Report of the 2nd NLP Homework: Sense Embeddings

Manuel Prandini ID: 1707827

I. INTRODUCTION

After the reading of [1], i have created a Sense Embeddings model for the BabelNet synset that are in Wordnet. The goal for this homework is that given a context of words surrounding the target word, the model has to predict the correct word sense represented as *'lemma_synset'*. Same as standard word embeddings but with word senses as outputs. I have utilized Python 3.6.5 and different libraries used to parse the dirty datasets and to do preprocessing. For this homework, i worked locally, using my PC, with 8GB RAM and Intel I7 CPU with 4 cores.

II. PREPROCESSING

I made several modifications to the files obtained from the parsing of the two datasets used because, by testing, I tried to see how to improve the score by trying to combine different data cleanings. I decided to create a function that would perform all the data cleanups for both the datasets used, even if each of them has different corrections to make. I used also the stemmization on the words through the *Porter's algorithm* and i remove the english stopwords, both using the *nltk library*.

A. EuroSense Corpus: High Precision

I utilized the EuroSense corpus, and i downloaded it from *http://lcl.uniroma1.it/eurosense/*. From it, i chosen the *"High Precision"* xml file. This file was quite big (22,6 GB), so i used iterparse function from *lxml.etree library* to read it. Each element of the xml file is composed by a sentence in different languages (I utilized only the english sentences) and below some annotations containing the word to be replaced within the sentence (anchor) with the right replacement (the word already lemmatized) and the BabelNet synset. Then i created a temporary file of sentences that contained inside the correspettive words replaced with their BabelNet synset. I replaced only the BabelNet synsets in the right format, using a regex match. I check also if the BabelNet synsets had a correspondence with WordNet synsets using a mapping file. Then i preprocess it. First of all, I removed all the string punctuations, and i lower the words for each sentence and i splitted they because is more easy to preprocess the singular words. I verified using grep command on the file, that the EuroSense corpus had a lot of mistakes. For example, i found words that finish with a dot, so i correct these word removing this last one. Many formats of the BabelNet synsets were wrong because they don't finish with part of speech tagging. Other words in the annotations were without the lemma so it didn't correspond to the anchor in the relative sentence. The file contained also the HTML character escapes like *'''*, so I decided to remove replacing they with blank character.

B. Train-On-Matic

I downloaded this dataset from *http://trainomatic.org/trainomatic.html#about*. The structure of this xml file is different from EuroSense in that a sentence is repeated several times along the corpus and each time a different word is noted. the words in the annotations need to be lemmatized. Dataset file does not contain BabelNet synsets but WordNet synsets, so to recover the corresponding BabelNet synset i had to create a mapping from Wordnet synsets to BabelNet synsets. Moreover, i removed the punctuation and the non-printable ASCII characters and also some apostrophes (') before a few words. Moreover, many words were not split well, ended with a dot and immediately started another word without putting a white space between them, so I tried to split the dot as well.

III. MODEL

I used a CBOW type word2vec model, from *gensim library*. It takes as input an array of arrays composed of sentences split into single words (created with preprocessing) and different parameters, such as the *window*, to calculate how many words surround the central word both on the right and left side, the *min_count* that discards the words that do not appear at least as long as the number indicated in the parameter and various other parameters such as the *learning rate*. The model produces the word embeddings, or the representation

of words in vector space. In the end, I filtered the embedding file leaving only the synsets inside.

IV. WORD SIMILARITY TASK

For each pair of words, I found the related synsets (using a map that has lemmas as keys and values for a set of related synset), and for each pair of synsets of the two words I got the highest score among these. The score was calculated based on the cosine similarity between two synsets. The OOV words are considered *-1* in the final score. The final score is given by the *Spearman* between the gold of the pairs of words and the cosine similarities calculated on them. It is very important to create the sense embeddings vocabulary in the right way because the result changes drastically.

