Machine Learning Engineer Nanodegree

Sentiment Analysis on IMDB Movie Review Dataset

Piyush Moolchandani
October 04, 2019

Definition

Project Overview

This project deals with the problem of sentiment analysis. IMDB's movie review dataset has been used to train the model in this project.

Sentiment analysis is contextual mining of text which identifies and extracts subjective information in source material, and helping a business to understand the social sentiment of their brand, product or service while monitoring online conversations. Sentiment analysis basically provides service provider a idea of polarity of user's views on the product by doing some processing on the text review/feedback given by the user. In today's competitive market, it is very important to analyse and act upon the reviews but with the ever growing data, it is impossible to get reviewed each and every review by an human, hence sentiment analysis algorithm becomes an crucial advancement in this field.

Problem Statement

The objective of this project is to create a model to predict whether a given review is of positive and negative view, and for training and testing this model, IMDB dataset from this ACL 2011 paper by Learning Word Vectors for Sentiment Analysis (http://www.aclweb.org/anthology/P11-1015 (http://www.aclweb.org/anthology/P11-1015)) is to be used.

A LSTM based RNN Network is used to solve this problem. Initially data is preprocessed to make it suitable to be used with the lstm network and then a lstm network with embedding layer is defined. This network is then trained using training dataset of IMDB dataset and after suitable training, it is tested on other half (testing dataset) of the data and we have a model with trained parameters for sentiment analysis.

Metrics

Accuracy metric is used in this project for performance evaluation.

it is a common metric for binary classifiers. It takes into account both true positives and true negatives with equal weight.

> accuracy = (True Positives + True Negatives) / Size of Dataset

Accuracy is a correct metric to evaluate the performance on this project because the most important thing to understand while evaluating the performance of a sentiment analysis algorithm is to know what fraction of reviews are correctly classified. Accuracy metric does exactly this as is visible in formula mentioned above.

Analysis

Data Exploration

The core dataset contains 50,000 reviews split evenly into 25k train and 25k test sets. The overall distribution of labels is balanced (25k pos and 25k neg). It also include an additional 50,000 unlabeled documents for unsupervised learning.

In the entire collection, no more than 30 reviews are allowed for any given movie because reviews for the same movie tend to have correlated ratings. Further, 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. In the labeled train/test sets, a negative review has a score <= 4 out of 10, and a positive review has a score >= 7 out of 10. Thus reviews with more neutral ratings are not included in the train/test sets. In the unsupervised set, reviews of any rating are included and there are an even number of reviews > 5 and <= 5.

Only the train and test sets are used in this project.

Examples from dataset

1. **From pos directory :** Bromwell High is nothing short of brilliant. Expertly scripted and perfectly delivered, this searing parody of a students and teachers at a South London Public School leaves you literally rolling with laughter. It's vulgar, provocative, witty and sharp. The characters are a superbly caricatured cross section of British society (or to be more accurate, of any society). Following the escapades of Keisha, Latrina and Natella, our three "protagonists" for want of a better term, the show doesn't shy away from parodying every imaginable subject. Political correctness flies out the window in every episode. If you enjoy shows that aren't afraid to poke fun of every taboo subject imaginable, then Bromwell High will not disappoint!

2. **From neg directory :** From the beginning of the movie, it gives the feeling the director is trying to portray something, what I mean to say that instead of the story dictating the style in which the movie should be made, he has gone in the opposite way, he had a type of move that he wanted to make, and wrote a story to suite it. And he has failed in it very badly. I guess he was trying to make a stylish movie. Any way I think this movie is a total waste of time and effort. In the credit of the director, he knows the media that he is working with, what I am trying to say is I have seen worst movies than this. Here at least the director knows to maintain the continuity in the movie. And the actors also have given a decent performance.

