapping phase are used to check if any candidate resource has a matching type with a domain or range restriction of a predicate. For instance, Hakimov et al. (2013) uses such features along with frequencies of surface forms to rank the candidates. TBSL (Unger et al., 2012) uses the string similarity, knowledge base fact checking and considers whether the combined URIs occur in a triple to rank possible interpretations. SemSek (Aggarwal and Buitelaar, 2012) extracts only resource URIs in the Phrase Mapping step. By traversing over candidate resource URIs, the predicates of each resource URI are compared with a substring from a question, and the predicate URIs that match a certain threshold (edit distance similarity, context similarity using distributional semantics) are extracted as candidates.

Systems that use learning models rely on different sets of features such as local features (string/context similarity), knowledge base consistency features (domain/range restrictions), POS-tag features (whether a subsequence should be mapped to a predicate or resource), dependency relation features (indicates whether a certain resource has a relation to a predicate), knowledge base features (whether certain resource and predicate URI exist in a triple), co-occurrence of surface forms with URIs, popularity of resource URIs (e.g. PageRank score of all resource URIs in a knowledge base). SINA (Shekarpour et al., 2015) trains a Hidden Markov Model (HMM) to disambiguate phrase mappings and select the best interpretation. DEANNA (Yahya et al., 2012) uses a Integer Linear Program (ILP) algorithm for disambiguation whereas Xser (Xu et al., 2014a) is based on a structured perceptron algorithm. Our approach AMUSE extracts similar features and trains a factor graph model that learns to output the best interpretation among all candidates.

Query Construction: this step involves constructing a query by combining the returned URIs from the Disambiguation step explained above. Some approaches rely on a pre-defined set of templates of SPARQL queries while others construct a query dynamically based on the retrieved URIs. TBSL (Unger et al., 2012) uses templates of SPARQL queries and fills the slots using ranked URIs. Another similar technique is followed by QAKiS (Cabrio et al., 2012), ISOFT (Park et al., 2014a), Intui2 (Dima, 2013) and PowerAqua (Lopez et al., 2012) where templates contain at most two triples. RTV (Giannone et al., 2013), gAnswer (Zou et al., 2014a) and QAnswer (Ruseti et al., 2015) use dependency parse tree relations to combine URIs into triples. These approaches use the syntax of a question to combine URIs after the disambiguation step. Other approaches use semantic parsing methods that consider the syntax of a question to construct the query. The approach proposed by Hakimov et al. (2015) uses the CCG grammar to guide query construction by combining lexical items in a question. A similar approach is followed by AMUSE where the syntax of a question (dependency relations between words) is used to construct a query. SINA (Shekarpour et al., 2015) uses URIs from the disambiguation step and generates a graph. Such graph is composed of resource and class URIs as nodes and the predicate URIs as edges between the connected nodes. This graph can be converted into a SPARQL query since the queries represent a graph structure as well. These types of approaches consider only the underlying semantic information in a knowledge base. Our approach AMUSE considers the syntax of a question and the knowledge base semantic information in a query construction step.

Distributed Knowledge: this technique involves executing the constructed query on single or multiple knowledge bases. The constructed queries are executed on a single or multiple knowledge bases depending on the retrieved URIs. Some approaches combine URIs from different knowledge bases (e.g.PowerAqua). If knowledge bases are interlinked then the URIs are transferred into the respective knowledge base URI using *owl:sameAs* links. If the knowledge bases are not connected then it results in multiple queries based on each knowledge base. Each query is executed separately and the results are combined using labels as in PowerAqua (Lopez et al., 2012). Our approaches AMUSE and Hakimov et al. (2013) use a single knowledge base, DBpedia.

In Table 4.2, all systems that participated in QALD benchmarks as of July, 2018 are shown. The table includes the participating systems and their performances on the respective QALD dataset.

TABLE 4.2: All systems participated in Question Answering on DBpedia of QALD challenges published till July 2018. * systems that did not participate in challenges directly

System	Lang	Total	Precision	Recall	F-measure	Reference
QALD-1						
FREyA (Damljanovic et al., 2010)	en	50	0.54	0.46	0.50	Lopez et al. (2013)
PowerAqua (Lopez et al., 2012)	en	50	0.48	0.44	0.46	Lopez et al. (2013)
TBSL (Unger et al., 2012) *	en	50	0.41	0.42	0.42	Unger et al. (2012)
QALD-2						
SemSek (Aggarwal and Buitelaar, 2012)	en	100	0.35	0.38	0.37	Lopez et al. (2013)
BELA (Walter et al., 2012) *	en	100	0.19	0.22	0.21	Walter et al. (2012)
QAKiS (Cabrio et al., 2012)	en	100	0.14	0.13	0.13	Lopez et al. (2013)
Hakimov et al. (2013) *	en	55	0.83	0.32	0.46	Hakimov et al. (2013)
QALD-3						
gAnswer (Zou et al., 2014a) *	en	100	0.40	0.40	0.40	Zou et al. (2014a)
RTV (Giannone et al., 2013)	en	99	0.32	0.34	0.33	Cimiano et al. (2013)
Intui2 (Dima, 2013)	en	99	0.32	0.32	0.32	Cimiano et al. (2013)
SINA (Shekarpour et al., 2015) *	en	100	0.32	0.32	0.32	Shekarpour et al. (2015)
DEANNA (Yahya et al., 2012) *	en	100	0.21	0.21	0.21	Yahya et al. (2012)
SWIP (Pradel et al., 2012)	en	99	0.16	0.17	0.17	Cimiano et al. (2013)
Zhu et al. (2015) *	en	99	0.38	0.42	0.38	Zhu et al. (2015)
QALD-4						
Xser (Xu et al., 2014a)	en	50	0.72	0.71	0.72	Unger et al. (2014)
gAnswer (Zou et al., 2014a)	en	50	0.37	0.37	0.37	Unger et al. (2014)
CASIA (He et al., 2014)	en	50	0.32	0.40	0.36	Unger et al. (2014)
Intui3 (Dima, 2014a)	en	50	0.23	0.25	0.24	Unger et al. (2014)
ISOFT (Park et al., 2014a)	en	50	0.21	0.26	0.23	Unger et al. (2014)
Hakimov et al. (2015) *	en	50	0.52	0.13	0.21	Hakimov et al. (2015)
QALD-5						
Xser (Xu et al., 2014a)	en	50	0.74	0.72	0.73	Unger et al. (2015)
QAnswer (Ruseti et al., 2015)	en	50	0.46	0.35	0.40	Unger et al. (2015)
SemGraphQA (Beaumont et al., 2015)	en	50	0.31	0.32	0.31	Unger et al. (2015)
YodaQA (Baudiš and Šedivy, 2015)	en	50	0.28	0.25	0.26	Unger et al. (2015)
QALD-6						
UTQA (Veyseh, 2016)	en	100	0.82	0.69	0.75	Unger et al. (2016)
UTQA (Veyseh, 2016)	es	100	0.76	0.62	0.68	Unger et al. (2016)
UTQA (Veyseh, 2016)	fs	100	0.70	0.61	0.65	Unger et al. (2016)
SemGraphQA (Beaumont et al., 2015)	en	100	0.70	0.25	0.37	Unger et al. (2016)
AMUSE (Hakimov et al., 2017) *	en	100	-	-	0.26	Hakimov et al. (2017)
AMUSE (Hakimov et al., 2017) *	de	100	-	-	0.16	Hakimov et al. (2017)
AMUSE (Hakimov et al., 2017) *	es	100	-	-	0.20	Hakimov et al. (2017)
QALD-7						
WDAqua (Dragoni et al., 2017)	en	215	0.16	0.16	0.14	Dragoni et al. (2017)
gAnswer2 (Zou et al., 2014a)	en	215	0.49	0.49	0.47	Dragoni et al. (2017)
AMAL (Dragoni et al., 2017)	fr	215	0.72	0.72	0.72	Dragoni et al. (2017)
QALD-8						
WDAqua-core0 (Diefenbach et al., 2017b)	en	219	0.39	0.40	0.39	Usbeck et al. (2017)
gAnswer (Zou et al., 2014a)	en	219	0.39	0.39	0.39	Usbeck et al. (2017)
QAKIS (Cabrio et al., 2012)	fr	219	0.06	0.05	0.06	Usbeck et al. (2017)

4.3 SimpleQuestions Systems

The SimpleQuestions dataset includes **simpler** questions where a single fact from Freebase is used to answer those questions. Over the years, researchers have focused on building neural network architectures since the number of training instances is more than 75K and such architectures need lots of data to train. These systems are listed in Table 4.3 with their respective scores obtained on the dataset.

TABLE 4.3: Systems evaluated on SimpleQuestions dataset ranked by the reported F1 measures, * systems evaluated on FB5M data, the rest were evaluated on FB2M data

System	Accuracy
Bordes et al. (2015)	0.63
Aghaebrahimian and Jurčíček (2016)	0.65
Golub and He (2016)	0.71
Lukovnikov et al. (2017)	0.71
Dai et al. (2016)	0.76*
Yin et al. (2016)	0.76
Ture and Jojic (2017)	0.88

Bordes et al. (2015) present the baseline for the dataset using Memory Networks (Weston et al., 2015). The approach generates candidate entities using n-grams from the question text that match some Freebase entity. The top two entities for each of the five longest matched n-grams are chosen as candidates. The main component of the model is scoring the candidate Freebase fact y and it is computed as follows:

$$S_{QA}(x,y) = cos(W_V g(y), W_S f(y)) \tag{4.1}$$

where W_V is the word embedding matrix and W_S is the knowledge base embedding matrix for Freebase. The approach corrupts the dataset to generate negative samples by assigning random questions from the datasets to Freebase entity and predicate pairs. The training objective is to maximize the score (Equation 4.1) between positive and negative samples.

Yin et al. (2016) proposed an approach that uses Convolutional Neural Networks (CNN) with attentive max pooling. They also presented entity linking methods to find the subject entity in the question: passive and active. The passive method includes querying all n-grams in the question text and retrieve spans that match some Freebase entity. The active linking includes training a system that learns to detect the span of an entity and retrieve Freebase entities using the detected surface form. The retrieved Freebase entities are ranked by the learned score for each method and the top-k candidates are returned. Active linking outperforms Passive linking with a 0.08 margin (0.81 vs 0.89) for top-20 candidates, which is the number that was set for the whole system. The system for predicting the entity and the predicate is depicted in Figure 4.3. The authors proposed to use character embeddings since this generalizes better in handling out-of-vocabulary (OOV) words. The left part shown in Figure 4.3 consists of a CNN layer on character-level embeddings to detect the subject entity. As shown on the right, the detected entity span is replaced by the special character (<e>) and the whole sequence of words in the question are fed into a CNN layer using word embeddings. This CNN layer also has an attentive max-pooling layer that learns to assign different scores to n-grams.

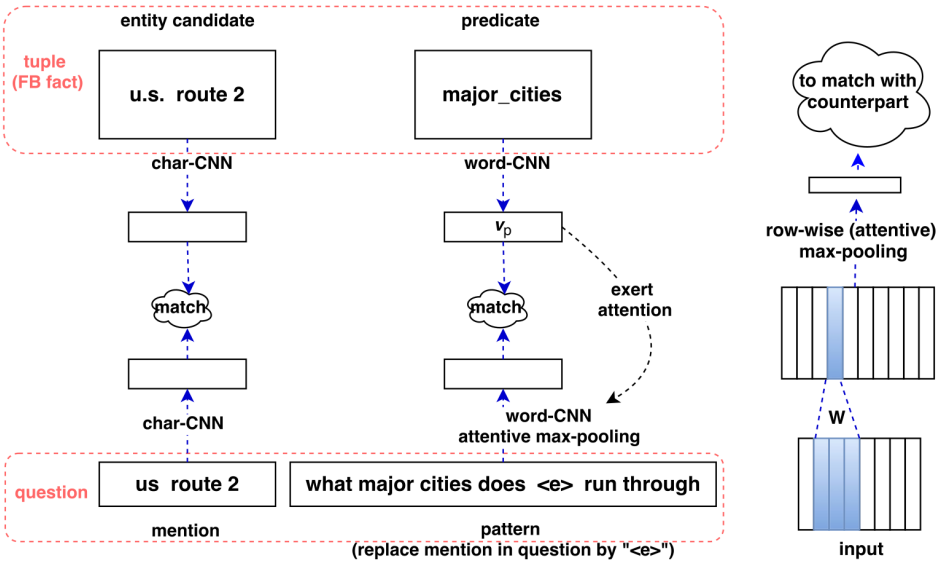


FIGURE 4.3: CNN with attention max-pooling proposed by Yin et al. (2016)

Lukovnikov et al. (2017) proposed a system to encode the question q, the subject entity s and the predicate p using word and character embeddings and learn a function that optimizes both subject and predicate assignments by introducing negative samples. The question is encoded by computing word embeddings of each token and feeding them into an RNN. The last output from the network is the representation of the question text as shown in Figure 4.4a. The subject entity is encoded by character embeddings and word embeddings. Each character in the label of the entity is fed into an RNN and word embeddings of type labels of the entity are fed into another RNN. An example of type labels for entity *hainan* are *chinese, province* as shown in Figure 4.4b.

Finally, the predicate is encoded using word embeddings as shown in Figure 4.4c. Predicates in Freebase are grouped under different domains. Each predicate URI consists of a label and hierarchy of domain labels. For instance, the predicate for the question "What cyclone affected Hainan" is *meteorology/affected_area/cyclone*, which consists of 3 hierarchies. Each label is fed into the RNN and the output from the last layer in the network is the representation of the predicate.

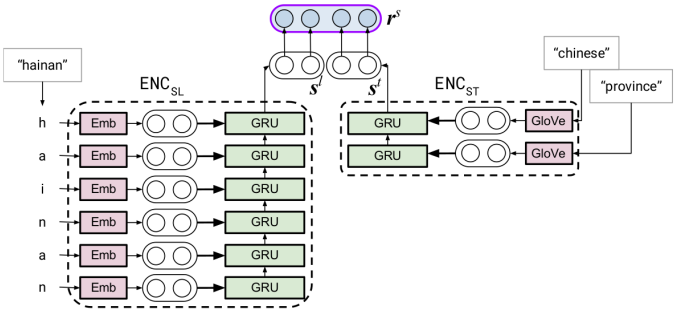
The similarity score of the subject entity and the predicate to the question is given below.

$$\hat{s} = \arg\max_{s_i \in C_s} S_s(q, s_i) \tag{4.2}$$

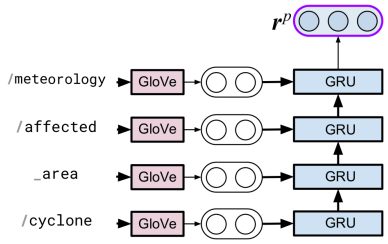
$$\hat{p} = \arg\max_{p_i \in C_p} S_p(q, p_i) \tag{4.3}$$

The functions S_s and S_p compute cosine similarity between the given question representation and the subject entity representation or the predicate representation. C_s includes candidate entities that are retrieved by matching n-grams in the question text and Freebase entity labels. C_p includes candidate predicates that are connected to entities in C_s and the predicate that have an entity from C_s in the subject position. Equation 4.2 selects an entity with the maximum score from the candidate entities in C_s. Equation 4.3 selects a predicate with a maximum score.

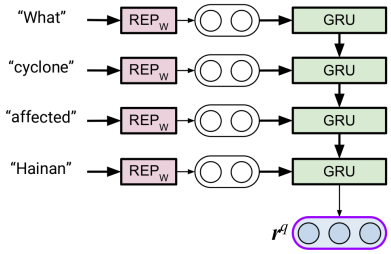
The approach is trained to optimize the scoring functions S_s and S_p using negative sampling, where they introduce questions with the corrupted subject and the predicate.



(A) Entity encoding using character-level (entity label) and word embeddings (entity type labels)



(B) Predicate encoding using word embeddings (each word in hierarchy)



(C) Question encoding using word embeddings

FIGURE 4.4: Encoding the question, the subject entity and the predicate proposed by Lukovnikov et al. (2017)

Golub and He (2016) proposed an approach that uses both LSTM and CNN encoders together with character-level embeddings. The question is encoded by representing all characters as one-hot vectors and feeding them into two-layered LSTM with an attention mechanism. Knowledge base entities and predicates are encoded again with character-level embeddings by extracting labels of each URI and feeding them into a CNN with two layers. The architecture is shown in Figure 4.5. The output from the system is computed using LSTM decoder with an attention mechanism. The decoder computes the similarity of the question to an entity and the similarity of the question to a predicate.

The input to the whole network is a pair of entity and predicate. The output from the network is a score for the given pair. The entity and the predicate pairs are generated similarly to Bordes et al. (2015). All n-grams are used to match a Freebase entity. The matching entities are extracted together with their corresponding spans in the question. Entities with spans that are part of bigger spans are filtered out. The remaining entities are ranked by the number of triples they have in Freebase, then the top 10 highest ranking entities are returned as candidate entities. All predicates for each entity in the top-10 list are extracted as candidate predicates. Candidate pairs of entity and predicate are generated using these lists. Training the model is done by feeding the given entity and predicate pair along with 50 randomly sampled negative pairs, similar to Lukovnikov et al. (2017). During prediction the candidate

pairs are generated as explained above and scored using the equation given below.

$$\hat{e}, \hat{p} = \arg\max_{e_i, p_j \in C_n} P(e_i) * P(p_j) \tag{4.4}$$

The important point in the evaluation suggests that character embeddings generalize better compared to word embeddings (0.78 vs 0.38). Moreover, the effectiveness of the model with an attention mechanism shows that the system learns to differentiate between spans of the predicate and the entity by assigning different weights to them.

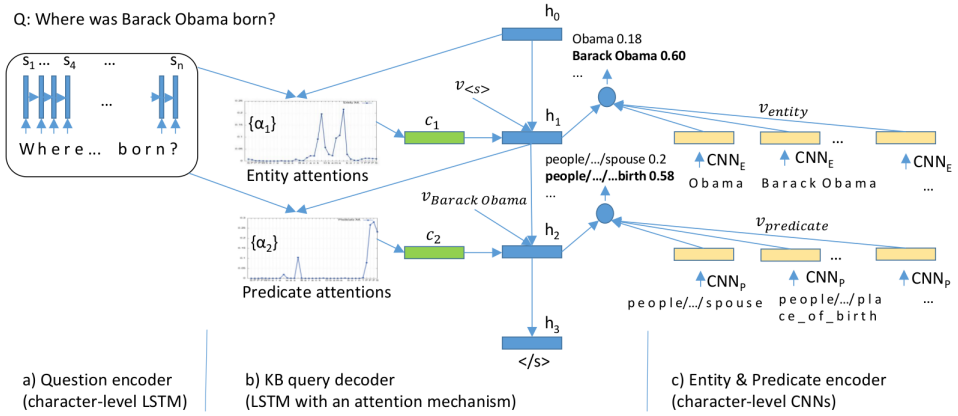


FIGURE 4.5: CNN with attention max-pooling proposed by Yin et al. (2016)

The results are given below.

TABLE 4.4: The results for passive and active Linking methods proposed by Yin et al. (2016)

N	Passive	Active
1	0.57	0.74
5	0.71	0.85
10	0.75	0.87
20	**0.81**	**0.89**
50	0.86	0.90
100	0.88	0.92

Ture and Jojic (2017) proposed a simpler model based on RNNs without any attention mechanism. They essentially propose to use a model with two BiGRU layers for prediction of predicates and a model with two BiLSTM layers to predict the span for the subject. The entity span detection model is similar to ours. The predicate prediction model is similar to our BiLSTM-Softmax in Chapter 8.

Chapter 5

Lexicon

In this chapter, we present resources for mapping natural language phrases into knowledge base entries by introducing an inverted index for retrieval of URIs. We combine several external resources and compare the performance on the created lexical inverted index.

5.1 Mapping from text to knowledge base entries

A key component in a question answering pipeline is the mapping of query terms to knowledge base entries. Consider the question *Who is the writer of The Hunger Games?* It seems to be a trivial task to link the query word *writer* to the appropriate identifier `dbo:author`, however it still requires prior knowledge about the semantics of the query word and the KB entry (e.g. that the writer of a book is the author).

QA systems need to handle the lexical gap to increase the coverage and performance. A survey on QA systems (Höffner et al., 2017) published within QALD workshops concluded that QA systems need to handle seven challenges and the **lexical gap** is one of them (see Section 4.2). Consider the following question and its expected SPARQL query.

How tall is Michael Jordan ?

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>
SELECT DISTINCT ?s
WHERE {
      dbr:Michael_Jordan dbo:height ?s.
}
```

A QA system needs to map the term *tall* to the knowledge base property *dbo:height* in order to answer this question. However, such a lexical entry does not exist in DBpedia. The lexical gap refers to the problem that the QA system can not answer this question only because of the missing lexical entry. As we focus on building a multilingual QA system, we need to overcome this challenge for multiple languages.

Another question and its expected SPARQL query is given below.

Give me all Danish movies.

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT DISTINCT ?s
```

```
WHERE {
      ?s rdf:type dbo:Film.
      ?s dbo:country dbr:Denmark.
}
```

It is even more challenging than the previous question because the term *Danish* here stands for the combination of the property *dbo:country* and the resource *dbr:Denmark*. The word *movies* should be mapped to the combination of the property *rdf:type* and the class *dbo:Film*. The property *rdf:type* is trivial in this case since all resources in DBpedia have the same property to express the class they belong to. The main challenge here is to map the word *movies* to the class *dbo:Film* using external dictionaries like synonyms, e.g. *movie* has a synonym entry *film* in WordNet (Miller, 1995b).

5.2 Inverted Index

To address the lexical gap, we rely on five resources to build the indexes for retrieval. As we explained in Section 2.3.3, we have five different DUDES types. For Resource, Property, Restriction and Class DUDES we created a separate inverted index to retrieve matching URIs given the query words. Our index is based on English, German and Spanish languages. Note here that these inverted indexes are based on DBpedia data. Next, we explain in detail how each index is created.

5.2.1 Resource Index

The resource index consists of resources in DBpedia and their different surface forms. Most of these resources are people with different name labels, e.g. nickname, full name, given name, surname etc. To capture as many labels as possible, we merge structured data available in DBpedia with Wikipedia anchor links. NERFGUN (Hakimov et al., 2016) is an entity linking (EL) system that disambiguates entities detected in a text to corresponding Wikipedia articles. We rely on the index of NERFGUN for retrieving candidate resources given a label to search for. The index is built by extracting anchor links in Wikipedia that refer to certain entities as candidate surface forms. On top of that, the information for resources from DBpedia properties such as *rdfs:label, dbo:givenName, dbo:surname, dbp:name, etc.* are also extracted. Surface forms from both sources are merged by aggregating the frequency of each surface form for a given resource. We give a sample from the index below. The data in the table indicates how often a surface refers to a DBpedia resource.

Surface form	DBpedia resource	Frequency
bielefeld	dbr:Bielefeld	25
bielefeld university	dbr:Bielefeld_University	18
uni bielefeld	dbr:Bielefeld_University	15
bielefeld	dbr:Bielefeld_University	5

5.2.2 Property Index

The property index is built based on lexicalizations of DBpedia properties extracted by M-ATOLL[1] (Walter et al., 2014), WordNet (Miller, 1995b) synonyms, machine translation (to translate English labels into other languages) and using word embeddings (to retrieve candidate properties for a given mention text). Next, we give an overview of each of these four resources.

[1] http://www.dblexipedia.org

Ontology Labels RDF Schema labels in English, German and Spanish for properties are extracted from the DBpedia Ontology. DBpedia has a property *rdfs:label* to express labels of resources, properties and classes in multiple languages.

M-ATOLL M-ATOLL(Walter et al., 2014) is a framework for obtaining lexicalization of ontology items. Currently, M-ATOLL supports English, German and Spanish languages. The lexicon of property type is added to the index.

Machine Translation We rely on the online dictionary `Dict.cc`[2] as our translation engine. We query the web service for each available English label and target language, then store the obtained translation candidates as new labels for the respective entity and language. While these translations are prone to be noisy without a proper context, we receive a reasonable starting point for the generation of candidate lexicalizations, especially in combination with the word embedding approach.

Word Embedding Retrieval Many word embedding methods such as the skip-gram method Mikolov et al. (2013) have been shown to encode useful semantic and syntactic properties. The objective of the skip-gram method is to learn word representations that are useful for predicting context words. As a result, the learned embeddings often display a desirable linear structure that can be exploited using simple vector addition. Motivated by the compositionality of word vectors, we propose a measure of semantic relatedness between a mention m and a DBpedia entry e using the cosine similarity between their respective vector representations $\vec{v}_m$ and $\vec{v}_e$. For this we follow the approach in Basile et al. (2016) to derive entity embedding vectors from word vectors: We define the vector of a mention m as the sum of the vectors of its tokens[3] $\vec{v}_m = \sum_{t \in m} \vec{v}_t$, where the $\vec{v}_t$ are raw vectors from the set of pre-trained skip-gram vectors. Similarly, we derive the vector representation of a DBpedia entry e by adding the individual word vectors for the respective label l_e of e, thus $\vec{v}_e = \sum_{t \in l_e} \vec{v}_t$.

As an example, the vector for the mention text *movie director* is composed as $\vec{v}_{movie\ director} = \vec{v}_{movie} + \vec{v}_{director}$. The DBpedia entry `dbo:director` has the label *film director* and is thus composed of $\vec{v}_{dbo:director} = \vec{v}_{film} + \vec{v}_{director}$. The vectors $\vec{v}_{movie}$, $\vec{v}_{film}$ and $\vec{v}_{director}$ are obtained from the skip-gram embeddings.

To generate potential linking candidates given a mention text, we can compute the cosine similarity measure between $\vec{v}_m$ and each possible $\vec{v}_e$ as a measure of semantic relatedness and thus produce a ranking of all candidate entries. By pruning the ranking at a chosen threshold, we can control the produced candidate list for precision and recall.

Using the cosine similarity to compute the vector similarity between mentions and entities, we can interpret this similarity score as a measure of semantic relatedness and thus as an indicator for a potential match between the mention text and the candidate entity. By computing the similarity of a mention to all possible KB entries, we can produce a ranking of these properties that places more likely candidates at the top positions of the rankings and unlikely candidates at the bottom.

