Learning Sense Embeddings

Michał Ostyk-Narbutt (1854051)
Prof. Roberto Navigli
Natural Language Processing Homework 2

June 6, 2019

SAPIENZA
UNIVERSITÀ DI ROMA

Contents

1 Introduction

This report tries to present the process of learning sense embeddings from the annotated EuroSense corpus to be used on a word similarity task.

2 Model and Approach

2.1 Dataset and tools

The dataset used was the high precision version of the Eurosense corpus [1]. It consists of a large `XML` file which consists of many multi-lingual sentences taken from the European Union parliament. All the processing steps used in this project were performed in Python 3, with he help of key external libraries. These were `Gensim` used for model training, `iterparse` used for parsing the large corpus, and `Scikit-Learn` for dimentionality. The code file structure can be found in the `README.md` file of the Gitlab repository [4].

2.2 Parsing

The parsing consisted of itertively reading the large corpus file, and processing it line by line. Using special tags, sentences were extracted along with their annotations. However, for the purposes of this assignment extraction was performed only for the English language. Each *sentence*, consisted of several *texts* for the English Language. However, not every *text* had corresponding anchored annotations. Hence, these texts not containing any *anchors* were discarded. Conversely, due to inconsistencies in the dataset, many sentences which had English *anchor* annotations but lacked *texts* were also discarded. Moreover annotations in this corpus file contain a certain ID called *synsetID* which was specified in BabelNet. In this task, only annotations which had a corresponding *synsetID* in WordNet were considered. Stop words were **not** removed, but certain punctuation marks were as they conflicted with some anchors. For example: "European Union-Ukraine Agreement", were there was an anchor for "European Union". Additionally prior to writing the parsed sentences to file, they were lowered as not to have duplicates of senses ie.: "**l**emma_synsetID" and "**L**emma_synsetID". In total, 1.87 million sentences were extracted from the dataset. Figure 1 portrays a histogram of annotation occurrences in each parsed sentence. The median number of annotation was **5**. This could be an indicator of how word senses affect each other depending on the number of annotations. In total, 1.87 million sentences were extracted from the dataset.

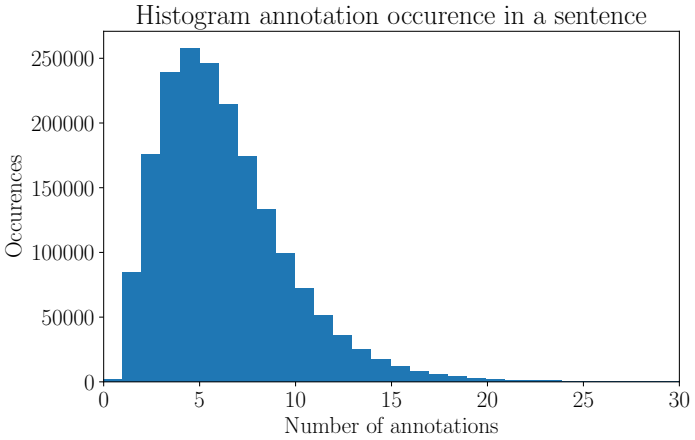

Figure 1: Histogram based on the number of annoations in each of the 1.87 m extracted sentences

2.3 Training

The training was performed using the the Word2Vec model from `gensim`. At first, only the *CBOW* version was used, but at the end *SKIPGRAM* was implemented for comparison.

2.3.1 Scoring: Similarity measure

A key target of training the model, was to evaluate it on a similarity measurement task. For this, I utilised cosine similarity between words. Then using Spearman correlation, I tested the results against a gold, human-annotated dataset. I did not incorporate any other similarity measures.

2.3.2 Hyperparmeters

The key hyperparamters explored when training the model were: α – learning rate, min count – minimum count of word occurrence in the vocabulary (of the parsed dataset), window, and the embedding size. These hyperparameters were explored using grid search in order to maximize the correlation score. As for for the grid search, I implemented a `Word2VecModel` class, which resembles a `scikit-learn` estimator. It consists of a `fit` function which builds the model, given a set of hyperparameters, then builds the vocabulary, and proceeds to train it on a set number of epochs (50). Then using `sklearn.model.selection.ParameterGrid`, I iteratively perform a grid seach of the best hyperparameters based on the cosine similarity score performance.

