Bachelor Thesis

A keyword query translator for Broccoli

Iradj Solouk

Summer 2014

University of Freiburg
Faculty of Engineering

Reviewer: Prof. Dr. Hannah Bast
Advisor: Florian Bäurle

Abstract

A keyword query translator that is used by the semantic text search engine BROCCOLI will be introduced. This keyword translator interprets a string, which is given by the user, in a way that considers the underlying data structure of BROCCOLI and tries to consider the context of the given information. A GOOGLE-like keyword search field will be used. The keyword translation leads to user friendliness in terms of not requiring knowledge about the concept of the underlying ontology and the possibility of using keywords or parts of phrases as an input. This process provides a service that can be used in place of the normal incremental query creation process.

The semantic interpretation of the input string requires natural-language-processing. WordNet, a lexical dabase for English words, and the Stanford part-of-speech tagger is used for part-of-speech tagging.

Zusammenfassung

Die vorliegende Arbeit beschäftigt sich mit einem für die Suchmaschine BROCCOLI erstellten Keyword Translator. Dieser Keyword Translator interpretiert die vom Benutzer eingegebene Phrase oder Worte so, dass die verwendeten Datenstrukturen des BROCCOLI-Systems berücksichtigt werden. Des Weiteren wird dabei auch die Bedeutung der Eingabe interpretiert. Ein Texteingabefeld wie das von GOOGLE wird verwendet. Dadurch dass kein Verständnis über die im Hintergrund verwendete Ontologie nötig ist, führt der Prozess der Übersetzung zu einer Erhöhung der Benutzerfreundlichkeit. Das System ermöglicht die Verwendung von Sätzen oder Schlagwörtern als Eingabe. Der Keyword Translator soll als Alternative für den manuellen Prozess der Suchanfragenerstellung dienen. Der manuelle Prozess der Suchanfragenerstellung erfolgt durch inkrementelles Aufbauen der Suchanfrage.

Das semantische Abbilden der Texteingabe gebraucht natürliche Sprachverarbeitung. Hierfür werden die lexikalische Datenbank WordNet und die Stanford Wortartenerkennung von englischer Sprache verwendet.

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Main challenge & line of attack . . . 2
- 1.3 Natural-language-processing . . . 2
  - 1.3.1 WordNet . . . 3
  - 1.3.2 Part-of-speech Tagger . . . 3
- 1.4 BROCCOLI . . . 3
  - 1.4.1 Definition & functionality . . . 3
  - 1.4.2 Semantic Full-Text Search . . . 4
  - 1.4.3 User Interface & query construction . . . 4
- 1.5 Structure . . . 6

2 Related Work **7**

3 Keyword Translation **9**
- 3.1 Initial point . . . 9
- 3.2 General problems with language . . . 10
- 3.3 Resource oriented problems . . . 11
  - 3.3.1 Ambiguous mapping . . . 11
  - 3.3.2 Missing relation . . . 12
  - 3.3.3 Dealing with stop words . . . 12
  - 3.3.4 Different vocabulary . . . 13
- 3.4 Algorithm . . . 13
  - 3.4.1 Instance as root . . . 14
  - 3.4.2 Class as root . . . 14
  - 3.4.3 Instance recognition . . . 18

4 Analysis **21**
- 4.1 Theoretical analysis . . . 21
  - 4.1.1 Analyzing instance and class recognition . . . 21
  - 4.1.2 Analyzing the inference of relations . . . 21
- 4.2 Empirical analysis . . . 22
  - 4.2.1 Problems . . . 22
  - 4.2.2 Evaluation setup . . . 22
  - 4.2.3 Quality & runtime performance . . . 23
  - 4.2.4 Discussion of the quality performance . . . 24
  - 4.2.5 Comparison to other approaches . . . 27

5 Outcome **28**
5.1 Future Work 28
5.2 Conclusion 30

A Appendix: SemSearch Queries **31**

1. Introduction

"People can't share knowledge if they don't speak a common language."

— Thomas Davenport

The thesis will deal with a special component of a certain type of search engines. This certain type of search engine is called semantic search engine. Our search engine deals with a large amount of data[2]. The data elements of the used search engine are structured and linked together according to the structure of an *ontology*. By *ontology* a knowledge base is meant. We think that using such a data structure is promising in terms of information provision. It permits complex querying in an intuitive way. The reason why the formerly mentioned special component of semantic search engines is developed will be presented in the following paragraph.

1.1 Motivation

There are different user interface elements for semantic search engines such as facets, graph browsers, breadcrumbs or simple input fields. The semantic search engine we work with is called BROCCOLI. Querying it with the current user interface can be performed incrementally via a facet-like box[4]. Let us suppose to query the following example: "Football players with goalkeeper position".
At the moment, one can manually select the class "football" player and add the relation "position" and add the instance "goalkeeper" via the facet-like box. The internal query consists of the search elements that are going to be presented in 1.4.3 . This procedure is the incremental and manual construction of the query where the user has to click and type in the terms of the elements. Therefore we develop a special component for BROCCOLI called the keyword translator. This translator has to handle the internal logic of the queries in respect to the input string. One can monitor the current query via a breadcrumb panel that visualizes the latter mentioned elements of the query. But for this kind of querying the user requires some understanding of the concept of ontology. Keyword search solely via a single input field like GOOGLE and YAHOO is still a common and effective way to find useful information on the internet. Most people are using text keyword search, which means that in order to provide an accessible method of querying, we think a keyword translator has to come up. The search application BROCCOLI combines different user interface elements and also makes use of a keyword translator on which we will focus.

At the moment, a prototype for the keyword translation is already in use. It uses a straightforward approach which does not take synonyms into account. Thus the keyword translation will be redone. It brought forth this thesis.

1.2 Main challenge & line of attack

The main challenge is the mapping of a string onto an internal query. How this is done and problems that one may face overcoming this challenge are presented in chapter 3. One of those problems is that the user has her/his own vocabulary or language when typing in the input string. He may use a sentence structure or just keywords and she/he may paraphrase what he/she wants to query. The quote from the beginning by Thomas Davenport states that a common language is necessary in order to share knowledge. Thus the quote explains the problem at its core. The resources which the user wants to query may be stored and named with terms that do not have morphological similarity with the vocabulary entered by the user.

The approach to solving this combines the analysis of phrase structures, semantic mapping and the communication with the resource container. Semantic mapping means the usage of the correlation of input words and words which are semantically related to them in order to find resources for these input words. The first two parts are perfomed while using a lower number of backend communications than are needed for the third part. The combination of them is a highly dynamic process and promises a good translation process. This is why the next subsections serve as a basis for understanding the problems and the approaches for solving them in detail.

1.3 Natural-language-processing

This section shall give a basis for the terms used in the linguistic analysis. Semantic relations are being used in order to map words onto a resource of the search engine's resource container. Some of these expressions will be mentioned later on. Now a list of semantic relations towards the target expression is presented[8].

Synonymy: The synonym of a target expression is another expression that can substitute the target expression without changing the truth value of a sentence in which the target expression has been used. The definition of this can be widened in respect to a certain context. E.g. "car" is a synonym for "automobile" and vice versa.

Antonymy: Examples are used to give the idea of an antonym. The expression "rise" can be an antonym of "fall". But "rise" cannot be an antonym of "descend". That antonyms have the opposite meaning towards each other is a necessary condition but not a sufficient one.

Hyponymy: An example is used to give the idea of a hyponym. The expressions "dog", "pig", "chinchilla", "baboon" and "elk" are hyponyms of "animal". A hyponym is a more specific, less abstract expression for the target expression.

Hypernymy: An example is used to give the idea of a hypernym. The expression "animal" is a hypernym of "dog", "pig", "chinchilla", "baboon" and "elk". A hypernym is a more abstract, less specific expression for the target expression.

There are even more relation expressions, but the use of them will not be mentioned in this work.

A common english sentence consists of a subject, predicate and an object. Consider the sentence “The snow is white”. The word “snow” is the subject, the word “is” is the predicate and the word “white” is the object.

1.3.1 WordNet

The (Princeton)WordNet [7] is a lexical database for the English language. The database contains more than 118,000 word forms. Colloquially it can be described as a “social network” for words. It is possible to find antonyms, hyponyms, hypernyms, synonyms, meronyms, troponyms and entailments for words that do have these. WordNet does not only get the semantically related words, one can also retrieve descriptions and usage phrases of certain words. Using WordNet amplified the finding of resources in the ontology which will be addressed in chapter 3.

WordNet is also used to receive the base form of a word. Even if a word is in past tense or has an irregular plural form WordNet will have the base form. E.g. for a word like ”cacti” or ”cactuses” the base form ”cactus” can be obtained.

1.3.2 Part-of-speech Tagger

Another tool used during the translation is the (Stanford)part-of-speech tagger [11]. One can pass a phrase to the part-of-speech tagger in order to find out what kind of part-of-speech a certain word in the phrase has. This simplifies the query translation and resource recognition.

1.4 Broccoli

1.4.1 Definition & functionality

Broccoli [2] is the semantic search engine which has been developed by the chair of algorithms at the University of Freiburg. It is the successor of Susi [6]. The combination of traditional full text search and ontology search together with its user interface provides a search application of its own kind. They use a modified version of the freebase ontology[3]. And for the text search part the english wikipedia is used. The search process works with four kinds of search elements:

- word
- class
- instance
- relation

Classes, instances and relations are used for the ontology search part, whereas the regular words are used for the text search part. There is a special kind of relation named *co-occurs with* which is used for the words element.
We will use the formerly mentioned elements in order to build a query. Such a query can be described as a tree, which uses the relations as arcs and an optional set of these elements as nodes. That tree will have a single root node which can have

arcs that only point away from the root node. This allows the user to query in an intuitive and easy way due to visual feedback.

