NLU Assignment 1: Implementing and analyzing Word2Vec

Aditay Tripathi
IISc / Bengaluru
aditayt@iisc.ac.in

1 Word2Vec Model

Word2Vec(Mikolov et al., 2013) model is based on skipgram model(Cheng et al., 2006). Skipgram model tries to predict the surrounding word given the current word. Given a sequence of words $w_1, w_2, ..., w_T$, skipgram model maximizes the following objective:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq i\leq c}\log p(\frac{w_{t+i}}{w_t}), \tag{1}$$

where, c is the context window. $p(\frac{w_{t+i}}{w_t})$ is given by Softmax function:

$$p(\frac{w_i}{w_j}) = \frac{\exp(u_{w_i}^T v_{w_j})}{\sum_{w=1}^{W}\exp(u_w^T v_{w_j})}, \tag{2}$$

where, v_{w_j} is the current word embedding and u_w is the context embedding for the context word w. But to calculate the probability in equation (2), summation need to be done over the whole dictionary of size $|w|$. Word2vec model tries to elevate this problem using negative samples rather than calculating the whole summation. The objective function in this case becomes:

$$\log\sigma(u_w^T v_{w_c}) + \sum_{i=1}^{k} E_{w_i\sim P_n(w)}[\log\sigma(-u_w^T v_{w_i})], \tag{3}$$

where, w_i is the set of negative samples, w_c is the context word. σ is the Sigmoid function defined as follows:

$$\sigma(x) = \frac{1}{1+\exp(-x)} \tag{4}$$

$P_n(w)$ is the unigram probability raised to power $3/4$. This objective tries to make embeddings of the current word similar to the context word embeddings and dissimilar to the negative word embeddings. Here, inner product is used as a similarity measure.

In large corpus, some words are very frequent. This may cause a bias in the learned embeddings. To counter this, each word w_i is discarded with the probability $1 - \sqrt{\frac{t}{count(w_i)}}$, where t is a threshold and it is fixed at 0.00001.

The objective function is optimized by using Gradient Ascent(Ruder, 2016). The gradient updates in this case would be:

$$u_w = u_w + lr * (1 - \sigma(u_w^T v_{w_c})) * v_{w_c} \tag{5}$$

$$u_w = lr * \sum_{i=1}^{k} E_{w_i\sim P_n(w)}(\sigma(-u_w^T v_{w_i}) - 1) * v_{w_i} \tag{6}$$

$$v_{w_c} = v_{w_c} + lr * (1 - \sigma(u_w^T v_{w_c})) * u_w \tag{7}$$

$$v_{w_i} = v_{w_i} + lr * E_{w_i\sim P_n(w)}(\sigma(-u_w^T v_{w_i}) - 1) * u_w, \tag{8}$$

where lr is the learning rate.

The final embedding of a word is constructed by concatenating input and context embeddings of the word.

2 Dataset and preprocessing

The word2vec model is trained on Reuters corpus(Yang et al., 1999). It has a total of $10,788$ documents of which $7,769$ are for training and $3,019$ are the test set. The training set is used for word2vec training. $7,769$ documents in training set are combined into a single document. All the words in the train document are converted to lower case before further preprocesing. Total number of words in this document are $1,253,696$. All the words with word length length less than 3 are removed. The document is then tokenized to make a train set. The numbers of tokens in this set are

$784,537$ and dictionary size is $24,418$. One set of word2vec experiments are performed on this dataset. The dataset can be further processed to remove stopwords and can be lemmatized. Wordnet(Miller, 1995) lemmatizer is used to lemmatize the dataset. This set has $603,678$ tokens and dictionary size is $22,131$. Window size of 5 is kept for the all the cases, (Mikolov et al., 2013) has window size of 5 for optimal embeddings.

3 Evaluation metric

Word2Vec model is evaluated in SimLex-999(Hill et al., 2015) word similarity task. SimLex-999 provides a way of evaluation of a model learning word similarity rather than relatedness or association. In SimLex-999 task there are word pairs and the corresponding similarity score given by a human in the range 0-10. To evaluate our word2vec model, cosine similarity is calculated for each pair of words and then Pearson coefficient is calculated between the human given scores and cosine similarity. Pearson coefficient is used final measure of model performance. Cosine similairy is given by the following formula:

$$CosineSimi(w_1, w_2) = \frac{< w_1, w_2 >}{||w_1|| * ||w_2||} \quad (9)$$

Pearson coefficient is a measure of linear correlation between two random variables. If word2vec is highly correlated with human given scores then it able to encode the word similarity on top of word association. It is calculated as follows:

$$\rho X, Y = \frac{cov(X, Y)}{\sigma_X \sigma_Y}, \quad (10)$$

where $cov(X, Y)$ is the covariance of X and Y and $\sigma(X)$ is standard deviation.