V. TRAINING AND TESTING

I made several changes to the data set cleanup during the tests, progressing gradually to see how much the performance improved. The window size of the word2vec model is setted to 5 for all the tests. I started the tests using only EuroSense and in the first test, using the word2vec parameters in Table I, I obtained a fairly low Spearman, of 0.16 because I didn't check the synset format (so I also left synset that didn't end with the POS tagging), but I checked only if the offsets were contained in the mapping from BabelNet to Wordnet and I only removed some types of punctuation, like commas, points, semicolons and replacing the HTML escape characters. Well reviewing the cleaning performed on the dataset I noticed the fact that some synsets did not have the following format *"lemma_bn:offsetPOStagging"* and I noticed that many sentences had other punctuation characters as question marks, gates etc, so I also removed them and checked the synset format via a regex. With the parameters indicated in Table II I got 0.19. In the third test I tried to decrease the learning rate and to increase the number of iterations as you can see in Table III, I also tried to reinsert the HTML escape characters, because I thought that removing them, it also removed meaning to sentences and this has increased the Spearman reaching 0.22. I also checked if one of the Wordet synonyms matched to the lemma of the BabelNet synset replaced within the sentences but this worsened the performance. Subsequently lowering the learning rate to 0.025, Table IV, removing special characters, punctuation but leaving the HTML escape characters inside, and furthermore realizing that many synsets started with a capital letter, I modified the regex and got an improvement reaching 0.25 of Spearman. Then, I tried to vary the word2vec parameters but the result got worse and so the most I could get from EuroSense was with those parameters and those cleanings on the data. So I decided to add the TOM dataset to train my word2vec model. TOM compared to EuroSense which is based on speeches of the European parliament, has many sentences with mathematical symbols so they have a different sentence structure, since the first is much more discursive. I decided to remove the stopwords, many initial apostrophes to the tokenized words, and split the words also for the dot, as two consecutive words were not split well and word2vec considered a single one, and also many words ended with a dot so I also removed the latter from them. Since TOM had no HTML escape characters I decided to remove them permanently. The parameters of the Table V and these changes, gave me Spearman 0.28. Last change that i made was stemming on all the words using the Porter's Algorithm, and I also wanted to add an extra parameter to the word2vec model, trying to set the sample parameter that is the threshold for configuring which higher-frequency words are randomly downsampled, as shown in Table VI, this led me to reach a Spearman 0.3247. In the end, the best result I get is 0.3323 also eliminating the non-printable ASCII characters and filtering the synsets with the right format in the sense embedding that I need to determine the score in the word similarity task. Finally, I tried to train the word2vec model with different parameters, increasing the min count, and the feature array but this only led to worse results.

VI. CONCLUSIONS

In this homework to improve performance it was very important how the datasets to be given as input to the word2vec model have been preprocessed, since not cleaning the words of the dataset correctly, many useful data for embedding are lost. Certainly, it is possible to create a model with a better Spearman using much more data, and of a different nature, or that deal with different topics. Due to lack of hardware resources I was unable to preprocess SEW, another dataset proposed in the slides, but despite this, using t-SNE with the parameters indicated in the Table VII was able to show, figure 1, that taking as an example the synsets similar to the two senses of the word *" bank "*, one intended as a *"river bank"*, and the other as *"a public institution or credit institution"*, they are divided linearly. In the figures below [1,2,3] it is possible to see other results obtained.

1st Test	
Dataset	EuroSense
Size	400
Min_count	2
Hs	default
iter	5
learning rate	default
sample	default
Spearman	0.16

TABLE I: Parameters of the first test based only on EuroSense dataset

2nd Test	
Dataset	EuroSense
Size	300
Min_count	2
Hs	1
iter	5
learning rate	0.05
sample	default
Spearman	0.19

TABLE II: Parameters of the second test. Respect to the previous test, size is changed and also the learning rate

3rd Test	
Dataset	EuroSense
Size	300
Min_count	2
Hs	1
iter	10
learning rate	0.03
sample	default
Spearman	0.22

TABLE III: Parameters of third test. Respect to the previous test, learning rate is lower and the iterations are more

4th Test	
Dataset	EuroSense
Size	300
Min_count	2
Hs	1
iter	10
learning rate	0.025
sample	default
Spearman	0.25

TABLE IV: Parameters of fourth test. Respect to the previous test, the learning rate is lower and it is last test with only EuroSense preprocessed.

5th Test	
Dataset	EuroSense and TOM
Size	300
Min_count	2
Hs	1
iter	10
learning rate	0.025
sample	default
Spearman	0.28

TABLE V: Parameters of fifth test. Respect to the previous test, has been added the TOM dataset

6th Test	
Dataset	EuroSense and TOM
Size	300
Min_count	2
Hs	1
iter	10
learning rate	0.025
sample	1 e-3
Spearman	0.3247 - 0.3323

TABLE VI: Parameters of sixth test. Respect to the previous test, has been added the sample paramater and in this table are reported two Spearman values, where the first is based on the test with the non printable ASCII characters inside, the second is based on removing these characters and review the right format when was build the sense embeddings vocabulary and shown the better result.