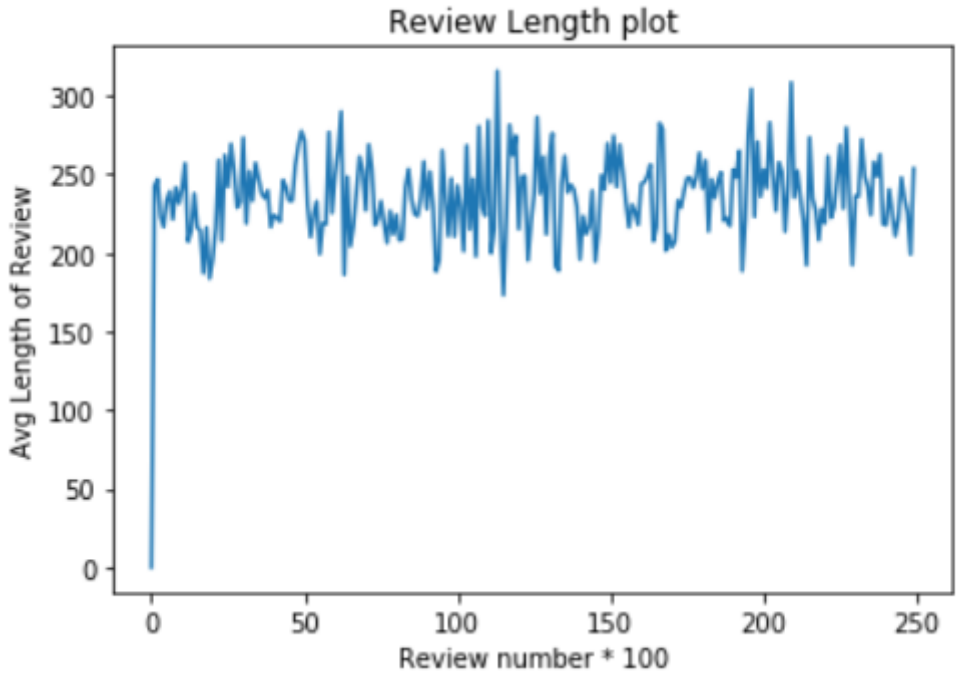
This dataset needs to be preprocessed before giving input to the network. Distribution of datasets:

1. Test data
2. Train data:
    - 80 % : actual train data
    - 20 % : validation data

Exploratory Visualization

Not all the reviews are of same length in the dataset but we need to make the input size fixed, so we will take somewhere around the average length as the fixed length for the input data. Following is a plot of lengths of all the reviews. Note: Data used for this plot is the data obtained after removing punctuations form the original

dataset.



Clearly majority of reviews lie between 200 to 300, after calculating average of lengths, 250 was decided as the fixed length and reviews are padded with zeroes or clipped accordingly.

Algorithms and Techniques

I have used **LSTM RNN Networks** to solve this problem and **Naive Bayes** classifier to benchmark my solution.

1. **Naive Bayes** is a good algorithm for working with text classification. The relative simplicity of the algorithm and the independent features assumption of Naive Bayes make it a strong performer for classifying texts. The Naive Bayes classifier uses the Bayes Theorem to select the outcome with the highest probability. This classifier assumes the features(in this case- words)are independent.

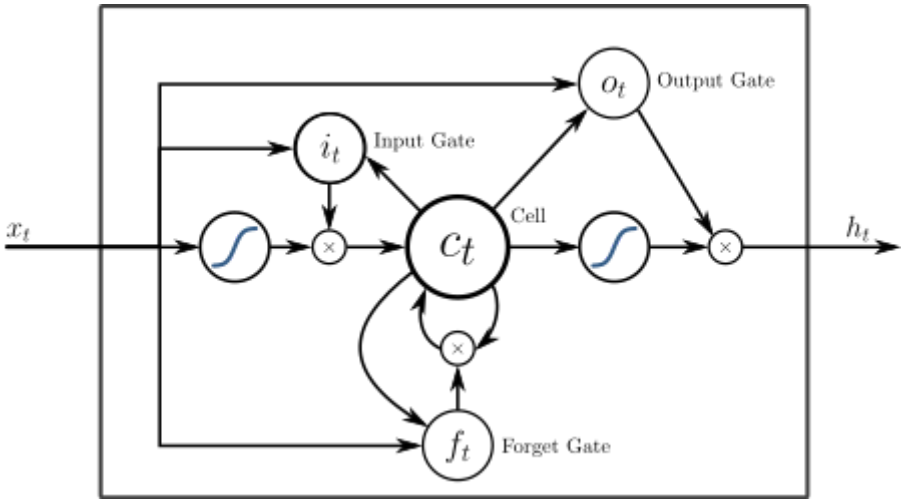
   The Naive Bayes classifier assumes the probability of the label (positive or negative) for the given review text is equal to the probability of the text given the label, times the probability a label occurs, everything divided by the probability that this text is found. It compares the probabilities of the labels and chooses the one with higher probability.

