Implementation of word2vec and Experiments

Lucky, lucky1@iisc.ac.in

Abstract

In this assignment, we are assigned with two tasks and one bonus task. First task is to implement a word2vec (skipgram) model from scratch. For it we are supoosed to use Reuters Corpus. After that task 2 is to capture relation between words using analogical reasoning task. Bonus task is to find any kind of bias present in embeddings learned in task 1.

1 Task 1

First task is to implement word2vec skipgram model (Mikolov et al., 2013).

1.1 Exploring Dataset

We are considering 'Reuters' Corpus. The Reuters Corpus contains 10,788 news documents totaling 1.3 million words. The documents have been classified into 90 topics, and grouped into two sets, called "training" and "test".

categories	file_count
earn	3964
acq	2369
money-fx	717
grain	582
crude	578
trade	485
interest	478
ship	286
wheat	283
corn	237
dlr	175
money-supply	174
oilseed	171
sugar	162
coffee	139
gnp	136

Figure 1: Distribution of categories in the corpus. Only few are shown in the image.

We will be using the same train-test split as provided int the corpus itself. We have 7769 training documents and 3019 test documents. Following operations are done in preprocssing the dataset:

1. Removing everything except english alphabet.
2. Removed too frequent words.
3. Converting everything to lower case.
4. Removing stopwords taken from nltk package.
5. Lemmatization.

We have used a python script for all of these operations already available.

1.2 Training

Initially We have mapped every word to an index and after that made an *one-hot vector* for all words in training data. After that training data is been prepared such that, from training corpus surrounding few words are taken as labels for each word. This process took around 34 minutes.

The loss function used is *Noise Contrastive Estimation loss* implemented by ourselves. Optimization function used is Stochastic Gradient Descent (SGD). Negative sampling is used to reduce training time. To sample out negative samples following formula is used:

$$p(w_i) = \frac{f(w_i)^{\frac{3}{4}}}{\sum_{j=0}^{n} f(w_j)^{\frac{3}{4}}}$$

1.3 Evaluation and Experiment settings

For evaluation we are supposed to use word-similarity task (Hill et al., 2014). SimLex999 dataset contains word-pairs and their comparisons on few metrics. We have extracted these word pairs and their SimLex999 score, and on these

words if they are present in Reuters Corpus calculated cosine similarity. After this, We have calculated correlation between these two measures. This is the final evaluation that we have used for the embeddings.

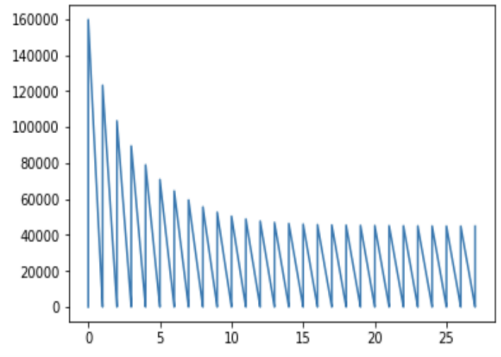

Figure 2: Plot of Epoch vs Loss

The best *pearson correlation* achieved is 0.34. The score is not descent probably because the training is done for less time due to some complications in the setting for running it.

	word1	word2	SimLex999	cosine
0	old	new	0.141066	0.015370
99	recent	new	0.712644	0.005441
115	south	north	0.205852	0.062724
171	tax	income	0.224660	-0.123312
195	business	industry	0.709509	-0.086094
235	area	zone	0.846395	0.013342
236	business	company	0.918495	-0.047260
239	money	capital	0.672936	-0.031491
270	strength	might	0.714734	-0.043502
311	disc	computer	0.310345	-0.043066
340	bean	coffee	0.514107	-0.016539

Figure 3: Word-pairs and their SimLex-999 and cosine similarity score

We have trained the model for 28 *epochs* and *window size* is 3. We have considered 5 *negative samples* and *embedding size* is 300.

2 Task 2

In task 2, we are supposed to verify the claims in the word2vec paper about capturing relationships between words through the analogical reasoning task. For that we are using a text file *questions-words.txt*.

```
[['houston', 'texas', 'minneapolis', 'minnesota'],
 ['high', 'higher', 'low', 'lower'],
 ['low', 'lower', 'high', 'higher'],
 ['good', 'best', 'strong', 'strongest'],
 ['strong', 'strongest', 'good', 'best'],
 ['france', 'french', 'japan', 'japanese'],
```

Figure 4: Examples of few question-answer pairs from Reuters Corpus in form of data in question-words.txt.

Accuracy achieved in this task is 40.6%. For few words it gave correct answers like, examples shown above but for many it was not mapping to incorrect words.

3 Bonus Task

Bonus task is to identify biases in the learnt word embeddings. For example, there may be a bias indicating man is generally doctor, programmer and a woman is homemaker or nurse. As If we search for words near to *new*, then word year appears, but if we search for words near *old* year does not appear. This might happen because *new year* is much more frequent then *old year*. For example, following are few examples:

```
zone 0.11758183176420586
zealand 0.09288599226462192
z 0.06988215697843438
york 0.07223962921018603
yield -0.04104659694571564
yet 0.09314488980754808
yesterday -0.015849778229978986
years 0.07249966249275344
yearend -0.07082143384103291
year 0.021367378260173795
```

Figure 5: Words near to word *new*

3.1 Few plots showing visualtion of embeddings

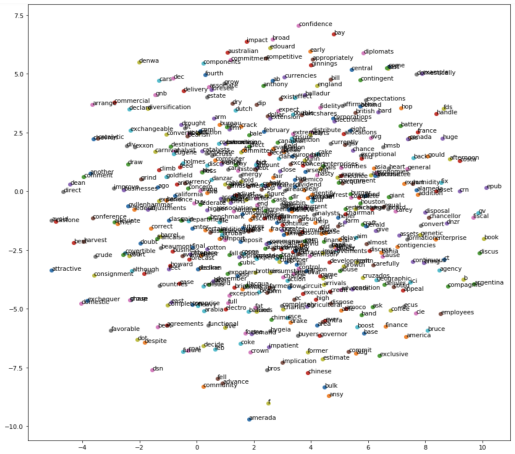

Figure 6: Target word Embeddings

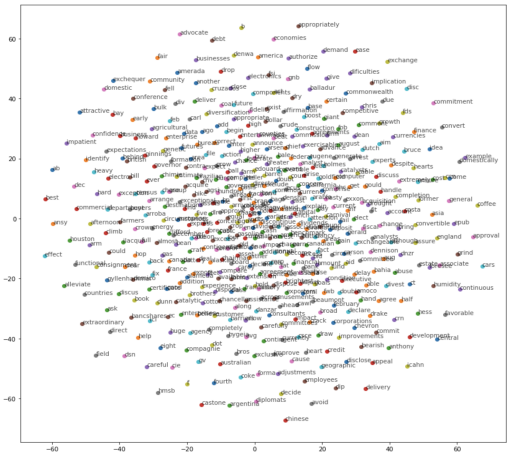

Figure 7: Context word Embeddings

4 Link to github code

This is the link for python code of implementation.

References

Felix Hill, Roi Reichart, and Anna Korhonen. 2014. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *CoRR*, abs/1408.3456.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546.