Report Natural Language Processing
Homework 3

Domenico Alfano

1 Introduction

The problem of Knowledge Acquisition lies at the core of Natural Language Processing.
Knowledge Acquisition is the process of acquiring that information, and its formalised structure, that will allow some particular task to be performed by a computer system.
In this report, consideration is given to Knowledge Acquisition from Natural Language, in other words, the acquisition of information from the written source, Wikipedia.
The main goal of the assignment is to implement a system that generate Question-Anwer pairs, using Information extraction, given a data set.
This implementation follows five steps:

- In the first step I have choosed five relations.
- In the second I have created the question patterns for each relation.
- In the third step I have created triples using Information extraction.
- In the fourth step I have disambiguated the words in the data set.
- In the last step I have created the file question-answer-paris.txt.

Following this report, it will be explained in detail how I implemented these steps.

2 Relations

As I said before, the fisrt step was to choose the relations between two concepts. The relations are used to filter the written source in order to extract only the desired information. Obviously, we could find in a written source, as Wikipedia, a big number of relations.
In my case, I choosed five relations:

- Specialization: describes what a concept is.
  For example:
  - Dog is Animal.
  - Atom Heart Mother is Album.
  - Pizza is Margherita.
- Size, describes what is the dimension of a concept.
  For example:
  - Eiffel Tower is high.
  - Central Park is big.
  - Ant is small.

- Time, describes when a concept was used or created.
  For example:
  - Google it has been created in 1998.
  - The atomic bomb was used in 1945.
  - The Stadio Olimpico was created in 1932.
- Purpose, describes how a concept could be used.
  For example:
  - Credit card is used to withdraw money.
  - Sporting center is used for training.
  - Smartphone is used to call.
- Similarity, describes what a concept could be confused with.
  For example:
  - Aswardby could be confused with Aswarby.
  - Intolerance could be confused with food allergies.
  - Periodic limb movement disorder could be confused with restless leg syndrome.

3 Patterns

After choosing the relations, I created, as requested, at least three question patterns for each relation:

- Specialization:
  - What is X?
  - Is X, Y?
  - What is the specialization of X?
- Size:
  - What is the dimension of X?
  - What is the size of X?
  - How is X?
- Time:
  - In which year it has been created X?
  - When X was created?
  - Is X created in Y?
- Purpose:
  - What X is used for?
  - For what could be used X?
  - Is X could be used for Y?

- Similarity:
  - What X could be confused with?
  - What X is similar to?
  - Could X be confused with Y?

As we can see, I used variables in the questions because they will be later replaced by the concepts associated.

4 Information Extraction

Now, I have a set of relations and a set of questions for each relation. The next step is to collect concepts to fit the relations, in other words, the next step is to extract information. To extract Information from written source, I chose the DEFIE method, an approach based on a syntactic-semantic analysis of textual definitions. In DEFIE the key idea is to leverage the linguistic analysis of recent semantically-enhanced Open Information Extraction techniques while moving from open text to smaller corpora of dense prescriptive knowledge. The aim is then to extract as much information as possible by unifying syntactic analysis and entity linking. This method consists of three phases:

- Syntactic Analysis: each textual definition is parsed to obtain a dependency graph.
- Semantic Analysis: entity linking and word sense disambiguation..
- Syntactic-Semantic Graph Construction: the information extracted by parsing and disambiguating a given textual definition is unified into a syntactic-semantic graph.

First of all, it is worth to specify that the format of the data provided is .xml, so I needed to parse them.
This process was implemented with the aid of the *lxml* library that allowed me to extract the corpus of the text and the disambiguated words. Having done that, I implemented the *parse_xml* function that is aimed at:

- "Cleaning" the text taken as input. This means that the function delete for every sentence all the text between brackets, because in my opinion the main information in a sentence is out of them.
- Collecting all the words that will be merged, saving for each of them the index of the first character of the word in the sentence where it appears.

