Natural Language Understanding
Assignment 1

Ponsuganth Ilangovan P
ponsuganthp@iisc.ac.in

1 Problem Statement

To train word embeddings by implementing a word2vec skipgram model (Mikolov et al., 2013) with the Reuters corpus from nltk package. Also word vectors are to be evaluated with the Simlex 999 (Hill et al., 2015) similarity task. An attempt to identify biases in the words embeddings is also made as mentioned in task 2.

2 Methodology

In word2vec each word is associated with a vector in n dimensional space and the similarity and differences between words is captured with the vectors learned form a very large corpus. The cosine between two vectors gives the similarity between those words. These vectors are learned from a very large corpus by likelihood maximization of the corresponding dot products of the vectors that are nearer to each other in the corpus and for this reason we will have to fix a word window which tells the algorithm how many word pairs it has to consider form the corpus. In the vanilla skipgram model we take the dot products of the corresponding word vectors and do a softmax it to compute the probability for that particular word in the pair. The probability of the word pair is given by,

$$p(w_k|w_j) = \frac{exp(c_k \cdot v_j)}{\sum_{i \in |V|} exp(c_i \cdot v_j)}$$

This model maximizes this probability. But the softmax is computed for the hwole vocabulary and for this reason the time taken for training will be high. To optimize we incorporate the negative sampling method in which we try to maximize association between the word pair in our window and also we disassociate a random sample of words from the corpus. The loss function in this case will be,

$$loss = log\sigma(c \cdot w) + \sum_{i=1}^{k} log\sigma(-w_i \cdot w)$$

Maximizing negative of sigmoid is equivalent to minimizing the probability. Also as mentioned in the word2vec official paper the performance is good if the distribution with which the words are sampled for negative sampling follows the unigram distribution raised to the three fourths. In that way words that occur many times in the corpus are chosen for disassociation so that the semantics of the words are captured in a better way.

3 Implementation

The model is implement with the deep learning library TensorFlow. Word2Vec is implemented as a neural network whose weights are the word vectors stacked up as a matrix. For every word pairs the center word embeddings and context word embeddings are fetched from the weights matrix by an embedding lookup call. Then the vectors are dotted directly in the case with negative sampling or the center embedding is multiplied with the whole context embedding matrix and softmaxed out to get the probabilities with each word pair with the center word in the case of vanilla skipgram model. The corresponding word pair probability is maximized and the negative log of that probability is defined as the loss function. The optimizer used was gradient descent optimizer which is available in the tensorflow package. In the case of skipgram model with negative sampling, the word pair probability is maximized and the similarity with some random samples from the corpus is minimized.

4 Experiments

The Reuters corpus from the nltk package consists over fifty thousand sentences. After prepossessing of the corpus such as removal of stop words, punctuation, etc the total number of words in the corpus is about one million. The words are lower cased and unique words are found which comes to about thirty thousand which gives us our vocabulary.

4.1 Vanilla Skipgram Model

Experiments for this model were with embedding size 50 and 100 and word window size of 3 (one on both sides of the center word), 5 (2 on both sides), 7 and 9. The experiments for skipgram were run for 8 epochs.

4.2 Negative Sampling

Skipgram with negative sampling was experimented with window sizes of 5, 7 and 11 and with embedding sizes of 50 and 100. The number of negative samples were varied as 16, 40 and 80. The experiments were run for 6 epochs and embeddings were stored for every 2 epochs.

5 Results

5.1 Task 1

The word vectors were trained with the corpus and a validation split was taken to find a good model from the set of models that were trained. The likelihood of the model on the validation dataset was calculated to choose the best model. The plots (Figure 1 and Figure 2) shows validation likelihood and spearman correlation score versus various models. To calculate the spearman correlation we make use of the Simlex-999 dataset. The word pairs are taken from the dataset and the cosine similarity is computed with the best model. The correlation between the cosine similarity and the simlex rating is calculated for comparison.

5.2 Task 2

The analogy task is done with the dataset from google word2vec and the questions-words text file in the dataset. Only the words in the vocabulary are considered and others are neglected. The first two word vectors are subtracted and added with the third word vector. the nearest neighbours for the resulting vector is found and if the fourth word in the dataset is in those nearest neighbours then a point is given to the model. Accuracy is calculated and the models are compared.

5.3 Other Results

The biases in the word embeddings were also explored in plots by the principal component analysis of the vectors. Some of the common biases such as the cities and countries (Figure 3), Days and Months (Figure 4) and subjects (Figure 5). These were done with the PCA of the word vectors.

References

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41.4:665-695.

Tomas Mikolov, Ilya Sutskever, Greg Corrado, Kai Chen, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*.