2. **LSTM** : Long Short term-Memory Networks

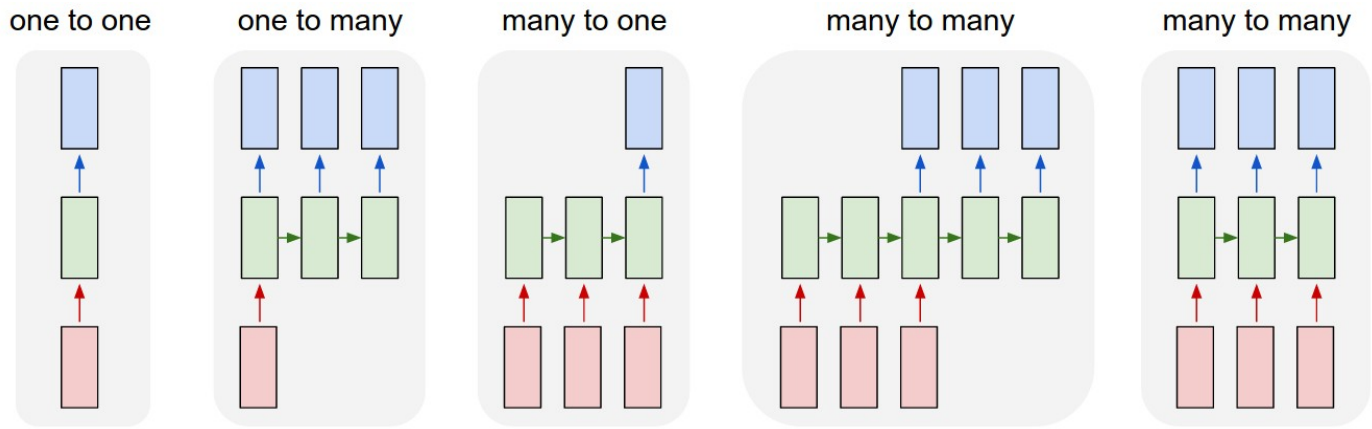
   Mostly in text data, words are not independent, they depend on the words surrounding them and may also make totally different sense when used with the complete sentence. So sentiment is dependent upon long semantics or feature detection. Each word in a sentence depends greatly on what came before and comes after it. In order to account for such dependencies, Recurrent Neural Network or Long Short-Term Memory come in handy.

   In RNNs, each word in an input sequence will be associated with a specific time step. In effect, the number of time steps will be equal to the max sequence length. RNN can be used when data is treated as a sequence, where the order of the data-points matter.

   RNN suffer from vanishing gradients problem and find it difficult to learn long-distance correlation in sequence. LSTM is a type of RNN which solves this problem, following is the internal structure of LSTM. This picture was obtained from https://en.wikipedia.org/wiki/Long_short-term_memory (https://en.wikipedia.org/wiki/Long_short-term_memory).



Many to one models are used for Sentiment Analysis. (http://karpathy.github.io/assets/rnn/diags.jpeg (http://karpathy.github.io/assets/rnn/diags.jpeg))



Inputs are provided to the network as tokenized vectors, which are then handled by embedding layers deployed just before LSTM layers.

Benchmark

Multinomial Naive Bayes Classifier is used as a benchmark model for this result. It was used because Naive Bayes is a good algorithm for working with text classification. The relative simplicity of the algorithm and the independent features assumption of Naive Bayes make it a strong performer for classifying texts.

But the problem is for review, words can not be considered independent from each other. Sentiment is dependent upon long semantics, each word in a sentence depends greatly on what came before and comes after it, so Naive Bayes algorithms are not standard for this type of problems, but can be used as a benchmark model.

When tested on this data, Naive Bayes gave accuracy of 50.088 %. Which will act as benchmark for LSTM network.

Methodology

Data Preprocessing Steps

1. Combine all test and training reviews into test and training review files respectively. Do the same for test and training labels with test and training labels files respectively. This is required to tokenize the reviews.