In this way I have two different lists: a list of all "cleaned" sentences and a list of a list of all the disambiguated words. The Syntactic Analysis is implemented with the support of the *Spacy* library that offers tokenization, sentence boundary detection, POS tagging and the syntactic parser.
This library required as input a text and deliver for each sentence the syntactic dependency between words. The Semantic analysis is based on Babelfy that uses a dense subgraph algorithm to identify high-coherence semantic interpretations of words and multi-word expressions across an input text. It was applied Babelfy to each textual definition, obtaining a sense mapping from surface text to word senses.
So, taking the list of all disambiguated words, This step was implemented using the *merge* function of *Spacy* to disambiguate the words. Finally, starting from the syntactic dependency between disambiguated words, as provided by the syntactic and semantic analysis, I built Syntactic-Semantic Graph with the aid of the *networkx* library. To do this, firstly, I created, with the function *get_edges*, the edges between all the nodes given their dependency. Semantic information from

the sense mappings was incorporated directly in the vertices of dependency graph by attaching available matches between words to the corresponding vertices.
In order to extract consistent information, subsets of vertices referring to the same concept or entity are merged in to a single semantic node, which replaces the subgraph covered in the original dependency structure.

4.1 Triples

At this step, all the information in a given textual definition has been extracted and encoded in the corresponding graph. Now, I have to consider those paths connecting entity pairs across the graph and extract the relation pattern between two entities and/or concepts, as the shortest path between the two corresponding vertices in Syntactic-Semantic Graph. This enables me to exclude less relevant information and reduce data sparsity in the overall extraction process.
Having done that, I considered only the shortest paths that connect a subject node to an object node or to an adjective node passing for a verb node, using the POS tagging provided by *Spacy*.
The triples have been obtained, with the function *get_triples*, by creating, for each sentence, lists of three elements where:

- The first element is the subject node.
- The second element is the path that connect the start node to the end node.
- The third element is the object node or the adjective node.

At this point, I needed to filter again all the obtained triples for each relation according to the question patterns. In fact:

- For the specialization relation, I considered only the triples where the first element of them started with an uppercase letter and it must not be a number. The second element of them has to be *is*. Obviously, I did not consider the triples where the third element of them was a possible adjective for the size relation. [1]
- For the time relation, I considered only the triples where the second element of them has to be *created in* and the third element has to be a number. [2]
- For the size relation, I considered only the triples where the second element of them has to be *is* and the third element of them had to be a possible adjective for the size relation. [3]
- For the similarity relation, I considered only the triples where the second element of them has to be $confused\ with$. [4]
- For the purpose relation, I considered only the triples where the first element of them started with an uppercase character and the second element of them has to be *used as* or $used\ for$. [5]

[6]

[1] Implemented in the function *spec_rel*.
[2] Implemented in the function *time_rel*.
[3] Implemented in the function *size_rel*.
[4] Implemented in the function *similarity_rel*.
[5] Implemented in the function *purpose_rel*.
[6] All the code for the Information Extraction task has been implemented in the file Homework3.py .

5 Question/Anwer pairs

Here, I approached to this task using a different technique. In fact, I used an informatic approach, instead of an Artifical Intelligence approach. The implementation is developed in four phases:

- Firstly, I saved all the triples in a text file. In order to do this, I created for each relation, a text file to save the triples and then I merged them into a unique file called *triples.tsv*, as requested. [7]
- Secondly, I saved all the sentences associated with the triples extracted in a text file. As I did before, I created, for each relation, a text file to save the sentences and then I merged them into a unique file called *examples.txt* [8]
- Thirdly, I wrote the question patterns in a text file called *patterns.tsv*, as requested.
- Finally, I generated a fourth text file that reads from the abovementioned three files. It is in this phase that I replaced the variables of the questions with the elements of the triples. To do this, I used the function *replace* that requires as input the variable to be substituted and the string that will take its place. The file generated is called *question-answer-pairs.txt*, as requested. [9]

[10]

[7] Implemented in the function *print_triples*.
[8] Implemented in the function *print_examples*.
[9] Implemented in the function *q_a_pairs*.
[10] All the code for the Question-Anwer pairs task has been implemented in the file output.py .