t-SNE Parameters	
Parameter	Value
iterations	+- 4000
n components	2
perplexity	10.0
learning rate	10.0

TABLE VII: Parameters of t-SNE. Thanks to these values has been possible to plot the different sense of the synsets of bank

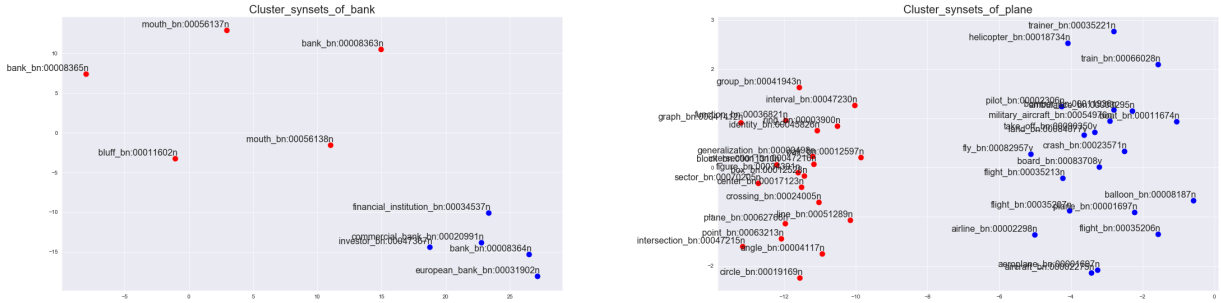

Fig. 1: In the figure on the left, red dots are synsets similar to the river bank synset, while blue dots are synsets similar to bank as credit institution synset. In the figure on the right, the cluster of the similar synsets of the two meaning of the word 'plane', where red dots indicate plane as geometric figure and the blue dots indicate plane as airplane.

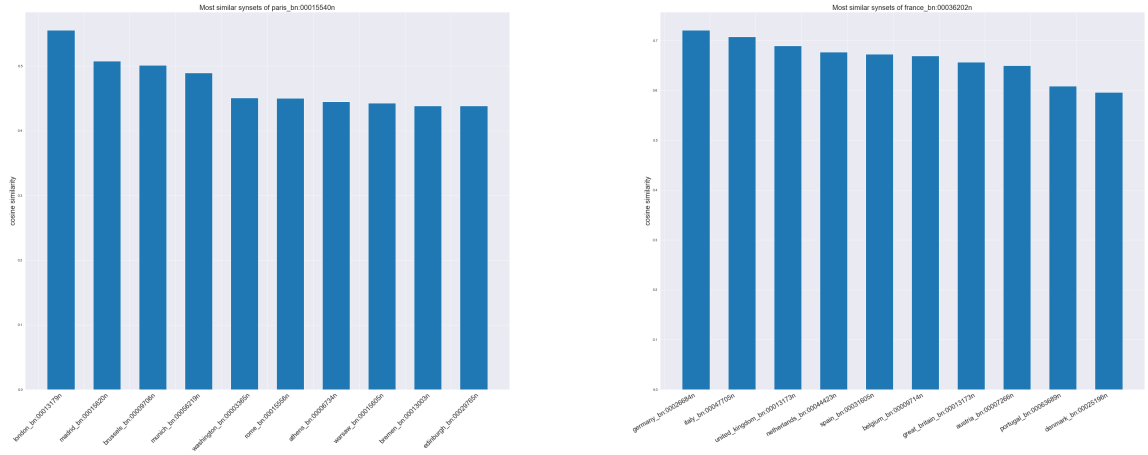

Fig. 2: These figures show the differences between words similar to France and Paris. For France are found names of other nations, for paris city names.

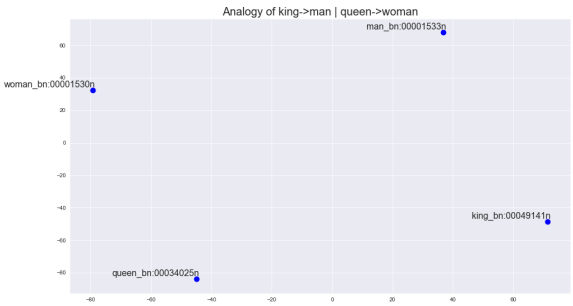

Fig. 3: The figure show the analogy between king and man with queen and woman.

REFERENCES

[1] Iacobacci, Pilehvar and Navigli, SENSEMBED: Learning Sense Embeddings for Word and Relational Similarity, 2015.