Deep Learning

Sentiment Classification using Word Embedding for IMDB Movie Review Data set

April 23, 2019

Lavanya Mandadapu

Contents

1 Introduction **3**

2 Data set description and Word2Vec Embedding **3**

3 Description and Explanation of Experiments **4**

4 Other Experiments **7**

5 Conclusions **9**

1 Introduction

Sentiment classification is a task of identifying whether a piece of text that a person is talking about falls into positive (person expresses liking emotions) or negative (person expresses disliking emotions). Sometimes, the person may express neutral emotion. However, in this exercise we are only concerned about positive and negative emotions/sentiment. Word embeddings are naturally used in many Natural Language Processing tasks because of their useful representation of the words and often lead to better performance. In this work, word embeddings are generated using word2vec and then passed to the embedding layer of the neural network for the sentiment classification. The source code and the results are available at `https://github.com/lavanya463/Word_embeddings.git`.

2 Data set description and Word2Vec Embedding

IMDB data set is for binary sentiment classification [1]. The original data has 25,000 movie reviews for training and 25,000 movie reviews for testing. However, for this work 40,000 movie reviews are considered for training and 10,000 are considered for testing. The data is also biased so the data is shuffled with fixed seed. The text reviews are labelled as 1 or 0 for positive and negative sentiment respectively.

Word embeddings for IMDB data set are generated using Gensim implementation of Word2Vec. To generate embedding, the data is prepared, first by creating word tokens (all words are converted to lowercase) using nltk word tokenizer and then by removing punctuations, stopwords and non alphanumeric words. Following parameters of word2vec are experimented to generate different embeddings.

- **Sentences**: List of the sentences we have in the corpus, to generate embeddings.
- **size**: It takes the embedding dimension. Embedding dimension is the number of dimensions that we want to represent our word; the size of the word vector.
- **min_count**: This is minimum frequency of the occurrence of the word to be considered into the model. If this value is equal to 1, then all the words will be considered.
- **window**: During training, the words that occur in the window neighbourhood are considered together. This is useful to capture the context of the word.

With appropriate values for above parameters, once the word embeddings are generated then the embedding matrix with word vectors is created. This embedding matrix can be directly fed into the embedding layer of our neural network. We have to set trainable parameter as false, as our embedding is already learned [2]. In figure 1 we can see the clusters of similar words produced using embeddings. The clusters are for the words present in bottom right in the picture. This plot is inspired and adapted form the medium post [3]. In the picture we can see that given *julia* as key, most similar words included are all the other actor/human names *kate*, *natalie* etc. And given *bollywood* as key, the

cluster includes *hindi* and some other region names. This tells us the relation of *bollywood* being connected with regions. That is the reason even though *portuguese* has nothing to do with *bollywood*, it is still in the same cluster as it is a region. Similarly, *season* is connected with *episode*, movie etc. It is also interesting to see how the word *baby* is related to *mama*, *dog*, *santa* and all other family related words. We can also see how the clusters of story and script are together, as they will be in similar context. This clusters has projected the strength of the generated word embeddings (with best combination, discussed in next section).

