

Youtube Comment Sentiment Analysis

Anas Sikder, Abrar Jahin, Ar-Raji Billah

160204021@aust.edu, 160204029@aust.edu, 160204053@aust.edu

Introduction

The task of analyzing opinion, emotion, sentiment etc by utilizing Natural Language Processing methods is called Sentiment Analysis or opinion mining. At present, social media influences the whole society to a great extent. Among various social media platforms, YouTube is a site where millions of people interact through uploading, watching videos and posting comments. The scarcity of manually annotated data makes the task of sentiment analysis on YouTube comments very complex.

A. Task definition with usage

The goal of this project is to predict sentiment from comments of youtube videos using NLP. The videos belong to different channels and are related to different categories. For our task, we have chosen the following categories:

- Sports
- Tech products
- Historical documentaries
- News
- Psychology
- Health

Example usage of our task is described below:

- If a company releases a new tech product, they would generally like to know how people feel about it. Analyzing the sentiment of comments can indicate some opinions regarding whether people are excited or disappointed.
- Sports clubs may learn about the popularity of different players. Sometimes, a top notch player may be out of form. Whether fans are still hopeful or angry at that player, fans blaming the club management or the player, these things can be noticed from the sentiments in the comments.
- The comment section of historical documentaries may be full of optimism, pessimism, wisdom or hate. The sentiments may help the channel admin post videos of more popular topics.
- Channels of news outlets can learn how much viewers trust or mistrust their contents. They may try being more transparent and neutral.

- Audience of psychology channels are mostly young. The comment sections represent their psychological characteristics. This may help psychologists enhance their expert system.
- Prevention of cyberbullying against youtube content creators.

B. Motivation

In the realm of NLP(Natural Language Processing), sentiment analysis is a subject undergoing intense studies.

Sentiment analysis is widely used in social media monitoring. There are real-time Social media monitoring tools. There are wide varieties of applications of sentiment analysis. Organisations all over the world are deploying systems to extract insights from data belonging to different social media platforms.

Sentiment analysis was used by the Obama administration to test public opinion for announcing policies ahead of the presidential election 2012. Forum posts, news articles etc are a vast source of information regarding public sentiment by utilizing which it is easier to strategize for the future.

Nowadays, market research and customer service methodologies deeply rely on it. It also shows plans of competitors, their innovations and goals. Tech giants like Google, Tesla, Microsoft, Apple etc are investing heavily in sentiment analysis for various purposes such as customer experience of new products. From the sentiment analysis in a moderation report, it can be figured out which community managers dealt with negative/positive sentiment and how rapidly they responded. According to this, the moderators who can handle negative sentiments better can be promoted.

The recent COVID-19 vaccine posts in social media received mixed reactions from people. It should be noted that hoax and panic inciting comments are heavily monitored by governments. Sentiment analysis from youtube comments is highly essential for these actions as comments in these videos tend to be very negative. Also people are discussing the observed side effects of the vaccine. This information may help medical researchers to a great extent.

Most of the recent sentiment analysis tasks are based on imdb movie dataset, hotel review, twitter tweets etc. Only a few papers for the task we are working on have been published. Most of those used a dataset with comments with similar topics.

Therefore, a system to predict sentiments from comments from videos based on diverse topics would amplify the knowledge in this domain.

C. Challenges

Some major challenges we may face while completing the task are:

- Not enough open source datasets. There is a notable corpus suitable for our task named SenTube [8] but is not open source.
- There are two highly upvoted open source datasets in kaggle but they are not annotated. [11],[12]
- Insufficient data may not yield decent results. Training DL models requires lots of data.
- Comments are from videos of different topics which makes the task further challenging.
- Accuracy fluctuates because of the dataset having texts with ambiguity and sarcasm. Humans themselves disagree a lot while assigning labels in sentiment analysis datasets.
- Using built-in libraries to assign labels will definitely not result in creating a Gold or Silver dataset. Moreover, such highly controversial practices are generally not appreciated by the ML/DL community.
- Difficult to find labeled data.

Related Works

A. Some related works

1. **Comment Abuse Classification with deep learning:** They worked on 115,846 comments. The dataset is from the wikipedia detox project. They achieved 93 and 94 accuracy with CNN with word embedding and LSTM with word embedding respectively.[1]
2. **Toxic comment classification:** Accuracy for Naive Bayes and LSTM are 48 and 67 respectively. The dataset consists of 159,000 comments.[2]
3. **Cooking is All about people, comment classification on cookery channels using BERT and Classification Models:** They gathered a total 4291 comments from two YouTube cooking channels. The authors tried both BERT/XLM and traditional ML models like RF,SVM, Naive Bayes, DT, KNN with different vectorizers like TF-IDF, count etc. RF with term frequency gained 63.59% accuracy which is the highest among ML models. XLM achieved the highest accuracy 67.31% among variations of BERT/XLM models.[3]
4. **Detecting Multilabel Sentiment and Emotions from Bangla Youtube Comments:** The Google translator was utilized to detect comment language. There are 5011 Bangla and 4189 English comments. Trainable skipgrams gained

the highest accuracy among word vectorizers (65.92%). Among LSTM, CNN, NB, SVM models, LSTM achieved the highest accuracy of 65.96%. [4]