In this application if one builds a query with only one element, e.g. the "film" class, then instances that are films will retrieved. This implies that classes can be regarded as sets of instances. These classes may have superclasses which denotes a hierarchy of classes[1]. A member of the "computer scientist" class is also a member of the "person" class unless intelligent non-humans lived on earth. Suppose we have this set of instances and would only like to query certain instances, e.g. instances of the class film, that are from a certain genre. Then we can add another search element to the query tree which in this case is the relation "genre". If one wants to query for "fantasy" films, then she/he can add the "fantasy" instance to the query tree. This search element combination can be regarded as an instance filter, which only allows film instances that are related to the "fantasy" relation. The query construction and the user interface are presented later on.

1.4.2 Semantic Full-Text Search

The analysis of web search query logs showed that most highly correlated items are constituents of phrases[10]. Thus regarding search words as parts of phrases may increase the quality of a search engine. Therefore the following described approach of BROCCOLI solves this problem.

In order to use the text search natural language processing has been done in the preprocessing step. For improvement of the quality *sentence constituent identification*, which is the decomposition of the sentence, is applied. It helps to find the context of the words. In this way the document of an instance(**rhubarb** in this case) like

> "The usable parts of **rhubarb**, a plant from the Polygonaceae family, are the medicinally used roots and the edible stalks, however its leaves are toxic."

which has been adapted from the paper about our system [2] will not return a matching context for the keywords ,,plants with edible leaves". This can be regarded as semantic interpretation. As a result the interpretation solves the problem of missing relations inside the ontology. The *co-occurs with*-relation can be used to simulate a relation with a class or an instance.

1.4.3 User Interface & query construction

For deeper understanding of the keyword translation the introduction of the user interface is inevitable. Figure 1.1 reflects the user interface of BROCCOLI . The boxes are color coded. Each color code represents one of the previously mentioned search elements. The red elements refer to a class, blue elements refer to an instance and green elements refer to a relation. The yellowish elements refer to the word-elements. An instance can also be a value which is handy if one wants to find people with a small height. Here one can see a constructed example query.

[1] This is called a taxonomy.

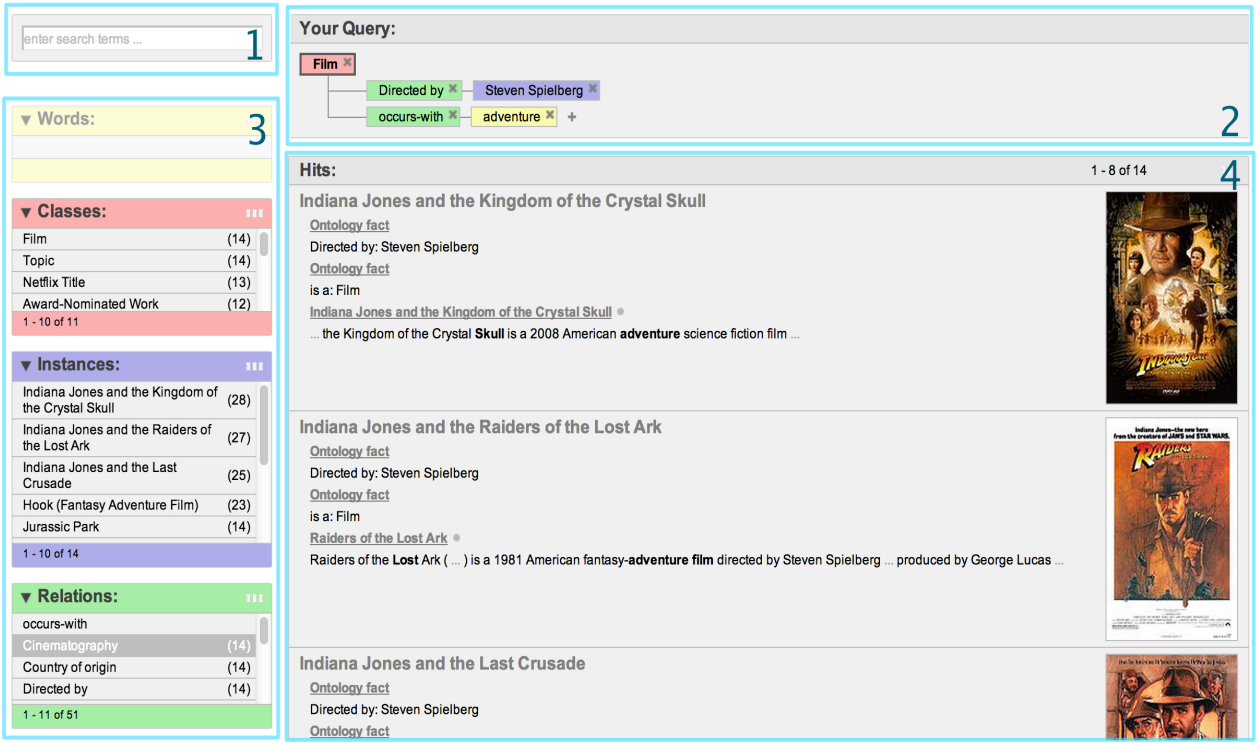


Figure 1.1: The BROCCOLI user interface is shown. The user interface is divided into four boxes. These boxes are numbered. The box (1) is the input field which allows keyboard input. Box (2) is the query panel which visualizes the current query. Box (3) shows boxes which propose possible search elements, that can be used to refine the query. The proposals update according to the keyboard input of the input field. Box number (4) depicts the actual results of the query.

The interface consists of:

1. *Input Field* : Here the user can either type in the strings she/he wants to find a single search element for or the whole keyword query.

2. *Query Panel* : This panel helps the user to monitor his own query. It depicts the dependencies of the single search elements. As soon as the query tree consists of at least one node the query has one active node. It is also possible to mark an element that is not the active node as the active element. From this active node on the query construction will be resumed. In Figure 1.1 the active element is the class named "Film", which is depicted by the black frame around the red box, that is labeled with "Film". And one can remove or manipulate single search elements if desired.

3. *Proposal Boxes* : Here proposals of search elements are shown. These proposals incorporate the current query of the active search element and the input which has been typed in by the user into the formerly mentioned input field.

4. *Hits Area* : The actual query results that are returned. Links to their freebase page, a picture if it is given and an excerpt of their corresponding

wikipedia documents are shown. These excerpts contain highlighted words which refer to the semantic context of the word search element via sentence decomposition.

The user interface has a clear design. A lot of possibilities to refine queries are offered as well. It supports visual feedback via breadcrumb panel. The facet-like box will help choosing the right elements and there is a single text input field which will provide the users input string to the keyword translator. For further investigation one can find the master's thesis from which the UI emerged in [4].

1.5 Structure

Chapter 1:
The main line of attack, semantic relations and applications that will help throughout the translation process have been introduced and BROCCOLI, its user interface and an example query have been shown.

Chapter 2:
Related work will be presented and compared to our work.

Chapter 3:
The actual keyword translator is introduced. The problems and how the attack of line is turned into an algorithm are described.

Chapter 4:
Theoretical analysis and an evaluation has been done here. Quality and runtime are measured and discussed. And the obvious high level approaches for solving our problem are described and compared to our approach.

Chapter 5:
Open ends and the amount of time implementing them are listed and a personal conclusion finalizes this work.

2. Related Work

In this section, other works dealing with the problems occurring within the scope of this thesis will be demonstrated and compared to this work. Thus this section will additionally define this thesis.

SPARK [14] is a semantic search system that offered an automatic translation of keyword queries into SPARQL[1] queries. In order to keep a consistent terminology, the technical terms used by the developers are stated and then explained. Their *formal query construction* module uses three major steps, that will be presented in the following. They call these steps: *term mapping*, *query graph construction* and *query ranking*.
The first step of the *formal query construction* module, called *term mapping*, is about assigning of terms, which were extracted from the input string, to the ontology resources. Such an ontology resource can pertain to the following types: class, instance, property or literal. For further simplification, literals will be treated as instances. Regarding morphological similarity of used terms via stemming, edit distance and semantic similarity via the application of thesauri like WordNet,[7] will increase the number of results. Zhou's Paper[14] mentions a *vocabulary gap* and a *lack of relation*. The *vocabulary gap* is the user's absence of knowledge about the exact names of the ontology resources. In order to consider the distinctions of direct mapping and semantic mapping, the direct one will have a higher rating. Probabilities are used for the rating. The *lack of relation* problem occurs when given the keyword query only non-relation mappings are found.
The second step of the *formal query construction* module called *query graph construction* is about collecting all combinations of possible resources. The sets in which they are collected only consists of mappings where each term has only been mapped once. Now the mappings, which are ontology resources, can be represented as nodes and arcs of a graph. The minimum spanning tree algorithm is deployed on these sets and afterwards SPARQL code is generated.
Finally the third step of the *formal query construction* module called *query ranking* is using a probabilistic model. The queries will be ranked with the help of *Baye's formula*.
Obviously, a keyword query translation depends on the input-string. This is where the *term mapping* is employed. Also morphological and semantic mapping is used for finding the matches. Graph construction and some ranking is also done but will be presented in chapter 3. By analogy, similar steps as the three elementary steps of the SPARK translator are performed in our work. But they are not used as three elementary steps, which are processed in series within the pipeline. For

[1] SPARQL is a query language for RDF. RDF stands for Resource Description Framework[13]. RDF is used in order to specify Metadata. The specification enables logical inference.

