Homework 3
Information Extractor

Marco Treglia
University of Sapienza
Master in Artificial Inteligents and Robotics
Natural Leanguage Processing

***Abstract*—In this homework, we have to excerpt information from a large set of Wikipedia's corpus. Precisely we have to extract at list twenty triple with the form $t = < a_i, r, a_j >$ which a_i and a_j arguments are the entity or concept and r is the relation pattern. We have to submit five relation wich in my case are shape, size, how to use, smell and sound. Then generate pairs of question-answer for each triple founded corresponding at the respective relation.**

1. Introduction

I first start building the DEFIE [1] approach, but at the end, I used differents approaches, respectively for each relation, on top of the dependency graph. Let's start from the beginning.

2. Manage the Data

For manage, the large dataset close to 19 GBytes compressed. This dataset comprising sixty hundred and sixty-six folder and each folder contains about six thousand and nine hundred XML file representing respectively Wikipedia's article. For easy oversee this huge dataset I first created the 'DataManager' class [Figure 1]. This first creates an index, then specifying the number of folder and file read the compressed gzip files and then convert it into a more manageable entity named **Record**. The latter class, in fact, have just two variables inside: *sentences* are all the phrase of the article, and *sm sentences* are all the babelfy's annotation for each phrase. We can now get into the game.

3. Dependency extraction

Having out textual definition d we can proceed to the syntactic and semantic analysis. Looking for the best solution, I came up with some different possibility (Stanford CoreNLP, nltk, spaCy). After some investigations, i chose the open-source library featuring state-of-the-art speed and accuracy spaCy [2].
I designed the **DependencyManager** class, wich takes as input a Record instance and convert each sentence into spacy.tokens.doc.Doc. This provides a nice syntactic analysis but an unprecise semantic parsing. For improve it, I

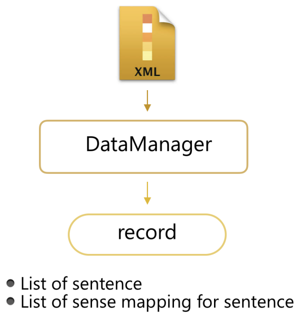

Figure 1. Scheme showing the extraction of data from 'babelfied-wikipediaXML'

merged the words in the spacy tree accordingly to the babelfy annotation. Unfortunately, this framework not provide a graph representation neither then the possibility to get distance or path between the word

4. Dependency graph

Hence for improve the maneuverability of moving between words in the dependency spacy's tree, I converted it into a graph, using the networkx [3] framework. Accordingly I created the **GraphManager** class. This class takes as input a DependencyManager instance, wich contains the disambiguated sentence, and create the graph representation. Furthermore, each word is mapped to the class **Node**, that will be the apt node of the graph.

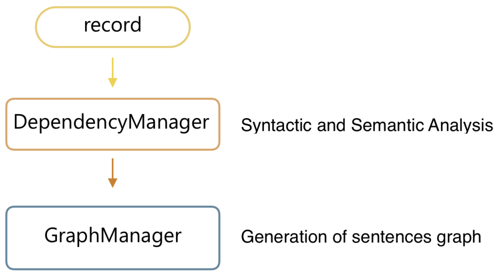
record
DependencyManager
Syntactic and Semantic Analysis
GraphManager
Generation of sentences graph

Figure 2. Scheme showing the generation of the graph

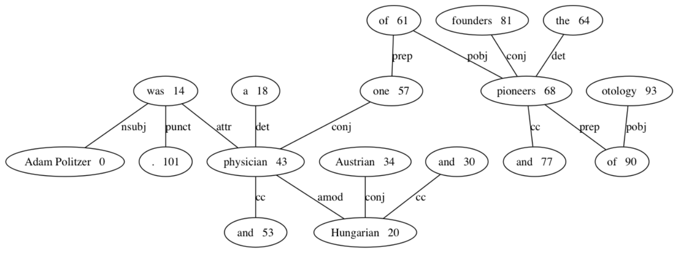

Figure 3. Example graph sentence with semantic tag as edge

5. Triples Extractor

This is the core of the project, and here comes the tricky thing of getting the right way for extract the information for the respectively relation assigned. The class **TriplesManager** do all this extraction task, taking as input the graph of the relation and the source text of the latter, then extract, manage and save each triple founded. For convenience, I used the class **Triple** for holding the discovered relation. Here the explanation for each relation.

