Sense Embeddings

Natural Language Processing, Homework 02

Ahmed El Sheikh
1873337

1. Abstract

The aim is to create Sense embeddings from Euro Sense corpus, to find similarity between 2 words contextual meaning, by making a model learn from sense annotated corpus. Sense Embeddings is to try to know the different contextual meanings of the words.

2. Data Loading

Starting with parsing High Precision Euro Sense dataset -which is an automatically annotated dataset, for every sentence each word is correlated with a sense, which is represented as set of characters and numbers known as "Synset"- resulting in approx. 1.9M -only English tagged- sentences, extending the data with train-O-matic dataset resulting in approx. 1.4K -only English tagged- sentences, as well as, using 41.3M sentences from S.E.W., resulting in total of approx. 43 million sentences used for training.

In all datasets a lot of sentences were either misannotated, or empty text tags, or empty annotation tags, so text tags with empty text were omitted, if text tag is not empty and annotations' tags were empty, sentence was added in normal format not in lemma_synset format.

Parsing SEW dataset was too big for the RAM to fit in, so, every 10K files parsed and processed were saved to a txt file, after 1st 10k files, every interval (10k files) was appended to this file.

3. Data Preprocessing

Before feeding the model the data, we need to clean the data and make sure it has a certain format -list of lists-, hence, I started with lowering all the characters, removing punctuation, removing stop words, and numbers, finally, stripping the sentence and removing multiple consecutive whitespaces. Afterwards, the sentence was tokenized.

4. Sense Annotations

Before annotating the tokens in every sentence, I made sure in EuroSense dataset to have the longest anchor (e.g. European Union was replaced with European_union_bn: xxxxxxx) in order not to lose contextual meaning. To replace by longest anchor, it was needed to get annotations and sort them based on anchor length then substitute the anchor with corresponding lemma_sysnetId. Same was done when replacing SEW mention (equivalent to anchor in Euro Sense). Using only the BabelNet synsets found in the provided mapping file, if synset was not present in this file, the word was not replaced by lemma_synset. For all datasets parsed, only the senses present in the provided file were used, to reduce embeddings matrix dimensionality. Parsing the train-o-matic dataset was simple, as well, following same steps as parsing EuroSense dataset.

Annotated sentence example: ['let', 'list', 'recent', 'headline_bn:00043320n']

5. Model

The model used was CBOW word2vec (provided by genism model), passing sentences in form of list of lists. Using default hyper parameters at first, then tuning the following params (window, alpha, sample, negative) using grid search, refer to table 3 for the parameters used. CBOW architecture predicts a word given a context within a specific meaning, which is composed of an input layer -expected to receive the sentence without the target word, followed by, a hidden layer, and at the end we have output layer, expecting to output the target word.

To train the model on such a huge dataset of 43M sentences, model was trained on batches of 5M sentences each. Train the model on 5M sentences and evaluate it, then rebuild model vocab and training it the remaining n times. After training model was saved in "embeddings.vec" containing only sense embeddings (word embeddings were removed).

Evaluation of the model was done using 2 measures, Cosine Similarity and Weighted Cosine Similarity, followed by calculating of Spearman Correlation between gold scores and computed ones, using word-similarity-353 'combined.tab' Same word might have more than 1 different contextual meaning, as demonstrate in table 4.

6. Grid Search

A grid search was implemented to tune the parameters of the model on params (window, alpha, sample, negative), as mentioned earlier, trying to reach the best results. Results of parameters tuning in Table 3, and plots of figure 1 is to show the correlation between parameters and model performance.

7. Experiments

Maintaining seed to make sure that the experiments are done properly.

1. In tables 1 and 2, it is demonstrated the effects of cleaning text, as well as the effects of window & sampling rate. After cleaning text, time as per training the model was smaller, however, data processing time was longer
2. Number of epochs (proportional to performance), longer training time, hence, better results
3. size of embeddings (proportional to performance), the dimensionality represents total number of features, the more (up to a scale), the better
4. Amount of data (proportional to performance)
5. Skip-gram and CBOW were used, skip-gram took x4 times more to train, produced worse results. [standalone experiment]
6. Negative sampling and Hierarchical SoftMax (hs) were used, hs is more computationally effective moving from O(vocab_size) to O(log(vocab_size)) producing better results. [standalone experiment]

As noticed the correlation score improves as per sampling decrease, and as window size increase, and cleaning of the text. Needless, to mention, that increasing number of epochs, increases the score, as well as, performance increases when using Hierarchical SoftMax, as it uses a tree of sigmoid nodes instead of one big SoftMax. Huffman coding ensures that the distribution of data points belonging to each side of any sigmoid node is balanced. Therefore, it helps eliminate the preference towards frequent categories comparing with using one big SoftMax and negative sampling.