Model	Word Window	Embedding Size	Batch	No. Neg Samples	Likelihood	Correlation
1	5	50	128	40	0.4648	0.0756
2	5	50	64	40	0.3791	0.0323
3	5	50	64	80	0.3364	0.0523
4	5	50	128	80	0.428	0.0283
5	7	50	64	40	0.3707	0.0561
6	7	50	64	80	0.3223	0.0452
7	7	50	128	40	0.4702	0.0740
8	7	50	128	80	0.3997	0.0802
9	11	50	1	16	0.5188	0.0828
10	11	50	1	40	0.3565	0.0523
11	11	50	1	80	0.2806	0.0532
12	11	50	64	40	0.3174	0.0731
13	11	50	64	80	0.2508	0.0325
14	11	50	128	80	0.3532	0.0831
15	11	50	128	40	0.3834	0.0592
16	5	100	64	40	0.3679	0.0582
17	5	100	64	80	0.3198	0.0606
18	7	100	64	80	0.3145	0.0171
19	7	100	64	40	0.3674	0.0676
20	11	100	64	40	0.3301	0.0755
21	11	100	80	64	0.2533	0.0675

Table 1: Results of training

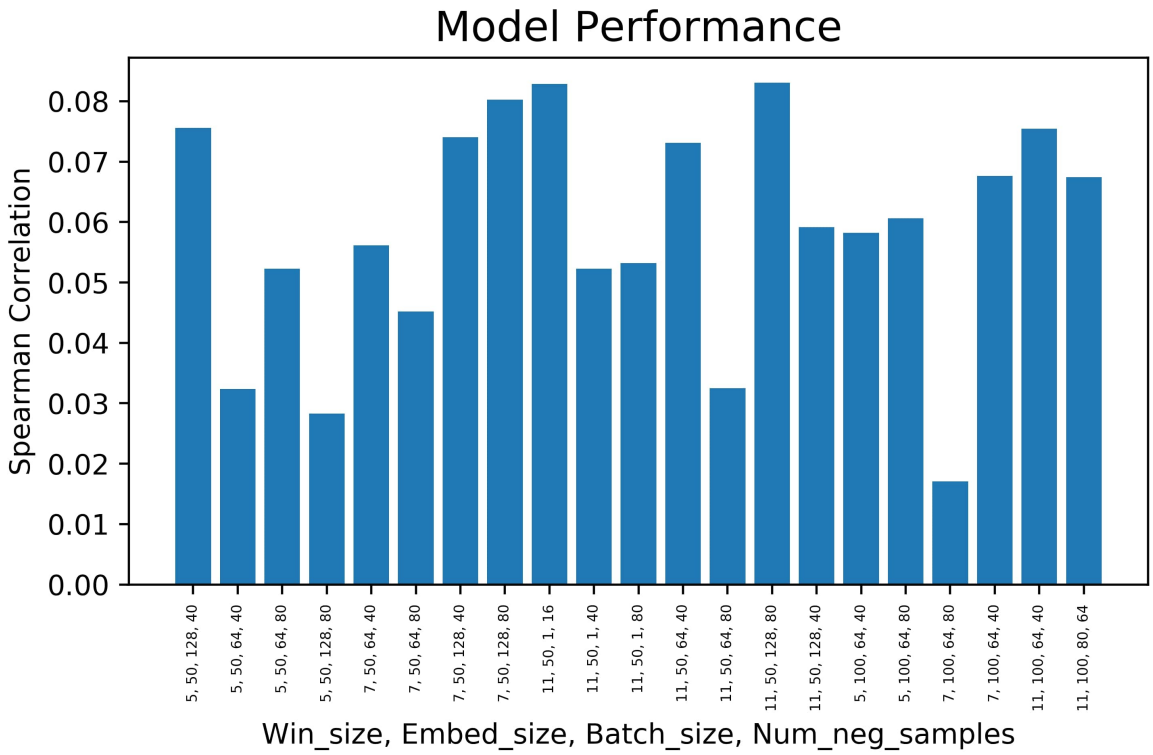

Figure 1: Spearman Correlation vs Models

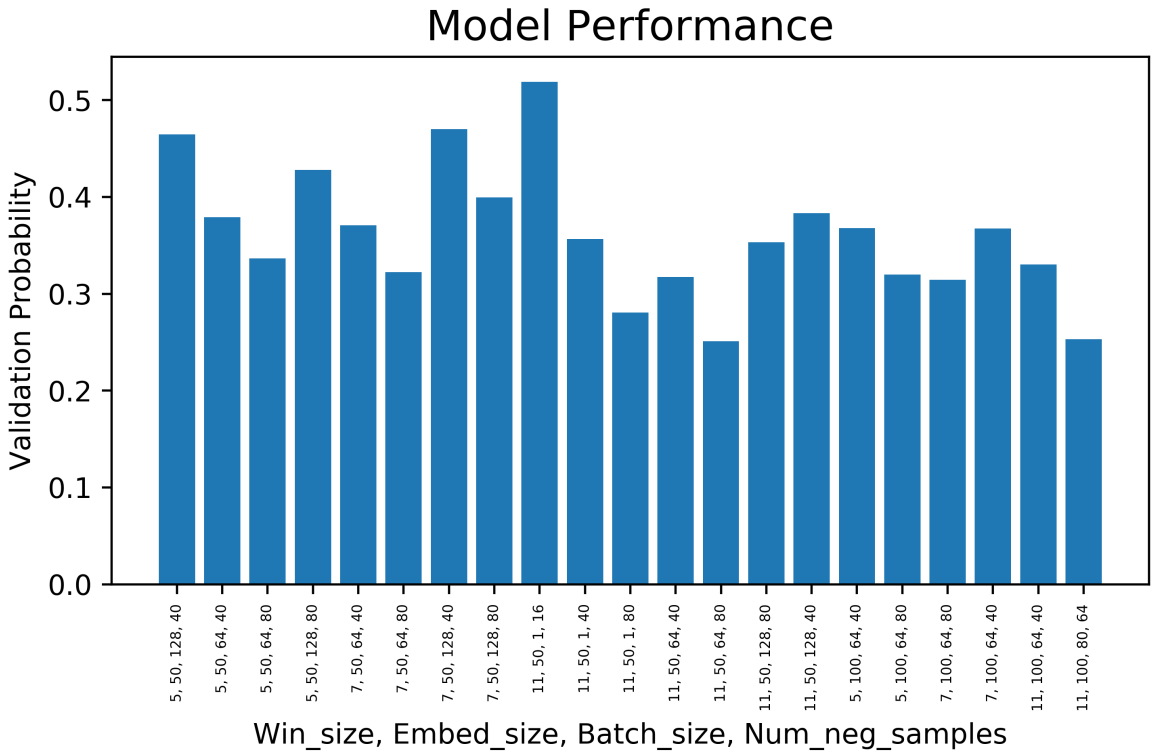

Figure 2: Validation Likelihood vs Models

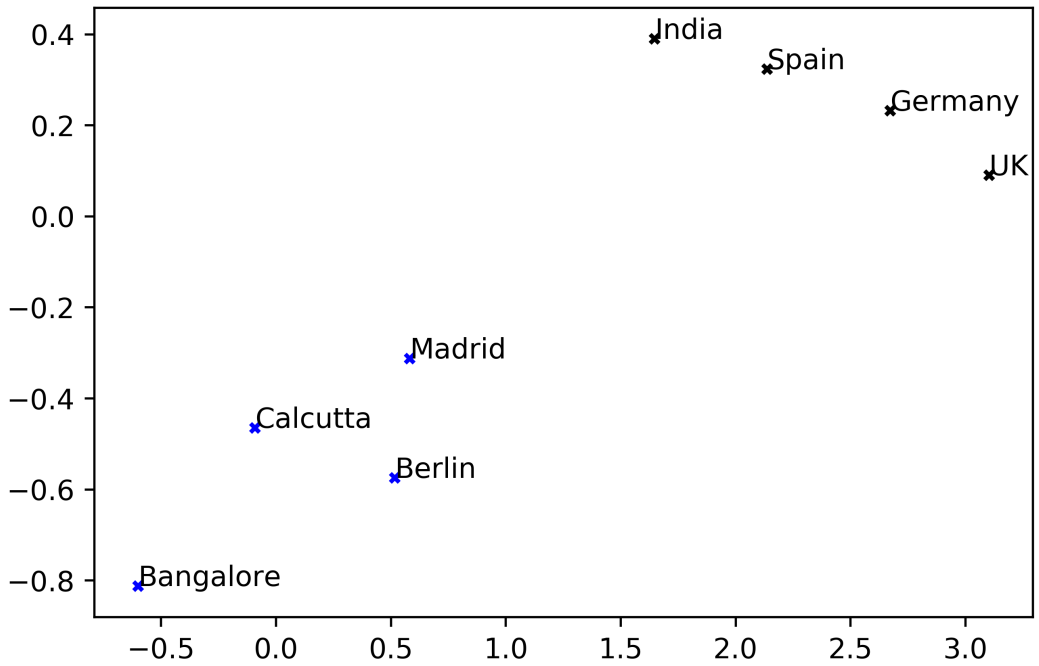

Figure 3: Countries and Cities

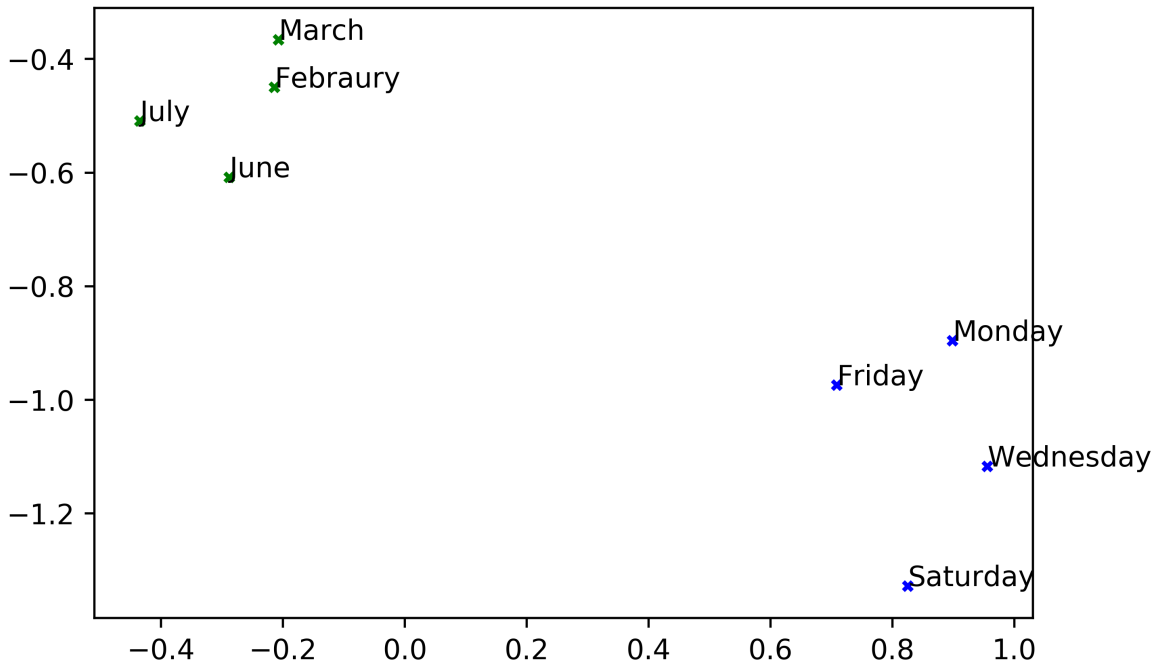

Figure 4: Days and Months

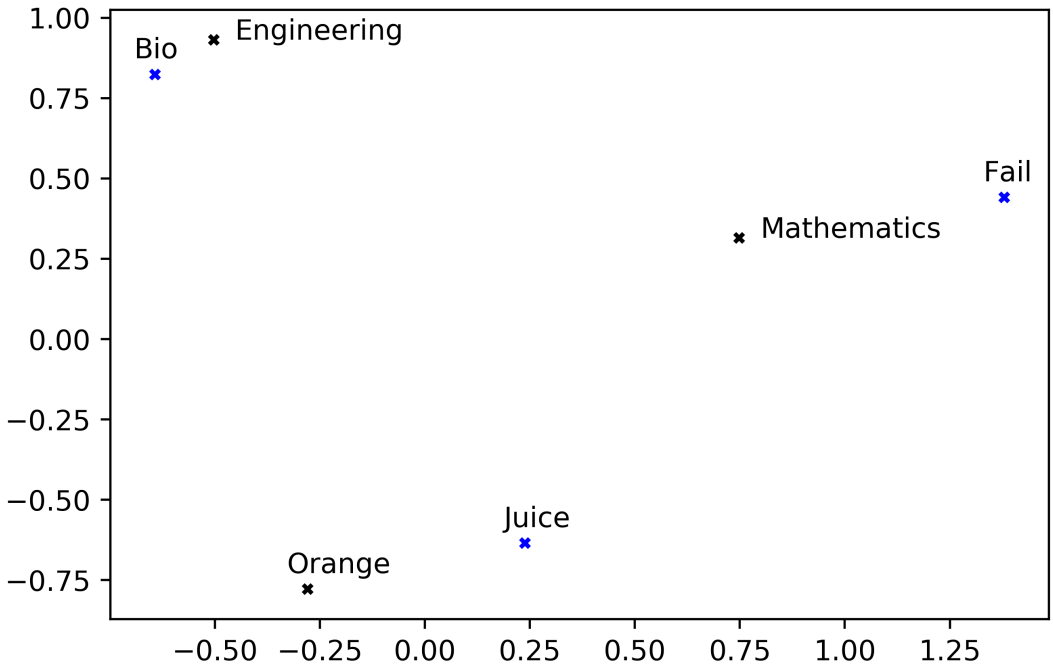

Figure 5: Subjects