BROCCOLI these three major steps of the SPARK translator are used iteratively and in an intertwined way. First all possible classes will be collected. Then for each of these classes the non-used words are mapped onto relations and instances. Throughout the mapping, morphological and semantic similarity is also considered.

In the following, we will refer to a more recently designed search engine system called SEMSEARCHPRO [12]. Their system supports features called *keyword translation*, *query visualisation* and *query refinement*. By analogy these features are also supported by BROCCOLI. Concerning the *keyword translation* of SEMSEARCHPRO three steps should be mentioned: *construction of the query space*, *top-k query graph exploration* and *query graph ranking*. The keyword translator we use is working in a similar way if the user wants to query the instances of a certain class. The query space is constructed based on morphologically or semantially matching keywords. If the second translation step is transferred onto BROCCOLI , then finding the top queries would contain some ranking. The results are ranked e.g. by the number of words matched and the BROCCOLI proposal-hit-count. The application of graph exploration onto queries makes this step expensive compared to the first and the third one. The proof for that is found in the theoretical analysis 4.1. Our translator returns a list of queries which went through ranking.

There is also an approach for *keyword search over relational databases*[5]. They use weighted scores for the ranking considering synonyms, hyponyms, hypernyms or high similarity of words. BROCCOLI also works with synonyms, hypernyms and words which occur in the description of the used word, for which we searched the synonym. Eventually one can type in "born" for the relation "date of birth" or "place of birth". BROCCOLI uses the synonym if a relation is found and a value or an instance can be attached to it, which corresponds to adjacent terms of the word we find a synonym for. Using this approach combinations of the root class, a relation and an instance or a value can be found, which corresponds to the user's input, which makes them probable.

The well known and minimalistic search application by GOOGLE also uses semantic web features. It is possible to query an instance like *"Bob Dylan"* and a relation like *"songs"* via keyword translation. Obviously, the keyword input for the previously mentioned query has been something like "bob dylan songs" or "songs bob dylan". Eventually a list of songs by Bob Dylan is retrieved. This is a straight and simple query which might be handy for the user and is also supported with our keyword translator.

Comparing related works and our work revealed common traits and distinctive traits that have been highlighted. The most defining trait of the BROCCOLI system is the use of the word search elements. The next chapter will introduce our own translation approach.

3. Keyword Translation

3.1 Initial point

The actual translation is the projection of the input string onto the internal query supported by BROCCOLI. The mapping of the input string onto the search query will be described in the following.

"films by steven spielberg" ↦ Film —— directed-by —— Steven Spielberg

The colored nodes depict an internal query(query tree). Defining the logics or the algorithm of this mapping is the actual task. Because given the internal query, Broccoli will return hits belonging to instances which fit to the query. These hits will appear as shown in Figure 1.1, which can be found in the first chapter. In the case of this example, obviously only films directed by Steven Spielberg will be listed.

A simplified structure of the BROCCOLI-User-Interaction system, in the case of keyword translation, will be shown in the following. Figure 3.1 represents a block diagram of the system. The components are user, UI, keyword translator, communicator and resource container. The user formulates a query in form of English words. These words build the input string which is sent to the UI. The UI passes the input string to the keyword translator. There the input string is processed. By splitting the input string into the occurring terms that are separated by a space a set of terms can be extracted. Furthermore the base form and alternate forms for each term in the extracted set of terms have also been derived throughout the processing. All these derived expressions will potentially be compared to the resources of the resource container. In order to do this the functionality of the communicator is used. Later this will be presented as the "semantic mapping". The keyword translator passes a string of a derived expression and a query to the communicator. The communicator processes the given string and query in order to ask the resource container for proposals. When the resource container provides the proposed resources, it considers the current query tree sent by the keyword translator to the communicator. The communicator parses the data, which will eventually become the list of proposals, from the resource container so that it can make the proposals available for the keyword translator. The mechanism used for comparing the derived terms with the resources in respect to given query trees is applied iteratively for each of the derived words. In this way a query tree may grow and can be sent to the UI so that the user can extract the result.

In the case of a user manually constructing the query tree via the UI, the UI component is directly connected to the communicator and the keyword translator box can be skipped.

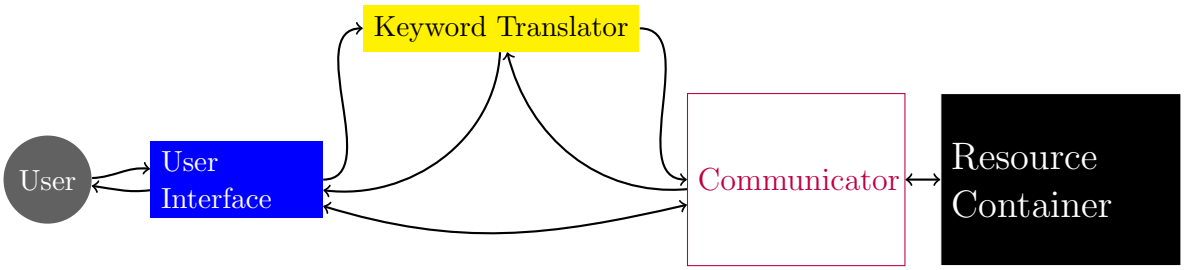


Figure 3.1: Block diagram of the system. The system consists of the user, the UI, the keyword translator, the communicator and the resource container. The user sends a string that is composed of keywords or is a phrase to the UI. The UI sends the string to the keyword translator. The keyword translator derives terms from the input string and sends them to the communicator where these terms are processed. The resource container responds to the communicator via provision of data which is parsed by the communicator. Eventually resources will be returned to the keyword translator. There the resources are integrated into the query if possible. This is done iteratively for each of the derived terms. Then after constructing possible queries, they will be returned to the user via the UI.

The final query tree should represent the input string in a "meaningful" way. If it is desired only to use keywords in comparison to natural language, even then some interpretation in reference to the keywords shall be done. Florian Bäurle, the advisor of this thesis, provided the functions for the communicator part. He also implemented the whole UI[4].

3.2 General problems with language

A problem-oriented section will start now because these were the problems which we faced while developing the translator. Some problems may be related to each other but within our frame of reference we will state them independently. Firstly, general problems in context of language and communication are mentioned, then resource oriented problems are presented.

We have to think about the general ambiguity of language within phrases or keywords, even if it describes what the author of this phrase thought of. For example a possible input string

"penalty for fighting dogs"

in which the *"fighting dogs"* may refer to: fighting with dogs, dogs that fight other dogs or a wild rock band. In fact it could refer to any thing,thought or idea that is named after it.

One additional problem which is inferred by the example is the missing predicate. In most cases a correct sentence consists of a subject, a predicate and optionally an object. The use of single keywords is a sort of casual language. If the user from the example wanted to know which penalty one should expect for keeping or training hunting dogs, then the predicate is missing. Syntactic and semantic ambiguity may be nice in arts but is a major problem for semantic search.

From the view of a communication scientist the user is the encoder of his message and the keyword translator is the decoder of the message[9]. They say that external noise is added to the message, which can be ignored in this case due to digital technology. But in Rusch's book[9] it is mentioned that individual actions and the intelligence of encoder and decoder will make things more complicated. The problem of interpretation is of subjective nature.

3.3 Resource oriented problems

The main resource we work with is the bag of search elements given by BROCCOLI. Starting from the input string, then communicating with the backend(feedback by the resource container) to retrieve the search elements will display the following problems:

3.3.1 Ambiguous mapping

The input string entered by the user could be the name of an instance, e.g. the title of a book. At the same time it could be mapped to a class-relation-instance query. For the string "arab states of the persian gulf" the backend finds an instance. One may try to decompose the query into some sort of location class and try to find instances for the unmapped words. If no proper instances can be found, then we can use a word node. Here the two possible queries are shown:

- Arab states of the persian gulf
- Country — occurs-with — arab* persian gulf

One approach to solve these problems is to process various possible interpretations which leads to increased runtime.
Suppose the user queries "films directed by Francis Ford". The resource container of the BROCCOLI system does support the following queries:

- Film — directed-by — Francis Ford
- Film — directed-by — Francis Ford Coppola

We have to wonder whether the well known Francis Ford Coppola is the right instance or the actor Francis Ford who was born in the century before Coppola has been conceived. Now the query has to be ranked. Different metrics can be used. One would be the score provided by BROCCOLI . It uses the term frequency[2]. This would represent the general popularity of the strings concerning the documents. Another metric would be the conformity of the substrings, which is good assuming the user utilizes the exact terms of the instances in the ontology.