Table 1: Table of hyperparemters used in the grid search with best ones in bold

window	4	**5**
min count	1	**3**
embedding size	**100**	300
α	0.001	**0.009**
Negative	**13**	
Epochs	**50**	

3 Results

Since *CBOW* works several times faster to train than skipgram, it was used for the grid search. Grid search was first performed on large number of combinations of hyperparamters, on 15 epochs. However, after noticing significant discrepancies in the correlation score, a selected few were selected to be trained in the final grid search. The hyperparamters are presented in Table 1, and the ones in bold refer to the best result of correlation = 0.27 (using *CBOW*). Using these best hyperapameters, a *SKIPGRAM* version of the word2vec model was also trained and resulted in a slightly worse of correlation = 0.26. The difference in performance seems consistent with the fact that *CBOW* has slightly better accuracy for the frequent words, since we used a min count of 3 but *SKIPGRAM* works better with rare phrases. A more graphical approach to visualizing the effect of hyperparamters on each other, and the ones which have the most effect on the final score is presented in Figure 2. The figure presents a swarm plot of the correlation score in regards to the hyperparamters.

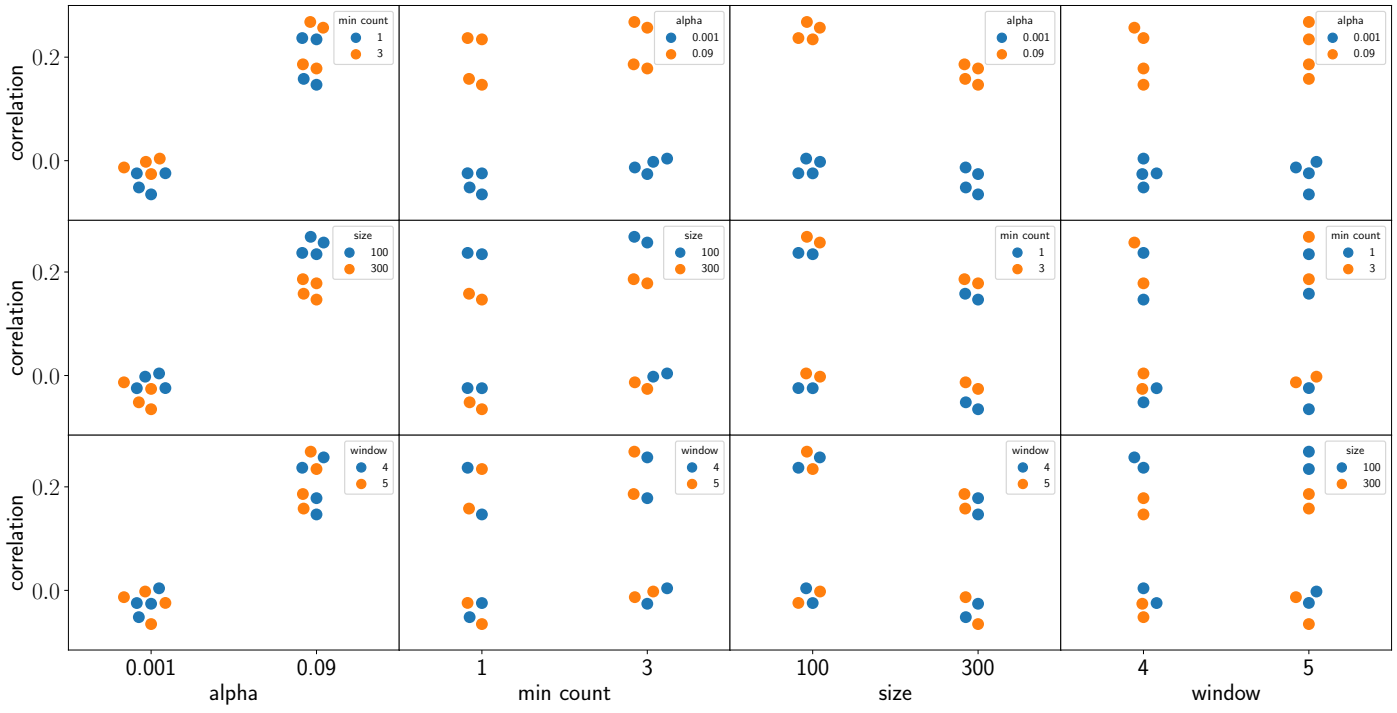

Figure 2: Grid search of the best hyperparamters based on Table 1. Visualization of the Correlation score vs each parameter depending on several others.