For this work, we trained 3 instances of the skip-gram model with each 100 dimensions on the English, German and Spanish Wikipedia respectively. Following this approach, the top ranking DBpedia entries for the mention text *total population* are listed below:

Mention	DBpedia entry	Cosine Similarity
total population	`dbo:populationTotal`	1.0
	`dbo:totalPopulation`	1.0
	`dbo:agglomerationPopulationTotal`	0.984
	`dbo:populationTotalRanking`	0.983
	`dbo:PopulatedPlace/areaTotal`	0.979

[2] `http://www.dict.cc`

[3] We omit all stopword tokens.

5.2.3 Class Index

The class index is created using DBpedia Ontology labels and WordNet synonyms. We extracted the labels for classes using the *rdfs:label* property. Additionally, we used the extracted labels as query terms to search for synsets in WordNet to extract them as additional resource.

5.2.4 Restriction Class Index

A restriction class is a special case of classes where a resource has a restriction to a certain property. For instance, the word *Swedish* corresponds to the following triple in DBpedia where the resource *dbr:Sweden* is bound to the property *dbo:country*.

?x dbo:country dbr:Sweden

The M-ATOLL extracts restriction classes for English with corresponding frequencies. Some samples are shown below.

Mention	DBpedia entry	Frequency
female	`dbo:gender dbr:Female`	238
catholic	`dbo:religion dbr:Catholic_Church`	18
german	`dbo:originalLanguage dbr:German_language`	139

5.3 Evaluation

We evaluate the proposed lexicon generation methods using machine translation and embeddings with respect to a lexicon of manual annotations that are obtained from the training set of the QALD-6 dataset. The manual lexicon is a mapping of mention to expected KB entry derived from the (question-query) pairs in the QALD-6 dataset. Since the M-ATOLL induces lexicalisations for the DBpedia Ontology properties, we restrict our word embedding approach to also only produce this subset of KB entities. Analogously, the manual lexicon is filtered such that it only contains word-property entries for DBpedia ontology properties to prevent the unnecessary distortion of the evaluation results due to unsolvable query terms.

The evaluation is carried out with respect to the number of generated candidates per query term using the Recall@k measure. Focusing on recall is a reasonable evaluation metric since the considered manual lexicon is far from exhaustive, but only reflects a small subset of possible lexicalizations of KB properties in natural language questions.

Figure 5.1, Figure 5.2 and Figure 5.3 visualize the retrieval performance using the Recall@k metric for English, German and Spanish languages respectively. The visualizations show performance values for M-ATOLL (matoll), word embeddings (w2v) separately and combination of both resources (w2v+matoll). For English, M-ATOLL reaches 0.3 at maximum whereas word embeddings reach 0.5 at maximum. Combination of both resources yields a better performance of 0.6. For German and Spanish M-ATOLL does not perform as good as for English whereas word embeddings perform better for both languages. The combining the M-ATOLL candidates with the word embedding candidates yields the strongest recall performance.

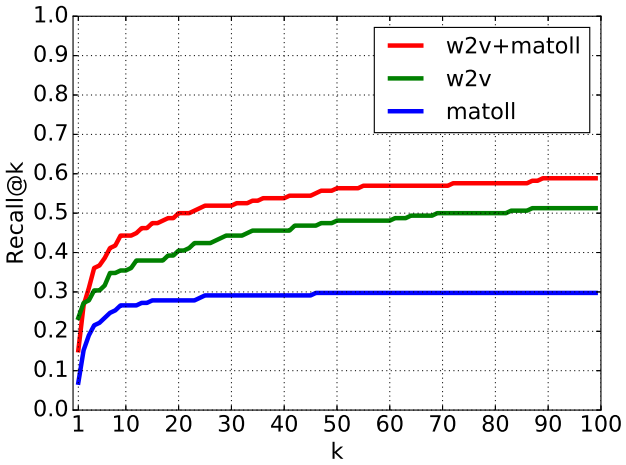


FIGURE 5.1: Retrieval performance on English with respect to the manual lexicon.

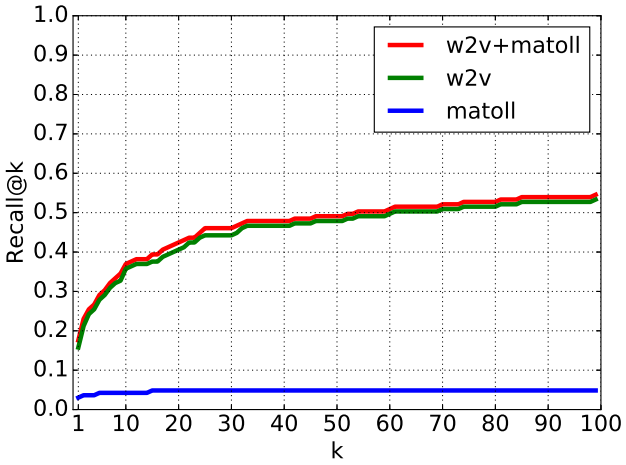


FIGURE 5.2: Retrieval performance on German with respect to the manual lexicon.

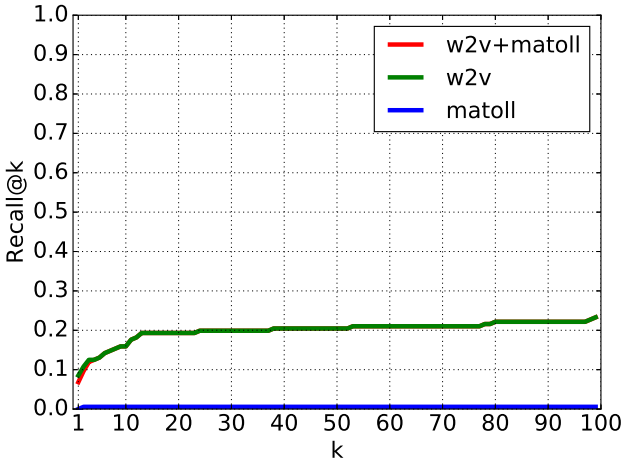


FIGURE 5.3: Retrieval performance on Spanish with respect to the manual lexicon.

Chapter 6

CCG-based Semantic Parsing Approach

In this chapter, we present a semantic parsing approach that uses Combinatory Categorial Grammar (CCG) for syntax and lambda calculus for semantics to build a QA system. The approach is applied to the QALD-4 (Unger et al., 2014) dataset on English questions. We give detailed information about the approach along with comparisons to other published systems. The content provided in this chapter is based on our previously published work (Hakimov et al., 2015).

6.1 Overview

We present a method for learning syntax and semantics jointly for question answering. The presented method is an adaptation of the CCG-based semantic parsing proposed by Zettlemoyer and Collins (2005) to the QALD-4 dataset. It is a monolingual approach as it depends on the CCG combination rules and the domain-independent lexicon for a specific language. The original approach requires rules for the function called GENLEX. We adapted the function to the QALD-4 dataset and added additional rules (see Table 6.2). The approach was evaluated on the QALD-4 dataset and compared to other state-of-the-art systems. In the next sections, we give a detailed description of the approach which then followed by the evaluation and discussion sections.

6.2 CCG-based Approach

As mentioned above, this chapter is about applying the semantic parsing approach proposed by Zettlemoyer and Collins (2005) on the QALD-4 dataset. We will refer to their approach as ZC05 throughout the thesis. ZC05 induces a grammar that maps sentences to logical forms. The grammar consists of entries that correspond to CCG syntax and semantics based on lambda calculus. The core part of the algorithm is the GENLEX function, which generates candidate entries for the grammar. Our implementation adds additional rules to this method (see Table 6.2). We give a detailed description of CCG in Section 2.2.1 also the lambda calculus used for semantic representations are described in Section 2.3.3.

Zettlemoyer and Collins (2005) added additional quantifying terms such as: *count*, *argmax* and *argmin* on top of functions for expressing predicates and individuals with the lambda calculus. The quantified expression *argmin* and *argmax* have the general form of $argmax(\phi, \psi)$ and $argmin(\phi, \psi)$, returning the first or the last item from the ordered list of items denoted by ϕ and aggregated by ψ. The quantified expression *count* has the form of $count(\phi)$ where the items are aggregated and it returns the number of items in the list. For example, the expression $\lambda x.argmax(robot(x), performance(x, y))$ returns a robot x that has the highest performance y.

In order to evaluate semantic parsing on the QALD-4 dataset, the provided SPARQL queries have to be converted to semantic representations using lambda calculus. For this conversion we define the following translation rules:

- Every resource in the query is translated into a constant.
- Every predicate in the query is translated into a function with two arguments.
- Every solution modifier `ORDER BY`, `LIMIT` and `OFFSET` is translated into an *argmax/min* quantifier.
- Every `COUNT` solution modifier is translated into the function constant *count*.

In the following we give some examples with their corresponding lambda calculus expressions.

1. Give me all islands that belong to Japan.

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT DISTINCT ?uri
WHERE {
      ?uri rdf:type dbo:Island .
      ?uri dbo:country dbr:Japan .
}
```

$\lambda x.\texttt{rdf:type}(x,\texttt{dbo:Island}) \wedge \texttt{dbo:country}(x,\texttt{dbr:Japan})$

2. Who is the youngest player in the Premier League?

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT DISTINCT ?uri
WHERE {
        ?uri dbo:team ?x .
        ?x dbo:league dbr:Premier_League .
        ?uri dbo:birthDate ?y .
}
ORDER BY DESC(?y)
OFFSET 0 LIMIT 1
```

$\lambda x.\mathit{argmin}(\texttt{rdf:type}(x,\texttt{dbo:Soccer\_Player})$
$\wedge\texttt{dbo:league}(x,\texttt{dbr:Premier\_League}),\texttt{dbo:birthDate}(x,y))$

3. How many films did Hal Roach produce?

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>
```

```
SELECT COUNT(DISTINCT ?uri)
WHERE {
        ?uri dbo:producer dbr:Hal_Roach .
}
```

$\lambda x.count(\texttt{rdf:type}(x,\texttt{dbo:Film}) \wedge \texttt{dbo:producer}(x,\texttt{dbr:Hal\_Roach}))$

6.2.1 Semantic parsing à la Zettlemoyer & Collins

In this section, we describe the semantic parsing approach proposed by Zettlemoyer and Collins (2005).

The input to the algorithm is a set of training examples (S_i, L_i) with $i = 1 \ldots n$, where each S_i is a sentence and each L_i is a corresponding semantic representation (*logical form*). The output is a pair (ϕ, θ), where ϕ is a set of features and θ is a vector of weights for those features.

The important part of the approach is the function called GENLEX(S,L). It takes as input a sentence S, a corresponding logical form L and generates a set of potential lexical items with syntactic categories and the corresponding semantics, and pairs them with all possible sub-strings of S. The input sentence is split into tokens. The logical form is also split into atomic CCG categories: noun, property and noun phrase. GENLEX generates all possible lexical entries paired with sub-strings from the sentence and atomic categories from the logical form. Additionally, domain-independent expressions, such as wh-words, auxiliary verbs, determiners, etc. must be manually defined by assigning a CCG category and lambda calculus expression. The wh-word *what*, for example, has the syntactic category (S/(S\NP))/N and the semantic representation $\lambda f.\lambda g.\lambda x.f(x) \wedge g(x)$. Example lexical entries are shown below in Table 6.1. These entries are generated by splitting the sentence "Barack Obama is married to Michelle Obama" and the logical form:

$\texttt{dbo:spouse}(\texttt{dbr:Barack\_Obama},\texttt{dbr:Michelle\_Obama})$

TABLE 6.1: Lexical entries generated by GENLEX with their syntactic and semantic representations for the input sentence "Barack Obama is married to Michelle Obama"

Phrase	Syntax	Semantics
Barack Obama	NP	dbr:Barack_Obama
Barack	NP	dbr:Barack_Obama
Obama	NP	dbr:Barack_Obama
Barack Obama	NP	dbr:Michelle_Obama
Barack Obama	(S\NP)/NP	$\lambda y.\lambda x.$dbo:spouse(x,y)
is	(S\NP)/(S\NP)	$\lambda f.\lambda x.f(x)$
married to	(S\NP)/NP	$\lambda y.\lambda x.$dbo:spouse(x,y)
married to	NP	dbr:Michelle_Obama
married to	NP	dbr:Barack_Obama
married	NP	dbr:Barack_Obama
to Michelle	NP	dbr:Michelle_Obama
Michelle Obama	NP	dbr:Michelle_Obama
Michelle Obama	NP	dbr:Barack_Obama
Michelle Obama	(S\NP)/NP	$\lambda y.\lambda x.$dbo:spouse(x,y)
...		

The logical form is split into 1 predicate and 2 constants: $\lambda y.\lambda x.\texttt{dbo:spouse}(x,y)$, `dbr:Michelle_Obama`, `dbr:Barack_Obama` where the variables x and y are replaced for the resources in the lambda expression. Each logical form has an assigned CCG category according to the definition of CCGs. For example, a predicate has the category (S\NP)/NP while a constant has NP. All such assignments are defined by the rules in GENLEX function.

The sentence is also split into tokens. GENLEX maps token(s) into a pair of CCG category and atomic lambda expression and generates all possible entries (lexical items) as shown above in Table 6.1. For simplicity we included only some samples for the input sentence. The domain-independent words such as *is* do not get assignments by GENLEX since their syntax and semantics is already pre-defined manually.

The generated lexical entries are then used in the actual semantic parsing step, employing a dynamic programming algorithm such as CYK parsing adapted for CCG along with a beam-search decoder. Semantic parsing using CCG grammar with lambda calculus is applied to all possible lexical entries as shown in Section 2.3.4. The parse tree is derived by applying CCG combination rules along with lambda calculus expressions throught the parse tree. It is syntactically correct when the final CCG category is S (stands for sentence) and it is semantically correct when the combined lambda calculus is equal to the expected one. In Figure 6.1, we show a parsing example for the sentence "Barack Obama is married to Michelle Obama" with the expected logical form:

$$\texttt{dbo:spouse}(\texttt{dbr:Barack\_Obama},\texttt{dbr:Michelle\_Obama})$$

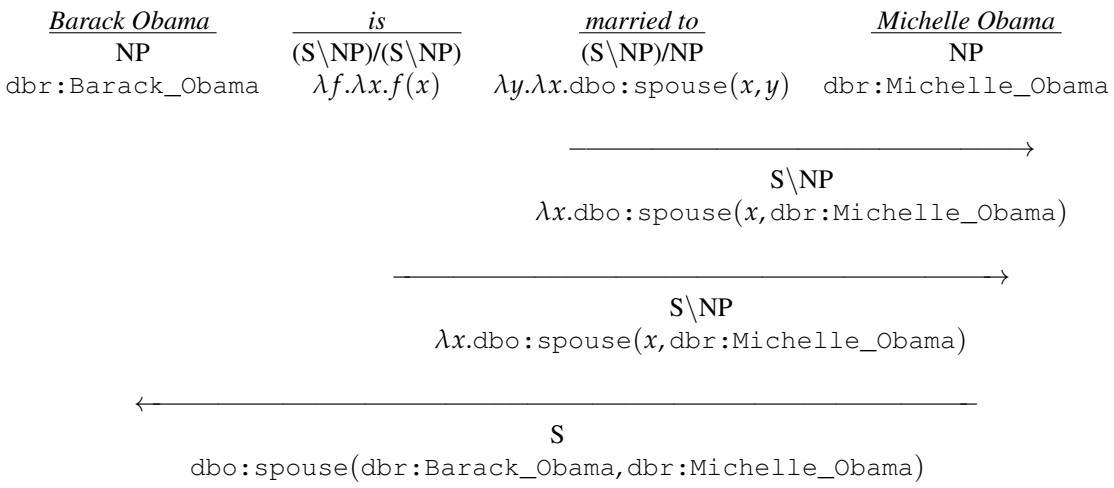


FIGURE 6.1: CCG parse tree with syntax and semantics for the sentence *Barack Obama is married to Michelle Obama.*

The semantic parsing step finally returns the highest scoring parse that derive the expected logical form L using all possible lexical items with weight vector θ for features. The parse tree shown in Figure 6.1 is valid because the derived logical form is equal to the expected and the final CCG category is S. The semantic parsing step may return multiple parse trees since some strings can be ambiguous. For example, the string *married to* has multiple assignments in Table 6.1 and different lexical items may result in the same final CCG category and the lambda calculus.

The algorithm returns the highest scoring parse tree T and the final logical form L for a given sentence S as shown in the equation below.

$$\underset{L}{argmax}\ P(L|S;\theta) = \underset{L}{argmax} \sum_{T} P(L,T|S;\theta) \tag{6.1}$$

where T is a parse tree (a sequence of steps that derive L), and *argmax* is taken over all T that produce L. Different T can be derived for the same S and L due to ambiguities in terms generated by GENLEX.

The probability of a particular pair (L, T) for a given S is defined by a log-linear model:

$$P(L, T|S; \theta) = \frac{e^{\bar{f}(L,T,S)\cdot\theta}}{\sum_{(L,T)} e^{\bar{f}(L,T,S)\cdot\theta}} \tag{6.2}$$

where $\bar{f}(L, T, S) = (\bar{f}_1(L, T, S) \ldots \bar{f}_d(L, T, S))$ is a function that maps triples (L, T, S) to a feature vector in R^d, with d being the number of features. For each lexical item in T there is a feature f_j that counts the number of times this item is used in T where $f_j \in \phi$. The lexical items in Figure 6.1 are examples of such features because the derived parse tree is valid. The sum in the denominator is over all valid parses for S under the induced CCG grammar.

To estimate the parameters θ, we train the model with a set of training examples $\{(S_i, L_i): i = 1 \cdots n\}$. Note that the derived parse tree T is not included. It is a hidden variable that is inferred.

Parsing is an iterative process: The first step uses all possible lexical items generated by GENLEX, and only those lexical items that were used in the successful parses are then passed to the second step of parsing, where parameter values are estimated. The output from the approach is the learned parameters θ and the induced lexical features ϕ.

6.2.2 Applying Semantic Parsing to QALD Dataset

In order to apply the ZC05 approach to the QALD-4 training dataset, comprising 200 natural language questions with corresponding SPARQL queries, all SPARQL queries were converted into lambda calculus expressions as explained earlier. Domain-independent expressions such as wh-words, prepositions, determiners, etc. were specified manually, based on similar domain-independent expressions used in ZC05. These expressions and the 200 training examples from QALD-4 are used as an input to the algorithm.

We re-implemented the algorithm following the descriptions in Zettlemoyer and Collins (2005), using CYK-style parsing with a stack decoder, and changing the parameter estimation step into perceptron updates. In Table 6.2, we show the updated GENLEX rules that we employed. Each entry in the table has an input trigger and the corresponding output category, shown together with an example. Each output category is matched with a syntactic and a semantic representation. Newly added input triggers are highlighted in **boldface**. Moreover, one-place predicates were removed, as in an RDF setting they are covered by the arity-two predicate with constant trigger (the predicate being `rdf:type`).

TABLE 6.2: GENLEX rules from Zettlemoyer and Collins, 2005 adapted to the QALD-4 dataset.

Input Trigger	Output Category and Example
Constant c	NP : c NP : dbr:Brooklyn_Bridge
Arity-two predicate p	(S\NP)/NP : $\lambda x.\lambda y.p(y,x)$ (S\NP)/NP : $\lambda x\lambda y.$dbo:author(y,x)
Arity-two predicate p	(S\NP)/NP : $\lambda x.\lambda y.p(x,y)$ (S\NP)/NP : $\lambda x.\lambda y.$dbo:starring(x,y)
Arity-two predicate p	(S\NP)/NP : $\lambda g.\lambda x.\lambda y.p(y,x) \wedge g(y)$ (S\NP)/NP : $\lambda g.\lambda x.\lambda y.$dbo:crosses$(x,y) \wedge g(y)$
Arity-two predicate p	N/NP : $\lambda x.\lambda y.p(x,y)$ N/NP : $\lambda x.\lambda y.$dbo:officialColor(x,y)
Arity-two predicate p	N/NP : $\lambda g.\lambda x.\lambda y.p(y,x) \wedge g(y)$ N/NP : $\lambda g.\lambda x.\lambda y.$dbo:capital$(y,x) \wedge g(y)$
Arity-two predicate p and constant c	N : $\lambda x.p(x,c)$ N : $\lambda x.$rdf:type$(x,$dbo:River$)$
Arity-two predicate p	(N\N)/NP : $\lambda x.\lambda g.\lambda y.p(y,x) \wedge g(y)$ (N\N)/NP : $\lambda x.\lambda g.\lambda y.$dbo:crosses$(y,x) \wedge g(y)$
Arity-two predicate p and constant c	N/N : $\lambda g.\lambda y.p(y,c) \wedge g(y)$ N/N : $\lambda x.$dbo:country$(x,$dbr:Germany$) \wedge g(x)$
argmax/min with second argument arity-two function f	NP/N : $\lambda g.\lambda x.$*argmax/min*$(g(x),f(x))$ NP/N : $\lambda g.\lambda x.$*argmax*$(g(x),\lambda d.$dbo:birthDate$(x,d))$

The learning procedure was run for 10 iterations over all 200 training examples. The output is a list of lexical items paired with a syntactic and semantic representation and a weighted score. The score of domain-independent expressions is initially set to 1, so that they always end up on the top-k stacks. Furthermore, these scores are not updated during the iterations. All other items generated by GENLEX start with a score of 0.

6.3 Evaluation

After training the ZC05 algorithm on the QALD-4 training set, the learned model was tested on the QALD-4 test set, comprising of 50 questions. We excluded questions that require YAGO classes, `UNION`s, `ORDER BY` statements and `FILTER`s, leaving 37 questions with respect to which the results produced by the semantic parsing approach were compared to the QALD-4 gold standard results. For each question q, precision, recall and F-measure were computed as follows:

$$Recall(q) = \frac{\text{number of correct system answers for } q}{\text{number of gold standard answers for } q}$$

$$Precision(q) = \frac{\text{number of correct system answers for } q}{\text{number of system answers for } q}$$

$$F\text{-}Measure(q) = \frac{2 \times Precision(q) \times Recall(q)}{Precision(q) + Recall(q)}$$

Since the QALD-4 training queries cover only a small part of the DBpedia vocabulary, we decided to increase lexical coverage of the system by adding a lexical item for each DBpedia

predicate and class on the basis of their label, according to the GENLEX rules in Table 6.2 as "ontology labels".

	Precision	Recall	F-measure
Learned lexicon + ontology labels	0.66	0.05	0.09
Learned lexicon + ontology labels + handcrafted items	0.93	0.70	0.80
Learned lexicon + ontology labels + M-ATOLL	0.70	0.18	0.30

TABLE 6.3: Evaluation results of applying ZC05 semantic parsing on the QALD-4 test dataset.

The test results are given in the first row of Table 6.3. Most prominently, recall turns out to be very low. This is because most of the expressions in the test questions do not appear either in the training data or among the DBpedia labels. Thus, the system lacks a great deal of lexical knowledge of expressions that were not seen during training. For example, to answer the question *Who was the first to climb Mount Everest*, the system would need a lexical item such as the following one:

$$\textit{first to climb} : \text{N/NP} : \lambda x.\lambda y.\texttt{dbo:firstAscentPerson}(x,y)$$

Such an item is not present in the induced lexicon, neither it is contained among the ontology labels. Therefore, we need external lexical resources in such cases to bridge the lexical gap. In order to test how much additional lexical knowledge is needed, we manually handcrafted lexical items for the test data. Some examples are given in Table 6.4.

Phrase	Syntax	Semantics
first to climb	N/NP	$\lambda x\lambda y.\texttt{dbo:firstAscentPerson}(x,y)$
artistic movement	N	$\lambda x\lambda y.\texttt{dbo:movement}(x,y)$
launched from	(S\NP)/NP	$\lambda x\lambda y.\texttt{dbo:launchPad}(y,x)$
extinct	N	$\lambda x.\texttt{dbo:conservationStatus}(x,\texttt{'EX'})$
German	N/N	$\lambda g\lambda x.g(x) \wedge \texttt{dbo:country}(x,\texttt{dbr:Germany}))$
taikonauts	N	$\lambda x.\texttt{rdf:type}(x,\texttt{dbo:Astronaut})$
		$\wedge\texttt{dbo:nationality}(x,\texttt{dbr:China})$

TABLE 6.4: Samples from manually hand-crafted lexical items for the QALD-4 test dataset.

In total we created 54 lexical items. The results using those additional lexical items are presented in the second row in Table 6.3, showing that recall is now significantly increased as well as an increase in precision is observed. Thus, the system shows remarkable improvements by using the handcrafted lexical items. However, for large domains the required manual effort is not always feasible. Therefore, we ran M-ATOLL (Walter et al., 2014), a system that automatically extracts lexicalizations for ontology elements from a text corpus, on the predicates used in the training dataset. It managed to find 10 of the required 54 lexical items. Results using lexical items per predicate that were automatically extracted by M-ATOLL are shown in the third row in Table 6.3. Note here that the additional recall is gained at the cost of a reduced precision.

Next, we compare results to the systems that participated in the QALD-4 challenge. The evaluation results of all these systems are given in Table 6.5, based on all 50 test questions (not just 37 as in Table 6.3 where the unhandled cases are taken as incorrect interpretation

in order to compare fairly). We added the three different settings of our system as explained above: *ZC05* uses learned lexical entries together with ontology labels, *ZC05+handcrafted* uses handcrafted lexical entries in addition, whereas *ZC05+M-ATOLL* uses entries generated by M-ATOLL in addition.

	Total	Proc.	Right	Part.	Recall	Precision	F-measure
Xser (Xu et al., 2014b)	50	40	34	6	0.71	0.72	**0.72**
ZC05 + handcrafted	50	28	26	0	0.52	0.93	**0.67**
gAnswer (Zou et al., 2014b)	50	25	16	4	0.37	0.37	**0.37**
CASIA (Shizhu et al., 2014)	50	26	15	4	0.40	0.32	**0.36**
Intui3 (Dima, 2014b)	50	33	10	4	0.25	0.23	**0.24**
ISOFT (Park et al., 2014b)	50	28	10	3	0.26	0.21	**0.23**
ZC05 + M-ATOLL	50	10	7	0	0.14	0.70	**0.23**
ZC05	50	3	2	0	0.04	0.66	**0.07**

TABLE 6.5: Comparing ZC05 semantic parsing approaches with the systems participated in QALD-4.

The evaluation results show that ZC05 can be applied for QALD datasets as long as there is a way to obtain lexical items. ZC05+handcrafted achieves 0.67 scores compared to 0.23 by ZC05+M-ATOLL. Comparing automatic systems with another system that uses handcrafted lexical items provides important insights to what is important in building QA system. We can see that adding more accurate lexical items can give better results.

6.4 Discussion

The approach can only be applied to a single language because it depends on two language-specific data: CCG combination rules and domain-independent lexicon. We showed that ZC05 can be applied to an open-domain dataset such as QALD-4 even though the underlying algorithm has been applied on closed-domain dataset such as Geoquery. However, as suggested by results shown in Table 6.5, the approach does not generalize well for unseen data. It works well for closed-domain datasets since the model learns to associate words from sentences with certain semantic representations. It is mainly due to the fact that the variability of natural language expressions appearing in test is not covered in the train data for the QALD-4 dataset. It is more challenging compared to closed-domain datasets where the similar natural language expressions can be found in both train and test splits.

Adding more training instances for the task can solve the issue of the lexical gap. However, to create more training instances and defining domain-independent lexica requires domain expertise and it can be costly. Thus, the approach did not generalize well compared to other systems. Adding lexical items from M-ATOLL improved the results. Adding handcrafted lexicon improved the results significantly from 0.23 to 0.67 since the lexicon derived by M-ATOLL did not have the adequate coverage needed for this experiment. It shows that the system suffers more from lexical gap rather than the errors occur from the pipeline components that are learned, e.g. syntactic parser or semantic parser. Therefore, we can conclude that it is possible to apply the ZC05 semantic parsing approach on any dataset that has either a high number of training instances or limited vocabulary. Moreover, the approach needs domain expertise in adapting the model to other languages since it is based on language-specific CCG combination rules and domain-independent lexical items.

Chapter 7

Dependency parse tree-based Semantic Parsing Approach

In this chapter, we present another semantic parsing approach for building question answering systems. The approach abstracts from the underlying language by using cross-lingual dependency parse trees from Universal Dependencies where a single pipeline can be built for multilingual setting. The pipeline is adapted for English, German and Spanish languages evaluated on the QALD-6 dataset. The content provided in this chapter is based on our previously published work (Hakimov et al., 2017).

7.1 Overview

We present another semantic parsing approach for question answering. The approach differs from the CCG-based Semantic Parsing Approach in how the semantics of the natural language questions is composed. The approach uses the dependency parse tree of a question as syntax and trains a model that learns the mapping from syntax to semantics while CCG-based Semantic Parsing Approach learns syntax and semantics in tandem. Specifically, this approach uses the dependency relations in the parse trees. This approach presents a multilingual architecture where the dependency parse tree syntax are based on Universal Dependencies (Nivre et al., 2016; Nivre, 2017). The approach is applied for English, German and Spanish and evaluated on the QALD-6 dataset. In the next sections, we give the detailed description of the approach then followed by the evaluation and discussion sections.

7.2 Dependency Parse Tree-based Approach

Our intuition for this system is that the interpretation of a natural language question in terms of a SPARQL query is a compositional process. Such process includes composing a meaning representation of a sentence by combining partial semantic representations of smaller units with each other in a bottom-up fashion along a dependency tree. The dependency parse tree of a question guides the semantic composition where the dependency relations between nodes and their POS tags define the result of composition. Instead of relying on hand-crafted rules guiding the composition, we rely on a learning approach that can infer such 'rules' from training data. We employ a factor graph model trained using a ranking objective and the SampleRank algorithm.

The model learns to prefer good over bad interpretations of a question. In essence, an interpretation of a question represented as a dependency tree consists of an assignment of several variables: i) a knowledge base identifier (KB ID) and semantic type to every node in the parse tree, and ii) an argument index (1 or 2) to every edge in the dependency tree specifying which slot of the parent node, subject or object, the child node should be applied

to. The input to our approach is thus a set of pairs (q, sp) of a question q and a SPARQL query sp.

Consider the following questions that ask the same information in English, German & Spanish respectively:

- *Who created Wikipedia?*
- *Wer hat Wikipedia gegründet?*
- *¿Quién creó Wikipedia?*

Independently of the language they are expressed in, the three questions can be interpreted with the same SPARQL query as given below.

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>
SELECT DISTINCT ?uri WHERE {
   dbr:Wikipedia dbo:author ?uri .
}
```

