Natural Language Processing, Homework 3 Report

Dorjan Hitaj, Matricola 1740478

11 June 2017

1 Abstract

Information extraction is the task of extracting structured information from unstructured or semi structured text. In this homework we had to extract information from various corpora for 5 specific relation pre-selected by us. I have chosen to extract information about: ***activity***, ***shape***, ***colorPattern***, ***howToUse***, ***similarity*** relations by following different information extraction techniques. The corpora I used are: babelfied wikipedia, and various internet sites.

2 Introduction

In a huge corpora such as the web there may exist various relations between various pairs of concepts. We want to extract these relations from these corpora in order to make possible performing computations on those data. This task was about extracting relation triples that fit to various categories. With those triples question-answer pairs have to be generated. To tackle this information extraction task I have followed 2 extraction paradigms, one presented in Navigli, Moro 2013 and a standard seeded information extraction technique. By means of heuristics and manual inspection I have filtered the information extracted to remove the ones extracted by mistake.

3 Information Extraction

Information extraction is the task of extracting structured information from unstructured or semi-structured text. The information will be extracted in the form of triples, which are composed of 2 concepts and the relation that exists between those two concepts. The triple is of shape **{source_concept, relation, target_concept}**. I have used the **Babelfied wikipedia** and various internet sites to extract information about the relations I have chosen.

For the first one I have followed the paper of Navigli and Moro, 2013, while for the second type of source I have used seeded information extraction.

3.1 Babelfied Wikipedia

3.1.1 File preprocessing

The babelfied wikipdia consists of babelfied wikipedia pages, whic completely extracted totaled in an amount of 220GB. Before starting to implement the Navgli, Moro algorithm I did some preprocessing of the files. The babelfied wikipedia folder was structured in 666 folders each of them containing on average 6500-6900 xml files(babelfied wikipedia pages). Reading them one by one would cause a toll on the computation time because of the location of those files in disk. They might be scattered through various memory segments so reading would be slow. To tackle this problem I merged the content of each folder's files into a single large file. This single large text file contains all those files contents. This resulted in a tremendous speed up in file processing time since now all those 6900 small files which were previously randomly scattered in the storage are now stored in a contiguous segment which is read very fast. This is a crucial optimization step in the overall runtime.

3.1.2 Triple extraction

The Navigli Moro algorithm proceeds as follows:

For each wikipedia page, **for each** sentence in that page get all the hyperlink pairs in the sentence. **For each** of these pairs get the text in between them and check if that phrase contains a verb then store in a list the ***phrase(p)*** and the ***relation instance {h1, p, h2}***. To check if there is a verb in the phrase I have used **nltk** part of speech tagger since it is fast and has high accuracy.

After the whole files have been processed and all the possible relation instances and relation phrases {h1, p, h2} have been found is the time for cleaning.

Cleaning is done in two phases:

Phase 1 We check if a relation phrase is frequent enough in the corpus. This frequency is compared to a ***neta***-value, and if the relation phrase frequency is lower we discard that relation phrase and all the relation instances {h1, p, h2} that contain that phrase(p). The value of ***neta*** I have chosen is 3 as the best, after a few tests with values from 2-5. Lower than this range would mean considering all phrases which is not appropriate and higher than this range the risk of discarding valuable relation instances increases. To count the frequencies of the relation phrases I have used python **Count** class which returns a dictionary of the phrases and their counts. This way of frequency counting gave a speed up impact of more than 8-10 times faster than usual counting with loops. This is a crucial optimization step in the overall algorithm run time.

Phase 2

Now for each relation phrase in the cleaned phrases, I perform dependency parsing, by means of Spacy python library, of the following fictional sentence: **d = depParser("x "+ phrase +" y")** and check:

If according to the dependency parser **x** is not a **subject** of a word in the phrase(p) and **y** is not a **object** of a word in phrase(p) then discard that phrase and all the relation instances that contain that phrase.

In the end what is left are the relation instances that have the highest chance to be correct and are syntactically well-built relations. In approximate numbers after phase 1 and phase 2 the number of relation phrases extracted remaining is about 26000 and the number of relation instances(h1, p, h2) is about 120000. These numbers refer to total extractions before filtering for only the relations I have chosen. Now these relations have to be filtered and mapped to the 5 relations that I have, which I explain in section 4 **Triple mapping**

This method was not enough to extract the triples I needed so to cope with that I had to perform some additional triple extraction methods like the one explained below which is Seeded web scrapping

3.2 Web scrapping

3.2.1 Automatized scrapping

The above mentioned method for triple extraction was not very fruitful in extracting triples for the relations I have chosen, reasons that are explained in the Triple mapping **problems** section.

