# Sentiment Analysis of Movie Reviews using a Deep Learning Convolutional Neural Network

Ahrim Han, Ph.D.

July 23, 2019

# Contents

# 1 Introduction

**Problem Statment.** Sentiment analysis (or opinion mining) is the task of identifying and classifying the sentiment expressed in a piece of text as being positive or negative. Given a bunch of text, sentimental analysis classifies peoples opinions, appraisals, attitudes, and emotions toward products, issues, and topics. The sentiment analysis has a wide range of applications in industry from forecasting market movements based on sentiment expressed in news and blogs, identifying customer satisfaction and dissatisfaction from their reviews and social media posts. It also forms the basis for other applications like recommender systems.

In the past years of studies, a review text was converted to fixed-length vector using bag-of-words and these vectors were later used to train the classifier such as Naive Bayes or Support Vector Machine. The major problem of the bag-of-words are the 1) lack of consideration of semantic relationship between words 2) data sparsity and high dimensionality. Moreover, as there are tons of reviews and posts on the web, there is a strong need for the more accurate and automated sentiment analysis technique.

In recent years, there has been a remarkable performance improvement to Natural Language Processing (NLP) problems by applying deep learning neural networks. Therefore, in this project, I will build deep learning models to classify the positive and negative movie reviews using the high-edge deep learning techniques. I will observe the performance improvements in the accuracy and compare models using various parameters.

**Expected Beneficiaries.** Automated and accurate sentiment analysis techniques can be used to detect fake reviews, news, or blogs and are becoming more and more important due to the huge impact on the business markets [1] [2]. We provide the potential beneficiaries of this work.

- Businesses can find consumer opinions and emotions about their products and services.
- E-commerce companies, such as Amazon and Yelp, can identify fake reviews. Fake reviews are

not only damaging both competing companies and customers, but they also lead to reduced trust in e-commerce companies and lower sales.

- Potential customers also can know the opinions and emotions of existing users before they use a service or purchase a product.

**Approach.** We prepare the movie review text data for classification with deep learning methods. We obtain the large data set of the movie reviews [3] [4]. We clean the documents of text reviews by removing punctuations, stopwords, stemming and removing non-frequent words to prevent a model from overfitting. In this pre-processing of documents, we use the more sophisticated methods in the NLTK python package.

To build a deep learning model, we basically use the sequential model of Keras.

1. First, the Embedding layer is located. There are two ways of setting the embedding layer: using the pre-trained word embedding or training new embedding from scratch. We use a pre-trained word embedding, GloVe (Global Vectors for Word Representation) embeddings [5].

2. Second, a series of convolution 1D and pooling layers are added according to typical CNN for text analysis. Then, after flattening layer, fully connected dense layers are added. Since this is a binary classification problem, we use the sigmoid function as an activation function for the final dense layer.

3. Finally, we will make the different deep learning models by adjusting the parameters and will find the best accurate model. We later will investigate the features affecting the accuracy.

## 2 Data Preprocessing

In this section, we perform cleaning documents. More details of the process for cleaning documents can be found in this IPython notebook.

## 2.1 Environment

For this project, we used **my own Linux machine having AMD Ryzen 7 2700X, 16GB Memory, Geforce RTX 2070**.

In addition, **Keras with Tensorflow backend** is used for making a deep learning model. Keras is a high-level API, written in Python and capable of running on top of TensorFlow, Theano, or CNTK deep learning frameworks. Keras provides a simple and modular API to create and train Neural Networks, hiding most of the complicated details under the hood. By default, Keras is configured to use Tensorflow as the backend since it is the most popular choice. Keras is becoming super popular recently because of its simplicity.

## 2.2 Data Loading

There is a large dataset for binary sentiment classification of movie reviews [3] [4]. It contains substantially more data than previous benchmark datasets. This data set contains 50,000 reviews which is evenly split into two groups: 25,000 highly polar movie reviews for training and 25,000 for testing. The train and test sets contain a disjoint set of movies, so no significant performance is obtained by memorizing movie-unique terms and their associated with observed labels. Therefore, we can assume that the validation result with testing data set can be applicable for other movie reviews.

Each group has the same number of positive and negative reviews: a positive review has a score from 7 to 10 while a negative review has a score from 1 to 4. The reviews having score 5 or 6 are excluded to avoid vagueness.

For building deep learning models, all the documents are loaded as in Figure 1. The data sets for training and testing are stored in `data/train` and `data/test`, respectively. For each data set, positive and negative reviews are stored in `pos` and `neg` sub-directories. I have attached the progress bars using the tqdm library, which is useful in dealing with large data by allowing us to estimate each time of the stages.

In [3]:

```python
# load all docs in a directory
def load_docs(directory):
    documents = list()
    # walk through all files in the folder
    for filename in tqdm(os.listdir(directory)):
        # create the full path of the file to open
        path = directory + '/' + filename
        with open(path, 'r') as f:
            # load the doc
            doc = f.read()
            # add to list
            documents.append(doc)
    return documents
# load all training reviews
print("Loading training-positive-docs")
global_train_positive_docs = load_docs('data/train/pos')
print("Loading training-negative-docs")
global_train_negative_docs = load_docs('data/train/neg')
# load all test reviews
print("Loading test-positive-docs")
global_test_positive_docs = load_docs('data/test/pos')
print("Loading test-negative-docs")
global_test_negative_docs = load_docs('data/test/neg')
```

```
Loading training-positive-docs
100% 12500/12500 [00:01<00:00, 6822.51it/s]
Loading training-negative-docs
100% 12500/12500 [00:03<00:00, 3314.74it/s]
Loading test-positive-docs
100% 12500/12500 [00:03<00:00, 3479.76it/s]
Loading test-negative-docs
100% 12500/12500 [00:00<00:00, 19914.30it/s]
```

Figure 1: Data Loading.