The approach consists of generating multiple states where the solution to the problem partially or fully exist. The learning model essentially learns to rank the generated states. In order to generate states, we apply an inference method in particular Markov Chain Monte Carlo (MCMC). Our inference method consists of two layers that we call Linking to Knowledge Base (L2KB) and Query Construction (QC). Each of these layers consists of a different state generation and factor graph optimized for different subtasks of the overall task. The first inference layer is trained using an entity linking objective that learns to link parts of the query to KB Identifiers. In particular, this inference step assigns KB Identifiers to open class words such as nouns, proper nouns, adjectives,verbs, etc. We use Universal Dependencies[1](Nivre et al., 2016; Nivre, 2017) to get dependency parse trees for three languages. The second inference layer is a query construction layer that takes the top k results from the L2KB layer and assigns semantic representations to closed class words such as question pronouns, determiners, etc. to yield a logical representation of the complete question. The approach is trained on the QALD-6 train dataset for English, German & Spanish questions to optimize the parameters of the model. The model learns mappings between the dependency parse tree for a given question text and RDF nodes in the SPARQL query. As an output, our system produces an executable SPARQL query for a given natural language question. All data and source code are freely available[2]. As semantic representations, we rely on DUDES, which are described in Section 2.3.3. In the next sections, we explain all of the steps mentioned above about inferencing, parameter learning as well as the evaluation of the proposed method with a separate discussion section.

7.2.1 Representation with Factor Graphs

We provide the formal definition of factor graphs in Section 2.4. In this section, we define how the factor graph model is used along with definition of observed and hidden (also known as latent) variables for our task. Input to our approach is a pair (W, E) consisting of a sequence of words $W = \{w_1, \ldots, w_n\}$ and a set of dependency edges $E \subseteq W \times W$ forming a tree. A state $(W, E, \alpha, \beta, \gamma)$ represents a partial interpretation of the input in terms of partial semantic representations. The partial functions $\alpha : W \rightarrow KB$ maps words to KB identifiers, $\beta : W \rightarrow \{t_1, t_2, t_3, t_4, t_5\}$ maps words to the five basic DUDES types, and

[1] http://universaldependencies.org/v2, 70 treebanks, 50 languages
[2] https://github.com/ag-sc/AMUSE

$\gamma : E \rightarrow \{1,2\}$ maps edges to indices of semantic arguments, with 1 corresponding to the subject of a property and 2 corresponding to the object, respectively.

Observed variables are nodes and edges in a parse tree as shown in Figure 7.1.

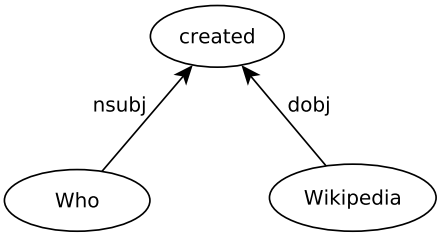


FIGURE 7.1: Observed variables : nodes and edges in a dependency parse tree

Figure 7.2 shows a schematic visualization of a question along with its factor graph. Hidden variables are depicted with dashed lines. Each node gets assigned a pair of Knowledge Base Identifier (KB ID) and a Semantic Type (DUDES type). Each edge gets a Slot Number, the ones connecting nodes with assigned KB ID and DUDES type. There can be also nodes without any KB ID and Semantic Types and edge Slot Number. For instance, determiners don't provide a meaning to the sentence in terms of the semantic representation.

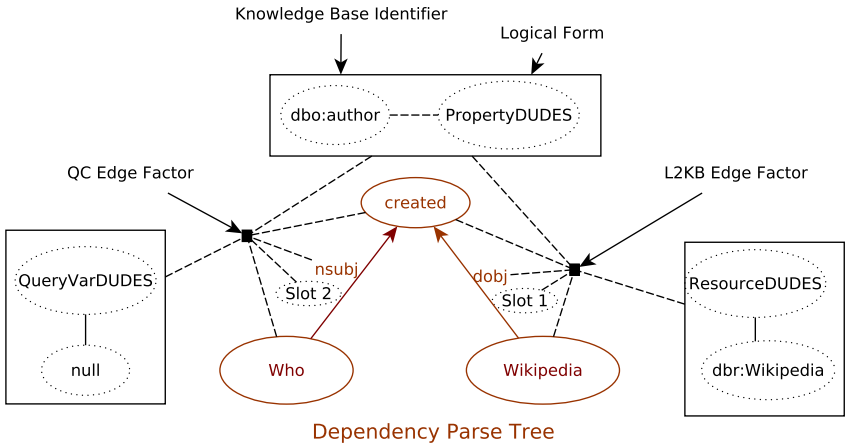


FIGURE 7.2: Factor graph for the question: *Who created Wikipedia?*. Observed variables are depicted as bubbles with straight lines; hidden variables as bubbles with dashed lines. Black boxes represent factors.

For a given input consisting of a dependency parsed sentence, the factor graph is rolled out by applying template procedures that match over parts of the input and generate corresponding factors. The templates are thus imperatively specified procedures that roll out the graph.

A template $T_j \in \mathcal{T}$ defines the subsets of observed and hidden variables (x', y') with $x' \in X_j$ and $y' \in Y_j$ for which it can generate factors and a function $f_j(x', y')$ to generate features for these variables. Additionally, all factors generated by a given template T_j share the same parameters θ_j. With this definition, we can reformulate the conditional probability as follows:

$$p(y|x;\theta) = \frac{1}{Z(x)} \prod_{T_j \in \mathcal{T}} \prod_{(x',y') \in T_j} e^{f_j(x',y') \cdot \theta_j} \tag{7.1}$$

where $Z(x)$ is the normalization function. We define a probability distribution over possible configurations of observed and hidden variables. This enables us to explore the joint space of observed and hidden variables in a probabilistic fashion.

7.2.2 Inference

We rely on an approximate inference procedure, in particular Metropolis–Hastings (Hastings, 1970; Metropolis et al., 1953) which is a powerful Markov chain method to simulate multivariate distributions. More detailed description on inferencing is given in Section 2.4. The method performs iterative inference for exploring the state space of possible question interpretations by proposing concrete changes to sets of variables that define a proposal distribution. The inference procedure performs an iterative local search and can be divided into the following steps:

1. Generatine possible successor states for a given state by applying changes
2. Score the states using the model score
3. Decide which proposal to accept as a successor state

A proposal is accepted with a probability that is proportional to the likelihood assigned by the distribution p (see Equation 7.1). To compute the logical form of a question, we run two inference procedures using two different models. The first model L2KB is trained using a linking objective that learns to map open class words to KB identifiers. The sampling process is run for m steps for the L2KB model; the top k states are used as an input for the second inference model called QC that assigns meanings to closed class words to yield a full fledged semantic representation of the question. This process is illustrated in Figure 7.3. Both inference strategies generate successor states by exploration based on edges in the dependency parse tree. We explore only the following types of edges: *Core arguments, Non-core dependents, Nominal dependents* defined by Universal Dependencies[3] and nodes that have the following POS tags: NOUN, VERB, ADJ, PRON, PROPN, DET. In both inference strategies, we alternate across iterations between using the probability of the state given the model and the objective score to decide which state to accept. Initially, all partial assignments $\alpha_0, \beta_0, \gamma_0$ are empty.

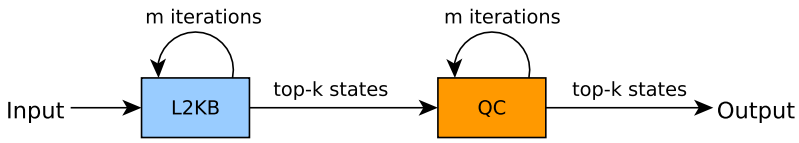


FIGURE 7.3: Inference Architecture

We rely on an inverted index to find all KB IDs for a given query term. The inverted index maps terms to candidate KB IDs for all three languages. We describe the details of this index in Section 5.2. Entries in the inverted index are grouped by DUDES types, so that it supports type-specific retrieval. The index stores the frequency of the mentions paired with KB ID. During retrieval, the index returns a normalized frequency score for each candidate KB ID. Next, we describe the two inferecing strategies.

[3] http://universaldependencies.org/u/dep/index.html

L2KB: Linking to Knowledge Base

Proposal Generation: The L2KB proposal generation proposes changes to a given state by considering single dependency edges and changing the following:

1. The KB IDs of parent and child nodes
2. The DUDES type of parent and child nodes
3. The argument index attached to the edge. The DUDES Type variables range over the five basic DUDES types defined, while the argument index variable ranges in the set {1,2}.

The resulting partial semantic representations for the dependency edge are checked for availability with respect to the knowledge base, pruning the proposal if it is not satisfiable. It means that the semantic representations that don't lead to an executable query are not passed into the next step. Figure 7.4 depicts the local exploration of the *dobj*-edge between *Wikipedia* and *created*.

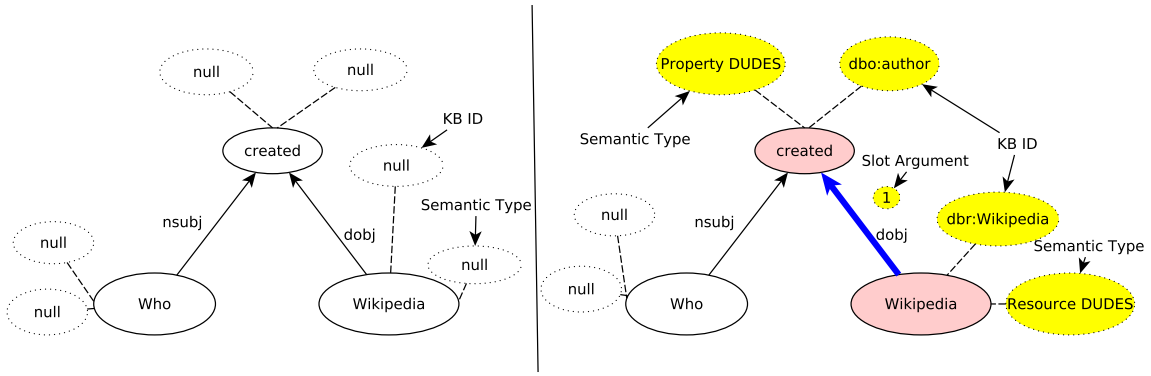
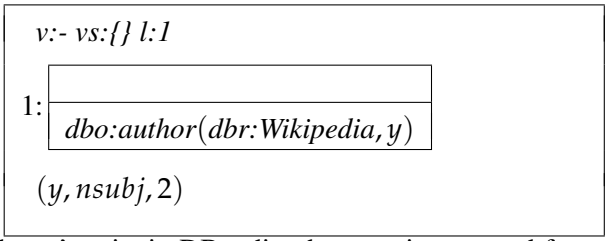


FIGURE 7.4: **Left**: Initial state based on dependency parse where each node has empty KB ID and Semantic Type. **Right**: Proposal generated by the LKB proposal generation for the question *Who created Wikipedia?*

The left image shows an initial state with empty assignments for all hidden variables. The right image shows a proposal that changes the KB IDs and DUDES types of the nodes connected by the *dobj* edge. The inference process has assigned the KB ID *dbo:author* and the *Property DUDES* type to the *created* node. The *Wikipedia* nodes gets assigned the type *Resource DUDES* as well as the KB ID *dbr:Wikipedia*. The dependency edge gets assigned the argument index 1, representing that *dbr:Wikipedia* should be inserted at the subject position of the *dbo:author* property.

These assignments are valid because once the KB IDs are combined it leads to the triple *dbr:Wikipedia dbo:author ?o.* that exists in DBpedia. As it is satisfiable, it is not pruned. In contrast, a state in which the edge is assigned the argument index 2 would yield the following non-satisfiable representation, corresponding to things that were authored by *Wikipedia* instead of things that authored *Wikipedia*:



As such query doesn't exist in DBpedia, the state is removed from candidate successor states.

We generate modified new states that differ from the current state s_t with three changes at most. Specifically, the modified state $s'_{ij} = (\mathbf{W}, \mathbf{E}, \mathbf{U'_{ij}}, \mathbf{S'_{ij}}, \mathbf{T'_{ij}})$ comprises the same observed variables $\mathbf{W}$ and $\mathbf{E}$, but has three changes on "hidden" variables $(\mathbf{U'_{ij}}, \mathbf{S'_{ij}}, \mathbf{T'_{ij}})$, URIs, Slots and Semantic Types.

Objective Function: As objective for the L2KB model, we rely on a linking objective that calculates the overlap between inferred KB IDs and KB IDs in the gold standard SPARQL query.

All generated states are ranked by the objective score. The top-k states are passed to the next sampling step. In the next iteration, the inference is performed on these k states. Following this procedure for m iterations yields a sequence of states $(s_0, \ldots, s_m)$ that are sampled from the distribution defined by the underlying factor graphs.

QC: Query Construction

Proposal Generation: Proposals in this inference layer consist of assignments of the type *QueryVar DUDES* to nodes for class words, in particular determiners, that could fill the argument position of a parent with unsatisfied arguments at the same time assigning the missing Slot Number depending on the parent nodes' DUDES Types.

Objective Function: As objective we use a function that measures the (graph) similarity between the inferred SPARQL query and the gold standard SPARQL query. The inferred SPARQL query is constructed using semantic composition on dependency parse trees using DUDES as semantic representations (see Section 2.3.4).

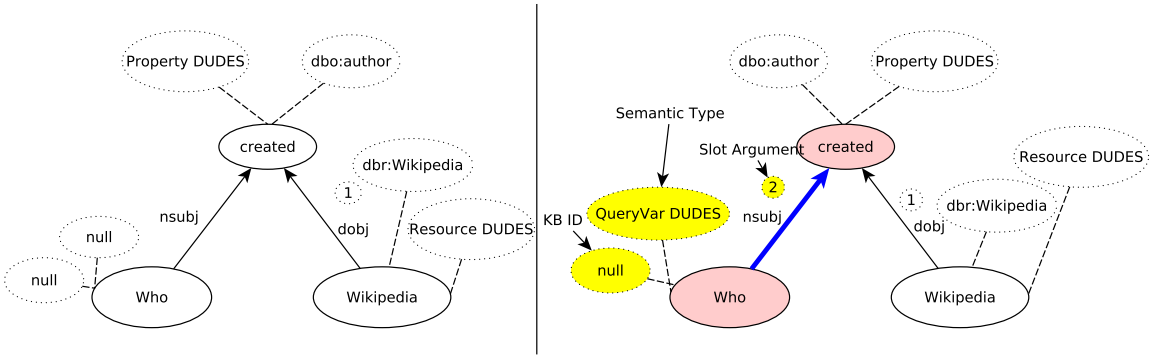
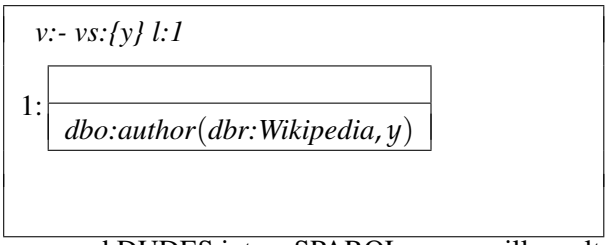


FIGURE 7.5: **Left**: Input state; **Right**: Proposal generated by the QC proposal generation for the question *Who created Wikipedia?*

Figure 7.5 shows an input state and a sampled state for the QC inference layer of our example query: *Who created Wikipedia?*. The initial state (see Left) has Slot 1 assigned to the edge *dobj*. Property DUDES have two slots by definition. The right figure shows a proposed state in which the argument slot 2 has been assigned to the nsubj edge and the *QueryVar DUDES* type has been assigned to node *Who*. We apply bottom-up semantic composition on the dependency parse tree and compose the following DUDES.



Converting the composed DUDES into a SPARQL query will result in the following:

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>
SELECT DISTINCT ?y WHERE {
     dbr:Wikipedia dbo:author ?y .
}
```

We apply two inference methods L2KB and QC as described above to generate a state that gives the correct query once we apply semantic composition. In the next section, we give a pseudo-code of a sampling method that generates states, which is used by L2KB and QC.

Candidate State Generation Algorithm

Below in Algorithm 3, we present the pseudo-code for sampling strategy that generates candidate states by exploring the edges of a dependency parse tree. The input to the algorithm is the current state s_t, all DUDES types and the dependency parse tree that contains a set of edges and words that edges connect to. The sampling process generates all possible candidate states. The algorithm explores every possible edge that connects two words in the parse tree (Line 3). Each edge connects two words w_i and w_j. We pair each word with a DUDES type t and query the inverted index to retrieve the matching list of candidate URIs (Lines 5-8). The list of URI candidates for a word, e.g. w_i, is paired with the other word's, e.g. w_j, URI candidates. Instead of creating all possible candidate pairs, we apply a pruning step that checks whether the pair results in a valid triple (Lines 9-19). The slot number is sampled during the pair generation where the slot number 1 indicates that the child node is in the subject position of the triple (Line 16-18). Similarly, the slot number 2 indicates that the child node is in the object position (Line 11-12). If the constructed triple exists in DBpedia then the candidate state s'_{ij} is generated. The candidate state s'_{ij} differs from the current state in three changes: the URI assignments for words w_i and w_j along with the slot number. The generated state is added to the list of candidate states S. The output from the algorithm is newly generated list of candidate states S.

Algorithm 3 Candidate state generation algorithm by traversing the dependency parse tree

Inputs: current state s_t, DUDES types T, dependency parse tree consisting of a sequence of words $W = \{w_1, \ldots, w_n\}$ and a set of edges $E \subseteq W \times W$
Output: sampled states S
for `the edge` e `in` E **do**
  `the edge` e `connects two words` w_i `and` w_j `in the parse tree`
  for `type` t_i `in` T **do**
    U_i = `query(`w_i`,` t_i`)` ▷ retrieves matching URIs based on the lemma of w_i and the given DUDES type t_1
    for `type` t_j `in` T **do**
      U_j = `query(`w_j`,` t_j`)` ▷ retrieves matching URIs based on the lemma of w_j and the given DUDES type t_j
      for `candidate URI` u_i `in` U_i **do**
        for `candidate URI` u_j `in` U_j **do**
          if the triple (?s, u_i, u_j) exists in DBpedia **then** ▷ the URI u_i is in the predicate and the URI u_j is in the object position. The subject position contains a variable ?s
            $slot_{ij} = 2$
            $s'_{ij} = (\mathbf{W}, \mathbf{E}, \mathbf{U'_{ij}}, \mathbf{S'_{ij}}, \mathbf{T'_{ij}})$ where $u_i, u_j \in \mathbf{U'_{ij}}, slot_{ij} \in \mathbf{S'_{ij}}, t_i, t_j \in \mathbf{T'_{ij}}$
            $S = S \cup s'_{ij}$
          end if
          if the triple (u_j, u_i, ?o) exists in DBpedia **then** ▷ the URI u_j is in the subject and the URI u_i is in the predicate position. The object position contains a variable ?o
            $slot_{ij} = 1$
            $s'_{ij} = (\mathbf{W}, \mathbf{E}, \mathbf{U'_{ij}}, \mathbf{S'_{ij}}, \mathbf{T'_{ij}})$ where $u_i, u_j \in \mathbf{U'_{ij}}, slot_{ij} \in \mathbf{S'_{ij}}, t_i, t_j \in \mathbf{T'_{ij}}$
            $S = S \cup s'_{ij}$
          end if
        end for
      end for
    end for
  end for
end for

7.2.3 Semantic Composition

As described in the two inference methods, the goal is to generate a state composed of observed and hidden variables. The sampled state can be converted into a SPARQL query by applying the semantic composition explained in Section 2.3.4. Below in Figure 7.6, we illustrate the state that gives the expected query.

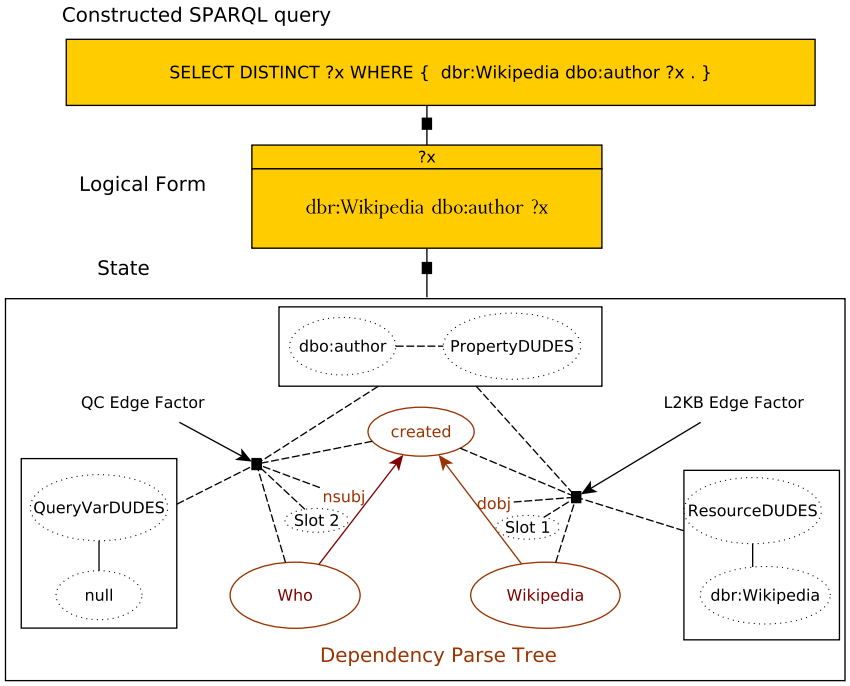
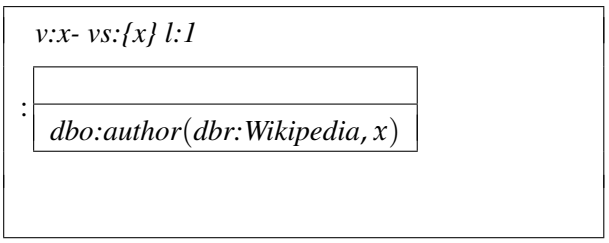


FIGURE 7.6: Factor graph for the question: *Who created Wikipedia?*. Observed variables are depicted as bubbles with straight lines; hidden variables as bubbles with dashed lines. Black boxes represent factors.

By applying the bottom-up semantic composition based on DUDES and dependency relations, we compose the following meaning representation.



This meaning representation is based on DUDES. This DUDES can be translated into the following SPARQL query as explained in Section 2.3.4.

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>
SELECT DISTINCT ?x WHERE {
      dbr:Wikipedia dbo:author ?x .
}
```

The inference methods sample a state that after being composed generates the expected SPARQL query for the given question "Who created Wikipedia?". In the next section, we describe features for the factor graph model.