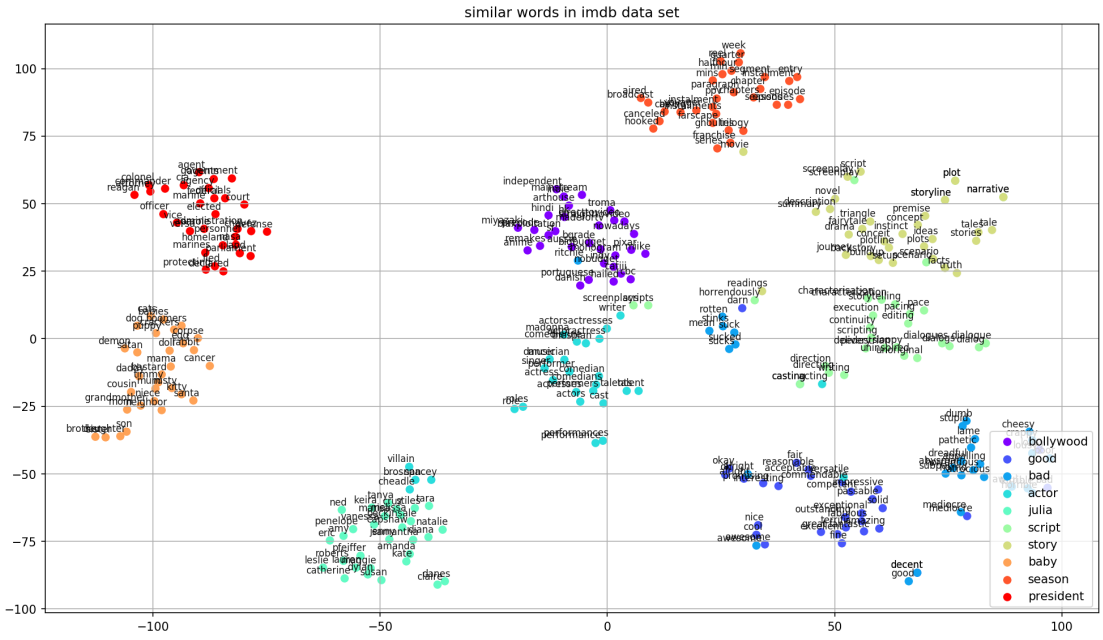

Figure 1: Clusters of similar words produced using embeddings

3 Description and Explanation of Experiments

A total of 27 combinations with small, medium and large values of embedding dimension, min count and window size are tested to understand their effect on the sentiment classification. Table 1 depicts these experiments and the resulted accuracy obtained from the neural network.

For all the experiments, a batch size of 256 and 10 epochs are considered. Figure 2 shows the neural network architecture used in this work. The trainable params will be changed based on the embedding dimension used. Single GRU layer with a dropout of 0.2 (to overcome overfitting), followed by a fully connected layer with sigmoid activation is considered. As the problem at hand is a binary classification problem, binary cross-

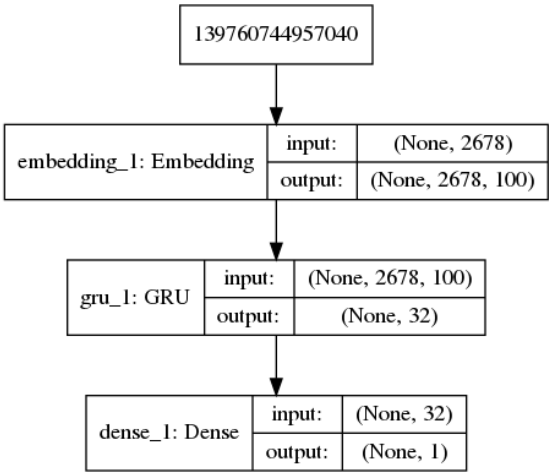

Figure 2: Neural Network model architecture used in this work

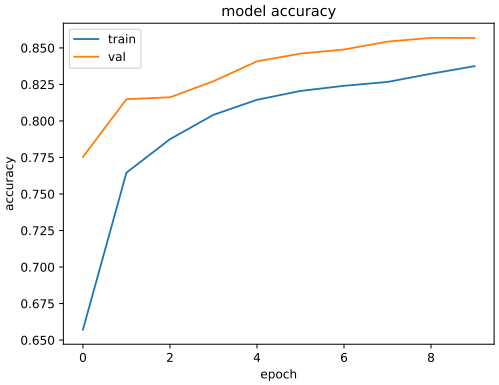

Figure 3: Accuracy plot

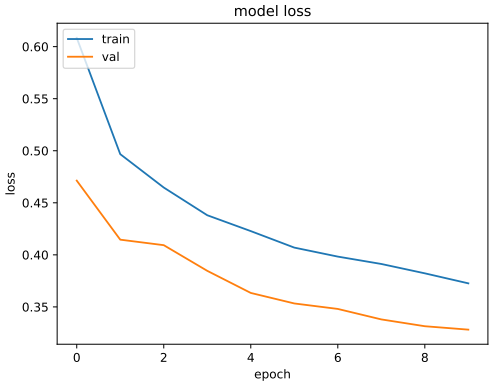

Figure 4: Loss plot

entropy is used for loss and adam is used as optimizer because of its high performance in similar tasks. Figures 3 and 4 shows the accuracy and loss plots of the neural network training after ten epochs when trained with the best combination 100 embedding dim, 5 window size and 1 min_count. The final test accuracy obtained with best combination after training for 25 epochs is 88.76.