## 2.3 Cleaning Documents

In most of NLP releated works, documents are normally pre-processed to get better performance. We tried to apply several techniques which are well-known as follows:

1. **Removing punctuations.** Normally punctuations do not have any meaning, but they exist for

understandability. Therefore, such punctuations should be removed. But, we did not remove the apostrophe mark (') since such removing caused the incorrect stemming.

2. **Removing stopwords.** We filtered out the stopwords. The stop words are those words that do not contribute to the deeper meaning of the phrase. They are the most common words such as: the, a, and is. `NLTK` python package provides a list of commonly agreed upon stop words for a variety of languages.

3. **Stemming.** The `PorterStemmer` is provided in `NLTK` python package. We made the words into lowercases and used the stemming method in order to both reduce the vocabulary and to focus on the sense or sentiment of a document rather than deeper meaning.

4. **Defining a vocabulary dictionary of preferred words and removing non-frequent words.** It is important to define a vocabulary of known words when using a bag-of-words or embedding model. The more words, the larger the representation of documents, therefore it is important to constrain the words to only those believed to be predictive.

   In this project, **we set up the vocabulary dictionary and remove the non-frequent words to prevent a model from overfitting.** Building code of the vocabulary dictionary is implemented in `vocab.ipynb` notebook. The process can be summarized as follows.

   - First, based on only reviews in the training dataset, we count the words' occurrences using `Counter` function. This is saved in `vocab_counter.txt` file and contains 52,826 words. Figure 2 shows the most frequent 30 words extracted from training data set.
   - Then, we filter out the words that have low occurrences, such as only being used once or none and set up the vocabulary dictionary. Thus, the vocabularies have the two or more occurrences. There are 30,819 number of words in the vocabulary dictionary and those are saved in the `vocab.txt` file.

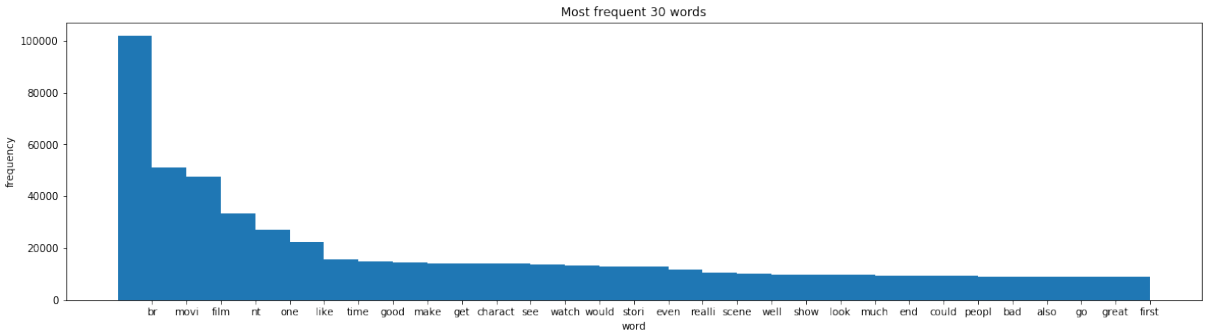


Figure 2: Most Frequent 30 Words in Training Data Set.

- Finally, using the the vocabulary dictionary, we remove the non-frequent words for cleaning documents. This is done by filtering out words that are not in the vocabulary dictionary and removing all words that have a length $<= 1$ character.

## 2.4 Multiprocessing

Pre-processing mentioned above requires heavy computation. To improve the speed, we parallelized the pre-processing using the **Pool module** in **multiprocessing package**. Since we use a CPU having 8 cores, the size of Pool is set as 8. By using this technique, **we could achieve 6 7 times speed up.** Using the single thread, it takes 10 12 minutes for cleaning up 12500 documents, whereas, using the multiple threads, it takes only 1 minute and 20 40 seconds.

## 2.5 Data Encoding

To use documents as an input of a model, each document is encoded as a sequence object of Keras. The function below encodes the documents as sequence objects as well as creates a list of labels: '0' for negative reviews and '1' for positive reviews. We do not need the one hot encoding process (a function called `to_categorical()` in Keras) because there is only two classes of positive and negative.

## References and Notes


[1] "Deep Learning Sentiment Analysis of Amazon.com Reviews and Ratings.", Shrestha, Nishit, and Fatma Nasoz, *arXiv preprint arXiv:1904.04096 (2019)*.

[2] Fake spot finding website, https://www.fakespot.com/.

[3] "Learning Word Vectors for Sentiment Analysis.", Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts, *The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011).*

[4] Stanford Large Movie Review Dataset, https://ai.stanford.edu/∼amaas/data/sentiment/.

[5] GloVe: Global Vectors for Word Representation, https://nlp.stanford.edu/projects/glove/.