7.2.4 Features

As features for the factors, we use conjunctions of the following information: i) lemma of parent and child nodes, ii) KB IDs of parent and child nodes, iii) POS tags of parent

and child nodes, iv) DUDES type of parent and child, v) slot number of an argument at edge, vi) dependency relation of edge, vii) normalized frequency score for retrieved KB IDs, viii) string similarity between KB ID and lemma of node, ix) rdfs:domain and rdfs:range restrictions for the parent KB ID (in case of being a property).

We have two templates that generate features for each inference method described before: Linking to KB and Query Construction. Each template generates a different set of features. Each feature captures different information about the explored edge, the parent and the child node. Some features include less information compared to others. This allows to create more specific and general features that scale well. We group features for each inference method as shown below.

L2KB Feature Template

In this section, we describe the template that generates features for linking phrases to knowledge base entries. We group features under four groups. Each group contains a set of features that contains information about the edge between a child and a parent node.

- **Group I**
  - Parent Lemma + Parent KB ID
  - Parent POS + Parent Semantic Type
  - Child Lemma + Child KB ID
  - Child POS + Child Semantic Type
  - Edge Dependency Relation + Slot Number
  - Parent POS + Parent Lemma + Child POS + Child Lemma
  - Parent POS + Parent Semantic Type + Child POS + Child Semantic Type
  - Parent POS + Parent Semantic Type + Child POS + Child Semantic Type + Dependency Relation + Slot Number
- **Group II**
  - Parent String Similarity >= Value + Parent Semantic Type
  - Child String Similarity >= Value + Parent Semantic Type
- **Group III**
  - Parent POS + Parent Semantic Type + Child POS + Child Semantic Type + Slot Number + Parent rdfs:range/domain
- **Group IV**
  - Parent & Child Nodes' String Similarity >= Value
  - Parent & Child Nodes' Knowledge Base Score >= Value

The argument *Value* is pre-defined. For every value that the condition holds, e.g. Parent & Child String Similarity >= Value, we add all possible feature buckets. For instance, if the pre-defined value is set to 0.7 then the following features are generated if the calculated string similarity for the parent and the child node is equal to 1.0 because all the conditions below hold since 1.0 is bigger than or equal to 0.7, 0.8, 0.9 and 1.0.

- Parent & Child String Similarity >= 1.0

- Parent & Child String Similarity >= 0.9
- Parent & Child String Similarity >= 0.8
- Parent & Child String Similarity >= 0.7

QC Feature Template

This is the template that generates features needed for constructing the query from a state. Each feature contains information about the edge that connects a parent node to the child.

- **Group I**
  - Parent POS + Parent Semantic Type + Child POS + Child Semantic Type + Dependency Relation + Slot Number
  - Parent POS + Parent Semantic Type + Child Lemma + Child POS + Child Semantic Type + Dependency Relation + Slot Number
- **Group II**
  - Parent POS + Parent Semantic Type + Child POS + Child Semantic Type + Dependency Relation + Slot Number + Parent rdfs:domain/range
  - Parent POS + Parent Semantic Type + Child POS + Child Semantic Type + Dependency Relation + Slot Number + Parent rdfs:domain/range + FIRST_TOKEN

As mentioned earlier, some features include more information than others. For instance, the feature with information such as "Parent Lemma + Parent KB ID" captures information only about the parent node whereas a feature such as "Parent POS + Parent Semantic Type + Child POS + Child Semantic Type" abstracts from actual lemmas of nodes and capture information in terms of POS tags and Semantic Types (DUDES type), which generalizes better since the same lemmas may not appear during prediction.

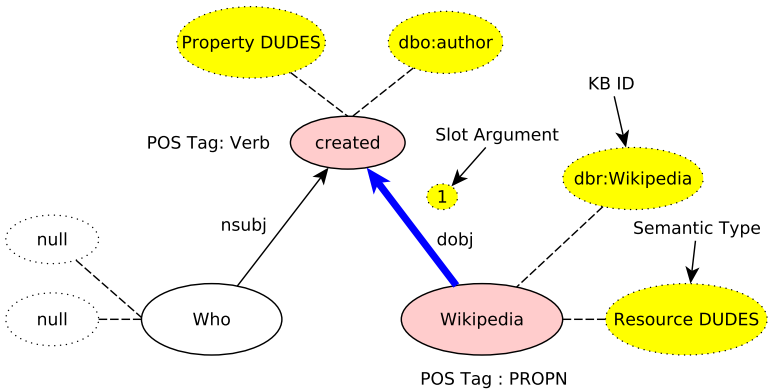


FIGURE 7.7: Features for Linking to KB task

L2KB Generated Features

In Figure 7.7, we have shown node information for the highlighted edge **dobj** between nodes *created* and *Wikipedia*. We generate the following features shown below from this edge using the L2KB Feature Template and all features are added in the same order as described above. Feature Group II generates features depending on the similarity score. As seen in the

example below, we added three features where the string similarity between the lemma of the child node "Wikipedia" and the assigned KB ID *dbr:Wikipedia* is a full match, 1.0. We add buckets of features starting from 0.7 up to the actual value of the similarity measure. Note here that, the similarity score between parent lemma and KB ID is not higher than 0.7 and that's why there aren not any features. Similarly, GROUP IV adds features based on the joint condition of parent & child nodes' string similarity and knowledge base score. Knowledge base scores for resources are normalized values retrieved from the surface form index and for properties it is either the M-ATOLL score or word embedding score depending on how the property was retrieved. In our example, we added only knowledge base score features on the joint condition because the child KB score and the parent KB score are higher than 0.7. GROUP III added the feature with additional information of parent KB ID's **rdfs:domain** restriction. This is bound to the slot number. If the slot number is 1 then the **rdfs:domain** is added, if the slot number is 2 then **rdfs:range** restriction is added. In this case, we added **rdfs:domain:** ***dbo:Work*** since the slot number was chosen as 1.

- **Group I**
  - Parent Lemma: *created* + Parent KB ID: *dbo:author*
  - Parent POS: *VERB* + Parent Semantic Type: *Property*
  - Child Lemma: *Wikipedia* + Child KB ID: *dbr:Wikipedia*
  - Child POS: *PROPN* + Child Semantic Type: *Resource*
  - Edge Dependency Relation: *dobj* + Slot Number: *1*
  - Parent POS: *VERB* + Parent Lemma: *created* + Child POS: *PROPN* + Child Lemma: *Wikipedia*
  - Parent POS: *VERB* + Parent Semantic Type: *Property* + Child POS: *PROPN* + Child Semantic Type: *Resource*
  - Parent POS: *VERB* + Parent Semantic Type: *Property* + Child POS: *PROPN* + Child Semantic Type: *Resource* + Dependency Relation: *dobj* + Slot Number: *1*
- **Group II**
  - Child String Similarity >= *1.0* + Child Semantic Type: *Resource*
  - Child String Similarity >= *0.9* + Child Semantic Type: *Resource*
  - Child String Similarity >= *0.8* + Child Semantic Type: *Resource*
  - Child String Similarity >= *0.7* + Child Semantic Type: *Resource*
- **Group III**
  - Parent POS: *VERB* + Parent Semantic Type: *Property* + Child POS: *PROPN* + Child Semantic Type: *Resource* + Slot Number: *1* + Parent rdfs:domain: *dbo:Work*
- **Group IV**
  - Parent & Child Knowledge Base Score >= 1.0
  - Parent & Child Knowledge Base Score >= 0.9
  - Parent & Child Knowledge Base Score >= 0.8
  - Parent & Child Knowledge Base Score >= 0.7

QC Generated Features

In Figure 7.8, we show a node information for the highlighted edge **nsubj** between nodes *created* and *Wikipedia*. We generate the following features shown below from this edge using the QC Feature Template and all features are added in the same order as described above. QC features are aimed to learn the structure of the query and fill the missing slots. As shown in Figure 7.8, we added **rdfs:range** restriction coupled with slot number *2* and other information on the edge. Additionally, we also mark the first token in the question, which might give clues on whether the query should be ASK or SELECT.

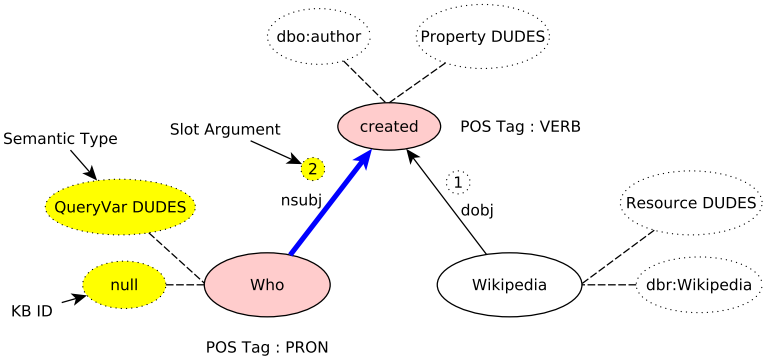


FIGURE 7.8: Features for Query Construction task

- **Group I**
  - Parent POS:*VERB* + Parent Semantic Type: *Property* + Child POS: *PRON* + Child Semantic Type: *QueryVar* + Dependency Relation: *nsubj* + Slot Number: *2*
  - Parent POS: *VERB* + Parent Semantic Type: *Property* + Child Lemma:*Who* + Child POS: *PRON* + Child Semantic Type: *QueryVar* + Dependency Relation: *nsubj* + Slot Number: *2*
- **Group II**
  - Parent POS: *VERB* + Parent Semantic Type: *Property* + Child POS: *PRON* + Child Semantic Type: *QueryVar* + Dependency Relation: *nsubj* + Slot Number: *2* + Parent rdfs:range: *dbo:Person*
  - Parent POS: *VERB* + Parent Semantic Type: *Property* + Child POS: *PRON* + Child Semantic Type: *QueryVar* + Dependency Relation: *nsubj* + Slot Number: *2* + Parent rdfs:range: *dbo:Person* + FIRST_TOKEN: *Who*

7.3 Learning Model Parameters

In order to optimize model parameters θ, we use an implementation of the SampleRank Wick et al. (2009) algorithm (see Section 2.4). The SampleRank algorithm obtains gradients for these parameters from pairs of consecutive states in the chain based on a preference function $\mathbb{P}$ defined in terms of the objective function $\mathbb{O}$ as follows:

$$\mathbb{P}(s', s) = \begin{cases} 1, & \text{if } \mathbb{O}(s') > \mathbb{O}(s) \\ 0, & \text{otherwise} \end{cases} \tag{7.2}$$

We have observed that accepting proposals only on the basis of the model score requires a large number of inference steps. This is due to the fact that the exploration space is huge considering all the candidate resources, predicates, classes etc. in DBpedia. To guide the search towards good solutions, we switch between model score and the objective score to compute the likelihood of acceptance of a proposal. Once the training procedure switches the scoring function in the next sampling step, the model uses the parameters from the previous step to score the states.

In the next section, we provide experiments for evaluating the performance of the proposed approach.

7.4 Evaluation

We present experiments carried out on the QALD-6 dataset comprising of English, German & Spanish questions. We train and test on the multilingual subtask. This yields a training dataset consisting of 350 and 100 test instances. We train the model with 350 training instances for each language from QALD-6 train split by performing 10 iterations over the dataset with learning rate set to 0.01 to optimize the parameters. We set k to 10. We use the top 5 candidates from the Resource index, 25 candidates from the Predicates index, 25 candidates from Classes index and 25 candidates from the Restriction Classes index during retrieval of KB IDs for mentions. We perform a preprocessing step on the dependency parse tree before running through the pipeline. This step consists of merging nodes that are connected with compound edges. This results in having one node for compound names and reduces the traversing time and complexity for the model. The approach is evaluated on two tasks: a linking task and a question answering task. The linking task is evaluated by comparing the proposed KB links to the KB elements contained in the SPARQL question in terms of F-Measure. The question answering task is evaluated by executing the constructed SPARQL query over the DBpedia, and comparing the retrieved answers with answers retrieved for the gold standard SPARQL query in terms of F1-measure.

In order to contextualize our results, we provide an upper bound for our approach, which consists of running over all instances in test using 1 epoch and accepting states according to an objective score only, which compares the accepted state to the ground. Thus, accepting states according to an objective score yields an oracle-like approach. We report test results based on training a model and performing an inference on test instances using the model score.

We report Macro F1 scores for this oracle in Table 7.1 together with the actual results on test. The oracle results are obtained using objective functions for both task while test results are obtained using the trained model scores.

We evaluate different configurations of our system in which we consider different source for lexical mapping as given below:

1. DBP: a dictionary derived only from DBpedia labels
2. M-ATOLL: additional dictionary entries derived from the M-ATOLL
3. Embed: entries inferred using cosine similarity in the embedding space
4. Dict: a manually created dictionary

TABLE 7.1: Macro F1-scores on test data for the linking and question answering tasks using different configurations

Language	Task	DBP	DBP + M-ATOLL	DBP + M-ATOLL + Embed	DBP + M-ATOLL + Dict
Oracle					
EN	Linking	0.05	0.22	0.46	0.59
EN	QA	0.05	0.21	0.30	**0.51**
DE	Linking	0.01	0.01	0.10	0.48
DE	QA	0.04	0.04	0.18	**0.44**
ES	Linking	0.02	0.04	0.10	0.51
ES	QA	0.04	0.06	0.22	**0.52**
Test					
EN	Linking	0.05	0.13	0.16	0.22
EN	QA	0.05	0.20	**0.26**	**0.34**
DE	Linking	0.01	0.01	0.10	0.27
DE	QA	0.04	0.04	**0.16**	**0.37**
ES	Linking	0.02	0.02	0.04	0.30
ES	QA	0.04	0.04	**0.20**	**0.42**

It is important to note that even the oracle does not get perfect results, which is due to the fact that the lexical gap still persists and some entries can not be mapped to the correct KB IDs. Furthermore, errors in POS tagging or in the dependency tree prevent the inference strategy to generate the correct proposals.

We see that in all configurations, results clearly improve when using additional entries from the M-ATOLL in comparison to only using labels from DBpedia. The results further increase by adding lexical entries inferred via similarity in embedding space (+Embed), but are still far from the results with manually created dictionary (Dict), showing that addressing the lexical gap is an important issue to increase performance of question answering systems over linked data.

On the linking task, while the use of embeddings increases performance as seen in the DBP + M-ATOLL + Embed vs. DBP + M-ATOLL condition, there is still a clear margin to the DBP + M-ATOLL + Dict condition (English 0.16 vs. 0.22, German 0.10 vs. 0.27, Spanish 0.04 vs. 0.30).

On the QA task, adding embeddings on top of DBP + M-ATOLL also has a positive impact, but is also lower compared to the DBP + M-ATOLL + Dict condition (English 0.26 vs. 0.34, German 0.16 vs. 0.37, Spanish 0.20 vs. 0.42). Clearly, one can observe that the difference between the learned model and the oracle diminishes the more lexical knowledge is added to the system.

7.4.1 Error Analysis

We analyzed the errors made during prediction and grouped them below.

- **Wrong resource**: 30%

  This errors occurs when the system predicts the wrong resource for a question or does not link to any resource. For instance, for the following question the model could not produce any linking for the entity “Boston Tea Party”.

  Question: When did the Boston Tea Party take place?

  Expected Query:

```
PREFIX dbr: <http://dbpedia.org/resource/>
```

```
PREFIX dbp: <http://dbpedia.org/property/>
SELECT DISTINCT ?d WHERE {
   dbr:Boston_Tea_Party dbp:date ?d
}
```

- **Wrong property**: 48%

  It is the most common error type since identifying the correct property is more challenging than finding the correct resource or a class. For instance, the model identified the wrong property for the following question.

  Question: Who wrote the song Hotel California?

  Expected Query:

```
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT DISTINCT ?o WHERE {
    dbr:Hotel_California dbo:writer ?o.
}
```

  Predicted Query:

```
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT DISTINCT ?o WHERE {
    dbr:Hotel_California dbo:musicalArtist ?o.
}
```

  The expected query is supposed to have the property *dbo:writer* instead the model predicts the property *dbo:musicalArtist*. Even though the predicted property makes sense in this case, the returned data from DBpedia is not the expected one. The expected query returns the following answer: Glen Frey, Don Henley, Don Felder. The inferred query returns the answer: Eagles (band). We can see in this case that the query makes sense because Glen Frey, Don Henley and Don Felder were members of the famous band Eagles. However, it counts as wrong interpretation.

- **Wrong Slot Number**: 10%

  As mentioned earlier the slot numbers define the position of the child node in parent node's semantic representation. Incorrect slot number would result in incorrect query. An example of such error is shown below.

  Question: How many people live in Poland?

  Expected Query:

```
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT DISTINCT ?o WHERE {
    dbr:Poland dbo:populationTotal ?o.
}
```

  Predicted Query:

```
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT DISTINCT ?v1 WHERE {
   ?v1 dbo:populationTotal ?v2.
   ?v1 dbo:residence dbr:Poland
}
```

We can see that the expected query has a property *dbo:populationTotal* that expects the resource *dbr:Poland* to be on subject position (or slot number 1). However, the predicted query has the resource on the wrong slot and it has an additional property *dbo:residence*. The slot number for the resource in the predicted query is 2.

- **Wrong Query Type**: 12%

  This error occurs when the type of queries do not match. The types of queries could be SELECT, and ASK queries. Some SELECT queries also include additional quantifiers such as COUNT or ORDER BY. We show an example of such error below.

  Question: Where does Piccadilly start?

  Expected Query:

```
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT DISTINCT ?s WHERE {
   ?s dbo:routeStart dbr:Piccadilly.
}
```

  Predicted Query:

```
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>
ASK WHERE {
   ?s dbo:routeStart dbr:Piccadilly.
}
```

  The model predicted the given question as ASK query where the expected query is SELECT. These type of errors are caused by not assigning QueryVar semantic type to any of the tokens in the sentences. Therefore, the constructed query does not include any return variables.

7.5 Discussion

We have evaluated the proposed model on three languages: English, German and Spanish. The results presented in Table 7.1 suggest that the model is capable of learning a multilingual semantic parser for all evaluated languages. The model was trained separately for each language using the same type of data: M-ATOLL lexicalizations, word embeddings, the DBpedia Ontology labels and the inverted index for resources. Moreover, using dependency parse trees from Universal Dependencies (UD) made it possible to use the same syntactic structures for each language because of cross-lingual syntax specifications.

By looking at the results for Linking to Knowledge Base task (Linking), we can see that for all languages there is a lexical gap even after adding lexical information from word embeddings, the M-ATOLL and the DBpedia Ontology labels. For German and Spanish this

lexical gap is even bigger because the lexicon from the M-ATOLL is not adequate to bridge the gap compared to English. If we add the hand-crafted lexicon the performance of the models increases. This shows that to achieve better results on QA tasks the model needs a lexicon with a better coverage. The oracle results suggest that the pipeline could not reach 1.0 even when added the hand-crafted lexicon, e.g. 0.59 on English. It can be explained with choice of the syntactic parser, the choice of sampling methods. As explained in Section 7.2.2, each sampling step requires assignments of URIs to both nodes connected by an edge, which generates more assignments than needed. It some cases not every node connected by an edge needs an assignment (URI and slot number) even though the edge represents a relation. An alternative sampling strategy could be sampling a single node along the dependency parse tree not sampling an edge that connects two nodes. However, the sampling process would take longer time to train and test since the number of candidate states would increase considering the number of resources in DBpedia (ca. 9 mil) and the pruning step can only be executed if at least two nodes have an assignment and an edge. Assigning two nodes and an edge would part be part of separate sampling steps since each sampling step would make a single change either on a node or an edge.

Similar analysis can be made about the Question Answering task (QA) reported in Table 7.1. Since L2KB is limited to a certain upper-bound, the QC step is affected by the performance of the L2KB step. The missed URIs from the previous step will result in the wrong SPARQL query after the QC step. It is due to the fact that the errors made during mapping the natural language phrases to KB IDs can not be recovered after the L2KB step is applied. We can see that adding word embeddings and M-ATOLL yields better results for English only. This means that the M-ATOLL lexicon for the QALD-6 questions for German and Spanish does not have a sufficient coverage. This is also shown in Chapter 5 where we evaluated the lexicon for each language without the QA pipeline. However, adding the lexicon from word embeddings improved the results for both of these languages as well as for English.

The sampling strategies described in Section 7.2.2 were chosen on the basis of empirical evaluation. The choice of making a joint assignment on an edge (child node URI, child Semantic Type, parent URI, parent Semantic Type and Slot Number) gave better performance than sampling over each assignment separately. Sampling each assignment separately suffered from the given huge search space. The search space is the whole DBpedia with approximately 6 million entities, over 2000 properties and around 800 classes. During sampling, sometimes the model pruned the needed state too early, which caused the whole process to not reaching the intended performance. Thus, we restricted each sampling step into making a joint decision on assignments for the connected nodes on an edge.

The choice of features given in Section 7.2.4 was based on the available information given a joint assignment. We can see that some features include general information such as the syntactic and semantic information without explicit lexical information (POS tag & Semantic Type) while others include more-specific in respect to the connection with lexical information (POS tag, Semantic Type & Lemma), this type of features are bound to the explored lexical information. This allows to train a model that generalizes better to unseen questions using features with more general information while at the same time mark the patterns seen on the given training data with more-specific features.

Chapter 8

Neural Network-based Semantic Parsing Approach

In this chapter, we present four different QA systems on the SimpleQuestions dataset (Bordes et al., 2015) that use the same environment for detecting named entities. Each architecture is similar to previously published systems and we compare them in terms of performance under the same environment. The content provided in this chapter is based on our previously published work (Hakimov et al., 2019).

8.1 Overview

We present four different approaches for building a question answering system. The systems are trained on the SimpleQuestions dataset (Bordes et al., 2015). The approaches differ from previously proposed ones in a way that they use word and character embeddings of words as features instead of manually defined indicator functions. Specifically, the systems are based on neural network architectures. All proposed architectures are evaluated in detail and compared to other similar methods. In the next sections, we give a detailed description of each model architecture which is then followed by the evaluation and discussion sections.

8.2 Methods

The task of Question Answering (QA) has received increasing attention in the last few years. Most research has concentrated on the task of answering factoid questions such as *Who wrote Mildred Pierced?*, yielding the answer *Stuart Kaminsky*. Typically, such answers are extracted from a knowledge base (KB). A frequently used dataset in this context is the SimpleQuestions Bordes et al. (2015) dataset, which consists of *simple* questions that can be answered with a single fact from the Freebase KB. For instance, the question above can be answered using the following triple from Freebase:

```
Subject:   m.04t1ftb (mildred_pierced)
Predicate: book.written_work.author
Object:    m.03nx4yz (stuart_kaminsky)
```

The system needs to identify the relevant entity (subject), i.e. *mildred_pierced* in the example question, and infer the appropriate predicate, i.e. *book.written_work.author*. In the case of SimpleQuestions, all questions involve a single triple, with the answer being the corresponding object. Thus, the task involves essentially predicting the subject and predicate of a triple. The answer to the given question is the object of the triple, m.03nx4yz (stuart_kaminsky).

Many different architectures have been proposed for this task, in particular many deep learning architectures. However, a systematic comparison of different architectural choices

has not been provided so far. In particular, different property predicting systems have used different approaches to identifying the entity, so that they are not directly comparable.

Using a common model for entity prediction based on an NER architecture, we consider four different architectures for the predicate prediction task:

- **BiLSTM-Softmax**: this architecture uses a standard BiLSTM softmax classifier to predict the property in a question where the output ranges over all properties seen during training.
- **BiLSTM-KB**: instead of using softmax layer output, this model predicts a low-dimensional representation of predicates that match to the closest predicate representation in pre-trained KB embeddings; the closest property is found using cosine similarity.
- **BiLSTM-Binary**: this architecture outputs a binary decision on whether a pair of subject and predicate matches for the given question q (true or false).
- **FastText-Softmax**: this architecture uses FastText[1] as a classifier to predict the property (Joulin et al., 2016).

The task of answering simple questions requires identifying the correct entity and the predicate in the question. In this section, we describe in detail the model for identifying the span of the entity and retrieving the matching candidates. Then, we describe four architectures for property prediction that build on this common entity prediction model. All four architectures rely on a candidate retrieval step that extracts candidate pairs of subject and predicate where only 1 pair constructs the query. The process is shown in Figure 8.3. In order to retrieve entity candidates we rely on an inverted index the construction of which we detail in the section below.

We present a partial Knowledge Base (KB) constructed with subjects and their related predicates. Each subject (blue colored) has a *type* relation, label and predicate relations to other subjects. We trained a Named Entity Recognizer (NER) system to identify the subject mention in a given question. By querying the identified subject mention we extract candidate subjects. This is achieved by matching the *label* values of each subject to the extracted mention. Retrieved candidate subjects are paired with predicates that exist within a triple in Freebase. These are predicates that connect two entities. For instance, the predicate *book.written_work.author* between *m.04t1ftb* and *m.03nx4yz*. Finally, the subject and predicate pairs are returned as candidates. We limit our candidate pairs to only those that appear in the Freebase-2M set of triples since this set of triples were used by human annotators to create the SimpleQuestions dataset.

Next, we explain each part in more detail starting with constructing an inverted index for entity retrieval, NER and finally Candidate Pair Generation.