3.3.2 Missing relation

This problem may be on hand of the input string or it may be on hand of the ontology meaning that no direct relation between the root node and the instance can be found. The missing relation problem due to the input string exist in the "film by steven spielberg" example. The predicate is missing. A naive approach for this problem would be to try the available relations and check whether an instance called "Steven Spielberg" is found, if yes then add this relation to the query tree. Although the naive approach is guessing it does work in many cases, because the proposals are sorted by the Broccoli score, so the more popular relations are the relations which are tested first. The missing relation on hand of the ontology, meaning that not a single relation pointing from the root node to the instance is also a hard challenge. Except for the usage of brute force either one will create a search element of the type word and save the effort of using any further backend operations or one will try to apply an instance-oriented approach. This approach aims to find the instance independently from the relation and tries to find an indirect relation. E.g., consider the input strings "female computer scientist" and "apollo astronauts". The query trees would appear as below:

- Astronaut – space mission – Event – space program – Apollo Program (Q1-3.3.2)
- Computer Scientist — gender — female

Regard the query tree Q1-3.3.2 . The constructed query consists of only four search elements. These are "Astronaut", "space mission", "space program" and the "apollo program". The "Event" element has not to be added. The logic of the backend takes care of it. While constructing the query an abstract class element has been added to the query tree. It is labeled with "Entity". It represents a set of possible classes. If via further containment, which means adding relations and so on to the query tree, a single class can be found that the queried instances belong to, then it will substitute the "Entity" class element.
Solving the issue of missing relation is a way to transfer the concept of fuzzy search for words to semantic search because the predicate has to be inferred. If a lot of relations have to be inferred it may lead to an increase of runtime.

3.3.3 Dealing with stop words

We process keywords and partially natural language. In order to do this we remove the stop words and only regard the leftover words. In this way less backend communication is done. E.g. finding the instance called "system of a down"(Band) would be then about mapping "system down" to the wanted instance. If the instance cannot be queried with "system down" we reverse the order of the terms which would be "down system".

One problem is about deciding which occurring words are stop words. If they do not provide information then we can ignore them. But if the input string of the user utilizes the syntax of natural language then it might be helpful to use these words. They can imply a delimiter for predicates or objects in a sentence. On the other

hand, if they are contained in the string of an instance, then stop words should not be treated as a delimiter. The instances “System of a down” and “The Who” would be hard to find.

3.3.4 Different vocabulary

This problem mainly occurs when mapping words to classes or relations. The user does not have to know in advance how a certain relation or class is named. E.g. if the user wants to find the class “soccer players” , but the class in the ontology that fits to “soccer player” is called “football player”, then different vocabulary has been utilized. Using a thesaurus could be useful. Thus we make use of the (Princeton) WordNet. We only use it for class names and relation names because for instance names synonyms are hardly used. The names of instances are more robust due to the fact that proper nouns are often used for them.

Finding a synonym for the word which should be mapped to a class is mostly easy. The WordNet API allows an easy retrieval of synonyms for words.Then these synonyms will be used to collect possible classes.

A more interesting approach is used when it is about finding the right vocabulary for the relations. Suppose the input string passed by the user is “astronaut born 1925-1930”. And the class named “astronaut” has already been detected. Then the leftover substrings are “born” and “1925-1930”. What we simply do is counting. We check the substring “born” in WordNet and collect all the words of the first two synonym sets and their description. Words like “give”, “birth” and “bear” are listed. Then the top-k relations, where k is an arbitrary number, of the “astronaut” class are obtaining a point for each word of the former collected words they contain. A threshold is set relative to the highest number of collected points. Then only the relations above this threshold are stored and assigned to the users substring. In our example “born” is mapped to “date of birth” and “place of birth”. Then the query construction continues. The leftover substring is “1925-1930”. Now the two relations are queried in combination with “1925-1930”. Now the type of instance that is expected for the “date of birth” relation and the “place of birth” relation is checked. In this way the “date of birth” relation will be confirmed and the mapping is done successfully. Although this easy approach is simply counting, it works quite well to some extend because WordNet already gathered semantically related information and because of the previously mentioned confirmation of the mapping. This will be explained in the following chapter.

3.4 Algorithm

This section will reveal the actual keyword translation algorithm. The root node of a query tree either has to be an instance or a class. This implies that there are only two kinds of queries. Therefore the algorithm is separated into two parts. The two parts are named “instance as root”(IAR) query and the “class as root”(CAR) query.

At the beginning we prepare the user input string. It will be formatted, filtered, trimmed and split. Another static adjustment of the input string is the replacement

of certain words at a certain position. This enables us to process queries that have an interrogative structure. The question "Where is Claude Debussy born?" is an example for that. The word "where" as the first word in the input string will be exchanged for the word "location" which is the name of a frequently used class. Also the word "who" as the first term in the input string will be replaced by the word "person" which is the name of a frequently used class. These adjustments are hard coded just to be able to process interrogative clauses. The two parts of the algorithm are working with the result of the string preprocessing.

3.4.1 Instance as root

The task of this part of the algorithm is to return a query with an instance as the root node.
We simply try to map the words to instances and compare their substrings. If the user queries "angela merkel spouse", an instance "angela merkel" will be found. But it is possible to find several instances. For each of these instances we proceed as follows. We try to map the words that have not been used to find the instance onto a relation. If in this way all leftover words have been consumed the IAR query has been constructed successfully. The query trees for the input strings "angela merkel spouse" and "rhine length" are:

- Angela merkel — spouse (Q1-3.4.1)
- Rhine — length (Q2-3.4.1)

The query tree (Q1-3.4.1) points to other instances that belong to the class "Person". Whereas (Q2-3.4.1) points to an instance of the type value. An instance of the type value can be compared by the backend.
If such a query could not be constructed, then the (CAR) query is executed. The class-as-root part of the algorithm will be presented in the following.

3.4.2 Class as root

This kind of query consists of a class as the root node of the query tree. The query tree has instances, that are are connected to a relation, as its leaves. We will show the functionality of the algorithm by the example of the user input:

"female computer scientist born in >=1970" (E1-3.4.2)

The tasks we have to cope with are done in succession. They are described in the following:

Finding of possible classes:

We start with a natural language approach. Usually the structure of a sentence is subject-predicate-object. So we assume that the class can be found within the first few words of the input string. The part-of-speech tagger helps us to find the first occurring word that is neither a noun nor an adjective. We use the found string as a separator to regard all the leading words as legal terms that can be mapped onto classes. With the help of WordNet we use the base

form of a word.

Now ranking and culling of the collected classes is done. They get a score of +1 per leading word that matches to the class name and score of -1 per class term that is not matching to the leading words. If several classes are found then the top-two, evaluated by their scores, are taken. Only two are taken in order to reduce runtime.

If no classes have been found we assume that the wrong expression for the class has been used. In this case we apply WordNet to obtain synonyms and collect classes for them. They will be culled and ranked in respect to the synonyms and the leading words. The outcome of this step is a list of classes culled into the top-two classes. Such a class will become the root node of the query tree. These query trees potentially grow in the two following steps.

How the terms of (E1-3.4.2) have been tagged can be seen here:

(E1-3.4.2)	*female*	*computer*	*scientist*	*born*	*>=1970*
tags(E1-3.4.2)	adjective	noun	noun	verb	number

The word "born" is the separator due to the fact that it is neither an adjective nor a noun. Then the leading string is "female computer scientist", for which the class "computer scientist" will be found. The stop word "in" has been filtered.

Consumption of mapped relation-instance combinations(semantic mapping):

This is a statistical approach. For each class C_i in the set of classes(the index i stands for the index in the set of classes, in order to differentiate the single elements) that are found we proceed as followed. Let W_f(the index "f" stands for "following"), be the set of words which occur after the words that have been used for the mapping of C_i. We aim to map words to relations, on the condition that words, that are adjacent to relation-mapped word, are mapped onto an instance. The words-to-relation mapping will not only be morphological, it will also be semantical. First all words except stop words from the top-two synonym sets of a word $W_{f,i}$ from W_f and their descriptions are collected. Let such a collection be W_s, where "s" stands for semantic or synonym. Add the word $W_{f,i}$ to its W_s.

Then for every word of W_f we rate all possible relations of the root class in respect to W_s. For each term in top-m relation of C_i contained in W_s the relations rate will be increased by one. The number m is the maximum number of relations which are checked. In other words, each relation of all relation from C_i will have an integer rate that is greater than or equal to zero that is assigned to a word of W_f.

If adjacent words of W_f have the same relations with scores greater than

zero, then they will be aggregated. In this way e.g. the two adjacent words of a user input string "date birth" will be aggregated into one relation. This is done in order to avoid the multiple use of the same relation.

The metric for the culling of the relations is presented in the following. Let $Score_{max}$ be the maximum rate of all words in W_f. Collect the relations for a word in W_f with a score greater than or equal to 75% of the highest score of all relations for a single word in W_f and greater than $\frac{Score_{max}}{20}$. Let these collected relevant relations be R_R.
To give an intuition of the culling parameters, the score of a relation in respect to the word which is mapped is only relevant if its value is in the range of the highest score for the relation in respect to that word. And it is only relevant if its value has reached a critical value compared to $Score_{max}$. If this kind of culling is not done, then a lot more relations have to be processed, which would lead to a higher runtime and may decrease the quality due to misinterpretation. Moreover, then we could have used brute[1] force in the first place. These parameters were found empirically.

For each word W_f from which a relation in R_R came let the relevant adjacent words be $W_{R,A}$. For each of those relevant relations relation $R_{R,i}$ in R_R try to map words of $W_{R,A}$ onto instances given a query that has a C_i as the root node and the relation $R_{R,i}$ connected to it.

