NLP
Assignment 2: Sense Embeddings

Hossam Samir Arafat - Matricola: 1803850

May 2019

1 Introduction

This report details the process of obtaining sense embeddings of English words by training a Word2Vec model on a sense-annotated resource. The sense embeddings are then tested on a word similarity task. The cosine similarity between the embeddings of each word is computed and the Spearman's rank correlation coefficient compares the maximum computed cosine similarity with the ground-truth value for each word-pair.

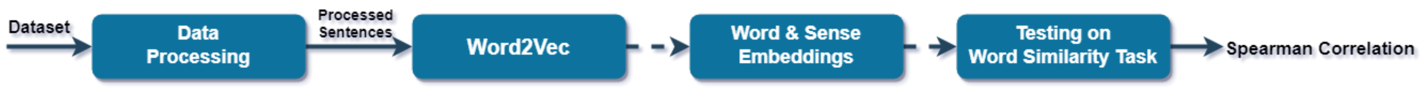

Figure 1: Overall Pipeline

2 Dataset Processing

2.1 Dataset

EuroSense is a sense-annotated resource which is built via the automatic disambiguation of the **Europarl parallel corpus**. The sense annotations are acquired from the multilingual sense inventory of **BabelNet**. **EuroSense** contains a total of about 1.9M sentences. Only BabelNet synsets with corresponding WordNet offsets were selected to reduce the amount of unique senses considered in training. The *"high-precision"* version of **EuroSense** was used in this implementation.

2.2 Processing

The main method used to find the *anchor* of every annotated word in the sentence and replace it the with its *lemma_sysnet* is the *replace()* method of Python's string class. This required careful processing to avoid the occurrence of the following errors:

1. If the annotated word is the first word in the sentence (with no space before it) or is followed by a punctuation sign without any spaces, it can incorrectly be ignored.
2. If one annotated word was a sub-string in another, for example: the anchor for ***"glove"*** can be incorrectly replaced with the synset for **"love"**, as such: ***"glove_bn:01"***.
3. If there are overlapping mentions, for example: the anchor for *"European"*, *"Commission"*, and *"European Commission"* can be incorrectly replaced with "European *Commission_bn:31"* where the longest annotation is **not** considered, as opposed to the correct "European_Commission_bn:40".

These pitfalls were avoided by implementing the following steps: First and foremost, the processing algorithm does not process a sentence until it meets the next sentence in the file. Even though this means that the algorithm ignores the final sentence in the file (negligible amount of 1.9M), it ensures that *all* annotations for a given sentence were collected before running any processing steps on it. After acquiring all the annotations for a sentence, a blank space is added before and after each anchor to be able to find only exact matches of this unique token; no sub-strings. The anchors are then ordered by length, so that the longest annotations are replaced first, for example, "European Commission" is correctly replaced while "European" and "Commission" are ignored. This ensures the algorithm does not fall into the 2nd and 3rd pitfalls. To avoid the 1st pitfall, all digits and punctuation symbols are replaced by a single blank space. Next, all duplicate (more than one) spaces are removed and a blank space is then added before the first and after the last word in the sentence.

Finally, after replacing each anchor word, all non-annotated stop-words are removed.
Tokens that appear no more than five times are ignored as to reduce the size of the vocabulary and hence training time. Moreover, five occurrences are not enough to sufficiently learn the context that a given word / sense occurs in.

3 Model

3.1 Architecture

The Continuous Bag of Words (CBOW) Word2Vec architecture from the Gensim library was used to obtain the sense embeddings. The network is trained to predict a target word / sense given its context. The context is defined by a window of neighbouring words on each side of the target word. In all training experiments, the following parameters were set as per (Mikolov et al., 2013a) recommendations on the original Word2Vec implementation: Negative sampling was used as the training algorithm and sub-sampling of frequent words was set to 10^{-3}. The values for the next group of parameters were found using a Grid Search Algorithm -described in the next section: the window size was set to 10 words, dimensionality of the vectors to 350 and the starting learning rate at 0.02.

3.2 Experiments

Grid Search sweeps the window size in through the three values $[5, 7, 10]$, the embeddings dinemnsionality through $[300, 350, 400]$, and finally, the learning rate through the values $[0.02, 0.03, 0.04]$.
At each iteration of Grid Search, the network is trained for 30 epochs on the current values of the hyper-parameters and the optimal values are simply the ones which yield the highest Spearman Correlation.
Results of the Grid Search algorithm can be found in Appendix A.2, Table 1.

4 Testing: Word Similarity

Given a set of word pairs, the implemented system should compute a similarity score for each pair, and these should be as close as possible to those given by humans. The sense embeddings were evaluated on the WordSim-353 dataset. Cosine Similarity measure was used to compare sense vectors:

$$Cosine\_Similarity(w_1, w_2) = \frac{w_1 \cdot w_2}{||w_1||||w_2||}$$

For each word pair in WordSim-353, the cosine similarity is computed between all the sense vectors of every word and the final similarity score is the one with the maximum cosine similarity. The Spearman correlation was used to quantify the system's performance against the human-chosen values.

5 Results

Appendix A.2, Table 1 shows the results of all 27 training experiments on the WordSim-353 similarity dataset. The Spearman correlation performance is shown in the right-most column.
The best result was obtained using a window size of 10, this is possibly due to the fact that **EuroSense** contains the proceedings of the European Union and most sentences are quite long and the larger the window, the more context is provided to the network for a given target word / sense. The best results obtained in this implementation is almost half of the SOTA obtained by SENSEMBED (Navigli, 2015). Due to lack of computing power, this implementation was not able to use more data, which would have surely improved the performance and doubled the Spearman correlation.