2. To randomize the data (so that no bias is there), while combining all reviews in step 1, after each review is inserted into the common file, randomly a review from pos or neg folders is chosen for the next insertion.

3. Read the data and convert the data to all lower caps and remove punctuations from it, as otherwise same words get counted many times.

4. Create list of all the individual words in complete data and give them token numbers. Use these token numbers to make tokenized vectors which can be given as inputs to the embedding layer.

5. All the outliers (zero size reviews) are removed from the data.

6. Fix the length of the review because neural network will accept only fixed size input, this is done using padding with 0's or clipping of the reviews for less than fixed size and more than fixed size inputs respectively. This fixed size is decided as 250 after calculating the average length of all the reviews.

Implementation

Proprocessed data is fed into a RNN network, which consists of following layers:

1. An embedding layer to converts our word tokens in integer form into embeddings of a specific size.

   > Embedding layer is an improvement over more the traditional bag-of-word model encoding schemes where large sparse vectors were used to represent each word or to score each word within a vector to represent an entire vocabulary. Instead, in an embedding, words are represented by dense vectors where a vector represents the projection of the word into a continuous vector space.

2. An LSTM layer defined by a hidden_state size and number of layers.

   > Standard RNNs (Recurrent Neural Networks) suffer from vanishing and exploding gradient problems. LSTMs (Long Short Term Memory) deal with these problems by introducing new gates, such as input and forget gates, which allow for a better control over the gradient flow and enable better preservation of “long-range dependencies”.

3. A Dropout layer.

   > This layer acts as a simple regularization step by dropping some nodes from the previous node. Whether to drop or not is decided based on drop probabilities.

4. A fully-connected output layer that maps the LSTM layer outputs to a desired output_size.

5. A sigmoid activation layer which turns all outputs into a value between 0 and 1.

The flow of training process

1. Define a sentiment analysis network class with following functions:
    - forward : feedforwards the data through network
    - initialize_hidden_layer : initializes the lstm layers
    - init : consturctor (initialize the variables and declares a network)
2. Create an instance of class defined in 1 with chosen values of various parameters.
3. Define optimizer, loss function.
4. Train the instance with train data, and check performance after every 100 steps on validation data.
5. Plot the graph of validation loss with number of steps taken so far. This plot should show validation loss decreasing with number of steps increasing.

Difficulties arise in maintaining the proper shape of inputs, and special care has to be taken care while reshaping or giving inputs to different layers. For convenience of maintaining proper shape of inputs, batch_first parameter was set to True, while defining LSTM layers.

References are mentioned at the end of the report.

Refinement

Initially I tried with following hyperparameters with few epochs:

```
lr = 0.01
clip = 1 and 10
embedding_dim = 1000
hidden_dim = 512
n_layers = 3
drop_prob = 0.25
```

I couldn't check every set of hyperparameters with large number of epochs because of limitation of resources.

After few epochs, I got an accuracy of 74% and 76% with clip as 1 and 10 respectively, then after trying various different set of hyperparameters, I finally trained my model with these set of hyperparameters, which I think are suitable for this model.

```
lr = 0.001
clip = 5
embedding_dim = 100
hidden_dim = 256
n_layers = 2
drop_prob = 0.4
```

With these parameter, I got 81 percent accuracy in the beginning, and I trained the network with these parameter for 100 epochs, which led to overfitting. So I reduced the number of epochs and added more variables to the embedding layer of model for better training and reduced the clip upper limit to 3. Following are the parameters used in final submission:

```
lr = 0.001
clip = 3
embedding_dim = 200
hidden_dim = 512
n_layers = 3
drop_prob = 0.4
```

I obtained 82.56% accuracy with these set of parameters.

Results

Model Evaluation and Validation

As mentioned above final model is the following LSTM Network:

Sentiment_Analysis_Network(
(embedding): Embedding(203684, 200)
(lstm): LSTM(200, 512, num_layers=3, batch_first=True, dropout=0.4)
(dropout): Dropout(p=0.4)
(fc): Linear(in_features=512, out_features=1, bias=True)
(sig): Sigmoid()
)

Network Features:

- Embedding layer generates a vector that maps all the distinct words to a continuous vector space. I tried different size of output vector from embedding layer ranging from 100 to 1000 but 200 seemed to work pretty well with the model.
- There are three LSTM layers with 512 outputs, which is a reasonable number of layers and number of output nodes for the given dataset. I have tried 128 and 256 also but model worked best with 512.
- The dropout probability is set at 0.4, it prevents the model from overfitting and is not large enough for causing the model to underfit.
- A fully connected layer is attached to these layers which maps the output of previous layer to one output and then a sigmoid layer is attached at the end which gives either 1 or 0 as the output.
- Output layer's size is 1 as the final output is a bipolar decision, a review is either Positive or Negative, for which 1 output node is enough.