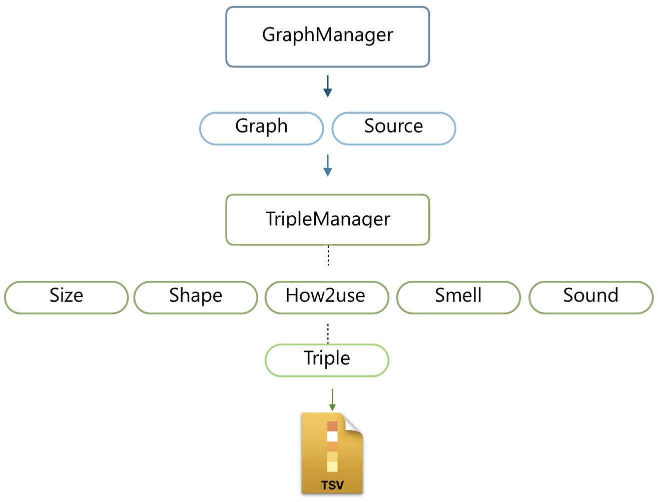

Figure 4. Scheme showing triple's extracion

5.1. Size

For the extraction of the triples for size relation I first looking in the sentence for all the noun singular and plural using the simpler Google Universal POS (part of speech) tag. Then check if in the sentence there are any adjective that matches with the pattern I used for express the size of something or taking the similarity measure using word2vec vector dimension from this adjective and the word shape. If the adjective satisfy the pattern or the similarity measure over a thresholder , i take the path between the noun and the adjective and takes only those that are lenght two or maximum three (taking one more word for better express the meanig of the relation with size).

5.2. Shape

For the extraction of the triples for shape relation, I used a similar approach for the size relation. Using the same POS noun tag as the first node to check, then verify the existence of an adjective second node which satisfies the similarity over a threshold with shape word or match with some pattern shape adjective. If the requirements are fulfilled, then takes the path between this two node with maximum length as two.

5.3. How to use

For the extraction of the triples for how to use relation, i used a different approach respect the previous relation. In fact with this, instead searching on the corpus the relation of the noun and a possible verb associate it, I realized that the frequency wich a possible noun (not pronoun) is associated with a verb of the pattern I created is much more likely to be correct respect all the feasible verb that could be found in corpus of reference. So at the beginning, i look for the noun using POS tag, then I measure the similarity between some category pattern I chose and this noun. If is greater than a threshold, I continue with comparing it with some verb pattern selected by me, and again if the similarity exceeds a certain tuned value I produce the triple with the best verb founded.

5.4. Sound

For the extraction of the triples for sound relation, i used a similar approach to the previous category. In fact, at the begin, i was searching for the phrase with some pattern and I come out with no triples or few incorrect one after several folders inspected. So I move to find a possible pos tag noun in the phrase, then measure the similarity with the pattern category I built. If exceed a threshold, carry on finding again the best fit with my sound adjective pattern. Finally the triple founded belong to the category I determined and contains the best suitable adjective.

5.5. Smell

For the extraction of the triples for smell relation I realize that was very similar to the sound one. Then using the same category pre-build pattern for making the first selection between the noun I proceed to find the best adjective to express the smell of that entity. I made another pattern for measure the similarity with a common smell-adjective and selected the best one wich, however, overcome the threshold.

6. Question Answer generation

Finding the triples the task was almost done. For generating the question answer I create a simple class **QuestionAnswerManager**. At the beginning, i used the triples.tsv file for generating it, but after the delivery requirement update (adding the babel id in the question-answer pair), I change it for generating the pair admitted directly from a Triple instance. In this way, since the Triple instance carries the Node classes, include also the babel id associated with it. I created three patterns for the generation of the question-answer pairs, randomly chosen and, as specify into the submission, I create the wrong question pair using the pattern of respective category.

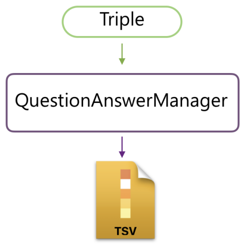

Figure 5. Scheme showing question-answer pair generation

7. Statistic

For easy get the statistic result, I made the **StatisticAnalysis** class. This reading from the triple founded extract the frequency distribution.

shape	size	how2use	smell	sound
3888	33687	473	99	542

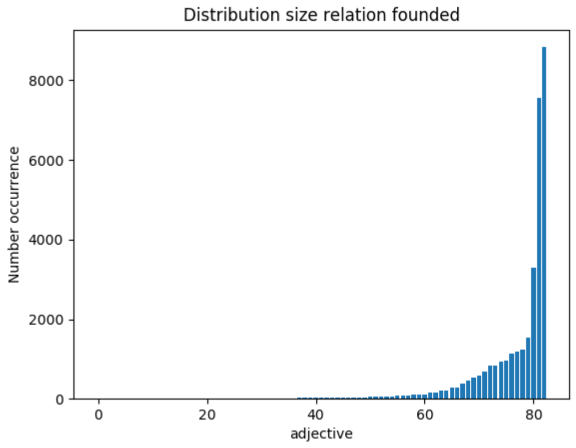

Figure 6. Graph showing size adjective distribution

Size top ten:

Adjective	Frequency
small	8834
large	7553
short	3300
little	1528
extensive	1238
big	1190
massive	1131
wide	953
huge	939
giant	826
vast	825