A Appendix

A.1 Processing

Before: Alphonsius (**Fons**) **van de Vijver** (born 1952) *is a* **Dutch** psychologist . *He* received *both his MA and his* PhD *in* Psychology *in* 1991 at the **Tilburg University** .

After: _ alphonsius **fons**_bn:14301494n **van_de_vijver**_bn:14301494n born **dutch**_bn:00101908a psychologist received phd psychology **tilburg_university**_bn:00188738n _

Figure 2: Example sentence before and after Processing. **Bold** words are the sense-annotated ones and their **lemma**_synset replacements. Punctuation symbols that are removed as well as the white spaces that the algorithm inserts in the beginning and end of the sentence are underlined. Removed stop-words are shown in Italics. Note how having removed the punctuation from the stop-word "'ma" (I am going to), the algorithm incorrectly replaces "MA" (Masters Degree). This was an intended side effect of the algorithm that was intentionally left; In our view, it does not harm the obtained sense embeddings as it removes non-annotated short words that resemble stop-words.

A.2 Grid Search Algorithm

Window Size	Embedding Size			Learning rate	Spearman Correlation		
5	300	350	400	0.02	0.2695	0.2845	0.2687
	300	350	400	0.03	0.2854	0.2833	0.2916
	300	350	400	0.04	0.2842	0.2834	0.2860
7	300	350	400	0.02	0.2784	0.2830	0.2876
	300	350	400	0.03	0.2885	0.2836	0.2931
	300	350	400	0.04	0.2804	0.2889	0.2839
10	300	**350**	400	**0.02**	0.2884	**0.3060**	0.2996
	300	350	400	0.03	0.3002	0.3016	0.3041
	300	350	400	0.04	0.2949	0.3030	0.2955

Table 1: Spearman correlation on WordSim-353 after each experiment of the Grid Search Algorithm

Window Size	Embeddings Size	Learning Rate	Spearman Correlation
10	350	0.02	0.3060

Table 2: Best Settings and Hyper-parameters of the Network

A.3 Results

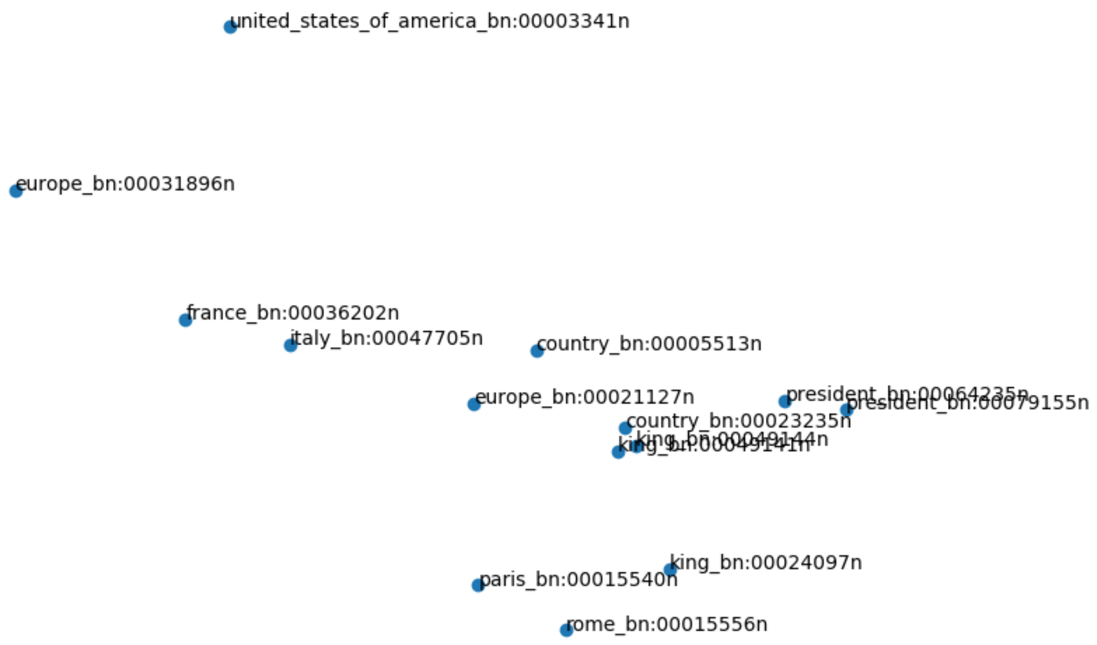

Figure 3: Principal Component Reduction (PCA) was used to reduce the dimensions of the sense embeddings to 2. The figure shows some sense of a "Political" nature / genre. It is clear that the senses for **country as "a particular geographic region"** and specific countries like **France**, **Italy**, and **Europe as "an organization of European countries"** are close together and are within almost a fixed distance from their capital cities **Paris** and **Rome** respectively. The senses for **King as "a ruler of a Kingdom"** and **President as "an executive officer of a firm or corporation" and as "ruler of the United States"** are close together and also close to the entities that these rulers rule, namely, countries and institutions. This shows that the network does indeed use distributional semantics to learn different representations for different senses depending on the context.