To extract relations about the relations {***activity***, ***shape***, ***colorPattern***, ***similarity***} I have targeted the animals and musical instruments category by scrapping Encyclopedia-like websites which had huge lists from A-Z of animals or instruments. I scrapped them by using the python libraries dryscrape and Beautifulsoup.

The scrapping works as follows:

First I initialize some arrays of seeds, with most common words you expect in a sentence for the above mentioned relations. For example in the **colorPattern** relation the suitable seeds are various color names and the word colour itself, for the **activity** some useful seeds are actions performed most commonly by animals like ***hunt, fly, swim, run, jump, hop, is able to, can*** and many more. In a similar way I instantiate seeds for the other relations.

One of the most fruit-full web sites for relations was ***"a-z-animals.com/animals/"***. This site contains almost all species of animals and for each animal has a specific page where the animal and its characteristics are explained in details.

The web site is structured in this way:

First there is a list of animals and to go to the animal specific page to the main url is concatenated the name of the animal. This feature was in my advantage for direct triple mapping since when I am parsing a single animals page I know all the time the source concept for which I am looking target concepts in the sentences of the page. This is very useful in sentences of the type: *It is white in color.* In this case I can easily define that the **It**, with high probability, refers to my source concept and I can take advantage of more sentences in the page, even though the concept is not directly mentioned in some sentence.

The web-page parsing is done in this way. I get all the links in the first page, and then in a loop I open all the animal-specific pages. By means of Beautifulsoup I strip the web page of its styling elements so I can focus only on the raw text information in the page. From this step I check all the sentences that contain the seeds and write them in a file in the format: *source concept(which is the site name), relation, target concept, sentence where I found the relation instance.* After that I check the file manually to remove the incorrect triples, but in fact the number of incorrect triples was less than 5% of the total number of triples extracted. In a similar way I tackled the triple extraction for the musical instruments. To perform this kind of information extraction I needed to first see how the site structure was, exploit that structure, and then write the code to fit the structure. In both cases was very efficient because the sites were very well-built and the seeds were straight to the point for the relations I needed.

3.2.2 Manual scrapping

The most difficult relation to extract triples was the **howToUse**-relation. The reason why is that most of the sentences found in the internet or in the babelfied wikipedia are slightly more towards the **purpose**(not in my relations). You can barely find sentences of the type: (You read a book; You drive a car) so to tackle this problem I extracted some of the triples manually for the **howToUse** relation.

3.3 Patterns

The patterns I designed for my relations are the most common and straightforward questions you ask about a concept to find out information about the concept in one of the relations that I had chosen. I also designed this patterns to be short and simple so that when generating Q/A pairs, the questions would be also well-formed.

Activity	**Shape**	**Similarity**	**colorPattern**	**howToUse**
What is X able to?	What is the form of X?	What can X be confused with?	What is the color of X?	How do you use a X?
What can X do?	How is X shaped?	What does X look like?	Is X Y?	What do you do with X?
Does X Y?	What shape does X have?	What does X resemble to?	Is Y the color of X?	Do you Y a X?
	What is X shaped like?	Is Y similar to X?	Is Y the pattern of X?	
	Is Y the shape of X?	Does Y look like X?		

4 Triple mapping

4.0.1 Problems

In the process of triple mapping I encountered some problems. The most significant one was that the triples extracted from the babelfied wikipedia were not for the relations I had to extract triples for. The were mostly for location and other simpler relations. The reason why I think there were almost no relations about my triples in the babelfied wikipedia is that the wikipedia sites were parsed by checking only the hyperlinks according to the Navigli, Moro algorithm, while the relations could exist even between a concept(which was a hyperlink) and another concept which was not a hyperlink.

For example: Lets consider a sentence I have used to extract a shape relation: *The Discus is a medium sized Fish, with a round, flattened body.* Sentences like this even if they were in wikipedia, the concept **round** in more than 90% of time is not a hyperlink so that triple, even if it existed in the babelfied wikipedia, it would not be extracted by the Navigli, Moro algorithm or any algorithm that is specifically based on only the concepts which are also hyperlinks in the web page. That is why I had to perform additional work on extracting triples for the relations that I had chosen.

4.0.2 Triples

Most of the triples I extracted by web scrapping, were automatically mapped to the appropriate relation because they were extracted by seeds that I designed specifically for that relation. Moreover I have manually skimmed all the triples extracted, to remove any incorrectly included triple in some relation, but they were very few to be removed. The rest of the triples, more specifically the triples of **howTouse**-relation I mapped them manually while searching because by using seeds was not very effective. The total number of triples extracted for each relation is displayed on the table below

	Number of triples
Activity	91
Shape	28
Similarity	44
colorPattern	74
howToUse	27
Total	**264**

5 Disambiguation

Disambiguation is very important to switch from superficial realization of relations to a more semantic and syntactic analysis. To perform this task I have used the **Babelfy** disambiguation system through its HTTP API. For each of the triples extracted I instantiate a call to the Babelfy API with the sentence where I have extracted the triple and get the babelnet_ids for the source and target concepts of the extracted triple for that relation. Throughout this process I noted that some concepts were not recognized by the Babelfy system and for those concepts I have left the babelnet_id empty as advised by instructors.