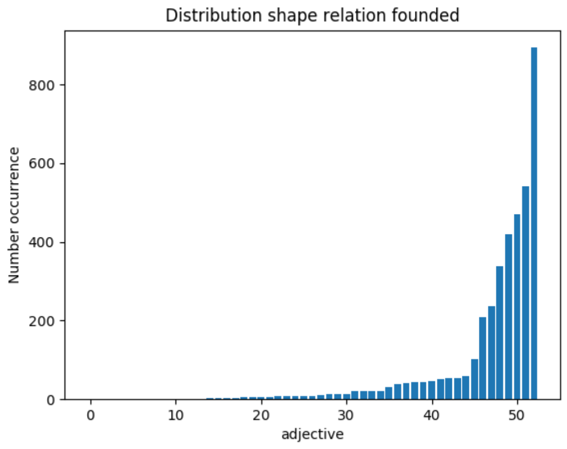

Figure 7. Graph showing shape adjective distribution

Shape top ten:

Adjective	Frequency
straight	894
flat	542
closed	470
round	419
solid	338
square	236
rugged	210
curved	102
sweeping	58
three-dimensional	54
oval	54

How to use top ten:

Verb	Frequency
eat	34
cook	27
sing	13
drink	7
read	6
detect	5
shopping	3
dance	2
plow	2
activate	0
love	0

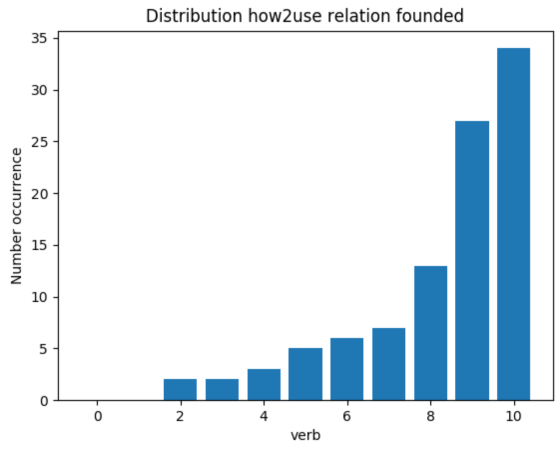

Figure 8. Graph showing how to use verb distribution

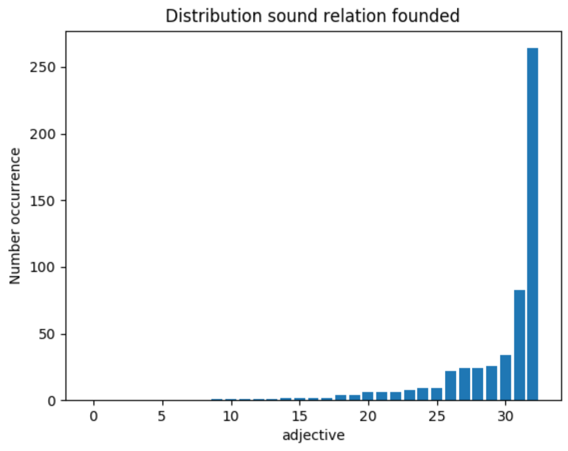

Figure 9. Graph showing sound adjective distribution

Sound top ten:

Adjective	Frequency
miaow	264
crunchy	83
strum	34
melodic	26
rhythmic	24
bark	24
bass	22
melodious	9
timbre	9
drone	8
mellifluous	6

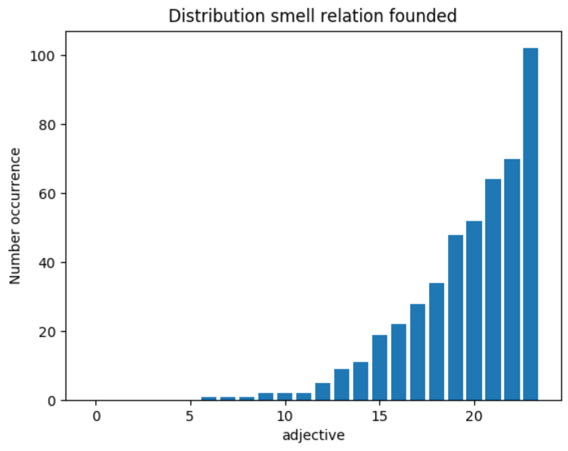

Figure 10. Graph showing smell adjective distribution

Smell top ten:

Adjective	Frequency
butterscotch	102
musky	70
aromatic	64
fragrant	52
funky	48
perfume	34
honeyed	28
smelly	22
luscious	19
stinky	11
perfumed	9

References

[1] Claudio Delli Bovi, Luca Telesca and Roberto Navigli Large-Scale Information Extraction from Textual Definitions through Deep Syntactic and Semantic Analysis .

[2] https://spacy.io Industrial-Strength Natural Language Processing

[3] https://networkx.github.io High-productivity software for complex networks