CBOW architecture goal is to predict a sense using its context. At the end, the model's only hidden layer activation function was hierarchical SoftMax instead of SoftMax.

Hierarchical SoftMax provides an alternative model for the conditional distribution $P(.|context)$ such that the number of parameters upon which a single outcome $P(word|context)$ depends is only proportional to the logarithm of $|word|$.

8. Embeddings Analysis

Working with such very high dimensional data, visualization of 400-dimensional vectors is difficult. Visualization of the data is needed to understand the relationships, so, the number of features must be reduced before plotting it, so I used t-SNE, which is a non-linear technique for dim reduction. I show my plots on the 7 keys and their most similar 5 words.

9. Results & Discussion

The best **correlation score** I was able to get was **0.5954** this was earned after several experiments and testing how model is affected by removing words, lowering all letters, removing numbers, and punctuation. The cleaner the data was the better the correlation score. Added to the increase of the contextual window in genism model resulted in better scores, same for negative sampling. Hyperparameters tuning did not give the expected results, I suggest this is due to that European Parliament documents are very domain specific, so I used SEW dataset for further development of my model and this actually helped me increase my score from 0.3348 on Euro sense only to 0.5954 as mentioned.

Plots & Tables

PARAMETERS (EPOCHS, WINDOW, SAMPLE)	CORRELATION SCORE	
	Before	**After**
30, 5, 1E-2	0.1852	0.2044
30, 5, 1E-3	0.1969	0.2420
30, 5, 1E-4	0.2030	0.2506

Table 1: Effects of window size & sampling rate without text cleaning

PARAMETERS (EPOCHS, WINDOW, SAMPLE)	CORRELATION SCORE	
	Before	**After**
30, 10, 1E-2	0.1977	0.2044
30, 10, 1E-3	0.1978	0.2420
30, 10, 1E-4	0.2190	0.2506

Table 2: Effects of window size & sampling rate with text cleaning

PARAMETER	VALUE
ALPHA	0.001
NEGATIVE	5
SG (CBOW)	0 (True)
SAMPLE	0.0001
WINDOW	10
HEIRARCHICAL SOFTMAX	1
CORRELATION SCORE	0.5954

Table 3: Best parameters found.

Bank as riverbank(bank_bn:00008363n)	*Tanimoto measure*	*Bank as money Bank (bank_bn:00008364n)*	*Tanimoto measure*
interest_rate_bn:00047085n	0.1968	interest_rate_bn:00047085n	0.9997
customer_bn:00019763n	0.1591	customer_bn:00019763n	0.9517
low_bn:00106206a	0.2172	low_bn:00106206a	0.9189
household_bn:00032892n	0.2368	household_bn:00032892n	0.8957