4 Discussion

4.1 General

From Figure 2, we can infer that alpha α, has the most significant impact on the final correlation score, as regardless of the comparing hyperparamter, $\alpha = 0.09$ always yields a better correlation score. Moreover, the difference is so substantial that for $\alpha = 0.001$ correlation scores swarm around 0, which would infer almost the lack of correlation. Additionally, we can see that the size of the embedding also plays a major role, as smaller (100) embeddings perform much better than larger (300). This is a consequence of the size of the vocabulary which is effected both by the min count parameter, but also more importantly by the contents of the corpus.

The corpus is very domain specific which influences correlation scores, with its context and also results in many out-of-vocabulary (OOV) words. A possible solution would be to simply add data from a wider domain, including non factual texts like novels. This however is not easy, as finding enough annotated data is quite difficult but could be partially overcome by combining the already made synsets and analysing them in different ways. On the other hand, it would be far safer to use a good corpus for a more wider domain, i.e: the Semantically Enriched Wikipedia (SEW) [2] corpus.

Furthermore, the overall performance of the model could be yet improved by considering the rising problem of sparsity in the data. One key driving factor of sparsity is the reduction of polysemy which yields a large number of synsets. To overcome this, one could possibly reduce the number of senses for a given lemma but different synset.

4.2 Embedding visualization

To further understand what kind of embeddings `word2vec` produces, embedding visualizations were performed. To do this, I reduced their dimensionality from $size = 100$ to 2D and 3D using the `sklearn` implementation of Principal Component Analysis (PCA) followed by T-distributed Stochastic Neighbor Embedding (t-SNE) and removed synset IDs for plotting purposes.

Figure 3 depicts a 2D plot of a few chosen word embeddings (legend) and their top 30 most similar words according to the model. From the depicted 4 words we can deduce that they are some are more closely matched than others. Whereas Figure 4 portrays a 3D visualization of all the 44 thousand words in the vocabulary of the model. The plot is basically just one giant cluster in 3D symbolizing that on a larger scale, how much of a domain-specific dataset Eurosense really is.

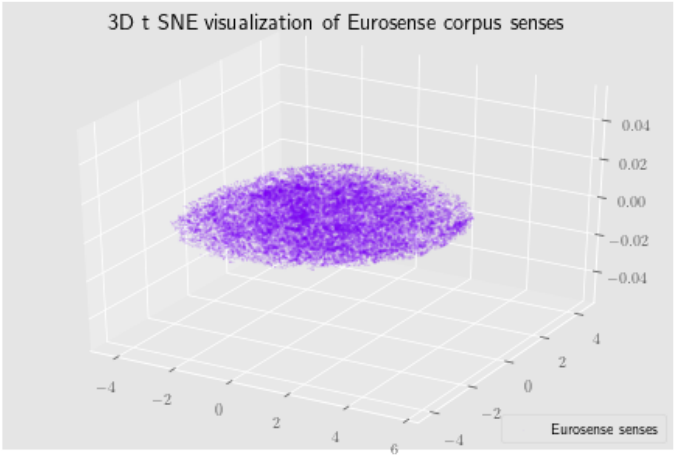

Figure 4: 3D T-SNE on the entire vocabulary of the model. *Perplexity* = 3 and *n_iter* = 300. First reduced by PCA to 10 dimensions then applied T-SNE on 10 nearesst neighbours. Plots inspired by [3]