8.2.1 Inverted Index Construction for Entity Retrieval

We extract all entity mentions from Freebase using *type.object.name* and *common.topic.alias* predicates. During the extraction process, we also counted how often a surface form occurs together with an entity. As a result, we generated a surface form index for each subject with an associated frequency value. Additionally, we merged a surface form index created for DBpedia entities using *owl:sameAs* links. NERFGUN (Hakimov et al., 2016) provides such an index of surface forms. We converted the DBpedia URIs into Freebase MIDs using the links provided by the DBpedia release of 2014[2]. The converted index was merged with the

[1] https://github.com/facebookresearch/fastText

[2] http://oldwiki.dbpedia.org/Downloads2014#links-to-freebase

index data extracted from Freebase. We aggregated the frequency values if the same surface form and Freebase URI (MID) existed in both indexes.

A sample from this index is given below. All surface forms in the index are normalized; they are converted into lowercase, punctuation as well as non-alpha-numeric characters are removed, etc.

Surface Form	URI	Frequency
mildred pierced	m.04t1ftb	11
mildred pierced	m.04t_038	8
mildred pierced	m.0cgv06r	7

8.2.2 Named Entity Recognition

We trained a Named Entity Recognizer (NER) system similar to the one proposed by Chiu and Nichols (2015) using weak supervision.[3] Since the dataset requires a single subject we adapted the NER to identify a single entity span.

The original approach is tailored towards identifying common named entity (NE) types: LOCATION, PERSON, ORGANIZATION, MISCELLANEOUS. Our goal is extract the single named entity span without doing any distinction between those types. We use a *IO* tagging scheme to mark tokens inside (I) and outside (O) of the single named entity of interest.

We merge the consecutive tokens that have *I* as an output. This process is illustrated in Figure 8.1. The predicted output shows that tokens *Mildred* and *Pierced* get assigned the output *I* while other tokens get *O* as an assigned label.

[3] We build on the code available at https://github.com/kamalkraj/Named-Entity-Recognition-with-Bidirectional-LSTM-CNNs

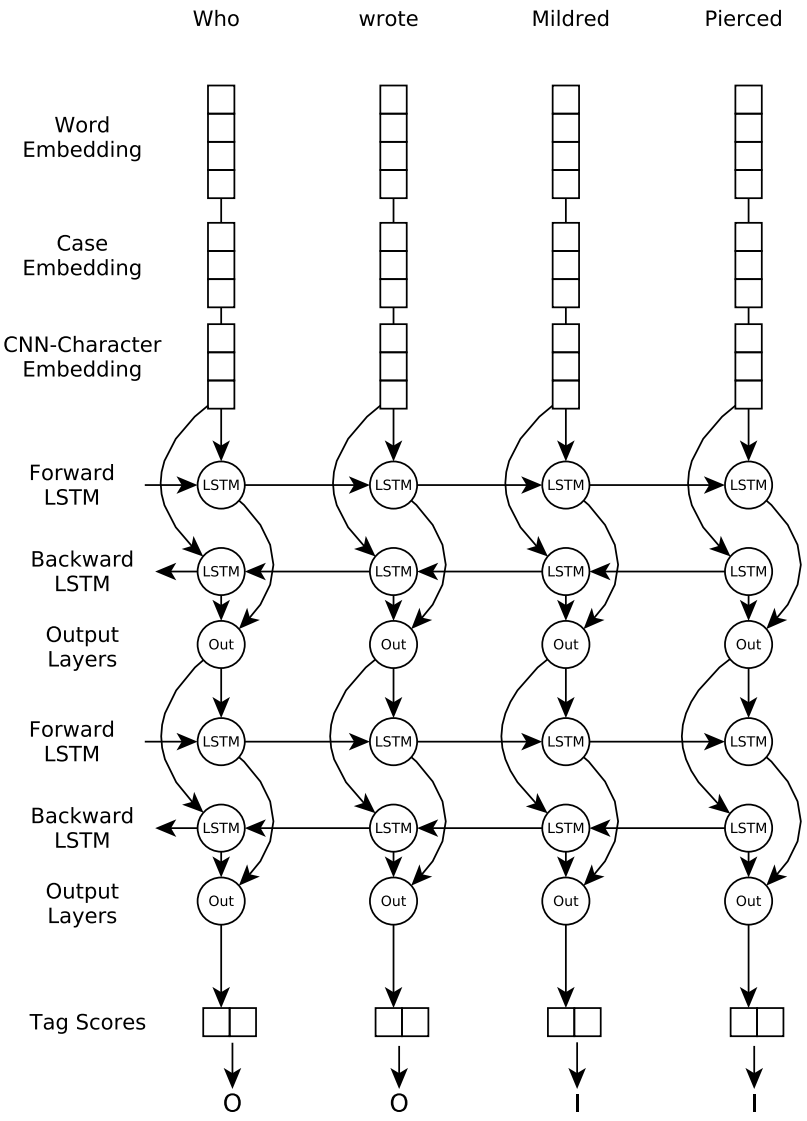


FIGURE 8.1: Named Entity Recognition using Bidirectional LSTM

The architecture is based on Bidirectional LSTMs (BiLSTM) (Graves et al., 2013). It is composed of two LSTM (Hochreiter and Schmidhuber, 1997) layers. The model uses words and characters as features along with case of words (lowercase, uppercase). These features are concatenated and fed into a neural network.

The input sentence is tokenized. Each token in the sentence is converted into a word embedding representation using Glove (Pennington et al., 2014) vectors (100 dimensional). Each token is also represented in terms of characters by converting the token into a matrix where each vector corresponds to a one-hot encoding vector of a character. The character matrix is fed into a Convolutional Neural Network (CNN) (LeCun et al., 1998). The CNN applies a convolution function to input vectors. We apply a Max-Pooling layer on the CNN output layer that represents the most important character embeddings given the token. The process is illustrated in Figure 8.2. A sigmoid function is applied to the output layer to infer the maximally scoring label for each token.

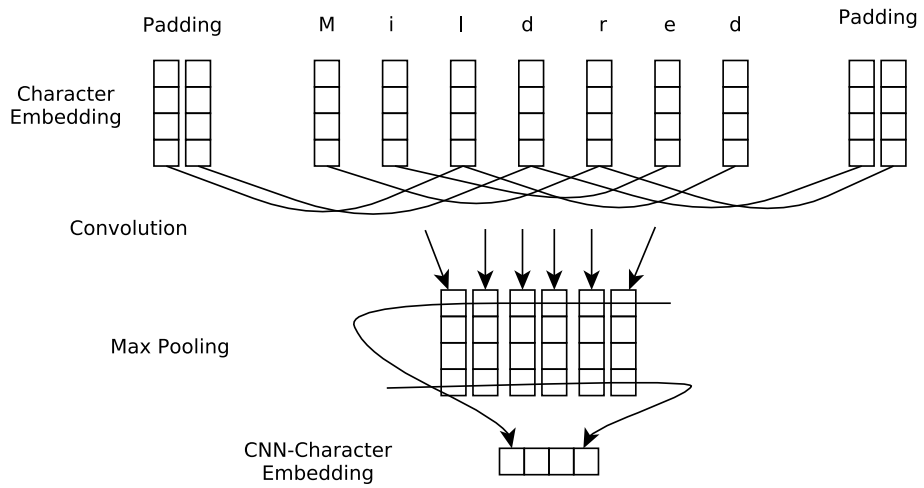


FIGURE 8.2: CNN max pooling operation on character embeddings

As the SimpleQuestions dataset does not explicitly provide the span of subjects, we rely on weak supervision to infer the subject during train. We infer the position of the subject by querying the inverted index for each n-gram in the question. We assume that the correct subject span is the one that matches the expected subject URI when queried on an index. The algorithm for inferring the span of a subject is given in Algorithm 4. These inferred token labels are used as expected output labels from the NER model.

Algorithm 4 Inferring Named Entity Spans

procedure FIND-SPAN(s, u, m) ▷ input sentence s, the expected URI u and maximum ngram size m
  $ngrams \leftarrow extract\_ngrams(s, m)$ ▷ extracts all possible n-grams from the input s
  $inferred = \emptyset$
  for each item n_i in $ngrams$ **do**
    $candidates$ = retrieve_candidates(n_i)
    if u is in $candidates$ **then**
      $inferred = n_i$
      Break ▷ stops the loop if the expected URI is found
    end if
  end for
  return $inferred$ ▷ The inferred span for a given URI
end procedure

The algorithm extracts all possible n-grams from a question by limiting the maximum n-gram size to 10. The process starts with bigger n-grams where each n-gram is queried against the surface form index. The index returns a list of candidates. If the subject (e.g. *m.04t1ftb*) is among the returned results the span of a named entity is assumed to be correctly inferred. The process will be stopped at this point. This procedure is applied to all questions in the SimpleQuestions train split.

The NER model is trained for 15 epochs, the embedding size of the BiLSTM was 300, the CNN networks uses 3 kernels.

Prediction: During prediction we give the question text into the model and obtain the output labels. We merge the tokens that are returned as *I*. As shown in Figure 8.3, we apply NER and extract the entity mention, i.e. *Mildred Pierced* in our example. Querying the

mention will result in some candidates because it matches the sample surface forms given in Section 8.2.1.

However, in some cases NER returns partial names where some tokens are missing. This is due to the fact that some questions are lowercased and some are not. It is challenging for the NER system to learn whether a certain word, e.g. preposition, is part of a named entity or not.

To overcome this drawback, first we find all n-grams from the question that match some candidate subject in the index. Then, we compute *edit distance similarity* between the extracted mention from NER and all n-grams that matched some subject. The n-gram with the highest similarity to the one proposed by NER is used as **inferred** subject mention m.

This way we can ensure that there is always a candidate mention m with candidate subjects. In cases where NER does not return any span or the maximum similarity between extracted mention and n-grams does not surpass a certain threshold (t=0.6), we pick the n-gram that covers the most tokens and matches to some candidates.

8.2.3 Candidate Pair Generation

As shown in Figure 8.3, we apply the trained NER system and extract the entity mention, i.e. *Mildred Pierced* in our example. The extracted mention m is queried on the surface form index. All matching entries are added to the set $S(m)$. Each entry contains a subject URI (Freebase MID) and a frequency value. For example, the following subjects are found: *m.04t1ftb*, *m.01d13qs*, *m.04t_038*, *m.0cgv06r*.

We define a KB as a set of triples of the form (s_i, p_i, o_i) that appear in the Freebase-2M dataset. Given a subject s_i we define the set $Pred(s_i)$ of all the properties that s_i has as

$$Pred(s_i) := \{p_i \mid \exists o_i (s_i, p_i, o_i) \in KB\} \tag{8.1}$$

We further define the set of candidate pairs for mention m as:

$$C(m) := \{(s_i, p_i) \mid s_i \in S(m) \wedge p_i \in Pred(s_i)\} \tag{8.2}$$

For example, the extracted candidate entity *m.01d13qs* has 2 predicates:
music.release_track.release, *music.release_track.recording*. By combining the predicate with the candidate entity we generate candidate pairs (see Figure 8.3).

The next step is to find a ranking function that takes an input question text (q), the identified mention m and candidate pairs ($C(m)$=$\{(s_1, p_1), (s_2, p_2), (s_3, p_3), \ldots, (s_n, s_n)\}$), and returns the highest ranking pair (s^*, p^*).

$$(s^*, p^*) = argmax_{(s_i,p_i)\in C(m)} P(s_i, p_i|q;\theta) \tag{8.3}$$

where $P(s_i, p_i)$ computes the probability of a pair s_i and p_i using the equation below.

$$P(s_i, p_i|q;\theta) = P(p_i|q;\theta) * P(s_i|q:\theta) \tag{8.4}$$

where $P(p_i|q;\theta)$ is the probability of predicate p_i as computed by our four predicate models described below. $P(s_i|q:\theta)$ is the probability of a subject s_i computed by normalizing the frequency scores retrieved for the mention m.

In the following sections, we describe our proposed approaches for the prediction of target predicates.

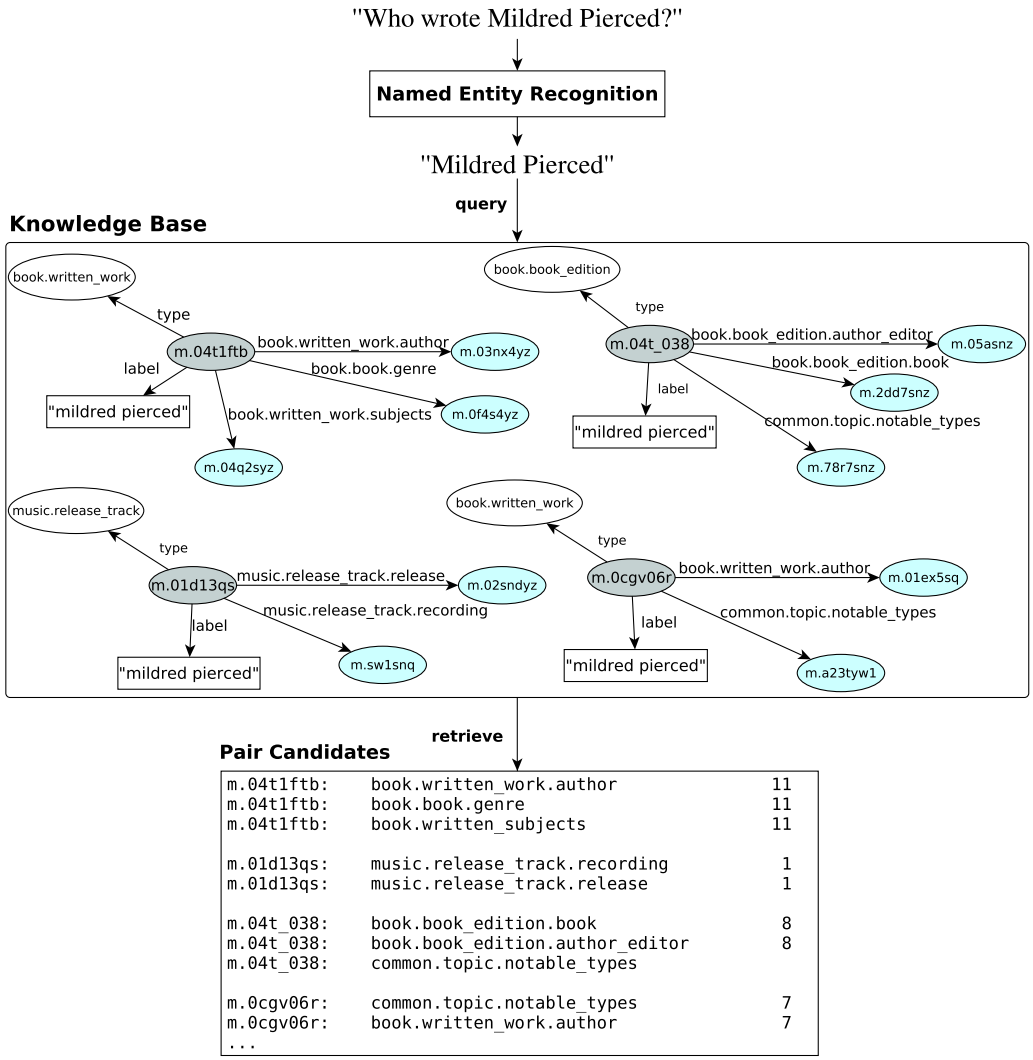


FIGURE 8.3: Candidate pair generation illustration for neural model architectures

8.2.4 Model 1: BiLSTM-Softmax

Our first model is a BiLSTM classifier that predicts the target predicate given the question text. This a standard model to predict multiple class labels using a softmax layer by encoding the input text using word and character embeddings. Before passing the question text to our network, we replace the entity name with a special placeholder token *e* (e.g. "Who wrote e?") that abstracts away the (inferred) subject mention. Moreover, the model is very similar to the one proposed by Ture and Jojic (2017).

Architecture

Similar to the NER model, the question text is encoded on the word and character level. Character-level word embeddings are computed by applying a CNN layer with Max-Pooling on the characters of each token. This process is the same as explained above in Figure 8.2. Word and character embeddings are concatenated and passed through a BiLSTM layer. The final states of the BiLSTM layers are concatenated and fed into a feed-forward layer with

softmax activation function, which calculates a probability distribution over a set of predicates. We identified 1629 predicates in the training split of the SimpleQuestions dataset. The model architecture is shown in Figure 8.4.

The model assigns a probability for each predicate. During candidate pair generation we extract subjects along with frequency values. These frequency values are normalized so that we yield a proper probability for each subject given a question q and a mention m. The score for each candidate pair is calculated by multiplying the probability score of the candidate predicate with the normalized frequency value of the candidate subject as given in Equation 8.3.

Hyper-parameters

The CNN layer uses an embedding size of 100, the LSTM layer uses 200 dimensions; Word embeddings are initialized using 100 dimensional Glove vectors and are retrained with the rest of the model. The model is trained for 100 epochs. These parameters are chosen based on trying different configurations of parameters on the development set.

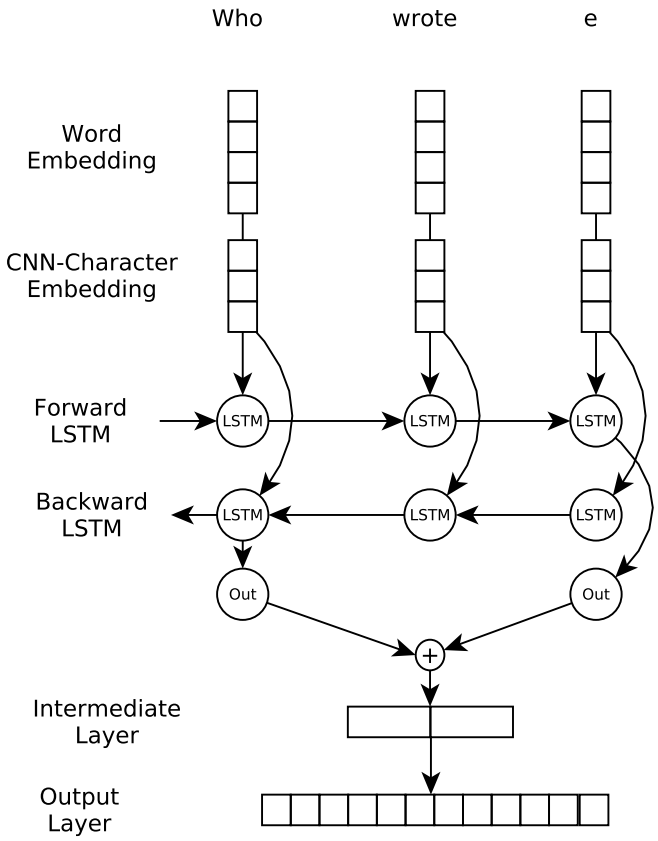


FIGURE 8.4: BiLSTM-Softmax model that computes probability distributions for predicates given only the question text

8.2.5 Model 2: BiLSTM-KB

In this subsection, we present a different approach for predicting a predicate from a given question that incorporates pre-trained graph embeddings into the classification process. Before we describe this model architecture, we first introduce how these graph embeddings are

computed.

Graph Embedding

There have been different approaches proposed over the years for computing embeddings for knowledge bases. RDF2Vec (Ristoski and Paulheim, 2016) is such a method. By performing a random walk on the graph, the algorithm records paths between pairs of entities. The resulting paths are considered as "sentences" and are fed into the popular word embedding algorithm word2vec which computes vector representations for vertices and edges.

TransE (Bordes et al., 2013) is another method for computing graph embeddings. The method is based on taking a single triple, e.g. (e_i, p, e_j) and creating corrupted triples from it by randomly replacing the subject e_i or the object e_j with a random entity from the KB. The objective of the method is to learn a ranking function that maximizes the margin between the score of an actual triple and the corrupted triples.

In this work, we compute KB embeddings using FastText (Joulin et al., 2016). We phrase the task of learning KB embeddings as a classification task. For each triple $t = (e_i, p, e_j)$ in the KB, we construct training samples for the FastText classifier by treating the predicate p and the object e_j as input tokens and subject e_i as the target class. To create embedding vectors that are aware of the role of an entity in a triple, we generate the training sample using role-specific embeddings: e_i^s, e_j^o and p^s. Here, e_i^s indicates that the target is an entity in the subject position, e_j^o is an input entity in the object position and p^s an input predicate used for predicting a subject entity. Analogously, we create a training sample with the object o being the target class. An example in the FastText format for the triple `Inferno, hasAuthor, Dan_Brown` is given below:

$$\texttt{\_\_label\_\_Inferno}^s\ \texttt{hasAuthor}^s\ \texttt{Dan\_Brown}^o$$

$$\texttt{\_\_label\_\_Dan\_Brown}^o\ \texttt{hasAuthor}^o\ \texttt{Inferno}^s$$

By training a FastText classifier on the generated training samples, we obtain vector representations for all entities and predicates with respect to their role in the triple[4]. We chose FastText as a classifier for its good performance on text classification tasks in terms of accuracy and speed.

Architecture

In the following, we describe a neural network model that uses the pre-trained graph embeddings to predict the target predicate given a question text. The intuition is that we can project the question text into the embedding space of the KB, thus supporting the learning process by utilizing the pre-trained, latent structure of that space. Additionally, the model is not limited to predicates seen during training whereas BiLSTM-Softmax outputs probability distribution to predicates that only appear in the training split.

Similar to the model in Figure 8.4, the question text is encoded using word and character level embeddings. The encoded text is fed into a BiLSTM layer that outputs a sequence of hidden states. We concatenate the last states of the forward and backward LSTM and pass it through a feed-forward layer which produces a fixed-sized output vector $\hat{p}$ of 200 dimensions. The network is trained to maximize the cosine similarity of the produced output vector $\hat{p}$ and the pre-trained embedding vector p^* of the target predicate.

[4]Due to the huge amount of target classes, training the classifier with a full softmax objective is not feasible. Instead, we use the negative sampling objective that is part of the FastText toolkit as an approximation to the softmax objective.

During prediction we compute the cosine similarity of the computed output vector to the embeddings of all predicates in Freebase-2M and normalize across all predicates to obtain a probability distribution.

The score for a candidate pair is computed as given in Equation 8.3. The model architecture is shown in Figure 8.5.

Hyper-parameters

The CNN layer has 100 dimensions, the LSTM has 400 dimensions. We use 100-dimensional Glove vectors.

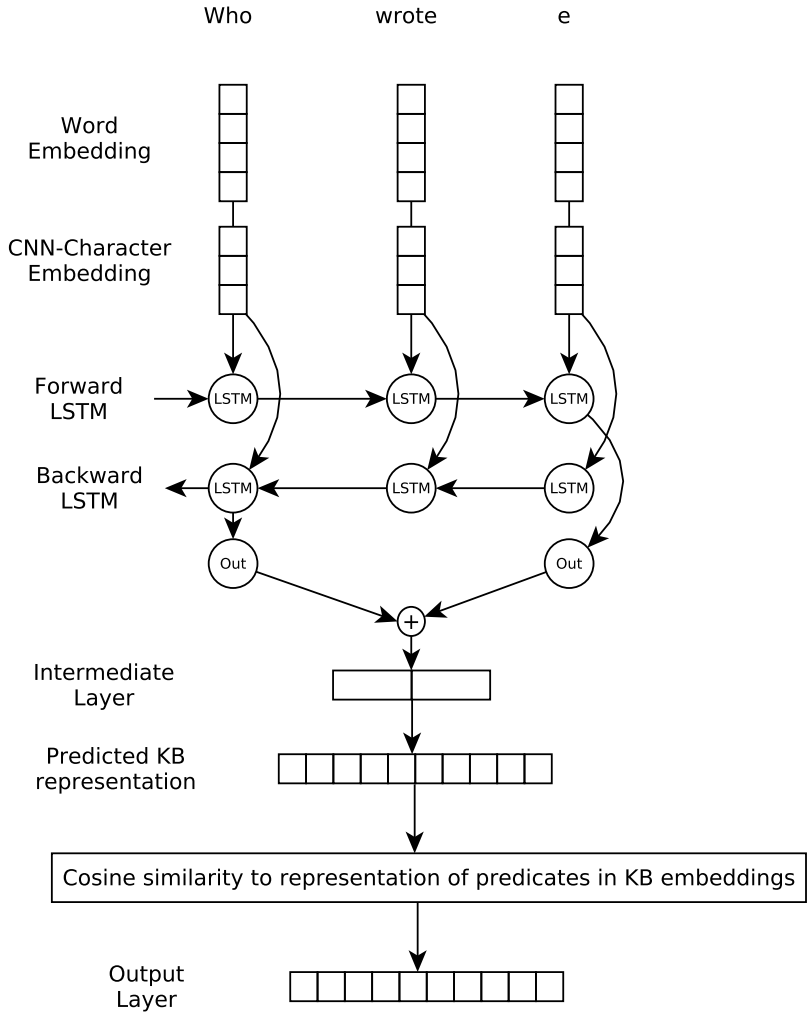


FIGURE 8.5: BiLSTM-KB model that computes probability distributions for predicates given only the question text

8.2.6 Model 3: BiLSTM-Binary

This model is different than the other 2 models explained above (see Section 8.2.4 and Section 8.2.5) in terms of the input to the model. While BiLSTM-KB introduces external knowledge about predicates from a knowledge base, this model learns to associate the question text with the tokens in the predicate URI. The input is composed of a question text q and the label of a single predicate p_i and the model outputs a binary decision (0 or 1) indicating if the

predicate is correct for the question. By giving the label of a predicate as an input feature, the model can potentially use the similarity between the question text (e.g. *Who **wrote** e?*) and the predicate label (e.g. *book.**written**_work.author*) to determine if the given predicate tokens matches the question text.

Architecture

The inputs q and p_i are tokenized and fed into encoding layer that uses word and character embeddings. These are shown as a separate components. The encoding is the same process explained in Section 8.2.4 (see Figure 8.4) where the tokens are represented by word and character embeddings and fed into 2-layer BiLSTM.

The tokenization of the predicate p_i is done by splitting the URI by *dot* and *underscore* characters. The latent embeddings are fed into an intermediate layer, which learns to score the compatibility between (embedded) question input q and predicate p_i. Finally, the output layer is a sigmoid function that outputs a binary decision in terms of probability. The model architecture is depicted on Figure 8.6.