Let us apply the algorithm for (E1). Then the set of classes would be $\{(C_1)\}$. Therefore we have only one class C_1, where C_1 is: Computer Scientist. W_f is {"born",">=1970"}. The W_s for "born" is {"born", "bear", "give", "birth", "deliver" , "birth"}. WordNet provided these words. The W_s for ">=1970" can be neglected due to the fact that WordNet did not find words for it. Here we denote the list of relations that can be found for our class C_1 and their scores for each of the words in W_f:

- Profession ◁▷

$\mathbf{W}_f$	"*born*"	">= 1970"
scores	0	0

- Gender ◁▷

$\mathbf{W}_f$	"*born*"	">= 1970"
scores	0	0

- Country of Nationality ◁▷

$\mathbf{W}_f$	"*born*"	">= 1970"
scores	0	0

- Date of birth ◁▷

$\mathbf{W}_f$	"*born*"	">= 1970"
scores	2	0

- Place of birth ◁▷

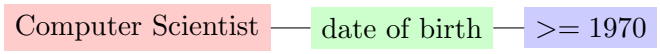
$\mathbf{W}_f$	"*born*"	">= 1970"
scores	2	0

- ....

There are more relations, but we neglect them now, because they will not change the result. The culling occurs via comparison of the scores. $Score_{max}$

[1] Using brute force in this context means the computation of all possible outputs, given the inputs. Depending on the problem it will lead to a solution, but will increase runtime.

equals two. Hence relations like Profession will be removed. Now the set of relevant relations R_R that were mapped by the word "*born*" is { Place of birth , Date of birth }. The set of adjacent strings of "born" is {">=1970"}. The one string will be mapped onto an instance of the type value in respect to the relevant relation and its class. Thus it fits to the Date of birth relation. All the strings of W_f have been consumed. The intermediate query would be:



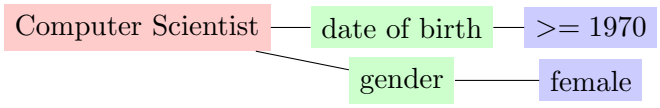
The semantic mapping of the relations via WordNet vocabularies is only working in combination with the mapping of the words that are adjacent to the words that were used to find the relation onto instances. This can be seen as a confirmation of a successful mapping.

Inference of relations via instance mapping:

The idea of overcoming this task is to map the leading words and the W_s, that have not been consumed for a query, to independent instances and try to find a connection between these instances and the root class. Finding the independent instances will be called "instance recognition" and will be explained later.

The search for a connection between the instances and the root class can be time consuming, due to the fact that we do not know how many relations are needed between the instance and the class in the query tree. This is why we set a limit for the number of relations allowed between instance and root class. We assume that the more relations used, the less probable it is that they semantically belong together. The limit we chose is two, because two relations have a reasonable distance and can be processed in reasonable time. Another problem concerning the runtime is that we do not know how many relations take the root class as the starting point. So we also limit the number of relations that will be taken into account.

For our example we would map the string "female" onto the instance that is called "female". Now we iterate over the top-k relations that are proposed by the backend in respect to a query with the instance female as the root node. If there is a relation that is connecting female and Computer scientist, then we have inferred the relation. All words have been consumed and the final query would look like this:



A connection between the instance and the root class is wanted. If not a single connecting relation can be found, then we would seek a pair of relations that connects female and Computer scientist. On top of each of those top-k relations we would seek the top-j relations to find the connecting pair. An example query for this is (Q1-3.3.2) given the input string "apollo astronauts".

There the pair of relations that connect the instance named "apollo program" and the class node named "Astronauts" are shown.

Appending unmapped words:

If after the previous processing there are strings in W_s or W_f of a class C_i found, then use the special relation called *occurs-with* which uses the word search elements for text search. Currently these word search elements are added to the root class node. One can improve this by analyzing the linguistic structure. Thus the word search element of a word that refers to a search element that has been found already and integrated into the query tree can be appended to it via marking of the search element as the active node.

Evaluate multiple queries:

The result of the preceding steps is a list of query trees. The quality of such a query tree is evaluated by the number of non-word search element nodes in the query tree. We assume that if it is possible to map independent terms of the input string to resources which can be combined in a low-depth query tree. Ideally an admissible query is found after these steps. The most vital step is the first one because finding the intended class will point into the right direction regarding the accessibility of the resources.

3.4.3 Instance recognition

As mentioned in the inference step of the class-as-root part, instances are found independently from a root node. The idea is to find the instances which have a name which uses more terms before finding the instances that use less terms for their name. Imagine someone wants to query "star wars IV". If one tries to find instances that have only two strings in their name before finding instances with more than two strings in their name, then the instance called "star wars" is found before the instance that fits better to the input string "star wars IV". We assume that the given class has been found and that we will process a list of strings which has no stop words like "a", "the" and "of".

Four stages are used. Suppose k is any number that is not greater than the number of all strings that is in the list that is being processed. The first stage uses all occurring units of k adjacent strings respecting their order in the list. We use these units only to point the ranking of the proposals in the right direction. What we do next is to obtain an arbitrary number j of proposals for a unit. Now we only regard the instances that have at least $k + 1$ strings in their name. Then each of these instances will be rated with a score. That score will be incremented by one if a string of the instance name is contained in the list of strings that we process. After that we remember whether the string has already been used, so that the score

is only incremented once for every occurrence in both the list and the strings of the instance. Then we only regard the instances with a score greater than k. Out of these instances we will regard the instances with the highest occurring score. Out of these highest scoring instances we will only regard the instances with the least number of strings in their name. Then we pick the first element of this selection, that represents the instance with the highest priority by the backend. In our example the list of words that we process is "star wars IV", k equals two and j equals 13. The first unit is "star wars". The 13 proposals for this unit that are sent by the backend are:

- Star Wars
- Star Wars The Old Republic
- Star Wars Episode I The Phantom Menace
- Star Wars Episode IV A New Hope
- Star Wars Episode III Revenge of the Sith
- Star Wars Episode VI Return of the Jedi
- Star Wars Knights of the Old Republic
- Star Wars Galaxies
- Star Wars Episode II Attack of the Clones
- Star Wars The Clone Wars
- Kinect Star Wars
- Star Wars The Force Unleashed
- Star Wars Episode V The Empire Strikes Back

The first proposal can be neglected, because it has less than $k + 1$ strings. Then after rating the other instances we have the instance with the highest score and that has a score greater than two, which is called "Star Wars Episode IV A new hope". Now all the strings that have been used for the ranking are removed from the list that we process. If no match would have been found for the unit "star wars" then the next unit would have been "wars IV".

Now we assume that all the instances with names consisting of at least $k+1$ strings are found. The second stage is about using k adjacent strings, in respect to their order in the list, to find instances with exactly k strings in their name.
The proposals of the backend are sensitive to the order of the strings that are typed in. If k equals two, which is used in our program, then the third stage uses units of two adjacent words but in reversed order. Then we will take the top proposal that has less than five strings in its name as the mapping. Although this stage allows the usage of instances with more than k(equals two in our work) it is there to map

two words onto instances with two words in their name. The reason for that is that we filtered the stop words before processing the list. Consider a user wants to find instances named “System of a Down”, “Simon and Garfunkel” or “Church of Ireland”. The filtering of stop words will return the lists “System”, “Down”, “Simon”, “Garfunkel” and “Church”, “Ireland”. Then the reversed order of these strings will help us find these instances.

The last stage uses units of single strings. They are mapped to instances with one or two strings in their name.

We could also try to use a brute-force approach which would be the usage of units that represent all combinations of the strings. But for performance reasons we use these units of k strings.

The algorithm of the keyword translation has been presented by an example. The next chapter is the analysis of the work.

4. Analysis

4.1 Theoretical analysis

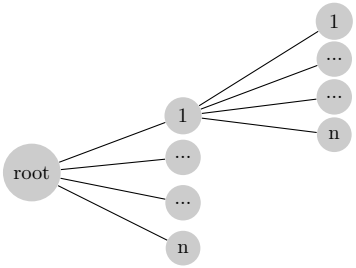
4.1.1 Analyzing instance and class recognition

Suppose that we have t terms within the input string. The number of operations needed for the class recognition is asymptotically the same as the number of operations needed for the instance recognition. This holds true because both steps iterate a constant number of times over all t terms which means that it is in $O(k \cdot t)$ where k is an arbitrary constant and stands for the number of substeps in the recognition algorithm. For example, one of those substeps in the instance recognition algorithm is the finding of instances which have a name consisting of more than two terms. The constant k can be neglected asymptotically indicating that the recognition steps are in $O(t)$.

4.1.2 Analyzing the inference of relations

For this analysis a single query construction process with given instances and root node has been assumed. Suppose a query is given which has a class as the root node. If the user input string does not contain information on the relation between the instance and the root class, then it may be that more than one relation is necessary in order to connect the instance and the root class. Whether such a connecting path exists depends on the completeness of the used ontology. But for now assume that the target instance and the root class are contained in the same connected component. Furthermore assume that each node of the tree except the leaves will branch out into n nodes.

Then the number of operations needed to find the connecting path is in $O(n^d)$, where d is the depth of the tree. This is why we limited d in our algorithm. The depth d is only two because we assume that with a higher d the semantic meaningfulness of the resulting query will decrease.



The running time of finding a connecting path highly depends on the taxonomy of the ontology. The reason for that will be presented in the following. Consider

the user enters an input string which contains the class and the instances the user thought of. If the ontology does not contain a directly connecting relation between those, then the distance of the connecting path increases. Hence the distance is the depth of the tree, the number of operations also increases.

This chapter stated the theoretical problems of performance. The next section will consist of an evaluation of the algorithm.