From table 1 experiments we can see that when the min count is larger (50 in our case) we get high accuracy in all the cases. This can be seen because we are taking into account only the words with high frequency (many occurrences), so all the irrelevant (less occurred) words are removed. This is making our embeddings to be more appropriate

Table 1: Different combinations of embedding dimension, window size and min count with accuracy

Embedding dimension	Window size	Min count	Test accuracy
32	2	1	74.721
32	2	10	78.8
32	2	50	81.26
32	5	1	78.95
32	5	10	79.29
32	5	50	81.38
83.9832	10	1	74.7
32	10	10	73.92
32	10	50	77.09
64	2	1	81.63
64	2	10	82.1
64	2	50	84.74
64	5	1	78.16
64	5	10	82.95
64	5	50	84.74
64	10	1	79.21
64	10	10	80.11
64	10	50	81.58
100	2	1	85.35
100	2	10	83.08
100	2	50	83.38
100	5	1	**85.76**
100	5	10	**85.52**
100	5	50	84.35
100	10	1	78.93
100	10	10	81.09
100	10	50	83.98

which in turn making our model learn better. Likewise, when we are considering all the words (with min count 1) we are considering garbage words or irrelevant words too, resulting in less accuracy. With medium min count (10) our accuracy is more or less closer to the accuracy obtained with min count 1.

One of the best way to choose embedding dimension is by using a False Nearest Neighbours (FNN's) [4]. This method examines the fraction of nearest neighbours as the function of the embedding dimension. The minimum embedding dimension is found when most of the nearest neighbours do not move apart. In our case, a small(32), medium(64) and large(100) dimensions are tested. With less dimensions(32) we got less accuracy, as the dimensions increased the accuracy has increased. This is because we are increasing

the size of word vector, we are allowing the word to be represented in many dimensions. Therefore, we are eliminating some of the weak dependencies between words, making our model to learn strongly related words, resulting in the increase of accuracy.

A window size of 2, 5, 10 are tested. With both small and high window size the accuracy is less compared with the medium window size. If we use small window size then our model cannot capture the context of the word. And if we have larger window size then we are distorting the context. In our particular corpus, the max words per sentence after pre-processing are not too large. Therefore, a medium window size will be sufficient to capture the context of the word, which proved to be correct by obtaining high accuracy.

From all the experiments the best combination is obtained with embedding dimension of 100(large), window size of 5(medium) and min count of 10(medium). However, the result obtained with large embedding dimension, medium window size and large min count is also higher than other combinations, apart from best. It is to be noted that with high dimensions even with a min count 1, the result is more or less similar to the best accuracy. This can be seen because, even if we consider all the words, when we are increasing the dimension, as said earlier, the unrelated words are being moved away in higher dimension, resulting in obtaining similar result to the min count medium. In the experiment results we can also see that with lower dimensions, even though the min count is higher, they cannot produce higher or similar accuracy than the accuracy obtained by using higher dimensions.

4 Other Experiments

Adapting the code provided in the class, some more general experiments are performed to get a better idea about how different combinations of min_count, window size and embedding dimension effects on obtaining the linear regularities of the simple analogy *King - man + women = queen*. Using the embeddings obtained on the combinations that are mentioned in previous section similar words for the word *horrible* are also generated, results obtained for this case are mostly similar to one other, but with a variation in the similarity score. Both Pairwise and Euclidean distance measures are tested, the results obtained with both are quite similar, however, pairwise has given better similarity score.

Regarding *King - man + women* analogy, the obtained word vectors has similar words to the word Queen, like Princess, Cinderella etc. However, all the word vectors obtained in the experiments does contains the word King. Table 2 dipicts the obtained words vectors on 27 combinations mentioned above. When the embedding dimension is large(100) we get to see good word vector with most of the words being related to the context Queen. When the embedding dimensions are less(32) and medium(64), we can see that some unrelated words like *Triangle*, *poe* present in the word vector. The reason behind this is that, with higher dimensions, most related words are surviving together (following the manifold concept). We can also see similar behaviour with the window size and min count, as they are increased more context is captured(with more window size), high occurrence words are considered, removing useless words for embeddings(with larger min

count), making word vectors containing related words to the analogy.

Table 2: Word vectors obtained for the analogy *King - man + women* with different combinations of embedding dimension, window size and min count