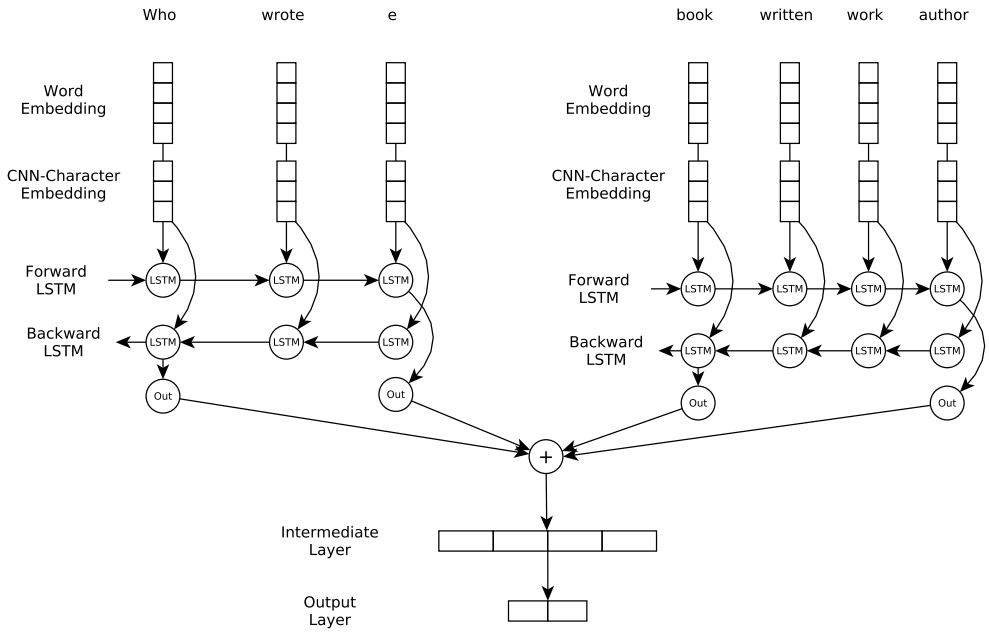


FIGURE 8.6: BiLSTM-Binary model that computes probability distributions as a binary decision given the question text and the predicate pair

During prediction we collect all predicates from each candidate subject and feed them into the model one at a time. The model outputs a probability for each predicate. The score for a candidate pair is computed using Equation 8.3. The highest scoring pair is selected as the final output.

Hyper-parameters

We use a CNN with 100 dimensions, and an LSTM with 400 dimensions. We use 100-dimensional Glove vectors. The model is trained for 100 epochs.

8.2.7 Model 4: FastText-Softmax

For our last model, we train a classifier that predicts the target predicate given the question text using FastText (Joulin et al., 2016). The FastText tool implements a linear classifier on top of a bag-of-n-gram representation of a text using word n-grams to preserve local word order and character n-grams for robustness against out-of-vocabulary words. The model outputs a probability for each predicate. The score for a candidate pair is computed using Equation 8.3. The highest scoring pair is selected as the final output. For a detailed description of the model architecture we refer to Joulin et al. (2016).

Hyper-parameters

Due to the moderate size of the target vocabulary[5] we can train the classifier with a full softmax objective. We trained the classifier for 50 epochs and a hidden layer size of 100. The classifier uses word n-grams of size 1 and 2 and character n-grams of size 5.

8.3 Evaluation

We provide evaluations on four models and the building components in isolation using the provided test split from the SimpleQuestions (Bordes et al., 2015) dataset as follows:

1. Named Entity Recognition: the evaluation shows the accuracy for extracting the correct mention from the question text.
2. Named Entity Linking: the evaluation shows in how many cases the subject can be retrieved by index look-up using the detected entity mention from the NER step.
3. Predicate Prediction: this evaluation shows how well the four models perform in predicting the correct predicate for the given question text.
4. Answer Prediction: this evaluation shows how well the proposed models perform on predicting the correct triple and how they compare to other systems on the SimpleQuestions dataset.

8.3.1 Named Entity Recognition

Training We trained a BiLSTM-CRF NER system on SimpleQuestions training split. The model was run for 100 epochs, with word embeddings from Glove 100-dimensional vectors, 200 dimension for LSTM layers.

Prediction During prediction, we queried all possible n-grams extracted from question text q on a surface form index. N-grams that returned a match were added to a set N. A question text q was given as an input to the trained NER system. The output from NER system was compared with each n-gram in N. The comparison is based on Edit distance similarity. The N-gram that is the most similar to the output is taken as a recognized subject mention m. In this way we can ensure that output mention from NER maps to some set of subjects. The system is regarded as having correctly identified a certain mention m if by looking up the mention in the index the correct subject is returned. For instance, in the Figure 8.3 the NER system identifies "mildred pierced" as an entity mention. By querying the entity mention we retrieve four subjects. The expected target subject *m.04t1ftb* is in the list. The NER component achieves an accuracy of 0.82 on the test split of the SimpleQuestions dataset.

[5] 1629 predicates in the training set.

8.3.2 Named Entity Linking

Once the subject mention m has been extracted from the NER system, the next step is to get all the matching subjects from the surface form index. We queried the mention m on an index and retrieved subjects with corresponding frequency values.

For evaluation, we ranked the subjects by their frequency values and calculated Recall@K. The system correctly links if the target subject is in the ranked list of K candidate subjects. The results are shown in Table 8.1.

TABLE 8.1: Named Entity Linking evaluation on test split using Recall@K

K	Recall
1	0.68
2	0.74
5	0.79
10	0.81
25	0.82
100	0.82
400	0.82

8.3.3 Predicate Prediction

All models described above compute probability distributions for predicates. To understand better the building blocks of each model, we evaluated the performance of each model for predicting the correct predicate. Below in Table 8.2, we listed the results for BiLSTM-Softmax, BiLSTM-KB, BiLSTM-Binary, and FastText-Softmax. We trained different models with different hyper-parameters. In Table 8.2, we listed only the best performing models of each type with their performance scores. The performance score is Accuracy and it was calculated by excluding the subject from the pair and comparing only predicted and expected predicates. As shown in Table 8.2, FastText-Softmax output performs all other systems while BiLSTM-Softmax and BiLSTM-Binary performed similarly.

TABLE 8.2: Evaluation of four models on predicate prediction task

Name	Accuracy
BiLSTM-Softmax	0.74
BiLSTM-KB	0.68
BiLSTM-Binary	0.73
FastText-Softmax	**0.79**

8.3.4 Answer Prediction

The task of question answering on the SimpleQuestions dataset requires a system to output a single triple consisting of a subject and a predicate. We evaluated the four proposed models on prediction of a triple consisting of a subject and a predicate. The predicated pairs are ranked using Equation 8.3.

Moreover, we compared our results with other published systems that evaluated using the same dataset. All results are shown in Table 8.3. None of the vanilla architectures could outperform the current state-of-the-art systems.

TABLE 8.3: Systems evaluated on SimpleQuestions dataset ranked by the reported accuracy measures, * systems evaluated on FB5M data, the rest were evaluated on FB2M data

System	Accuracy
Bordes et al. (2015)	0.63
Aghaebrahimian and Jurčíček (2016)	0.65
Golub and He (2016)	0.71
Lukovnikov et al. (2017)	0.71
Dai et al. (2016)	0.76*
Yin et al. (2016)	0.76
Ture and Jojic (2017)	0.88
BiLSTM-Softmax	0.66
BiLSTM-KB	0.61
BiLSTM-Binary	0.66
FastText-Softmax	0.68

8.3.5 Error Analysis

We choose BiLSTM-Softmax to perform error analysis and highlight the errors the model makes. In Table 8.4, we report the pair prediction results for BiLSTM-Softmax using Recall@K. We extract K top-ranking pairs as given by the model and evaluate how well the system performs on pair prediction. Additionally, we evaluate separately how the subject in the predicted pair compares to the subject of the expected pair. We perform the same evaluate on predicates as well.

We can observe that BiLSTM-Softmax predicts the correct predicate with 0.74 for Recall@1 and 0.8 for Recall@2. The predicate prediction has an upper-bound of 0.84, which was obtained by Recall@20. Subject prediction has a higher performance than pair prediction (0.67 vs 0.74 for Recall@1). Subject prediction has an upper-bound of 0.82 as explained in the previous section (Section 8.3.2). Overall results for pair prediction suggest that the model has the highest margin between Recall@1 and Recall@2. It means that the system could easily reach 0.74 if the ranking function improved.

TABLE 8.4: Recall@K values for BiLSTM-Softmax in Pair Prediction task

K	Pair	Subject	Predicate
1	0.67	0.74	0.74
2	0.74	0.78	0.80
3	0.77	0.80	0.81
4	0.78	0.80	0.82
5	0.79	0.81	0.83
10	0.80	0.81	0.83
20	0.80	0.82	0.84

Next, we analyzed the type of errors systems do as reported in Table 8.5. In total the system predicted 7206 wrong pairs and 14481 correct pairs out of total 21687 test instances. We reported the following type of errors:

- **Only Wrong Predicate**: If the predicted predicate is incorrect where the predicted subject is correct compared to the target subject and predicate pair. These errors could be caused by

- **Only Wrong Subject**: If the predicted subject is incorrect where the predicted predicate is correct compared to the target subject and predicate pair. These errors could be caused by NER or the frequency value of a subject.
- **Wrong Subject & Predicate**: If both predicate subject and predicate are incorrect compared to the target subject and predicate pair.
- **Empty Prediction**: If both predicate subject and predicate are empty.

By picking the highest ranking pair from predictions we compare it to the target pair, if there was a predicted pair. We can see that the majority of errors (0.29) are caused by not predicting any pair. The next biggest error mass is in predicting the pair wrong with 0.26. Finally, the system made more errors while predicting the predicate rather than the subject (0.23 vs 0.22).

TABLE 8.5: Error analysis for BiLSTM-Softmax in Pair Prediction task

Error Type	Count	Percentage
Only Wrong Predicate	1642	0.23
Only Wrong Subject	1591	0.22
Wrong Subject & Predicate	1911	0.26
Empty Prediction	2062	0.29
Total	7206	1.0

8.4 Discussion

We have shown that our NER step is reasonably accurate at detecting the subject span with an accuracy of 0.82. We have seen that in some cases NER picked the wrong span when the question contains some proper name which is not part of a target span, e.g. “where is mineral hot springs, colorado?” the expected span is “mineral hot springs” while the NER system recognizes the span “springs, colorado”. Similarly, during entity candidate extraction we have seen that sometimes the target subject has a frequency of 1, which affects the candidate pair score.

The models BiLSTM-Softmax and BiLSTM-Binary performed similarly on predicate prediction while BiLSTM-Binary had a margin of 0.6. FastText-Softmax outperformed all models on predicate prediction. For the answer prediction, BiLSTM-Softmax, BiLSTM-Binary and FastText-Softmax performed similarly even though FastText-Softmax had the best performance on predicate prediction with a margin more than 0.5.

While none of the model architectures could outperform the current state-of-the-art systems for the overall answer prediction, we evaluated the building blocks of a question answering system and showed how they perform in isolation. It shows how well each component performs and highlights the importance for comparing different models not just on the overall output performance but also the individual small components. Comparing to the state-of-the-art systems, our models lack on average 0.10 on accuracy.

For comparison with approaches presented in previous Chapters (6 and 7) we choose BiLSTM-Softmax since the performance is comparable to other models presented in this chapter or in some cases better.

Chapter 9

Discussion

In this chapter, we discuss the results obtained with the three different semantic parsing approaches proposed in this thesis and highlight differences in terms of the role of syntax and semantics, the multilinguality aspect of these approaches, the manual effort required for adapting to another domain or language and finally analyse the effects of the datasets on these approaches.

9.1 Overview

We described three different semantic parsing approaches for Question Answering in previous chapters: Chapter 6, Chapter 7 and Chapter 8. From Chapter 8 we have chosen BiLSTM-Softmax for comparison. In this chapter, we compare these methods based on the following dimensions:

- Manual Effort: what manual effort is required for each approach in order to adapt them?
- Syntax and Semantics: what is the role of syntax and semantics for each approach?
- Multilinguality: how does each proposed approach deal with multilinguality?
- Cross-domain Transferability: what are the requirements to adapt the approaches to other domains?
- Training Data Size and Search Space: how are the approaches affected by the given training data and the search space of the target domain?

9.2 Manual Effort

The presented approaches have different requirements in their architectures in terms of the needed lexical resources, the training procedure or the manual effort that is needed to run these methods. In the next section, we highlight the parts of each approach where manual effort is required and compare them against each other.

9.2.1 CCG-based Semantic Parsing Approach

The approach is built on CCG syntax and lambda calculus. For each language one needs to define CCG combination rules (see Section 2.2.1) as well as the *domain independent* lexicon for each domain and language. Defining such a lexicon requires not only manual effort but also domain expertise in CCG syntax and lambda calculus.

9.2.2 Dependency parse tree-based Semantic Parsing Approach

The approach builds on cross-lingual dependency parse trees from Universal Dependencies (UD) and compositional semantics based on DUDES. The manual effort required for this approach consists of the definition of features for the model. All of these features have to be designed carefully and evaluated properly in order to understand the impact and obtain a functioning system.

9.2.3 Neural Network-based Semantic Parsing Approach

This method, unlike others, does not need any manually defined features, lexicon or grammar entries. The model architecture is based on neural networks where the learning process is end-to-end. The words in sentences are used as features. The method does not depend on any syntactic information. It rather learns such syntactic and contextual dependencies using BiLSTM layers in the network.

9.2.4 Comparison

In terms of manual effort the Neural Network-based Semantic Parsing Approach does not require any additional resources for the pipeline to run. The approach is trained end-to-end where the model performance relies heavily on the amount of training data. The Dependency parse tree-based Semantic Parsing Approach requires hand-engineered features to be defined but the same features can be used for other languages as well. Lastly, the CCG-based Semantic Parsing Approach requires the most manual effort in adapting the system.

9.3 Syntax and Semantics Relationship

The syntax and the semantics are building blocks for semantic parsing approaches. In this section, we explain briefly how syntax and semantics is used in each proposed approach and highlight the differences and compare the proposed approaches.

9.3.1 CCG-based Semantic Parsing Approach

The CCG-based Semantic Parsing Approach learns syntax and semantics jointly from the training data. The approach learns to generate syntactic parse trees that depend on CCG combination rules. The approach depends on lambda calculus for defining semantics of words. The model learns the syntax and semantics jointly but it is limited to the given training data, which is the reason why it works better on closed-domain datasets. The meaning representation of a natural language question is obtained using semantic composition as explained in Section 2.3.4 where the syntax guides the semantics using CCG combination rules along with lambda calculus composition.

9.3.2 Dependency parse tree-based Semantic Parsing Approach

This approach uses UD dependency parse trees for syntax and DUDES for expressing semantics. UD includes parse trees for over 60 languages and new annotation data is being added regularly by the community. It uses compositional semantics based on DUDES (see Section 2.3.4) and the dependency relations on a parse tree to obtain the meaning representation of natural language questions.

9.3.3 Neural Network-based Semantic Parsing Approach

The approach does not rely on syntactic information about words in sentences. The syntactic and contextual information is rather learned. The BiLSTM layers in the network are capable of learning inter-dependencies between words. The input is represented as word and character embeddings. Word embeddings provide lexical and contextual information while character embeddings encode an additional information for the lexical coverage. The approach has a simple semantic representation. The task of answering questions in SimpleQuestions (Bordes et al., 2015) requires a system to predict a single triple with a predicate and a subject entity.

9.3.4 Comparison

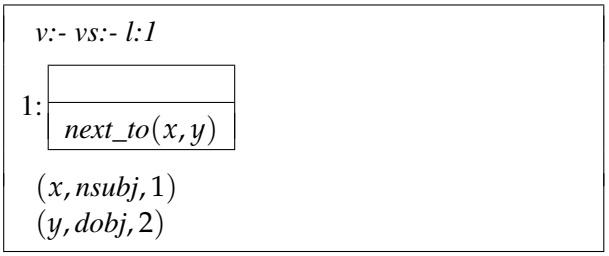
The CCG-based Semantic Parsing Approach learns the syntax and semantics from the training data whereas Dependency parse tree-based Semantic Parsing Approach uses a pre-trained dependency parser for syntax and learns only the mapping from parse trees into semantic representations. Both methods have semantic representations for aggregation, ordering, entities, classes and predicates. The Dependency parse tree-based Semantic Parsing Approach is the only system that uses dependency relations among nodes in the parse tree.

The Neural Network-based Semantic Parsing Approach differs from the other the two methods in terms of syntax and semantics. The syntactic and contextual information are rather part of a learning process using word and character embeddings with BiLSTM layers. All questions in the SimpleQuestions dataset follow the same rule where questions do not include aggregation, ordering etc. For such tasks, we only need a semantic representation for an entity and a predicate where these fill the subject or the object slot in a triple. The composition of semantics for a given question in fact does not require any machinery. It is a straightforward slot filling approach consisting of inserting the predicted Freebase entity into the subject position and the predicted Freebase property into the predicate position.

The CCG-based and the Dependency parse tree-based Semantic Parsing approaches both use a compositional semantics approach to obtain a meaning representation of a sentence using *bottom-up* semantic composition methods described in Section 2.3.4. They differ in order and criterion for composition. The lambda calculus is based on a certain order of application. For instance, the following lambda expression expects the variable λx to be combined before the variable λy.

$$\lambda x.\lambda y.\texttt{next\_to}(x,y)$$

However, the semantic composition with DUDES (see Section 2.3.3) does not necessarily depend on the order of application. It depends on the dependency relation with the child and the parent node. If the criterion is met then the application proceeds. The following semantic representation states that the variables x or y will be replaced if the given arguments have *nsubj* or *dobj* relation to the parent node in a dependency parse tree.



9.4 Multilinguality

Multilinguality is an important aspect to consider when building a QA system. Approaches that scale to other languages are preferred since building such systems require sometimes manual effort or domain expertise as explained above. In this section, we provide an overview on the multilinguality aspect of each approach and compare them.

9.4.1 CCG-based Semantic Parsing Approach

The approach is intended to be built for a single language, which is English. As explained above, the method relies on CCG grammar definition for each language and some additional manual lexicon. The application to another language is possible if these specific resources are provided.

9.4.2 Dependency parse tree-based Semantic Parsing Approach

This approach uses cross-lingual dependency parse trees from Universal Dependency (UD) as syntactic representations that allows the system to be defined as a *multilingual* method since other components of the method do not incorporate any data that restricts it to a specific language. The UD comprises of corpora for over 60 languages[1], which allows to build language-independent pipelines by abstracting from specific languages.

9.4.3 Neural Network-based Semantic Parsing Approach

The Neural Network-based Semantic Parsing Approach is different from the other two methods in the way that it does not depend on any syntactic representation specific to a language. The method rather learns such information from a large number of training instances. Word embeddings are used in this method to provide contextual information. Character embeddings are also used in order to provide lexical information.

9.4.4 Comparison

The important aspect to consider here is the lexicon for mapping natural language expressions to knowledge base representation, e.g. predicates, resources or classes. Such a lexicon is a language-specific resource. As we showed in Section 5.3, the availability of such data with high quality and the required variability is not comparable across all languages. Furthermore, we showed that by using only word embeddings the system can still achieve reasonable results where the multilingual lexica is not available or they lack wide coverage, e.g. the German and Spanish lexical items from the M-ATOLL. Considering these points, extending the Dependency parse tree-based Semantic Parsing Approach to other languages is possible and easier compared to extending the CCG-based Semantic Parsing Approach.

The Neural Network-based Semantic Parsing Approach uses word and character embeddings as resources to learn syntactic and contextual information. These resources are available for any language. Therefore, all components of the architecture can be adapted to another language without any changes to the pipeline.

[1] https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-2515

9.5 Cross-domain Transferability

Another important aspect to consider when building QA systems is the transferability of our proposed approaches to other domains. In this section, we compare the approaches and explain how each one of them can be transferred to another domain.

9.5.1 CCG-based Semantic Parsing Approach

The approach is built on DBpedia and applied to the QALD-4 dataset. It relies on external linguistic resources such as M-ATOLL and the lexicon that is learned by the model. Additionally, manual effort is required to adapt the method to another domain as explained above.

9.5.2 Dependency parse tree-based Semantic Parsing Approach

This approach can be applied on other domains without any changes to the model. The only drawback would be the availability of high quality dependency trees for the selected domain. Universal Dependencies (UD) is a community effort that has been growing rapidly in the recent years. However, the annotated dependency parse trees are based on news articles, web documents, etc. For instance, applying to a specific domain such as *biomedical* would require UD corpora to contain parse trees in the same domain.

9.5.3 Neural Network-based Semantic Parsing Approach

This approach does not depend on any domain knowledge and can be applied without any changes to the architecture. The method requires only character and word embeddings as an input, which can be obtained for other domains as well.

9.5.4 Comparison

The CCG-based Semantic Parsing Approach and the Dependency parse tree-based Semantic Parsing Approach were built on DBpedia data. The approaches rely on an inverted index of lexicon to retrieve matching knowledge base entries. Transferring such information to another domain is not always feasible. For instance, in order to apply these methods to the *biomedical* domain either one needs to build such an inverted index or the knowledge base must already include these lexicalizations. As previously described, the CCG-based Semantic Parsing Approach is restricted to a certain domain because of the need to manually define some lexical entries. Even though these can be fixed for any specific language, it is still a drawback because not every entry can be found in the domain where the approach has been applied before.

The CCG-based Semantic Parsing Approach works well only for closed domains. If the variability of natural language expressions in the selected domain is limited and it is possible to create enough training instances it would be an easier choice. However, if the knowledge base includes a vast amount of data then the Dependency parse tree-based Semantic Parsing Approach would be more suitable since it learns patterns for associating natural language expressions to knowledge base entries. Finally, if the selected domain includes a large number of training instances then the Neural Network-based Semantic Parsing Approach could be easily extended considering the fact that it is an end-to-end system.

9.6 Training Data Size & Search Space

The availability of training data defines the choice for a model architecture. The performance of each proposed model architecture depends on the size of the training data and the search

space for the approach to explore. The search space is essentially the number of knowledge base entries that the approach explores during the mapping of natural language tokens in questions. If the search space is large and the number of training examples are low then a system can not capture variabilities in natural language from training data. The M-ATOLL (Walter et al., 2014) is a tool that generates the lexicalization of DBpedia properties. It is a needed resource because the QALD datasets do not include many training instances to learn from. Our proposed Dependency parse tree-based Semantic Parsing Approach uses such external resources in order to minimize the lexical gap in the natural language.

Below in Table 9.1, we provide statistics on the size of the training data and approximate the number of knowledge base entries for each respective dataset. Note here that the search space size for each dataset is approximated since the data in the knowledge bases can vary. The assumption here for the search space is the total number of unique entries in each knowledge base.

TABLE 9.1: Training data size for each dataset together with approximate number of distinct entries in each knowledge base

Dataset	Knowledge Base	Training Instances	Number of Distint IDs
Geoquery	US Geography data	600	750
QALD-6	DBpedia	300	10 mil.
SimpleQuestions	Freebase	75,910	2 mil.

The search space for Geoquery (Tang and Mooney, 2001) is composed of entities such as cities, states, rivers, mountains and handful properties such as *border*, *population*, etc. The search space is approximated to 750 items in total.

The search space for the QALD-6 dataset is essentially DBpedia data with approximately 10 million resources, more than 2000 predicates and around 800 classes. We approximated the search space for this dataset as 8 million items to search from.

The SimpleQuestions dataset is based on a subset of triples from Freebase. The counted number of individuals and predicates is approximated to 2 million entries.

We can see that the QALD-6 dataset has the largest search space with the lowest number of training instances. This suggests that the dataset is more complex than others also considering different types of questions, e.g. aggregation. Next, the SimpleQuestions dataset has a large number of entries to search from. However, this dataset includes 75,910 training instances where all questions are factoid questions with a single fact asked about. The dataset does not include aggregation for instance, unlike the QALD-6 dataset.

Finally, the Geoquery dataset has the lowest number of items in its search space. The dataset is domain-specific where a limited vocabulary is used for training and test instances. It makes the learning process easier compared to the QALD-6 or the SimpleQuestions datasets. It also includes more training examples than QALD-6 (600 vs. 300).

By looking at these results, we can see that QA systems running on the QALD-6 dataset have a more complex task to solve by searching in a larger space while training the system using only 300 instances. It poses a bigger challenge for our Dependency parse tree-based Semantic Parsing Approach to generalize from available number of training instances.

Chapter 10

Conclusion

In this chapter, we sum up the thesis with ideas for the future work. We provide answers for the research questions that are addressed in this thesis.

10.1 Conclusion

In this thesis, we presented three different semantic parsing approaches that aim to build Question Answering (QA) systems. These systems provide an interface where users could query the data available in knowledge bases using a natural language.

Chapter 6 describes a method for building a QA system that works well for closed-domain datasets. The approach is based on the method introduced by Zettlemoyer and Collins (2005) that uses CCG syntax and the lambda calculus for expressing the semantics. They evaluated the method on the Geoquery (Tang and Mooney, 2001) dataset. We adapted their method to the QALD-4 dataset which is an open-domain dataset that includes questions about various topics. We showed that training on such a dataset can be challenging since the vocabulary in training and test documents can vary significantly where the vocabulary seen during prediction does not occur in the training instances, unlike closed-domain datasets where the vocabulary is limited and used interchangeably. This missing vocabulary reduces the performance of systems since there is a gap in lexical knowledge that is learned during training.

The contribution of this approach is to show how the lexical gap in mapping a natural language question into knowledge base queries can be closed. It is shown that using external ontology lexicalisations extracted by M-ATOLL (Walter et al., 2014) can provide additional lexical information to bridge the gap as shown by our evaluation results. The system achieved 0.09 F1 score on QALD-4 dataset using only the learned lexicon and the DBpedia Ontology labels. The performance of the system increased to 0.30 when we added ontology lexicalisations from M-ATOLL. It shows that the systems trained on open-domain datasets could incorporate such lexicon in order to bridge the gap.