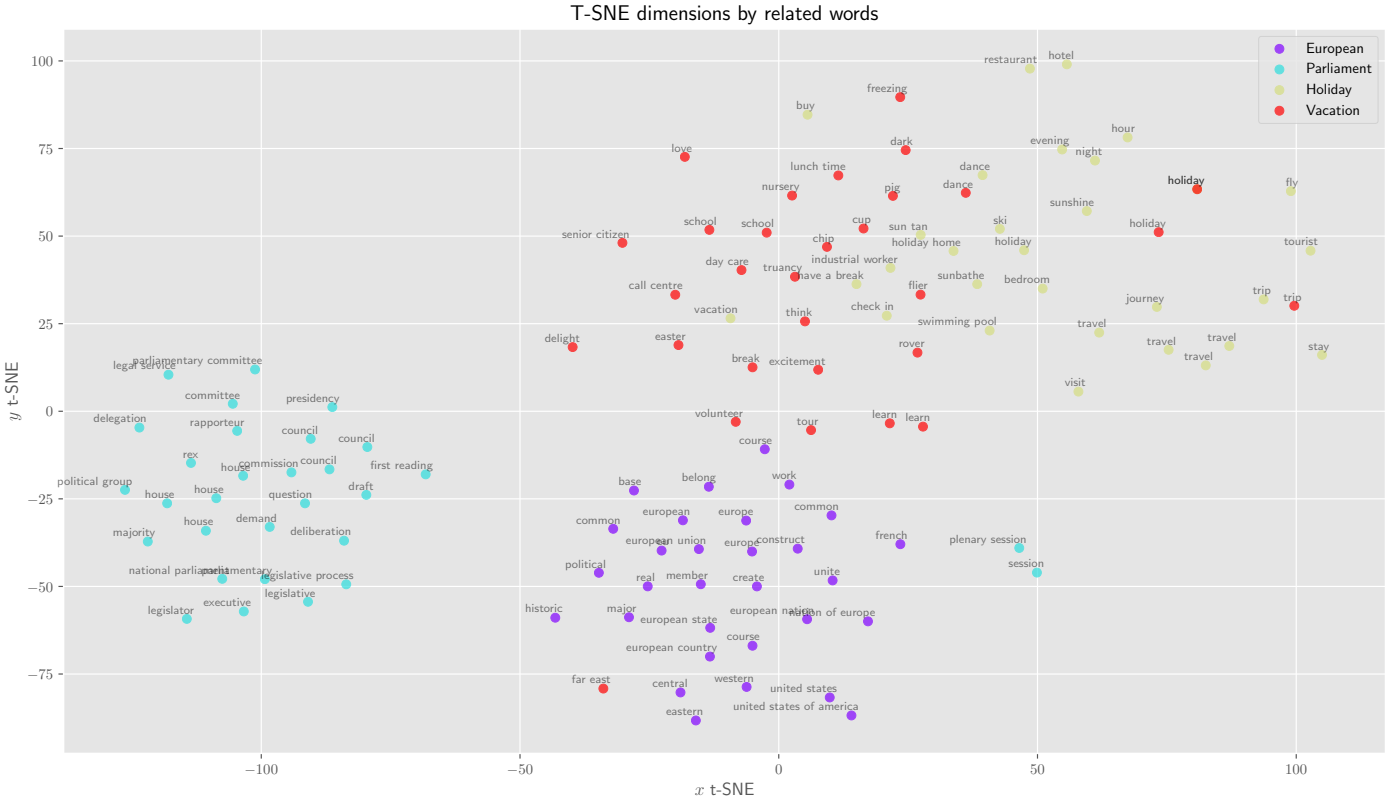

Figure 3: T-SNE visualization of the top 30 similar words to 'European' , 'Parliament', 'Holiday', and 'Vacation' . T-SNE with $perplexity = 20$ and $n\_iter = 5000$. Plots inspired by [3]

5 Conclusion

The best model, when stacked on a word similarity task yielded a correlation score of = 0.27. This score, could possibly be futher improved by a couple of factors. Namely, adding more data from more general and less domain dependent datasets. However, this arising issue of sparsity would also have to addressed in that case. Moreover, in the task similarity task, the OOV words significantly decreased the correlation score as there were OOV words in 48 out of the 353 pairs of words analyzed. To conclude, sense embeddings are an effective way of dealing with polysemy. In this approach we created sense embeddings though anchoring specific synsets to lemma's of words in text. The world similarity measurement which compares our results with manual human scoring should yield far better results, as almost always the case in Deep Learning, with more and better data.

References

[1] Claudio Delli Bovi, José Camacho Collados, Alessandro Raganato and Roberto Navigli. EuroSense: Automatic Harvesting of Multilingual Sense Annotations from Parallel Text. Proceedings of 55th annual meeting of the Association for Computational Linguistics (ACL 2017), pages 594–600, Vancouver, Canada, 30 July-4 August 2017. `http://lcl.uniroma1.it/eurosense/`

[2] Alessandro Raganato, Claudio Delli Bovi and Roberto Navigli. Automatic Construction and Evaluation of a Large Semantically Enriched Wikipedia. Proceedings of 25th International Joint Conference on Artificial Intelligence (IJCAI-16), pages 2894–2900, New York City, New York, USA, 9-15 July 2016. `http://lcl.uniroma1.it/sew/`

[3] `https://github.com/sismetanin/word2vec-tsne`

[4] `https://gitlab.com/Ostyk/michal_ostyknarbutt_1854051_nlp19hw2`