Table 4: Tanimoto measure between a word with 2 diff contextual meanings

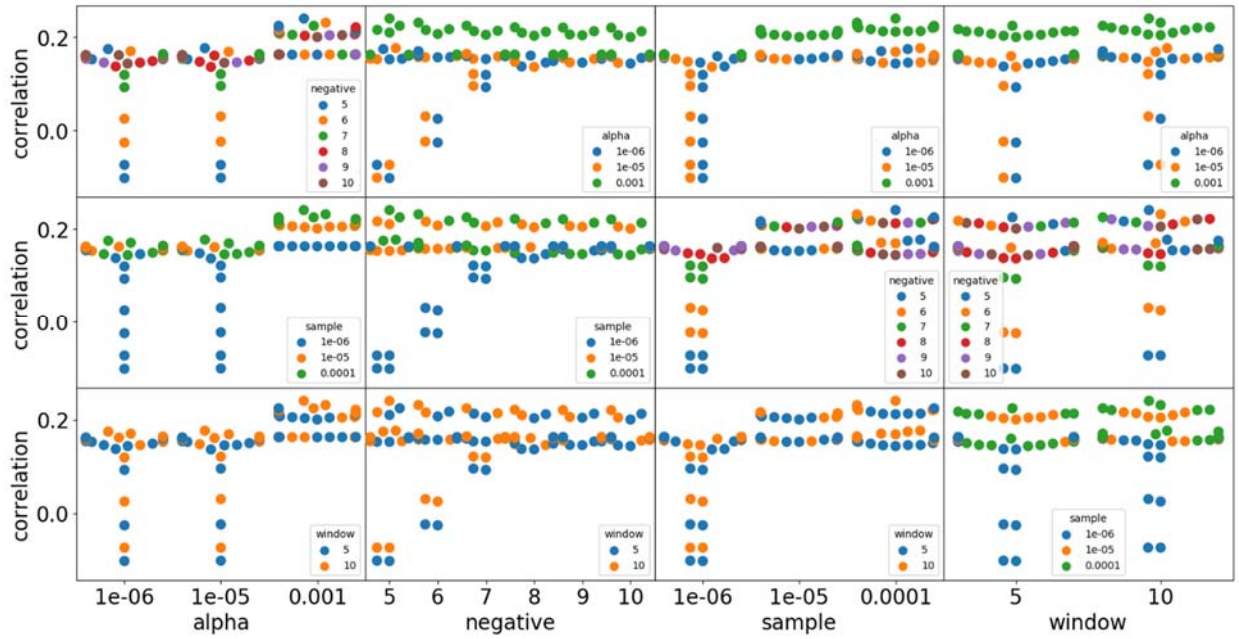

Figure 1: Effect of different parameters on our model

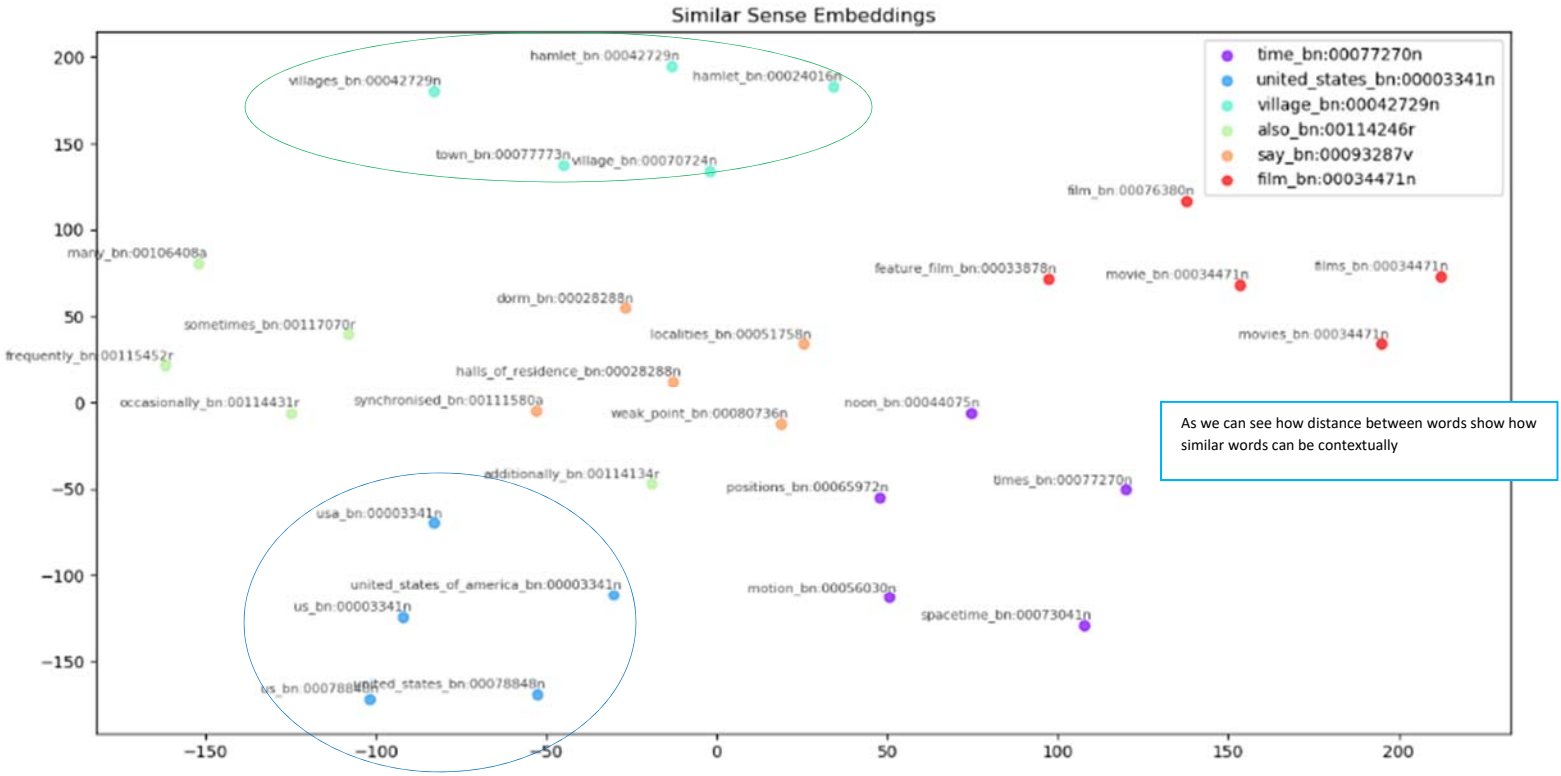

Figure 2: Visualization of sense embeddings, with 2 circles around only 2 keys to demonstrate how simple words are too close together compared to the rest