Chapter 7 describes a semantic parsing approach that uses dependency parse trees and DUDES (Cimiano, 2009) for expressing the semantics. The main contribution of this approach is showing how to build a semantic parsing approach for a multilingual QA system that is based on factorized graphical models. We showed that the same architecture can be applied to multiple languages without changing the pipeline components where we train a model that learns to map a syntactic structure to knowledge base queries in a supervised fashion. We used dependency parse trees from Universal Dependencies (UD) (Nivre et al., 2016; Nivre, 2017) as a syntax of the given questions. UD is an important community effort in creating multilingual NLP systems since the same set of dependency relations and POS-tags are used for all supported languages. As a proof-of-concept, we built a QA system that can handle questions in English, German and Spanish by training the model on the QALD-6 dataset. We showed how to build a multilingual QA system that is based on a semantic parsing approach.

Another contribution of this approach is to show how the mapping of natural language questions to knowledge base queries can be performed. We built an inverted index that combines lexical resources from WordNet, M-ATOLL and dictionary terms that are extracted from word embeddings. The choice of the evaluated languages were based on the languages that were supported by M-ATOLL at the time of writing. We showed that using only a dictionary from word embeddings can already give adequate results. This is the case for German and Spanish where the model based on the learned lexicon and M-ATOLL lexicalisations did not perform well. The approach achieved 0.04 Macro F1 score on Spanish and German in a question answering task. By adding an indexed dictionary from word embeddings we increased the performance by far. The performance of the approach increased to 0.20 for Spanish and 0.16 for German. This suggests that we can build multilingual QA systems for other languages as well even when ontology lexicalisations for some languages is missing or lacking adequate performance.

Chapter 8 describes a different set of architectures that do not use symbolic representations compared to the other two approaches explained above. This chapter evaluates four model architectures that are inspired by the state-of-the-art systems published in the same domain. All of these architectures are evaluated on the SimpleQuestions (Bordes et al., 2015) dataset. The main contribution of this chapter is to highlight the importance of comparing such architectures under the same environment because each published system can vary in many ways and it is sometimes hard to compare these systems by the reported performances only. Comparing these architectures under the same environment allowed us to understand the strengths and weaknesses in more detail.

In particular, we focused on the predicate prediction task and compared the four model architectures. Additionally, we evaluated each sub component of the pipelines such as named entity recognition and linking, relation prediction and joint answer prediction. We showed that the performance of the trained NER system affected all architectures since the detection and disambiguation of named entities constitutes half of the job for answering questions. The contribution of this chapter can be put in other words as identifying bottlenecks in the overall QA pipeline in order to shed light on the impact of different architectural choices to guide the future research on the QA task.

All proposed model architectures above have their own advantages and disadvantages, depending on the use case and the applied domain for QA. We compared them against each other on the points such as multilinguality, manual effort required to run, usage of syntax and semantics together in the process of interpretation and cross-domain transferability. We can conclude that the CCG-based Semantic Parsing Approach works best for domains where the vocabulary is restricted but it requires the most amount of manual effort in order to run it. It requires domain expertise in defining manually the domain-independent lexicon together with CCG combination rules.

The Neural Network-based Semantic Parsing Approach can also perform quite well in closed and open-domain datasets since the pipeline does not require any additional resources to run. It heavily depends on the training data size since optimizing the model parameters requires a large number of training examples. The Dependency parse tree-based Semantic Parsing Approach is a good choice for open-domain use cases where the number of training examples are limited. This approach incorporates multilingual lexical resources and language-agnostic dependency parse tree structures in order to build a multilingual QA system.

10.2 Research Questions

In this section, we provide answers to the research questions addressed in this thesis.

Question 1: How to map natural language phrases into knowledge base entries for multiple languages? Which linguistic resources can be used?

Using provided labels in an ontology of a knowledge base, external tools such as M-ATOLL (Walter et al., 2014) that generate ontology lexicalizations and contextual embeddings of words, phrases could be used to map natural language expressions to knowledge base entries.

Knowledge base ontologies sometimes provide labels in multiple languages. Additionally, as mentioned in Chapter 5, M-ATOLL lexicalizations can be used for multiple languages. Contextual word embeddings can be trained for any language considering the amount of text data available on the Web. In Chapter 5, we showed how we built an inverted index for retrieving URIs. We mapped each entry in the index to a certain semantic type, e.g. DUDES. This index combines data from different resources for multiple languages. The approach presented in Chapter 7 uses the index for mapping natural language phrases into knowledge base entries in multiple languages. In Chapter 6, the approach uses the labels from the knowledge base, learned lexicon and M-ATOLL lexicalisations to do the mapping for a single language, which is English.

The approach in Chapter 8 uses contextual information of words in order to do the mapping which is learned from training examples rather than using an inverted index for direct mapping.

Question 2: How to disambiguate URIs when multiple candidates are retrieved from mapping natural language tokens into knowledge base entries?

All proposed approaches have a model that is trained in a supervised fashion with an objective to disambiguate URIs and construct a meaning representation that is translated into a valid query. The disambiguation process is learned as a part of the training process of each proposed approach. Each approach outputs a single meaning representation that is the highest-ranking solution for a given question.

The CCG-based Semantic Parsing Approach deals with a restricted vocabulary where the ambiguity occurs less often. The method learns to rank certain URIs based on features that include lemma and CCG category information. The approach outputs the highest-ranking parse tree with its semantic representation for a given question where the disambiguation is handled by the score of such a parse tree.

The Dependency parse tree-based Semantic Parsing Approach uses factor graphs with features that express a certain edge relation between a parent and a child node (see Section 7.2.4). These features incorporate string similarity information, frequency value of a surface form to link phrase to URI, edge relation along with syntactic and semantic information. The method learns to rank the candidate URIs that construct a valid query by optimizing the weight for the defined features. There is also a pruning step where the candidates that do not lead to a valid query are removed early in the sampling process. This reduces the complexity for the algorithm to disambiguate.

Similarly, the Neural Network-based Semantic Parsing Approach also trains a supervised model to rank pairs of predicates and resources that appear in a question. The disambiguation is done on the basis of scores returned by the model for a predicate and a subject entity given a question. The chapter describes four different architectures that learn to rank a single

predicate and a subject entity using neural networks.

Question 3: How to use syntactic information of a natural language question together with semantic representations of entries in a knowledge base?

Semantic parsing approaches built on top of syntactic parse trees use semantic composition methods that take into account the underlying syntax and the ontology of the target knowledge base. In Chapter 7, we consider dependency relations between two nodes and the DBpedia Ontology, e.g., *rdfs:domain* and *rdfs:range* restrictions of properties, in order to learn the right mapping between syntax and knowledge base representation. The approach in Chapter 6 trains a model to learn a joint mapping of a question syntax and the provided knowledge base.

In Chapter 7, we presented a method for building a multilingual semantic parsing approach that uses compositional semantics of dependency tree structures to obtain a semantic representation. The approach uses a bottom-up semantic composition method that uses the dependency relations between words in a question and DUDES (Cimiano, 2009), which is a formalism for expressing the semantics of words or phrases in a sentence. The bottom-up semantic composition method described in Section 2.3.4 traverses each dependency relation and composes the semantics based on the parent and the child node where the syntax of a sentence guides the process to obtain the semantics.

The approach described in Chapter 6 learns the syntax and semantics jointly. The approach learns to generate the syntactic parse tree together with the semantics of a given question. Similar to the one described above, the syntax of a question guides the composition of semantics that is based on CCG combination rules and lambda calculus.

Question 4: What are the advantages and the disadvantages of a multilingual QA system vs. a monolingual system built for each language?

In Chapter 7, we present a multilingual semantic parsing approach that uses Universal Dependencies (UD) to obtain dependency parse trees of questions. Using such a resource enables us to abstract from languages because all parse trees use the same set of Part-Of-Speech (POS) tags and dependency relations. The main advantage for this approach is that the architecture can be extended for other languages without changing the pipeline. We showed this by applying the proposed approach on English, German and Spanish. It shows that the application of the approach to other languages is possible. In addition to that, having such systems enables other non-domain experts to build a question answering system for multiple languages as well as extending the approach to other domains.

The main disadvantage of a multilingual system could be seen in the amount of language-specific resources available for languages. Since not every linguistic resource is equally available or it can be limited for some languages, focusing on building a monolingual system for a chosen language can give better performance as shown by the method by Zettlemoyer and Collins (2005) with its application on the closed-domain dataset.

Question 5: What effort is required to adapt our QA pipelines to another language?

Approaches that are built on the syntax of a single language require more effort in order to adapt to another language. The method described in Chapter 6 is an example for such a system. It requires a domain expert to manually define CCG combination rules and domain-independent lexicon.

Considering the recent efforts in the research community for building Universal Dependencies (UD), the systems can focus only on the underlying mapping from syntax to semantics because all languages use the same set of grammar entries such as dependency relations and POS tags. In Chapter 7, we focused on this idea. The requirements for this approach for adapting to another language is extracting a lexicon that maps natural language expressions to knowledge base entries. Such a lexicon can be extracted from the embeddings of words in vector space, ontology lexicalizations provided by tools such as the M-ATOLL or the given ontology labels of knowledge bases. Word embeddings are openly available and trainable resources, e.g. word embeddings for 157 languages provided by Fasttext.[1] The same approach can be applied to other languages without any changes to the pipeline considering the fact that training instances will be provided in those languages.

In Chapter 8, we introduced four different model architectures that are built on language-independent pipelines. These model architectures use word and character embeddings of a given question as input. These resources as mentioned earlier are available for all languages. Thus, these architectures can be applied to other languages without any changes to the pipeline. However, these methods are heavily dependent on the amount of training data. The requirement for such architectures is acquiring enough training data in order to reach reasonable performance.

10.3 Limitations

This work compares three different semantic parsing approaches for building a QA system. The comparison of these approaches is described in detail in Chapter 9. The advantages and disadvantages of each system are highlighted. However, the work described in this thesis has the following limitations.

- Applying all systems on the same datasets.

  Even though approaches described in Chapter 6 and Chapter 7 were applied to QALD datasets, they were still different benchmarks. The CCG-based Semantic Parsing Approach was evaluated on the QALD-4 and the Dependency parse tree-based Semantic Parsing Approach was evaluated on the QALD-6. The Neural Network-based Semantic Parsing Approach was evaluated only on the SimpleQuestions dataset. Similarly, application of the Dependency parse tree-based Semantic Parsing Approach to SimpleQuestions dataset is another limitation.

- Applying the Neural Network-based Semantic Parsing Approach to QALD datasets.

  The presented neural network based approach can predict a single triple given a question. This limitation makes it harder to apply it to QALD like datasets properly since the queries are not limited to a single triple and some include quantifiers, etc.

- Applying the Dependency parse tree-based Semantic Parsing Approach on all languages that are available in QALD datasets.

  As a proof of concept, we trained on three languages and compared the results. These languages were chosen because at the time of the implementation these were the languages that M-ATOLL (Walter et al., 2014) supported at the time of writing the thesis.

[1] https://fasttext.cc/docs/en/crawl-vectors.html

10.4 Future Work

We plan to improve the following points and adding additional components in order to improve the quality of predictions for the model architectures presented in Chapter 7 and Chapter 8. We mentioned earlier that the limitation of approaches lies on evaluating all of the proposed model architectures on the same dataset.

10.4.1 Dependency parse tree-based Semantic Parsing Approach

The quality of predictions increases rapidly if the provided lexicon has a bigger coverage or a larger number of training examples. We plan to incorporate *deep contextualized word vectors* (Peters et al., 2018) that take into account not only the context of words but the syntactic structures of the whole sentence. Such a difference might improve the fact that sometimes word embeddings give similar results for words with opposite meanings, e.g. *youngest* and *oldest*. It would be an interesting research question to consider comparing traditional and deep contextual word embeddings for the task of QA.

Another future development can be considered in doing transfer learning. The idea is to train a model on a single language and apply the same model to other languages without explicitly training the system of those languages. Since our proposed approach in Chapter 7 has a language-independent pipeline, we could apply the learned model on a single language to others. The main change to the model would be the hand-engineered features that need to abstract from any language specific information such as lexical information.

Sequence-to-Sequence (seq2seq) (Sutskever et al., 2014b) could be another architecture to consider for this task. It has been proven in machine translation systems that such models can perform better, given enough training data. Another idea for such an architecture could be training a seq2seq model on the QALD dataset. This would be a challenging task since QALD datasets have limited training instances (200-300). Transfer learning from another task such as NER, POS tagger where there are more training instances could decrease the need of huge corpora for QA.

As a proof of concept, we have shown that we could use the pipeline for building QA system for other languages. We have chosen English, German and Spanish languages. However, QALD datasets contain questions in more than seven languages. It could be a future improvement of the pipeline to apply the system to all available languages.

10.4.2 Neural Network-based Semantic Parsing Approach

This work has been evaluated only on the SimpleQuestions (Bordes et al., 2015) dataset. The approach could be extended to other datasets such as QALD or WebQuestions (Berant et al., 2013).

The current model architecture uses words and characters as inputs. It can be extended into using the underlying dependency parse tree of a question along with some linguistic information such as POS tags. Moreover, the Named Entity Recognizer (NER) component can be further improved considering the same additional syntactic information such as dependency parse trees.

Another aspect to consider here would be a model that performs the prediction jointly. For instance, we trained a separate NER model, separate model for relation prediction and separate component that ranks the subject entities. The idea is to learn these subtasks in a joint fashion.

Appendix A

QALD Dataset

A.1 QALD-6 Instances from Training Data with Aggregation, Answer Types: Resource, Date, Number, Boolean


```
{
  "dataset": {
    "id": "qald-6-train-multilingual"
  },
  "questions": [
    {
      "question": [
        {
          "language": "en",
          "string": "Which German cities have more than 250000 inhabitants?",
          "keywords": "city, Germany, inhabitants, more than 250000"
        },
        {
          "language": "de",
          "string": "Welche deutschen Städte haben mehr als 250000 Einwohner?",
          "keywords": "Stadt, Deutschland, Einwohner, mehr als 250000"
        },
        {
          "language": "es",
          "string": "¿Qué ciudades alemanas tienen más de 250000 habitantes?",
          "keywords": "cuidad, Alemania, habitantes, más de 250000"
        },
        {
          "language": "it",
          "string": "Quali città tedesche hanno più di 250000 abitanti?",
          "keywords": "città, Germania, abitanti, più di 250000"
        },
        {
          "language": "fr",
          "string": "Quelles villes allemandes ont plus de 250000 habitants?",
          "keywords": "villes, Allemagne, habitants, plus de 250000"
        },
        {
          "language": "nl",
          "string": "Welke Duitse steden hebben meer dan 250000 inwoners?",
          "keywords": "stad, Duitsland, inwoners, meer dan 250000"
        },
        {
          "language": "ro",
          "string": "Ce oraşe germane au mai mult de 250000 de locuitori?",
          "keywords": "oraş, Germania, locuitori, mai mult de 250000"
```

```
        }
      ],
      "answertype": "resource",
      "query": {
        "sparql": "SELECT DISTINCT ?uri WHERE {
{ ?uri rdf:type dbo:City . }
UNION { ?uri rdf:type dbo:Town . }
?uri dbo:country dbr:Germany .
?uri dbo:populationTotal ?population .
FILTER ( ?population > 250000 ) } "
      },

      "aggregation": "true",
      "onlydbo": "true",
      "id": "101",
      "hybrid": "false"
    },
    {
      "question": [
        {
          "language": "en",
          "string": "What is the second highest mountain on Earth?",
          "keywords": "mountain, second highest"
        },
        {
          "language": "de",
          "string": "Was ist der zweithöchste Berg der Erde?",
          "keywords": "Berg, zweithöchster"
        },
        {
          "language": "es",
          "string": "¿Cuál es la segunda montaña más alta de la tierra?",
          "keywords": "montaña, seguna más alta"
        },
        {
          "language": "it",
          "string": "Qual è la seconda montagna più alta sulla Terra?",
          "keywords": "montagna, seconda più alta"
        },
        {
          "language": "fr",
          "string": "Quelle est la deuxième plus haute montagne de la Terre?",
          "keywords": "montagne, deuxième plus haute"
        },
        {
          "language": "nl",
          "string": "Wat is de op één na hoogste berg ter wereld?",
          "keywords": "berg, op één na hoogste"
        },
        {
          "language": "ro",
          "string": "Care este al doilea cel mai înalt munte de pe Pământ?",
          "keywords": "munte, al doilea cel mai înalt"
        }
      ],
      "answertype": "resource",
      "query": {
```

```
        "sparql": "SELECT DISTINCT ?uri WHERE {
?uri rdf:type dbo:Mountain .
?uri dbo:elevation ?elevation . }
ORDER BY DESC(?elevation)
OFFSET 1 LIMIT 1 "
      },
      "aggregation": "true",
      "onlydbo": "true",
      "id": "105",
      "hybrid": "false"
    },
    {
      "question": [
        {
          "language": "en",
          "string": "Is proinsulin a protein?",
          "keywords": "proinsulin, protein"
        },
        {
          "language": "de",
          "string": "Ist Proinsulin ein Protein?",
          "keywords": "Proinsulin, Protein"
        },
        {
          "language": "es",
          "string": "¿La proinsulina es una proteina?",
          "keywords": "proinsulina, protein"
        },
        {
          "language": "it",
          "string": "La proinsulina è una proteina?",
          "keywords": "proinsulina, proteina"
        },
        {
          "language": "fr",
          "string": "La pro-insuline est-elle une protéine?",
          "keywords": "pro-insuline, protéine"
        },
        {
          "language": "nl",
          "string": "Is proinsuline een proteïne?",
          "keywords": "proinsuline, proteïne"
        },
        {
          "language": "ro",
          "string": "Este proinsulina o proteină?",
          "keywords": "proinsulină, proteină"
        }
      ],
      "answertype": "boolean",
      "query": {
        "sparql": "ASK WHERE {
dbr:Proinsulin rdf:type dbo:Protein .
} "
      },
      "aggregation": "false",
      "onlydbo": "true",
```

```
    "id": "12",
    "hybrid": "false"
  },
  {
    "question": [
      {
        "language": "en",
        "string": "Are tree frogs a type of amphibian?",
        "keywords": "tree frog, amphibian"
      },
      {
        "language": "de",
        "string": "Sind Laubfrösche Amphibien?",
        "keywords": "Laubfrosch, Amphibie"
      },
      {
        "language": "es",
        "string": "¿Son las ranas verdes un tipo de anfibio?",
        "keywords": "rana verde, anfibio"
      },
      {
        "language": "it",
        "string": "Le rane verdi sono un tipo di anfibio?",
        "keywords": "rana verde, anfibio"
      },
      {
        "language": "fr",
        "string": "Sont les grenouilles arboricoles un type d'amphibiens?",
        "keywords": "grenouilles arboricoles, amphibien"
      },
      {
        "language": "nl",
        "string": "Zijn boomkikkers een soort amfibie?",
        "keywords": "boomkikker, amfibie"
      },
      {
        "language": "ro",
        "string": "Sunt broaștele de copac un tip de amfibieni?",
        "keywords": "broască de copac, amfibian"
      }
    ],
    "answertype": "boolean",
    "query": {
      "sparql": "ASK WHERE {  dbr:Hylidae dbo:class dbr:Amphibian . } "
    },
    "aggregation": "false",
    "onlydbo": "true",
    "id": "13",
    "hybrid": "false"
  },
  {
    "question": [
      {
        "language": "en",
        "string": "Give me all cosmonauts.",
        "keywords": "cosmonauts"
      },
```

```
        {
          "language": "de",
          "string": "Gib mir alle Kosmonauten.",
          "keywords": "Kosmonauten"
        },
        {
          "language": "es",
          "string": "Dame todas las cosmonautas.",
          "keywords": "cosmonautas"
        },
        {
          "language": "it",
          "string": "Dammi tutte le cosmonaute.",
          "keywords": "cosmonaute"
        },
        {
          "language": "fr",
          "string": "Donnes-moi tous les cosmonautes.",
          "keywords": "cosmonautes"
        },
        {
          "language": "nl",
          "string": "Geef alle kosmonauten.",
          "keywords": "kosmonauten"
        },
        {
          "language": "ro",
          "string": "Dă-mi toţi cosmonauţii.",
          "keywords": "cosmonauţi"
        }
      ],
      "answertype": "resource",
      "query": {
        "sparql": "SELECT DISTINCT ?uri WHERE {
?uri rdf:type dbo:Astronaut .
{ ?uri dbo:nationality dbr:Russia . }
UNION { ?uri dbo:nationality dbr:Soviet_Union . }
} "
      },
      "aggregation": "false",
      "onlydbo": "true",
      "id": "1",
      "hybrid": "false"
    },
    {
      "question": [
        {
          "language": "en",
          "string": "In which country does the Ganges start?",
          "keywords": "Ganges, start, country"
        },
        {
          "language": "de",
          "string": "In welchem Land entspringt der Ganges?",
          "keywords": "Ganges, entspringen, Land"
        },
        {
```

```
          "language": "es",
          "string": "¿En qué país nace el Ganges?",
          "keywords": "Ganges, país, orígen"
        },
        {
          "language": "it",
          "string": "In quale stato nasce il Gange?",
          "keywords": "Gange, stato, origine"
        },
        {
          "language": "fr",
          "string": "Dans quel pays commence le Gange?",
          "keywords": "pays, commence, Gange"
        },
        {
          "language": "nl",
          "string": "In welk land ontspringt de Ganges?",
          "keywords": "ontspringt, Ganges"
        },
        {
          "language": "ro",
          "string": "Din ce țară izvorăște Ganga?",
          "keywords": "Gangele, izvor, țară"
        }
      ],
      "answertype": "resource",
      "query": {
        "sparql": "SELECT DISTINCT ?uri WHERE {
dbr:Ganges dbp:sourceCountry ?l .
?uri rdfs:label ?l .
?uri rdf:type dbo:Country .
}"
      },
      "aggregation": "false",
      "onlydbo": "true",
      "id": "10",
      "hybrid": "false"
    },
    {
      "question": [
        {
          "language": "en",
          "string": "When did Michael Jackson die?",
          "keywords": "Michael Jackson, die"
        },
        {
          "language": "de",
          "string": "Wann starb Michael Jackson?",
          "keywords": "Michael Jackson, gestorben"
        },
        {
          "language": "es",
          "string": "¿Cuándo murió Michael Jackson?",
          "keywords": "Michael Jackson, muerto"
        },
        {
          "language": "it",
```

```
          "string": "Quando è morto Michael Jackson?",
          "keywords": "Michael Jackson, morto"
        },
        {
          "language": "fr",
          "string": "Quand mourut Michael Jackson?",
          "keywords": "Michael Jackson, mort"
        },
        {
          "language": "nl",
          "string": "Wanneer overleed Michael Jackson?",
          "keywords": "overleden, Michael Jackson"
        },
        {
          "language": "ro",
          "string": "Când a murit Michael Jackson?",
          "keywords": "Michael Jackson, muri"
        }
      ],
      "answertype": "date",
      "query": {
        "sparql": "SELECT DISTINCT ?date WHERE {
dbr:Michael_Jackson dbo:deathDate ?date .
} "
      },
      "aggregation": "false",
      "onlydbo": "true",
      "id": "174",
      "hybrid": "false"
    },
    {
      "question": [
        {
          "language": "en",
          "string": "Give me the birthdays of all actors of the television show Charme
          "keywords": "television show, Charmed, actor, birthday"
        },
        {
          "language": "de",
          "string": "Gib mir die Geburtstage von allen Darstellern der Fernsehserie Ch
          "keywords": "Fernsehserie, Charmed, Darsteller, Geburtstag"
        },
        {
          "language": "es",
          "string": "Dame los cumpleaños de los actores de la serie de televisión Char
          "keywords": "serie televisiva, Charmed, actores, cumpleaños"
        },
        {
          "language": "it",
          "string": "Dammi le date dei compleanni di tutti gli attori della serie tele
          "keywords": "serie televisiva, Charmed, attore, compleanno"
        },
        {
          "language": "fr",
          "string": "Donnes-moi les dates de naissance des acteurs de la série télévis
          "keywords": "série télévisée, Charmed, acteurs, date de naissance"
        },
```

```
        {
          "language": "nl",
          "string": "Noem de verjaardag van alle acteurs uit de televisieserie Cha
          "keywords": "verjaardag, acteur, televisieserie, Charmed"
        },
        {
          "language": "ro",
          "string": "Dă-mi zilele de naștere a tuturor actorilor din serialul de t
          "keywords": "serial de televiziune, Charmed, actor, zi de naștere"
        }
      ],
      "answertype": "date",
      "query": {
        "sparql": "SELECT DISTINCT ?date WHERE {
dbr:Charmed dbo:starring ?actor .
?actor dbo:birthDate ?date .
} "
      },
      "aggregation": "false",
      "onlydbo": "true",
      "id": "2",
      "hybrid": "false"
    },
    {
      "question": [
        {
          "language": "en",
          "string": "What is the birth name of Angela Merkel?",
          "keywords": "birth name, Angela Merkel"
        },
        {
          "language": "de",
          "string": "Was ist der Geburtsname von Angela Merkel?",
          "keywords": "Geburtsname, Angela Merkel"
        },
        {
          "language": "es",
          "string": "¿Cuál es el nombre de soltera de Angela Merkel?",
          "keywords": "nombre de soltera, Angela Merkel"
        },
        {
          "language": "it",
          "string": "Qual è il nome da nubile di Angela Merkel?",
          "keywords": "nome da nubile, Angela Merkel"
        },
        {
          "language": "fr",
          "string": "Quel est le nom de jeune fille d'Angela Merkel?",
          "keywords": "nom de jeune fille, Angela Merkel"
        },
        {
          "language": "nl",
          "string": "Wat is de meisjesnaam van Angela Merkel?",
          "keywords": "meisjesnaam, Angela Merkel"
        },
        {
          "language": "ro",
```

```
          "string": "Care este numele de domnișoară al Angelei Merkel?",
          "keywords": "nume de domnișoară, Angela Merkel"
        }
      ],
      "answertype": "string",
      "query": {
        "sparql": "SELECT DISTINCT ?string WHERE {  dbr:Angela_Merkel dbo:birthName ?s
      },
      "aggregation": "false",
      "onlydbo": "true",
      "id": "130",
      "hybrid": "false"
    },
    {
      "question": [
        {
          "language": "en",
          "string": "Give me all B-sides of the Ramones.",
          "keywords": "Ramones, B-sides"
        },
        {
          "language": "de",
          "string": "Gib mir alle B-Seiten der Ramones.",
          "keywords": "Ramones, B-Seiten"
        },
        {
          "language": "es",
          "string": "Dame todas las caras B de los Ramones.",
          "keywords": "Ramones, cara B"
        },
        {
          "language": "it",
          "string": "Dammi tutti i lati B dei Ramones.",
          "keywords": "Ramones, lato B"
        },
        {
          "language": "fr",
          "string": "Donnes-moi tous les faces B des Ramones.",
          "keywords": "faces B, Ramones"
        },
        {
          "language": "nl",
          "string": "Geef alle B-kantjes van singles van de Ramones.",
          "keywords": "B-kant, single, de Ramones"
        },
        {
          "language": "ro",
          "string": "Dă-mi toate laturile B ale Ramones",
          "keywords": "Ramones, laturi B"
        }
      ],
      "answertype": "string",
      "query": {
        "sparql": "SELECT DISTINCT ?string WHERE {
?x dbo:musicalArtist dbr:Ramones .
?x dbo:bSide ?string .
} "
```

```
  },
  "aggregation": "false",
  "onlydbo": "true",
  "id": "196",
  "hybrid": "false"
},
{
  "question": [
    {
      "language": "en",
      "string": "How many students does the Free University in Amsterdam have?
      "keywords": "Free University, Amsterdam, students"
    },
    {
      "language": "de",
      "string": "Wieviele Studenten hat die Freie Universität in Amsterdam?",
      "keywords": "Freie Universität, Amsterdam, Studenten"
    },
    {
      "language": "es",
      "string": "¿Cuántos estudiantes tiene la Universidad Libre de Amsterdam?
      "keywords": "Universidad Libre, Amsterdam, estudiantes"
    },
    {
      "language": "it",
      "string": "Quanti studenti ci sono nella Libera Università di Amsterdam?
      "keywords": "Libera Università, Amsterdam, studenti"
    },
    {
      "language": "fr",
      "string": "Combien d'étudiants a l'université libre d'Amsterdam?",
      "keywords": "Université libre, Amsterdam, étudiants"
    },
    {
      "language": "nl",
      "string": "Hoeveel studenten heeft de Vrije Universiteit in Amsterdam?",
      "keywords": "Vrije Universiteit, Amsterdam, studenten"
    },
    {
      "language": "ro",
      "string": "Câți studenți are Universitatea Liberă din Amsterdam?",
      "keywords": "Universitatea Liberă, Amsterdam, studenți"
    }
  ],
  "answertype": "number",
  "query": {
    "sparql": "SELECT DISTINCT ?num WHERE {  dbr:VU_University_Amsterdam dbo:n
  },
  "aggregation": "false",
  "onlydbo": "true",
  "id": "104",
  "hybrid": "false"
},
{
  "question": [
    {
      "language": "en",
```

```
          "string": "How tall is Michael Jordan?",
          "keywords": "tall, Michael Jordan"
        },
        {
          "language": "de",
          "string": "Wie groß ist Michael Jordan?",
          "keywords": "groß, Michael Jordan"
        },
        {
          "language": "es",
          "string": "¿Qué altura tiene Michael Jordan?",
          "keywords": "altura, Michael Jordan"
        },
        {
          "language": "it",
          "string": "Quanto è alto Michael Jordan?",
          "keywords": "altezza, Michael Jordan"
        },
        {
          "language": "fr",
          "string": "Quelle est la taille de Michael Jordan?",
          "keywords": "taille, Michael Jordan"
        },
        {
          "language": "nl",
          "string": "Hoe lang is Michael Jordan?",
          "keywords": "lang, Michael Jordan"
        },
        {
          "language": "ro",
          "string": "Ce înățime are Michael Jordan?",
          "keywords": "înălțime, Michael Jordan"
        }
      ],
      "answertype": "number",
      "query": {
        "sparql": "SELECT DISTINCT ?num WHERE {  dbr:Michael_Jordan  dbo:height ?num
      },
      "aggregation": "false",
      "onlydbo": "true",
      "id": "120",
      "hybrid": "false"
    }
  ]
}
```