4.2 Empirical analysis

4.2.1 Problems

An element of scientific work is the evaluation. Evaluating the translation is not trivial. There is no "ground truth" for the queries and we cannot use the commonly used metrics: precision and recall. Hence, one has to face different problems when building a query tree:

1. Coverage of the data the user wants to find.
2. Consistent taxonomy.
3. Several ways to construct a query tree that returns hits that satisfy the user's request.

The first problem is about whether the classes and instances do exist in the ontology. Not all instances that should be returned by a query do exist. In one case the second problem is apparent when direct relations between the resources are missing. Then word search elements can help or sometimes more relations between the target resources are necessary. In another hand the second problem is apparent when instances do not belong to classes they should belong to. The first and second problem are dependent on the ontology. The third problem is about the fact that a query is adjustable even if a certain relation is missing. This adjustment is done via the word search elements. If they are used, then one has to think of which strings are used for the search element.

These problems of building a query tree which satisfies the user's query, lead to a highly dynamic behavior when a user utilizes the manual query construction.

4.2.2 Evaluation setup

This subsection is about how the translations are measured. A result is determined by comparing the manually constructed query and the translated one. In order to compare the quality of a result, manually constructed queries have been made to the best of our knowledge and belief.

The data set used for testing is comprised of 50 input strings from the SemSearch Benchmark[1]. Some of their queries contain names of instances spelled in a slightly different way than the names used for these instances in the ontology. These spelling

mistakes have been adjusted while testing. This adjustment occurred to SemSearch-21, for which the name "Lilli Allen" has been changed to "Lili Allen" and SemSearch-48, for which the name "Jackie Weaver" has been changed to "Jacki Weaver". The table of the manually constructed queries and the translated queries can be found in Appendix A . The first line of an entry contains the query identifier and the input string. The second line shows the translated query and the third line shows the manually constructed query. Remarks are found in the fourth line. Whether the query input has been adjusted is denoted in the remark line of the table. The benchmark inputs are classified into three types:

Optimal These results are in accordance with their manual and translated query.

Sufficient These results return similar hits for their manual and translated query.

Insufficient These results do not return similar hits for their manual and translated query.

Impossible No manual query could be constructed for this input.

The input queries that are of the type "impossible" will be neglected, because of limited resources. But even for query inputs that are classified as impossible, the translated query, if it exists, can be found in the appendix.

Let N be the total number of queries minus the number of impossible ones. Let N_O be the number of results that are classified as optimal. Let N_S be the number of results that are classified as sufficient. Let N_I be the number of results that are classified as insufficient. Then we use the following performance metrics:

$$O\text{-to-}N\ ratio = \frac{N_O}{N}$$

$$OS\text{-to-}N\ ratio = \frac{N_O + N_S}{N}$$

$$I\text{-to-}N\ ratio = \frac{N_I}{N}$$

The finding of the right root node can be used as the metric for a sufficient result, because the hits returned in the end are utterly dependent to the root node. But in this case we use the finding of a query tree that returns similar or nearly the same results as the manual constructed query tree. The next subsection will present the performance of the translation.

4.2.3 Quality & runtime performance

First, the quality performance is presented. The manual query construction of the SemSearch query inputs has not been possible in eight cases. After the assignment of results to the different types the results are aggregated into the following parameters:

N_O	N_S	N_I	N
14	7	21	42

The values of the formerly defined metrics are:

O-to- N ratio	*OS-to-N ratio*	*I-to-N ratio*
0.3333	0.5	0.5

The *O-to- N ratio* states that 33.33% of the translations are optimal. The *OS-to-N ratio* states that 50% of the translations are admissible. In this context "admissible" means that the manual and the translated query return nearly the hits. The *I-to-N ratio* states that 50% of the queries either could not be translated properly or could not be translated at all. The reasons for that will be discussed in the next subsection 4.2.4 .

Another target that was measured is the runtime of the translations. The program was executed on a local machine. Peaks of several hundred milliseconds for one backend operations occurred. Another cause of increased runtime may be the use of this specific part-of-speech tagger. The main specifications of the used machine are: 4GB RAM, Intel(R) Core(TM) i3-2310M CPU @ 2.10GHz. The download rate is 149.148 $kbits/s$ and the upload rate is 15.574 $kbits/s$. The provider is MyWire.

All translations have been executed three times. Then the arithmetic mean of the three values is the value that has been used as the runtime for that translation. The arithmetic mean of the runtime of all translations that have been measured is 4.18 seconds. This runtime is higher than expected. There are several reasons for that. The runtime of a translation that led to a IAR query tree is always lower than the runtime for the translations that led to a CAR query. The reason for this is because the IAR part of the algorithm is executed before the CAR part. Another reason for increased runtime occurs when using synonym mapping in order to find the root class. Then for each of the words that are used as semantically related words of the used terms that are contained in the input string a backend communication operation will be done. Another reason for this occurrence is due to the necessity of a large number of operations in order to compute a pair of relations which connects the root node and an instance.

4.2.4 Discussion of the quality performance

There are different reasons for the translation failure of certain queries. Some problems may overlap with each other, but still they will be presented as they are, because they can be regarded as adequate samples of their kind of problem. Several reasons and a solution for their problem will be presented.

The first reason is that no semantic mapping is done if the morphological string mapping of the leading words is successful. This explains why the translation for the input string of SemSearch-4("Axis powers ...") , SemSearch-9("Degrees of ..") and SemSearch-17("Houses ...") has failed. There are classes that are named "Axis" and "Degree". Hence they are chosen as the root node. More feedback from the backend or a stronger use of WordNet(semantic mapping) could be used to avoid this failure. Using more feedback from the backend means, in one case, that we detect that the query returns no instances and so on. Then the root node can be

discarded and a new approach can be used. One new approach would be the usage of a highly abstract class as the class named "Topic". Then appending the exact input string as a word search element via an "occurs-with" relation may return instances. Then an arbitrary number of the top instances can be analyzed by its class. If most of these instances are from one class it may be the wanted class. For SemSearch-4("Axis powers ...") this approach would find the class called "Country" as the most likely class. This may increase the runtime. This exact same approach would work with the input string of SemSearch-50("Wonders of the ancient World"). Then the most likely class would be the class called "Location". The input strings of SemSearch-4, SemSearch-9, SemSearch-50, SemSearch-22("major league...") and SemSearch-41("four of the companions of the prophet") have in common that they consist of specific expressions that are often used in its arrangement and that do not necessarily denote a class node that fits. Hence the usage of the word search element will solve the problem.

The second reason is that the semantic mapping for the root class is only performed via the usage of hypernyms and synonyms of individual words. And each word is processed individually. Hence they do not affect each other. Consider the input SemSearch-19("Kenya's Captain...") where the semantic mapping of "Kenya" is responsible for finding a class that contains "republic". Consider the input SemSearch-22("Major league...") where this phenomenon also occurs. Only the semantic mapping of "league" that is responsible for finding a class that contains "conference". Aggregated semantic mapping may be necessary. Rating class proposals for the occurrence of synonyms, hypernyms and other related words of the words that are mapped and then aggregating the rating for same topics would be the approach. This might solve the problem of mapping the input SemSearch-24("tv series ... ") onto the class named "TV Program".

The third reason is that the morphological mapping is not efficient enough or that semantic information about the other instances that are mentioned is not used. Consider the input of SemSearch-32("Presidents depicted on mount rushmore..."). Either the president may be mapped morphologically onto the class named "US President" or the instance called "Mount Rushmore" is used for finding the class named "US President". Another way would be the usage of WordNet that has information about Mount Rushmore and who is carved in it. There are multiple ways to solve this.

The fourth reason is that the analytic approach for finding the subject is prone to errors. Consider the input of SemSearch-41("four of the companions of the prophet"). Here the approach cuts the words after the first "of" away and tries to find a class for the leading words. Either a more sophisticated use of natural language processing may help here or an independent search for resources in respect to the order of the words may solve the problem.

The fifth reason is that the consumption of the relations does not use semantic mapping processing for the instance-as-root queries. Consider the inputs of SemSearch-18("John Lennon, parents"), SemSearch-21("Lilli Allen parents") and SemSearch-23("Manfred von Richthofen parents"). In these three cases the "parents" should be mapped onto the relation named "Children(reversed)". The "(reversed)" in a relation implies a logical negation of it. In case a reversed relation is used the antonym of a word should be used for the mapping.
Now consider the input of SemSearch-48("who has jackie weaver been married to"). The relation for "married to" is called "Spouse(or domestic partner)". To solve this related terms for the terms of each relation has to be looked up and compared to the related terms of the words that are being mapped.

The sixth reason is that conjunction words are not processed in a logical way. Consider the input of SemSearch-15("Henry II's brothers and sisters") or SemSearch-33("Provinces and territories ..."). Here we seek terms that describe several individual words that occur in the conjunction with a single idea or concept. The intersection of more abstract hypernyms of the individual words that are the targets of the conjunction will solve the problem. This step should be at the beginning of the class-as-root part of the algorithm.

The seventh reason is that no acronym recognition is done for the common acronyms. This problem occurs when translating the input of SemSearch-31("permanent members of the UN Security Council") or SemSearch-7("Branches of the US military"). Looking up the acronym in WordNet would be a possible solution. If there are several possibilities for mapping the acronyms then obtaining feedback from the ontology with word search elements can be used.

The eighth problem occurs if semantic mapping is needed in order to find the root for a class-as-root query. Consider the inputs of SemSearch-7("Branches of the..."), SemSearch-12("First targets ...") and SemSearch16("Hijackers ..."). Right now direct hypernyms are used for the semantic mapping. The usage of more abstract hypernyms will solve the problem.