Other Features:

1. After trying various learning rates like 0.01, 0.005, 0.001, 0.0005, 0.0001, I choose 0.001 as the final learning rate for training the model.
2. BCE(Binary Cross Entropy) Loss is chosen because for binary classification it is used instead of cross entropy loss.
3. Adam is used as the optimizer as Adam combines the best properties of the AdaGrad and RMSProp algorithms to provide an optimization algorithm.
4. 20 % data from train dataset is used as a validation dataset.
5. All gradients are clipped to a maximum value of 3 to remove the problem of exploding gradients.

Model works well even if the input data is tweaked a little bit such that even if fixed input size is changed to 230 or 270, accuracy doesn't decreases by more than 2 to 3 percent.

Results from this model can be trusted and can be used for real world analytics as around % is acceptable accuracy for a sentiment analysis model, as long as it does not involve huge monetary or human losses.

Justification

The model I choose for benchmark was Multinomial Naive Bayes Classifier, which gave an accuracy of 50.088%. And this model gives an accuracy of 82.56%.

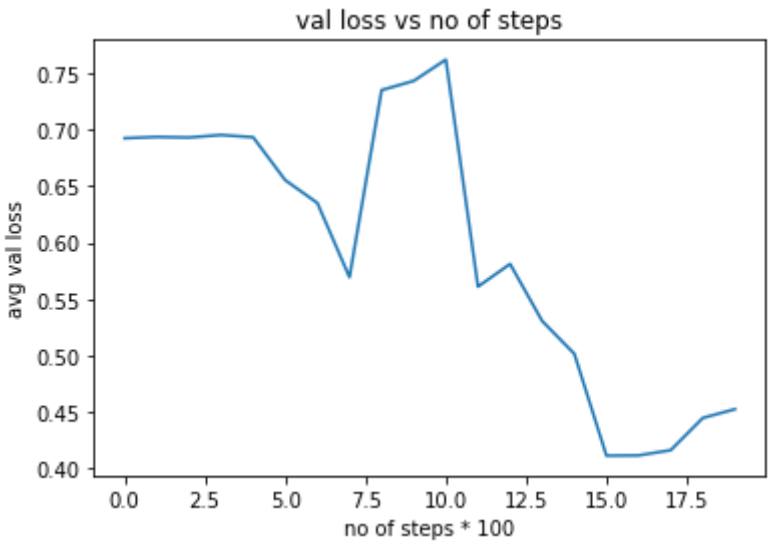
So clearly, this sentiment analysis model with lstm layers gives better results than the benchmark model presented earlier. There is a significant performance improvement over the benchmark, and even small change in accuracy is significant in the real world. Moreover, a LSTM model makes more sense than a Binomial Naive Bayes classifier in real world, NBC assume that each word in independent but LSTM considers the fact that sequential order of words is important, and output of one is effected by the presence of other, which is actually the reality of Reviews or any other text data.

Final solution is good enough to solve the real world problems, but can be further improved using recent advancements in the field of Natural language processing and Deep Learning like GF-RNNs and with the help of enough resources.

V. Conclusion

Free-Form Visualization

Following is the plot of changing Validation loss with the increasing number of training steps performed.



It is clearly visible in the graph that even though validation accuracy may not decreases constantly, it has decreased significantly over long term. Decrement of validation loss is important to know that out model is training correctly. The plot proves that the model is training in the correct direction which is decreasing the performance on a data on which it hasn't been trained on.

Reflection

I knew about the domain of Sentiment Analysis, but never previously had got the chance to explore about it. But I got a chance to make a sentiment analysis model through this project, and I enjoyed exploring the field and making a model myself.