Bibliography

Abele, Andrejs et al. (2017). *Linking Open Data cloud diagram (2017)*.

Aggarwal, Nitish and Paul Buitelaar (2012). "A system description of natural language query over dbpedia". In: *Proc. of Interacting with Linked Data (ILD 2012)[37]*, pp. 96–99.

Aghaebrahimian, Ahmad and Filip Jurčíček (2016). "Open-domain factoid question answering via knowledge graph search". In: *Proceedings of the Workshop on Human-Computer Question Answering*, pp. 22–28.

Artzi, Yoav and Luke Zettlemoyer (2011). "Bootstrapping semantic parsers from conversations". In: *Proceedings of the conference on empirical methods in natural language processing*. Association for Computational Linguistics, pp. 421–432.

Auer, Sören et al. (2007). "Dbpedia: A nucleus for a web of open data". In: *The semantic web*, pp. 722–735.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2014). "Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473*.

Banarescu, Laura et al. (2013). "Abstract meaning representation for sembanking". In: *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pp. 178–186.

Basile, Valerio et al. (2016). "Populating a Knowledge Base with Object-Location Relations Using Distributional Semantics". In: *Proc. of EKAW*, pp. 34–50.

Baudiš, Petr and J Šedivy (2015). "QALD challenge and the YodaQA system: Prototype notes". In:

Beaumont, Romain, Brigitte Grau, and Anne-Laure Ligozat (2015). "SemGraphQA@ QALD5: LIMSI participation at QALD5@ CLEF." In: *CLEF (Working Notes)*.

Berant, Jonathan et al. (2013). "Semantic Parsing on Freebase from Question-Answer Pairs." In: *EMNLP*. Vol. 2. 5, p. 6.

Berners-Lee, Tim (2006). *Linked data, 2006*.

Berners-Lee, Tim, James Hendler, Ora Lassila, et al. (2001). "The semantic web". In: *Scientific american* 284.5, pp. 28–37.

Bollacker, Kurt et al. (2008). "Freebase: a collaboratively created graph database for structuring human knowledge". In: *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. AcM, pp. 1247–1250.

Bond, Francis and Ryan Foster (2013). "Linking and Extending an Open Multilingual Wordnet." In: *ACL (1)*, pp. 1352–1362.

Bordes, Antoine et al. (2013). "Translating embeddings for modeling multi-relational data". In: *Advances in neural information processing systems*, pp. 2787–2795.

Bordes, Antoine et al. (2015). "Large-scale simple question answering with memory networks". In: *arXiv preprint arXiv:1506.02075*.

Cabrio, Elena et al. (2012). "QAKiS: an open domain QA system based on relational patterns". In: *Proceedings of the 2012th International Conference on Posters & Demonstrations Track-Volume 914*. CEUR-WS. org, pp. 9–12.

Chiu, Jason PC and Eric Nichols (2015). "Named entity recognition with bidirectional LSTM-CNNs". In: *arXiv preprint arXiv:1511.08308*.

Cho, Kyunghyun et al. (2014). "On the properties of neural machine translation: Encoder-decoder approaches". In: *arXiv preprint arXiv:1409.1259*.

Church, Alonzo (1936). "An unsolvable problem of elementary number theory". In: *American journal of mathematics* 58.2, pp. 345–363.

Cimiano, Philipp (2009). "Flexible Semantic Composition with DUDES". In: *Proceedings of the 8th International Conference on Computational Semantics (IWCS)*, pp. 272–276.

Cimiano, Philipp, Anette Frank, and Uwe Reyle (2007). "UDRT-based semantics construction for LTAG – and what it tells us about the role of adjunction in LTAG". In: *Proceedings of the 7th International Workshop on Computational Semantics (IWCS)*, pp. 41–52.

Cimiano, Philipp et al. (2013). "Multilingual question answering over linked data (qald-3): Lab overview". In: *International Conference of the Cross-Language Evaluation Forum for European Languages*. Springer, pp. 321–332.

Cimiano, Philipp, Christina Unger, and John McCrae (2014). "Ontology-based interpretation of natural language". In: *Synthesis Lectures on Human Language Technologies* 7.2, pp. 1–178.

Clarke, James et al. (2010). "Driving semantic parsing from the world's response". In: *Proceedings of the fourteenth conference on computational natural language learning*. Association for Computational Linguistics, pp. 18–27.

Cocke, John (1970). "Programming languages and their compilers: Preliminary notes". In:

Dai, Zihang, Lei Li, and Wei Xu (2016). "Cfo: Conditional focused neural question answering with large-scale knowledge bases". In: *arXiv preprint arXiv:1606.01994*.

Damljanovic, Danica, Milan Agatonovic, and Hamish Cunningham (2010). "Identification of the Question Focus: Combining Syntactic Analysis and Ontology-based Lookup through the User Interaction." In: *LREC*.

De Marneffe, Marie-Catherine et al. (2014). "Universal Stanford dependencies: A cross-linguistic typology." In: *LREC*. Vol. 14, pp. 4585–92.

Deng, Jia et al. (2009). "Imagenet: A large-scale hierarchical image database". In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. Ieee, pp. 248–255.

Diefenbach, Dennis et al. (2017a). "Core techniques of question answering systems over knowledge bases: a survey". In: *Knowledge and Information systems*, pp. 1–41.

Diefenbach, Dennis, Kamal Singh, and Pierre Maret (2017b). "Wdaqua-core0: a question answering component for the research community". In: *ESWC, 7th Open Challenge on Question Answering over Linked Data (QALD-7)*.

Dima, Corina (2013). "Intui2: A Prototype System for Question Answering over Linked Data." In: *CLEF (Working Notes)*.

— (2014a). "Answering Natural Language Questions with Intui3." In: *CLEF (Working Notes)*, pp. 1201–1211.

— (2014b). "Answering Natural Language Questions with Intui3". In: *CLEF 2014 Working Notes Papers*.

Dragoni, Mauro, Monika Solanki, and Eva Blomqvist (2017). *Semantic Web Challenges: 4th SemWebEval Challenge at ESWC 2017, Portoroz, Slovenia, May 28-June 1, 2017, Revised Selected Papers*. Vol. 769. Springer.

Evang, Kilian and Johan Bos (2016). "Cross-lingual learning of an open-domain semantic parser". In: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pp. 579–588.

Firth, John R (1957). "A synopsis of linguistic theory, 1930-1955". In: *Studies in linguistic analysis*.

Ge, Ruifang and Raymond J. Mooney (2005). "A Statistical Semantic Parser that Integrates Syntax and Semantics". In: *Proceedings of CoNLL-2005*. Ann Arbor, Michigan.

Gerber, Daniel and Axel-Cyrille Ngonga Ngomo (2012). "Extracting multilingual natural-language patterns for rdf predicates". In: *International Conference on Knowledge Engineering and Knowledge Management*. Springer, pp. 87–96.

Giannone, Cristina, Valentina Bellomaria, and Roberto Basili (2013). "A HMM-based Approach to Question Answering against Linked Data." In: *CLEF (Working Notes)*.

Goldwasser, Dan and Dan Roth (2011). "Learning from natural instructions". In: *Proceedings of International Joint Conference on Artificial Intellingence*.

Golub, David and Xiaodong He (2016). "Character-level question answering with attention". In: *EMNLP*.

Graves, Alex, Abdel-rahman Mohamed, and Geoffrey Hinton (2013). "Speech recognition with deep recurrent neural networks". In: *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE, pp. 6645–6649.

Hakimov, Sherzod et al. (2013). "Semantic question answering system over linked data using relational patterns". In: *Proceedings of the Joint EDBT/ICDT 2013 Workshops*. ACM, pp. 83–88.

Hakimov, Sherzod et al. (2015). "Applying semantic parsing to question answering over linked data: Addressing the lexical gap". In: *International Conference on Applications of Natural Language to Information Systems*. Springer, pp. 103–109.

Hakimov, Sherzod et al. (2016). "Combining textual and graph-based features for named entity disambiguation using undirected probabilistic graphical models". In: *Knowledge Engineering and Knowledge Management: 20th International Conference, EKAW 2016, Bologna, Italy, November 19-23, 2016, Proceedings 20*. Springer, pp. 288–302.

Hakimov, Sherzod, Soufian Jebbara, and Philipp Cimiano (2017). "AMUSE: multilingual semantic parsing for question answering over linked data". In: *International Semantic Web Conference*. Springer, pp. 329–346.

— (2019). "Evaluating Architectural Choices for Deep Learning Approaches for Question Answering over Knowledge Bases". In: *International Semantic Computing Conference*.

Hastings, W. K. (1970). "Monte Carlo sampling methods using Markov chains and their applications". In: *Biometrika* 57.1, pp. 97–109.

He, Shizhu et al. (2014). "CASIA@ V2: A MLN-based Question Answering System over Linked Data." In: *CLEF (Working Notes)*, pp. 1249–1259.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long short-term memory". In: *Neural computation* 9.8, pp. 1735–1780.

Höffner, Konrad et al. (2017). "Survey on challenges of Question Answering in the Semantic Web". In: *Semantic Web* 8, pp. 895–920.

Joulin, Armand et al. (2016). "Bag of Tricks for Efficient Text Classification". In: *arXiv preprint arXiv:1607.01759*.

Kamp, Hans and Uwe Reyle (1993). *From Discourse to Logic; Introduction to the Modeltheoretic Semantics of natural language*. Kluwer, Dordrecht.

Kasami, Tadao (1965). *An efficient recognition and syntax-analysis algorithm for context-free languages.* Tech. rep. Hawaii University, Honolulu Department of Electrical Engineering.

Kate, Rohit J. and Raymond J. Mooney (2006). "Using String-Kernels for Learning Semantic Parsers". In: *ACL 2006: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*. Morristown, NJ, USA: Association for Computational Linguistics, pp. 913–920.

Kate, Rohit J., Yuk Wah Wong, and Raymond J. Mooney (2005). "Learning to Transform Natural to Formal Languages". In: *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*. Pittsburgh, PA, pp. 1062–1068.

Krishnamurthy, Jayant and Tom M Mitchell (2012). "Weakly supervised training of semantic parsers". In: *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, pp. 754–765.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*, pp. 1097–1105.

Kwiatkowski, Tom et al. (2010). "Inducing probabilistic CCG grammars from logical form with higher-order unification". In: *Proceedings of the 2010 conference on empirical methods in natural language processing*. Association for Computational Linguistics, pp. 1223–1233.

Kwiatkowski, Tom et al. (2013). "Scaling semantic parsers with on-the-fly ontology matching". In:

Lafferty, John, Andrew McCallum, and Fernando CN Pereira (2001). "Conditional random fields: Probabilistic models for segmenting and labeling sequence data". In:

LeCun, Yann et al. (1998). "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11, pp. 2278–2324.

Liang, Percy, Michael I Jordan, and Dan Klein (2011). "Learning dependency-based compositional semantics". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, pp. 590–599.

Lopez, Vanessa et al. (2012). "Poweraqua: Supporting users in querying and exploring the semantic web". In: *Semantic Web* 3.3, pp. 249–265.

Lopez, Vanessa et al. (2013). "Evaluating question answering over linked data". In: *Web Semantics: Science, Services and Agents on the World Wide Web* 21, pp. 3–13.

Lukovnikov, Denis et al. (2017). "Neural Network-based Question Answering over Knowledge Graphs on Word and Character Level". In: *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, pp. 1211–1220.

Mendes, Pablo N et al. (2011). "DBpedia spotlight: shedding light on the web of documents". In: *Proceedings of the 7th international conference on semantic systems*. ACM, pp. 1–8.

Metropolis, Nicholas et al. (1953). "Equation of state calculations by fast computing machines". In: *The journal of chemical physics* 21.6, pp. 1087–1092.

Mikolov, Tomas et al. (2013). "Distributed representations of words and phrases and their compositionality". In: *Advances in neural information processing systems*, pp. 3111–3119.

Miller, George A (1995a). "WordNet: a lexical database for English". In: *Communications of the ACM* 38.11, pp. 39–41.

— (1995b). "WordNet: a lexical database for English". In: *Communications of the ACM* 38.11, pp. 39–41.

Montague, Richard (1970a). "English as a formal language". In:

— (1970b). "Universal grammar". In: *Theoria* 36.3, pp. 373–398.

Nakashole, Ndapandula, Gerhard Weikum, and Fabian Suchanek (2012). "PATTY: a taxonomy of relational patterns with semantic types". In: *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, pp. 1135–1145.

Navigli, Roberto and Simone Paolo Ponzetto (2012). "BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network". In: *Artificial Intelligence* 193, pp. 217–250.

Nivre, Joakim et al. (2016). "Universal Dependencies v1: A Multilingual Treebank Collection." In: *LREC*.

Nivre, Joakim et al. (2017). *Universal Dependencies 2.0*. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University.

Park, Seonyeong, Hyosup Shim, and Gary Geunbae Lee (2014a). "ISOFT at QALD-4: Semantic similarity-based question answering system over linked data." In: *CLEF (Working Notes)*.

— (2014b). "ISOFT at QALD-4: Semantic similarity-based question answering system over linked data". In: *CLEF 2014 Working Notes Papers*.

Pennington, Jeffrey, Richard Socher, and Christopher D. Manning (2014). "GloVe: Global Vectors for Word Representation". In: *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543.

Peters, Matthew E. et al. (2018). "Deep contextualized word representations". In: *Proc. of NAACL*.

Pradel, Camille, Ollivier Haemmerlé, and Nathalie Hernandez (2012). "A semantic web interface using patterns: the SWIP system". In: *Graph Structures for Knowledge Representation and Reasoning*, pp. 172–187.

Rabiner, Lawrence R (1989). "A tutorial on hidden Markov models and selected applications in speech recognition". In: *Proceedings of the IEEE* 77.2, pp. 257–286.

Rastegari, Mohammad et al. (2016). "Xnor-net: Imagenet classification using binary convolutional neural networks". In: *European Conference on Computer Vision*. Springer, pp. 525–542.

Reddy, Siva et al. (2016). "Transforming Dependency Structures to Logical Forms for Semantic Parsing". In: *Transactions of the ACL* 4, pp. 127–140.

Reddy, Siva et al. (2017). "Universal Semantic Parsing". In: *Proceedings of EMNLP*.

Reyle, Uwe (1993). "Dealing with ambiguities by underspecification: Construction, representation and deduction". In: *Journal of Semantics* 10.2, pp. 123–179.

Ristoski, Petar and Heiko Paulheim (2016). "Rdf2vec: Rdf graph embeddings for data mining". In: *International Semantic Web Conference*. Springer, pp. 498–514.

Ruseti, Stefan et al. (2015). "QAnswer-Enhanced Entity Matching for Question Answering over Linked Data." In: *CLEF (Working Notes)*.

Shekarpour, Saeedeh et al. (2015). "Sina: Semantic interpretation of user queries for question answering on interlinked data". In: *Web Semantics: Science, Services and Agents on the World Wide Web* 30, pp. 39–51.

Shizhu, He et al. (2014). "CASIA@V2: A MLN-based Question Answering System over Linked Data". In: *CLEF 2014 Working Notes Papers*.

Srivastava, Nitish et al. (2014). "Dropout: a simple way to prevent neural networks from overfitting". In: *The Journal of Machine Learning Research* 15.1, pp. 1929–1958.

Steedman, Mark (1996). "Surface structure and interpretation". In:

— (2000). "The Syntactic Process". In: *Computational Linguistics* 131.1, pp. 146–148.

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

Sutskever, Ilya, Oriol Vinyals, and Quoc V Le (2014a). "Sequence to sequence learning with neural networks". In: *Advances in neural information processing systems*, pp. 3104–3112.

— (2014b). "Sequence to sequence learning with neural networks". In: *Advances in neural information processing systems*, pp. 3104–3112.

Sutton, Charles, Andrew McCallum, et al. (2012). "An introduction to conditional random fields". In: *Foundations and Trends in Machine Learning* 4.4, pp. 267–373.

Tang, Lappoon R and Raymond J Mooney (2001). "Using multiple clause constructors in inductive logic programming for semantic parsing". In: *European Conference on Machine Learning*. Springer, pp. 466–477.

ter Horst, Hendrik, Matthias Hartung, and Philipp Cimiano (2018). "Cold-start knowledge base population using ontology-based information extraction with conditional random

fields". In: *Luxembourg Logic for AI Summit: LuxLogAI 2018, Luxembourg, September 17-26, 2018*.

Ture, Ferhan and Oliver Jojic (2017). "No Need to Pay Attention: Simple Recurrent Neural Networks Work!" In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 2866–2872.

Unger, Christina et al. (2012). "Template-based question answering over RDF data". In: *Proceedings of the 21st international conference on World Wide Web*. ACM, pp. 639–648.

Unger, Christina et al. (2014). "Question Answering over Linked Data (QALD-4)". In: *Working Notes for CLEF 2014 Conference*. Ed. by Linda Cappellato et al.

Unger, Christina et al. (2015). "Question Answering over Linked Data (QALD-5)". In: *CLEF*.

Unger, Christina, Axel-Cyrille Ngonga Ngomo, and Elena Cabrio (2016). "6th open challenge on question answering over linked data (qald-6)". In: *Semantic Web Evaluation Challenge*. Springer, pp. 171–177.

Usbeck, Ricardo et al., eds. (2017). *Proceedings of the Joint Proceedings of BLINK2017: Benchmarking Linked Data and NLIWoD3: Natural Language Interfaces for the Web of Data (BLINK2017-NLIWoD3)* (Vienna, Austria, Oct. 21, 2017–Oct. 22, 2017). CEUR Workshop Proceedings 1932. Aachen.

Vanderwende, Lucy, Arul Menezes, and Chris Quirk (2015). "An AMR parser for English, French, German, Spanish and Japanese and a new AMR-annotated corpus". In: *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pp. 26–30.

Veyseh, Amir Pouran Ben (2016). "Cross-Lingual Question Answering Using Common Semantic Space." In: *TextGraphs@ NAACL-HLT*, pp. 15–19.

Vossen, Piek (1998). *A multilingual database with lexical semantic networks*. Springer.

Vrandečić, Denny and Markus Krötzsch (2014). "Wikidata: a free collaborative knowledgebase". In: *Communications of the ACM* 57.10, pp. 78–85.

Walter, Sebastian et al. (2012). "Evaluation of a layered approach to question answering over linked data". In: *The Semantic Web–ISWC 2012*, pp. 362–374.

Walter, Sebastian, Christina Unger, and Philipp Cimiano (2014). "M-ATOLL: A Framework for the Lexicalization of Ontologies in Multiple Languages". English. In: *The Semantic Web – ISWC 2014*. Vol. 8796. Lecture Notes in Computer Science. Springer International Publishing, pp. 472–486.

Weizenbaum, Joseph (1966). "ELIZA—a computer program for the study of natural language communication between man and machine". In: *Communications of the ACM* 9.1, pp. 36–45.

Welty, Chris et al. (2010). "Large Scale Relation Detection". In: *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*. Los Angeles, California: Association for Computational Linguistics, pp. 24–33.

Weston, Jason, Sumit Chopra, and Antoine Bordes (2015). "Memory Networks". In: *ICLR*.

White, Aaron Steven et al. (2016). "Universal decompositional semantics on universal dependencies". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1713–1723.

Wick, M. et al. (2009). "SampleRank. Learning preferences from atomic gradients". In: *NIPS Workshop on Advances in Ranking*, pp. 1–5.

Winograd, Terry (1971). *Procedures as a representation for data in a computer program for understanding natural language*. Tech. rep. MASSACHUSETTS INST OF TECH CAMBRIDGE PROJECT MAC.

— (1972). "Understanding natural language". In: *Cognitive psychology* 3.1, pp. 1–191.

Wong, Yuk Wah and Raymond J Mooney (2006). "Learning for semantic parsing with statistical machine translation". In: *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the ACL*. ACL, pp. 439–446.

Woods, William A, Ronald M Kaplan, and Bonnie Nash-Webber (1972). *The lunar sciences natural language information system*. Bolt, Beranek and Newman, Incorporated.

Xu, Kun et al. (2014a). "Answering natural language questions via phrasal semantic parsing". In: *Natural Language Processing and Chinese Computing*. Springer, pp. 333–344.

Xu, Kun, Yansong Feng, and Dongyan Zhao (2014b). "Answering Natural Language Questions via Phrasal Semantic Parsing". In: *CLEF 2014 Working Notes Papers*.

Yahya, Mohamed et al. (2012). "Natural language questions for the web of data". In: *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, pp. 379–390.

Yin, Wenpeng et al. (2016). "Simple question answering by attentive convolutional neural network". In: *26th International Conference on Computational Linguistics (COLING)*.

Younger, Daniel H (1967). "Recognition and parsing of context-free languages in time n3". In: *Information and control* 10.2, pp. 189–208.

Zelle, John M and Raymond J Mooney (1996). "Learning to parse database queries using inductive logic programming". In: *Proceedings of the national conference on artificial intelligence*, pp. 1050–1055.

Zettlemoyer, Luke S and Michael Collins (2005). "Learning to Map Sentences to Logical Form : Structured Classification with Probabilistic Categorial Grammars". In: *21st Conference on Uncertainty in Artificial Intelligence* x, pp. 658–666. arXiv: 1207.1420.

— (2007). "Online Learning of Relaxed CCG Grammars for Parsing to Logical Form." In: *Proceedings of EMNLP*, pp. 678–687.

— (2009). "Learning context-dependent mappings from sentences to logical form". In: *Proceedings of ACL*, pp. 976–984.

Zhu, Chenhao et al. (2015). "A Graph Traversal Based Approach to Answer Non-Aggregation Questions Over DBpedia". In: *Joint International Semantic Technology Conference*. Springer, pp. 219–234.

Zou, Lei et al. (2014a). "Natural language question answering over RDF: a graph data driven approach". In: *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. ACM, pp. 313–324.

Zou, Lei et al. (2014b). "Natural language question answering over rdf: a graph data driven approach". In: *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. ACM, pp. 313–324.