Another problem that can be solved via data feedback refers to the word search elements way of working . A lot of natural language processing is done in order to consider the context of words in a phrase when using word search elements. This implies that the precise use of words in a word search element may have a drastic effect on the hits that will be displayed by the backend. Consider the input of SemSearch-12("First targets of the atomic bomb"). Suppose we already mapped "target" to the class named "location". The usage of the keywords "first target atomic bomb" as a word search element will return different hits than the usage of "first target nuclear bombing" as word search element. In this case the latter used keywords will lead to the receipt of the correct hits. Suppose that the user knows for certain which hits are wrong. If the user's feedback is used then several translations could be done. Synonyms and word proposals of the backend can be used. Then the "atomic" is replaced by "nuclear", because it is a synonym, and the backend proposal suggests using "bombing" instead of bomb.

4.2.5 Comparison to other approaches

Another approach for solving the problem of keyword translation is to collect first ontology resources independent from each other. The advantage would then be the quality performance in case a good rating formula is used for the ramification of the resources. The disadvantage would be increased running time. This approach is less prone to syntax errors of the input string.

The other extreme is a rather naive approach. It is about integrating any ontology resource that is found into the main query as soon as it is found. Less rating has to be done and less possible query trees are qualified. The advantage of this approach is less running time. The disadvantage is a lower quality performance. This approach may be more dependent on the syntax of the input string. The formerly used keyword translator used this approach.

Our approach is combining both approaches to some extent. For the queries that have a class as the root our approach is to find the words for the class mapping in the first words which are neither nouns nor adjectives. In this way not all words are used for the mapping and we assume a certain syntax of the input string. But still for the inference step it is trying to find the instances independently. And the gain of the semantic mapping of the words that are mapped onto relations is that less relations have to be inferred due to the fact that the information about which relation should be picked is in the input string.

5. Outcome

5.1 Future Work

This section is about possible improvements. The various approaches will be listed and ordered by importance. The estimated amount of time needed to realize the approach, if only one person is conducting the taks, is found at the end of a paragraph.

Provide several queries

In fact, the algorithm returns a query tree for each of the classes that the mapping found. Consider the user enters an input string that contains "football player" in the beginning . Then it will compute queries for root classes that are named "american football player" and "football player". Providing both queries is a convenient way to let the user choose which one she/he thought of. Furthermore the quality of semantic mapping onto the relations and the inferred relations of two queries can be monitored. The realization of this will only take a small amount of time, because the feature only has to be integrated into the user interface. An estimation of the required amount of time is less than a few hours.

Adjust translation pipeline to improved quality

If the pipeline starts with an accurate and efficient entity recognition step, then the quality of the queries could be improved. Instances can have atypical names like "the Who"(band) or"System of a Down"(band) or long names like "Star Wars Episode IV A New Hope". Applying natural language processing on user inputs containing mentioned names of instances is prone to errors. Filtering common stop words may totally remove the instance "the who". Working with elementary stages may increase the running time, which is why other improvement methods should be taken into account. Some helpful improvements for this approach are also going to be listed below. Combining this approach with the semantic mapping of the words onto relations via WordNet, which is done in our algorithm, will prove to be very powerful. This idea of adjusting the pipeline is based on the pipeline of the SPARK system. Conducting this *change of pipeline* would require one to rewrite a large amount of code and could potentially take several weeks

Instance recognition via offline database

E.g., WordNet stores a lot of names for instances. Recognizing instances in the input string with offline databases would mean that less backend communication has to be done. This would lead to the reduction of running time. Given that the

translation pipeline has a step where all operations that depend on the instance are processed, realizing this will take a few hours. Otherwise it will be hard to estimate the effort.

Apply more natural language processing

Using more natural language processing(NLP) could be helpful. It might be helpful for resolving the references in a phrase of words like "it"(pronouns) and "and"(conjunction). A lot of those stop words might help. Prepositions can denote relations.

A noun in combination with the by-agent may refer to a missing relation. An example for that is "composition by Bob Dylan". Here the predicate is missing, "made" would roughly fit. Thus analyzing relations with WordNet may work. WordNet would find for the relation called "composer" a derivational form of type verb, which is "compose" and its hypernym is "make". This is the base form of the missing predicate "made".

The article "the" can imply an instance which, for example, exists in instances that have a name like "The Who"(Band)

The occurrence of numbers in the input string may either imply the part of an instance name or an instance of the type value.

Whether a class or an instance is the root node of a query has to be worked out. NLP in combination with instance recognition can help with that. A noun in combination with a by-agent in the input string may suggest that the root node is a class. Otherwise with a pipeline that uses a certain series of stages the query recognition may become redundant. Working on that improvement may take a few days.

How much time the realization of such a natural language processing engine takes depends on its capabilities. For building a powerful NLP engine from scratch at least a few weeks may be necessary.

Semantically mapped relation instance-as-root query

Our algorithm did semantic mapping of words onto relations only in the class-as-root queries. This can also be done for instance-as-root queries. Doing this should be precise, because for this type of query tree there will be no instance on top of the relation that is confirming the mapping. If this improvement is implemented, then a formula has to be developed to compare the class-as-root query and the instance-as-root query. Implementing this may take a few days.

Using ontology feedback

The idea of this improvement is to work more closely with the proposals of BROCCOLI. One way to do this is the use of the word search element. Suppose we have the user input string "composition by System of a Down" and we found that "composition" should be a class, then building a query in the document class with the rest of the words will return an instance named "System of a Down". This also works with

the input string “books of Jewish Canon”. If it is entered as a word search element then the top instance is “Hebrew Bible” which corresponds to the input string, although morphologically they look totally different. But using WordNet may find hypernyms for “bible” in order to find out that both input strings should return the same instance.

Without doing that the algorithm would return a query with the class named “book” as the root node. This is actually not wrong, but a single instance may also be expected.

Implementing, testing and tuning this may take a few days.

5.2 Conclusion

The reason why a keyword translator should be used, the benefit of it, has been exposed. Several non-trivial problems concerning the keyword translation have been revealed. An algorithm which combines several approaches has been introduced as the main core of the work. The semantic mapping has been introduced which uses the occurrence of words that are semantically related to each other in order to find a proper relation instance pair. It uses one expensive operation which is the retrieval of the relations by the backend. The text processing is done offline and is rather cheap compared to the receipt of relations by the backend. The decomposition of a sentence in a more sophisticated way will also improve the quality of the translation.

An evaluation has been done. This brought up the problem of how to measure the translations. After that several kind of issues that are supposed to be addressed have been revealed. Out of that numerous ideas and approaches to solve the occurring problems have been mentioned. I think that improvement of the class recognition and a stronger utilization of resource feedback via search elements on the part of the backend will cover a lot more cases. The improvement of the recognition class would in many cases only be the use of more abstract hypernyms. They should have a certain degree of concreteness. But research could be done about this. The application of the resource feedback in general may also be very helpful. In fact it may be of help in various creative ways.

The pros and cons of using the lexical database WordNet have been implied. But the utilization of a thesaurus-like database is essential for semantic keyword translation.

A. Appendix: SemSearch Queries

SemSearch-1	Apollo astronauts who walked on the Moon
Translated query	Astronaut — Space Mission — Entity — Space Mission — Apollo Program occurs-with — walk* moon
Manual query	Astronaut — Space Mission — Entity — Space Mission — Apollo Program occurs-with — walk* moon
Remark	optimal translation
SemSearch-2	Arab states of the persian gulf
Translated query	Administrative Division — occurs-with — persian gulf
Manual query	Administrative Division — occurs-with — arab state persian gulf
Remark	sufficient translation
SemSearch-3	Astronauts who landed on the moon
Translated query	Astronaut — occurs-with — land* moon
Manual query	Astronaut — occurs-with — land* moon
Remark	optimal translation
SemSearch-4	Axis powers of world war II
Translated query	Axis — occurs-with — world war
Manual query	Country — occurs-with — axis power world war II
Remark	insufficient translation
SemSearch-5	Books of the jewish canon
Translated query	Book — occurs-with — jewish canon
Manual query	Book — occurs-with — jewish canon
Remark	optimal translation
SemSearch-6	Boroughs of new york city
Translated query	Administrative Division — contained by — New York City
Manual query	Administrative Division — contained by — New York City
Remark	optimal translation
SemSearch-7	Branches of the US military
Translated query	Administrative Division — occurs-with — military

Manual query	Organisation — occurs-with — united states military
Remark	insufficient translation
SemSearch-8	Continents in the world
Translated query	Continent — occurs-with — world
Manual query	Continent
Remark	sufficient translation
SemSearch-9	Degrees of eastern orthodox monasticism
Translated query	Degree — occurs-with — eastern orthodox monasticism
Manual query	-
Remark	impossible relation
SemSearch-10	Did nicole kidman have any siblings
Translated query	Nicole Kidman – Sibling
Manual query	Nicole Kidman – Sibling
Remark	optimal translation
SemSearch-11	dioceses of the church of ireland
Translated query	Religious Jurisdiction - Organisation — Church of Ireland
Manual query	Religious Jurisdiction - Organisation — Church of Ireland
Remark	optimal translation
SemSearch-12	First targets of the atomic bomb
Translated query	-
Manual query	Location – occurs-with — first target nuclear bombing
Remark	insufficient translation
SemSearch-13	five great epics of Tamil literature
Translated query	-
Manual query	-
Remark	impossible relation
SemSearch-14	gods who dwelt on mount olympus
Translated query	Deity — occurs-with — olympus
Manual query	Deity — occurs-with — olympus
Remark	optimal translation
SemSearch-15	Henry II's brothers and sisters
Translated query	-
Manual query	Henry II of England — sibling
Remark	insufficient translation
SemSearch-16	Hijackers in the september 11 attacks