- I started by watching a lecture on RNNs by Justin Johnson, in the Stanford's Convolutional neural networks for visual recognition class (https://youtu.be/6niqTuYFZLQ?list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv) (https://youtu.be/6niqTuYFZLQ?list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv)).

- Then I asked the mentors if I can do sentiment analysis for my Capstone project, and requested them to provide some reading material for this project, as I have never recurrent neural networks before.

- I read different blogs on sentiment analysis classifier which include the following:
  - The Unreasonable Effectiveness of Recurrent Neural Networks (http://karpathy.github.io/2015/05/21/rnn-effectiveness/ (http://karpathy.github.io/2015/05/21/rnn-effectiveness/))
  - Deep Learning #4: Why You Need to Start Using Embedding Layers (https://towardsdatascience.com/deep-learning-4-embedding-layers-f9a02d55ac12 (https://towardsdatascience.com/deep-learning-4-embedding-layers-f9a02d55ac12))
  - Sentiment Analysis using LSTM (Step-by-Step Tutorial) (https://towardsdatascience.com/sentiment-analysis-using-lstm-step-by-step-50d074f09948 (https://towardsdatascience.com/sentiment-analysis-using-lstm-step-by-step-50d074f09948))
  - Application of RNN for customer review sentiment analysis (https://towardsdatascience.com/application-of-rnn-for-customer-review-sentiment-analysis-178fa82e9aaf (https://towardsdatascience.com/application-of-rnn-for-customer-review-sentiment-analysis-178fa82e9aaf))
  - How to Use Word Embedding Layers for Deep Learning with Keras (https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/ (https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/))

- There were two main parts of the projects:

  1. Preprocessing : This part was interesting as well as challenging mainly because text wasn't preprocessed one, each review was in a separate file and I needed to combine them in one file and also with randomize the order. Further data needed to be pre-processed (removing punctuation, tokenizing) to make it suitable as an embedding layer input.
  2. Training and testing : This part was interesting, there were few points to consider in this section, such as initializing hidden layers after every epoch. Finding appropriate learning rate, dropout probability, clipping value and sizes of layers in network was also tricky and time-consuming, but worth the effort.

Final model meets my expectation. Even though there can be improvements, this model can also be used to solve sentiment analysis problem in real world.

Improvement

Following improvement are possible for this project:

1. With enough resources, hyperparameter can be tuned in better way. Combinations of different layer sizes and no of LSTM layers(2 or 3) can be tried and the best model can be selected.
2. Lexical normalization can be performed on dataset. For example: goooood and good need to be counted as one word but this model counts them as two different words. This also reduces computational complexity of the model.
3. GF-RNN networks can be used instead of LSTM networks, both LSTM AND GF-RNN have been cited by most papers on Sentiment Analysis.
4. Using Word2Vec for Embedding layer might also improve the performance of the model.

This model can be used as a benchmark model for a model implementing above mentioned improvement steps.

References

1. http://karpathy.github.io/2015/05/21/rnn-effectiveness/ (http://karpathy.github.io/2015/05/21/rnn-effectiveness/)
2. http://blog.echen.me/2017/05/30/exploring-lstms/ (http://blog.echen.me/2017/05/30/exploring-lstms/)
3. https://www.youtube.com/watch?v=6niqTuYFZLQ (https://www.youtube.com/watch?v=6niqTuYFZLQ)
4. https://towardsdatascience.com/deep-learning-4-embedding-layers-f9a02d55ac12 (https://towardsdatascience.com/deep-learning-4-embedding-layers-f9a02d55ac12)
5. https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-ker (https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-ker)
6. https://stats.stackexchange.com/questions/222584/difference-between-feedback-rnn-and-lstm-gru (https://stats.stackexchange.com/questions/222584/difference-between-feedback-rnn-and-lstm-gru)
7. https://towardsdatascience.com/sentiment-analysis-using-lstm-step-by-step-50d074f09948 (https://towardsdatascience.com/sentiment-analysis-using-lstm-step-by-step-50d074f09948)
8. https://towardsdatascience.com/application-of-rnn-for-customer-review-sentiment-analysis-178fa82e9aaf (https://towardsdatascience.com/application-of-rnn-for-customer-review-sentiment-analysis-178fa82e9aaf)
9. https://en.wikipedia.org/wiki/Sentiment_analysis (https://en.wikipedia.org/wiki/Sentiment_analysis)