4 Results and Discussion

In all our experiments, window size is fixed at 5. Gradient ascent is performed with $lr = 0.1$ with learning rate annealing applied. The lr is halved after 8 epochs. The model is trained for different hyper-parameters like number of negative samples and embedding dimension. Embedding dimension of 100 is optimal with 10 negative samples is optimal for the case of training set without stop words and lemmatization applied. Lemmatization reduces the inflection forms and derivatively related forms. It makes use of a dictionary and morphological analysis of words to reduced the inflection forms. For small training data number of negative samples are recommended to be in range 5-20 (Mikolov et al., 2013) and our optimal value comes out to be 10. The results are given in Table 1 and Fig. 3 and Fig. 4. For word2vec model trained without lemmatization, the optimal model gives Pearson coefficient of 0.1627 with 5 negative samples and 100 dimensions. Results are given in Table 2 and Fig. 1 and Fig. 2

# Neg Samples	Dim	Pearson Coff
5	100	0.1948
10	100	**0.2076**
15	100	0.1888
5	200	0.1886
10	200	0.1491
15	200	0.1584

Table 1: This result is for the training set without stop words and rest of the words are lemmatized to reduce the embedding size. It considerably reduces the training time.

# Neg Samples	Dim	Pearson Coff
5	100	**0.1627**
10	100	0.1415
15	100	0.1088
5	200	0.1295
10	200	0.1167
15	200	0.1379

Table 2: This result is for the training set with stop words and without lemmatization. Its training time is considerably higher. Lemmatization gives better Pearson coefficient score.

In both the cases, 100 dimensions are sufficient to embed the word vectors. The dictionary size is small around $22k$ and number of tokens is also small around $600k$-$700k$. Therefore, increasing dimension size leads to performance degradation. The Pearson coefficient of 200 dim vectors for different # negative samples are lesser than that of 100 dim vectors. Lemmatization leads to increased performance because multiple derivatives and inflections of a word are merged together in a meaningful way which leads to more data for each basic form.

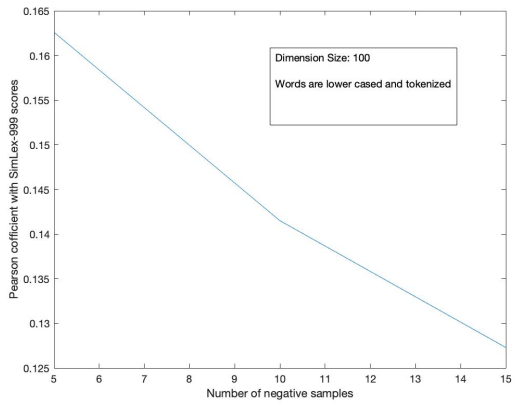

Figure 1: Embedding dim is 100 in this case. The optimal # negative samples in this case is 5 and gives score of 0.1627. Words are lower cased only

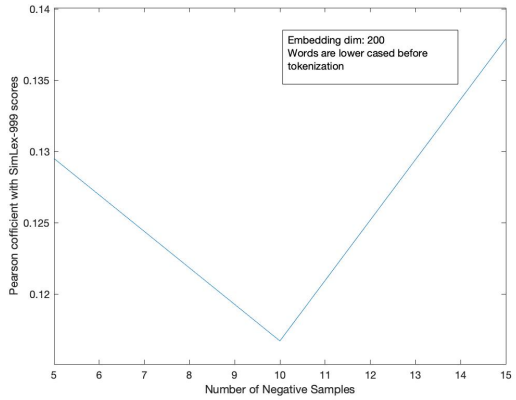

Figure 2: Embedding dim is 200 in this case. The optimal # negative samples in this case is 15 and gives score of 0.1379. Words are lower cased only