Translated query	Agent — occurs-with — september 11 attack
Manual query	Person — occurs-with — hijacker september 11
Remark	insufficient translation
SemSearch-17	Houses of the russian parliament
Translated query	House — occurs-with — russian parliament
Manual query	Governmental Body — Jurisdiction — Russia
Remark	insufficient translation
SemSearch-18	John Lennon, parents
Translated query	-
Manual query	John Lennon — Children(reversed)
Remark	insufficient translation
SemSearch-19	Kenya's captain in cricket
Translated query	Russian republic — occurs-with — cricket captain
Manual query	Sport team captain — occurs-with — cricket kenya
Remark	insufficient translation
SemSearch-20	Khublai Khan siblings
Translated query	Kublai Khan — Sibling
Manual query	Kublai Khan — Sibling
Remark	optimal translation
SemSearch-21	Lilli Allen parents
Translated query	-
Manual query	Lili Allen — Children(reversed)
Remark	insufficient translation , the subterm "Lilli" has been adjusted to "Lili" for this query.
SemSearch-22	Major leagues in the united states
Translated query	Conference Event — occurs-with — united states
Manual query	Sports Association — occurs-with - united states major league
Remark	insufficient translation
SemSearch-23	Manfred von Richthofen parents
Translated query	-
Manual query	Manfred von Richthofen — Children(reversed)
Remark	insufficient translation
SemSearch-24	Matt Berry TV series
Translated query	Broadcast
Manual query	TV Program — Starring TV role (reversed) — Matt Berry
Remark	insufficient translation

SemSearch-25	Members of u2?
Translated query	U2 — Member of(reversed)
Manual query	U2 — Member of(reversed)
Remark	optimal translation
SemSearch-26	Movies starring Erykah Badu
Translated query	Film — Film performance(reversed) — Erykah Badu
Manual query	Film — Film performance(reversed) — Erykah Badu
Remark	optimal translation
SemSearch-27	Movies starring Joe Frazier
Translated query	Film — Film performance(reversed) — Joe Frazier
Manual query	Film — Film performance(reversed) — Joe Frazier
Remark	optimal translation
SemSearch-28	Movies starring Rafael Rosell
Translated query	Film — Film performance(reversed) — Rafael Rosell
Manual query	Film — Film performance(reversed) — Rafael Rosell
Remark	optimal translation
SemSearch-29	Nations where portuguese is an official language
Translated query	Country — Official Language — Portuguese Language
Manual query	Country — Official Language — Portuguese Language
Remark	optimal translation
SemSearch-30	orders (or 'choirs') of angels
Translated query	-
Manual query	-
Remark	impossible relation
SemSearch-31	permanent members of the UN Security Council
Translated query	Military Unit — occurs-with — un security council
Manual query	-
Remark	impossible relation
SemSearch-32	Presidents depicted on mount rushmore who died of shooting
Translated query	Business Executive — occurs-with — rushmore shooting
Manual query	US President — Dedication — Mount Rushmore National Memorial Cause of death — Assasination
Remark	insufficient translation
SemSearch-33	provinces and territories of Canada

Translated query	Domain — occurs-with — territory canada*
Manual query	Canadian Territory
Remark	insufficient translation
SemSearch-34	ratt albums
Translated query	Book
Manual query	Musical Album — Artist — Ratt
Remark	insufficient translation
SemSearch-35	republics of the former Yugoslavia
Translated query	Russian republic — occurs-with — former Yugoslavia
Manual query	Administrative Division — occurs-with — former Yugoslavia
Remark	insufficient translation
SemSearch-36	revolutionaries of 1959 in Cuba
Translated query	Revolutionary — Spouse — Person — Country of nationality — Cuba occurs-with — 1959
Manual query	Revolutionary — occurs-with — Cuba 1959
Remark	sufficient translation
SemSearch-37	standard axioms of set theory
Translated query	-
Manual query	-
Remark	impossible relation
SemSearch-38	states that border oklahoma
Translated query	Administrative Division — occurs-with — border oklahoma
Manual query	Administrative Division — adjoins — Oklahoma
Remark	sufficient translation
SemSearch-39	ten ancient Greek city-kingdoms of Cyprus
Translated query	-
Manual query	Location — occurs-with — ancient greek Contained by — Cyprus
Remark	insufficient translation
SemSearch-40	the first 13 american states
Translated query	-
Manual query	-
Remark	impossible relation
SemSearch-41	the four of the companions of the prophet
Translated query	-

Manual query	person — occurs-with — companion prophet
Remark	insufficient translation
SemSearch-42	twelve tribes or sons of Israel
Translated query	-
Manual query	-
Remark	impossible relation
SemSearch-43	what books did paul of tarsus write?
Translated query	Book — occurs-with — paul tarsus write*
Manual query	Book — occurs-with — paul tarsus
Remark	sufficient translation
SemSearch-44	what languages do they speak in afghanistan
Translated query	Human Language — Main Country — Afghanistan occurs-with — speak*
Manual query	Human Language — Main Country — Afghanistan
Remark	sufficient translation
SemSearch-45	what tv shows has thomas jane been in
Translated query	Entertainment Lawyer · Profession · Entity · People With This Profession · Thomas Jane
Manual query	TV Program · Starring TV role(reversed) – Thomas Jane
Remark	insufficient translation
SemSearch-46	where the British monarch is also head of state
Translated query	Location — occurs-with — british monarch state
Manual query	-
Remark	impossible relation
SemSearch-47	who created stumbleupon
Translated query	Person — occurs-with — stumbleupon
Manual query	Person — occurs-with — stumbleupon
Remark	sufficient translation
SemSearch-48	who has jackie weaver been married to
Translated query	Person — Profession · Entity · People With This Profession – Jacki Weaver occurs-with – marry*
Manual query	Jacki Weaver — Spouse
Remark	insufficient translation , the subterm “Jackie” has been adjusted to “Jacki”
SemSearch-49	who invented the python programming language
Translated query	Person — occurs-with — python programming language
Manual query	Person — Developer(reversed) — Python (Programming Language)

Remark	sufficient translation
SemSearch-50	wonders of the ancient world
Translated query	-
Manual query	Location — occurs-with — wonders of the ancient world
Remark	insufficient translation

Bibliography

[1] Krisztian Balog and Robert Neumayer. "A test collection for entity search in dbpedia". In: *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. ACM. 2013, pp. 737–740.

[2] Hannah Bast et al. "Broccoli: Semantic full-text search at your fingertips". In: *arXiv preprint arXiv:1207.2615* (2012).

[3] Hannah Bast et al. "Easy access to the freebase dataset". In: *Proceedings of the companion publication of the 23rd international conference on World wide web companion*. International World Wide Web Conferences Steering Committee. 2014, pp. 95–98.

[4] Florian Bäurle. "A user interface for semantic full text search". In: *Masterarbeit, Albert-Ludwigs-Universität Freiburg, Lehrstuhl für Algorithmen und Datenstrukturen* (2011).

[5] Sonia Bergamaschi et al. "Keyword search over relational databases: a metadata approach". In: *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*. ACM. 2011, pp. 565–576.

[6] Björn Buchhold. "Susi: Wikipedia search using semantic index annotations". In: *Masterarbeit, Albert-Ludwigs-Universität Freiburg, Lehrstuhl für Algorithmen und Datenstrukturen* (2010).

[7] George A Miller. "WordNet: a lexical database for English". In: *Communications of the ACM* 38.11 (1995), pp. 39–41.

[8] George A Miller et al. "Introduction to wordnet: An on-line lexical database*". In: *International journal of lexicography* 3.4 (1990), pp. 235–244.

[9] Gebhard Rusch. "Kommunikation und Verstehen". In: *Die Wirklichkeit der Medien*. Springer, 1994, pp. 60–78.

[10] Craig Silverstein et al. "Analysis of a very large web search engine query log". In: *ACm SIGIR Forum*. Vol. 33. 1. ACM. 1999, pp. 6–12.

[11] Kristina Toutanova and Christopher D Manning. "Enriching the knowledge sources used in a maximum entropy part-of-speech tagger". In: *Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics-Volume 13*. Association for Computational Linguistics. 2000, pp. 63–70.

[12] Thanh Tran, Daniel M Herzig, and Günter Ladwig. "SemSearchPro–Using semantics throughout the search process". In: *Web Semantics: Science, Services and Agents on the World Wide Web* 9.4 (2011), pp. 349–364.

[13] W3C. Accessed: September 2014. URL: http://www.w3.org/RDF/.

[14] Qi Zhou et al. "The Semantic Web". In: ed. by Karl Aberer et al. Vol. 4825. Lecture Notes in Computer Science. Springer, 2007. Chap. SPARK: Adapting Keyword Query to Semantic Search, pp. 694–707.

Declaration

I hereby declare that I am the sole author and composer of this thesis and that no other sources or learning aids, other than those listed, have been used. Furthermore, I declare that I have acknowledged the work of others by providing detailed references of said work.
I hereby also declare that this thesis has not been prepared for another examination or assignment, either wholly or excerpts thereof.

Place, Date

Signature