6 Q/A Generation

To generate the question answer pairs I read all the question patterns that I have created and split them into 5 lists (one list contains the question patterns for one relation). After that I read the triples I have previously extracted along with the sentence where the triple is found, and also the source and target concepts that are previously disambiguated, and perform a similar splitting into 5 lists as above, thus meaning that all the triples of the same relation will be on the same list. One line read is of this shape: *source, relation, target, sentence, source(babelnet id), target(babelnet id)*

I have generated positive and negative question answer pairs. After generating all the Q/A pairs I write them in the file according to the required format **q, a, r, s, source::babelnet_id, target::babelnet_id**. For the negative question answer pairs I have not included the ::babelnet_id for the source and negative_target.

Positive Q/A pairs

For each relation I call a function with all the question patterns and all the triples of that relation and proceed as follows: For each of the question patterns, I iterate in all the triples and substitute X with the source concept and Y with the target concept. In questions patterns that contain both X and Y the answer is **Yes** while the answer in question patterns containing only X is the target concept of the corresponding triple. ***Ex:*** Consider the triple: ***angelfish shape triangular*** for the shape relation: The questions generated according to the patterns for shape(table 1) are like this:

What is the form of angelfish? triangular, How is angelfish shaped? triangular, What shape does angelfish have? triangular, Is triangular the shape of angelfish? Yes.

Negative Q/A pairs

For each relation I call a function with all the question patterns and all the triples of that relation and proceed as follows: For each question pattern that contains both X and Y, I substitute X with the current triple in the loop source concept, and for Y I generate a random integer in the range [0, length of current relation triples list] and from that random triple I get the target concept and place it in place of Y in the question. The answer of this question is with high probability **"NO"**. Since the target concept is random the index might occur to be the index of the current triple in the loop, case in which the questions answer would be positive, so I also check this case to prevent producing incorrect negative questions. ***Ex:*** Consider the triple: *angelfish shape triangular* for the shape relation: The questions generated according to the patterns for shape(table 1) the output is like this: Is triangular the shape of angelfish? Yes. By generating a random index lets assume I get the triple *Concertina shape hexagonal.* The question answer becomes: Is hexagonal the shape of angelfish? No.

Q/A statistics In the table below the number of generated question answer pairs for each relation is displayed.

	Positive Q/A	Negative Q/A
Activity	273	81
Shape	140	26
Similarity	220	87
colorPattern	296	209
howToUse	81	26
Total	**1010**	**429**

Total Q/A (negative + positive)	**1439**

7 Source Code

In the **/src** directory I have included the python files I wrote to handle this task. I will briefly name them here and tell what is their function.

navigli_moro_information_extraction.py : In this file I have implemented the information extraction algorithm presented in the Navigli, Moro paper named "Integrating Syntactic and Semantic Analysis into the Open Information Extraction Paradigm". The source corpora with this code I used is the babelfied wikipedia.

web_scrapping.py : Is the code I wrote for seeded information extraction from various internet sites, by means of the Beautifulsoup library. Small part of code takne from Beautifulsoup documentation.

disambiguate_babelfy.py : Used to perform disambiguation of the triples by using the babelfy disambiguation system through its HTTP API. Most of the code is taken from the documentation of the Babelfy HTTP API and modified to cope my needs.

question_answer_generator.py : Code that generates positive and negative Q/A pairs for the extracted triples.

8 Conclusion

This task was about information extraction. Information extraction is the task of extracting structured information from unstructured data. The information had to be extracted in the form of relations that occur between two concepts. To solve this task I used two different approaches. One presented in Navigli, Moro 2013 and the other was seeded information extraction. Throughout this process I encountered difficulties regarding the babelfied wikipedia corpora since there, only a hyperlink based information extraction was performed according to Navigli, Moro. There were many triples in the babelfied wikipedia but most of them were about relations that were not mine to extract. To extract triples for the relations I had I performed some extra task. I used seeded web scrapping to targeted internet sources that helped me gather a lot of qualitative triples. After extracting all the triples I performed disambiguation by means of the Babelfy-disambiguation system. From that I built Q/A pairs both positive and negative. Throughout this process I noted that for some specific relations like the ones I had chosen, a more target-specific procedure was needed in order to extract useful information. I extracted a total of 264 triples and from them I generated a number of 1439 Q/A pairs.