Combination	Test accuracy
1	['king', 'christie', 'lion', 'bride', 'edgar', 'prince', 'wells', 'princess', 'dickens', 'agatha']
2	['king', 'princess', 'bride', 'lion', 'jungle', 'prince', 'juliet', 'dickens', 'emperor', 'victoria']
3	['king', 'princess', 'juliet', 'romeo', 'queen', 'rice', 'bride', 'lion', 'carmen', 'prince']
4	['king', 'prince', 'romeo', 'princess', 'lion', 'christie', 'jungle', 'bride', 'jane', 'rochester']
5	['king', 'princess', 'queen', 'juliet', 'prince', 'triangle', 'carmen', 'romeo', 'lord', 'emperor']
6	['king', 'princess', 'lion', 'juliet', 'lord', 'prince', 'romeo', 'edgar', 'queen', 'poe']
7	['king', 'princess', 'juliet', 'romeo', 'queen', 'rice', 'bride', 'lion', 'carmen', 'prince']
8	['princess', 'king', 'queen', 'fairy', 'prince', 'alice', 'juliet', 'cinderella', 'castle', 'bride']
9	['king', 'princess', 'queen', 'cinderella', 'lord', 'prince', 'bride', 'carmen', 'fairy', 'lion']
10	['king', 'romeo', 'emperor', 'christie', 'dickens', 'prince', 'princess', 'bride', 'legend', 'carmen']
11	['king', 'emperor', 'princess', 'bride', 'romeo', 'prince', 'christie', 'jungle', 'legends', 'shanghai']
12	['king', 'emperor', 'princess', 'tarzan', 'bride', 'rice', 'dickens', 'romeo', 'christie', 'roman']
13	['princess', 'king', 'juliet', 'prince', 'romeo', 'queen', 'bride', 'cinderella', 'alice', 'triangle']
14	['princess', 'king', 'queen', 'juliet', 'prince', 'bride', 'romeo', 'carmen', 'lion', 'emperor']
15	['king', 'queen', 'princess', 'juliet', 'lover', 'carmen', 'romeo', 'fairy', 'bride', 'carol']
16	['princess', 'queen', 'king', 'juliet', 'romeo', 'bride', 'prince', 'carmen', 'cinderella', 'alice']
17	['king', 'christie', 'romeo', 'princess', 'dickens', 'emperor', 'carmen', 'bride', 'rice', 'photographer']
18	['king', 'emperor', 'princess', 'bride', 'romeo', 'wells', 'rice', 'dickens', 'guinea', 'mcnally']
19	['king', 'romeo', 'princess', 'carmen', 'emperor', 'dickens', 'prince', 'furst', 'juliet', 'agatha']
20	['king', 'princess', 'queen', 'juliet', 'romeo', 'bride', 'carmen', 'prince', 'fairy', 'alice']
21	['king', 'princess', 'queen', 'prince', 'juliet', 'romeo', 'bride', 'carmen', 'hilton', 'rice']
22	['king', 'princess', 'queen', 'romeo', 'juliet', 'bride', 'prince', 'alice', 'wedding', 'carmen']
23	['king', 'princess', 'queen', 'juliet', 'romeo', 'bride', 'carmen', 'prince', 'fairy', 'alice']
24	['king', 'princess', 'queen', 'prince', 'juliet', 'romeo', 'bride', 'carmen', 'hilton', 'rice']
25	['king', 'princess', 'queen', 'romeo', 'juliet', 'bride', 'prince', 'alice', 'wedding', 'carmen']
26	['queen', 'king', 'princess', 'prince', 'juliet', 'fairy', 'romeo', 'bride', 'cinderella', 'castle']
27	['queen', 'king', 'princess', 'juliet', 'prince', 'cinderella', 'romeo', 'bride', 'carmen', 'fairy']

The combinations numbers given in the table 2 are the combinations of embedding dimension, min count and window size, these combinations follows the combinations of the table 1.

5 Conclusions

Word embeddings can be used for a variety of tasks in deep learning such as sentiment analysis, syntactic parsing etc., Embedding dimension, Window size and Min count effect on the embeddings and the resulted classification accuracy is tested in this work. It is relatively important to tune this parameters to obtain good embeddings for the task at hand. There is lot of data that is being generated day to day like in twitter or any other social media platforms. It is necessary and useful if we use this data to generate embeddings, which in turn can be used to classify or cluster the sentiment or the words.

References

[1] Stanford:. http://ai.stanford.edu/~amaas/data/sentiment/.

[2] Javaid Nabi:. https://towardsdatascience.com/machine-learning-word-embedding-sentiment-classification-using-keras-b83c28087456.

[3] Sergey Smetanin:. https://towardsdatascience.com/google-news-and-leo-tolstoy-visualizing-word2vec-word-embeddings-with-t-sne-11558d8bd4d.

[4] Barbara Cannas Massimo Camplani. The role of the embedding dimension and time delay in time series forecasting. https://pdf.sciencedirectassets.com/314898/1-s2.0-S1474667015X60885/1-s2.0-S1474667015367859/main.pdf?x-amz-security-token=AgoJb3JpZ2luX2VjENL