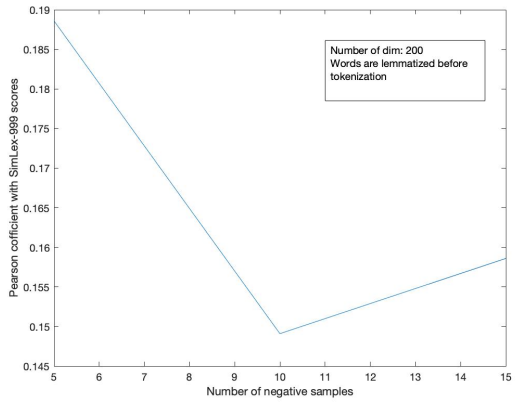

Figure 3: Embedding dim is 200 in this case. The optimal # negative samples in this case is 5 and gives score of 0.1948. Stopwords are removed and rest are lemmatized

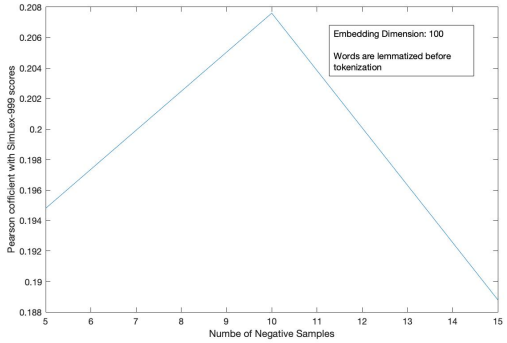

Figure 4: Embedding dim is 100 in this case. The optimal # negative samples in this case is 10 and gives score of 0.20767. Stopwords are removed and rest are lemmatized

5 Empirical Analysis

Given a test word, its cosine similarity is calculated with every word and the top-K candidates are collected. The top-k candidates shows correlation with the test word. Few of the examples are: Test word: ”would”, Top-10: [(’benefit’, 0.557), (’way’, 0.558), (’time’, 0.562), (’issue’, 0.56), (’also’, 0.58), (’plan’, 0.59), (’could’, 0.60), (’market’, 0.61), (’government’, 0.62), (’would’, 1.0)]

Test word: ”next”, Top-10: [(’autumn’, 0.59), (’higher’, 0.60), (’due’, 0.60), (’normal’, 0.60), (’since’, 0.60), (’starting’, 0.616), (’opening’, 0.61), (’week’, 0.64), (’month’, 0.65), (’next’, 1.0)]

In the top-10 neighbors for ”next” are words like ”month”, ”week”, ”opening”, which could be because of these words occurring together like ”next month”. Similar is the case with ”would”. Word ”queen” has ”elizabeth” in its top-5 neighborhood and ”king” has ”burger” in its neighborhood. There is some correlation between test word and its neighbors. However the original word2vec model(Mikolov et al., 2013) is trained on very large dataset therefore it is able to capture the correlation strongly. Reuters corpus has limited vocabulary, word like ”woman” is not present in the corpus. (Mikolov et al., 2013) has performed analysis like syntactic similarity and semantic similarity but reuters(Yang et al., 1999) is too small to conduct these type of analysis. Since ”queen” and ”king” comes in different contexts in the dataset and has different kind of neighbors, as discussed above, it is difficult to do the syntactic and semantic analysis on these words. To perform such anal-

ysis large dataset is required.

For the test words like 'male' and 'female' following are the neighbors:

Test word 'male': [('dunlop', 0.62), ('unemployement', 0.62), ('kilograms', 0.64), ('nurse', 0.65), ('commencing', 0.66), ('moerbeke', 0.67), ('sex', 0.68), ('inflorescences', 0.74), ('female', 0.78)] Test word 'female':

[('benguet', 0.6348872552254287), ('span', 0.64), ('vanilla', 0.65), ('unsold', 0.66), ('commencing', 0.66), ('inflorescences', 0.68), ('dips', 0.69), ('unemployement', 0.69), ('male', 0.78)]

Embeddings for 'male' and 'female' are highly correlated with each other. However, correlation of 'female' with 'unemployment' is coming out to be higher than 'male' and 'unemployment' which indicates a bias in the dataset.

References

Winnie Cheng, Chris Greaves, and Martin Warren. 2006. From n-gram to skipgram to concgram. *International journal of corpus linguistics*, 11(4):411–433.

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, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Sebastian Ruder. 2016. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Yiming Yang, Xin Liu, et al. 1999. A re-examination of text categorization methods. In *Sigir*, volume 99, page 99.