B. Important prior works

1. **Sentiment Analysis of YouTube Movie Trailer Comments Using Naïve Bayes:** The writers achieved 81% accuracy. They experimented with TF-IDF feature extraction and Naive Bayes classifier. [5]
2. **Sentiment analysis on youtube: A brief Survey :-** Lexicon based sentiment analysis, negation detection and Social Media Aware Phrase Detection (SMAPD) these three issues are addressed to detect sentiment polarity. [6]
3. **Sentiment Analysis on Youtube Comments: A brief study:-** Focused only on the polarity of the comments, not subjectivity. They used TextBlob to detect polarity. [7]
4. **SenTube: A Corpus for Sentiment Analysis on YouTube Social Media:-** Dataset for YouTube comments annotated using web based annotation tool. Dataset consists of 35887 English comments. They used BOW vectorizer and SVM model. [8]
5. **Textblob and VADER:** Textblob is a python library for processing textual data which provides API for sentiment analysis. VADER is a lexicon and rule based tool specially tuned for social media sentiments. [15], [16]

The issues with the related recent works are:

- There are not many commercial applications based on these studies.
- VADER works better with slangs, emojis etc and Textblob is more suitable for formal language usage. But YouTube comments are mixed of both formal and informal comments.
- Most of the papers didn't use diverge and controversial topics having ambiguous comments when building the dataset.
- Paper 2 in the above list used 11 different categories of videos but they trained only SVM. They didn't train any DL model.
- Though some of these recent works have various word vectorizer based experiments, they completely missed comparing different GloVe, Word2Vec, FastText etc word embedding models.
- Majority of the related publications focused on sentiment analysis tasks for other social media platforms like Twitter etc, not on YouTube comments.
- The SenTube corpus is not open source.

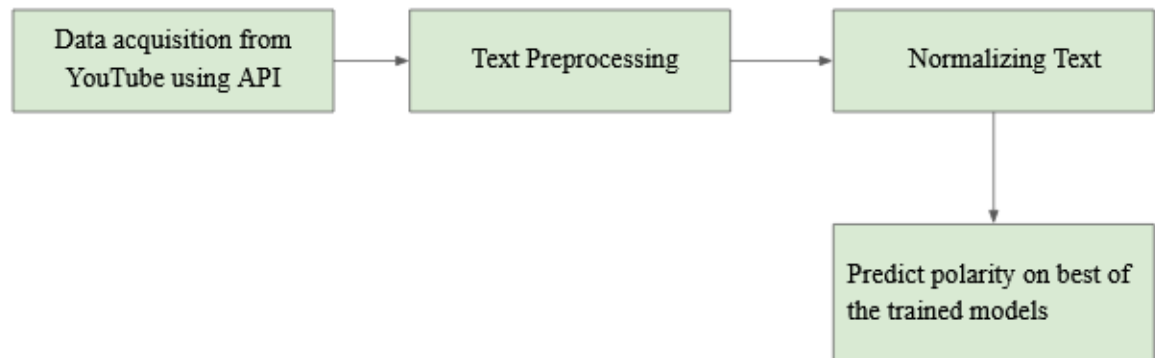
C. Steps to be taken to address the issues

- While building the dataset, we would choose videos with wide varieties of topics and videos which have difficult to interpret comments along with videos with simple comment sections.
- In most of the related works they didn't use simple RNN models, rather proceeded to training complex CNN and LSTM models. Though basic RNN has limitations, it is capable of performing remarkably in the tasks we are trying to complete.
- We would experiment with different pre-trained word embedding models for feature extraction of our basic Deep Learning models.
- We would try to implement web or android applications based on the experiments in this report in the future.

Project Objective

A. Subtasks of the System:

- I. The subtasks of this sentiment analysis task are data acquisition, text processing, text normalizing and polarity prediction.
- II. The flow of the subtasks that would be performed by the system is shown below:



Here, 1 is for positive and 0 is for negative polarity.

B. Dummy input and output of the system:

Comment no.	Comment	Predicted Polarity
1	Just upgraded from a Galaxy J7 Prime to a iPhone 11 2 days ago, and I love it	1
2	But you should add that overstimulated immune system is also a bad thing	0
3	Congratulations!! What a Journey 2018/2019 season!!	1

Methodologies

- A. After properly gathering and annotating the data, we are going to solve the problem by using word vectorizers like Bag of Words,TF-IDF or pre-trained word embedding models such as FastText,GloVe etc for feature extraction and then feeding the extracted features into ANN or RNN models. The libraries and methods for preprocessing and text normalizing procedures and the baseline models of this experiment are described below:

Libraries

NLTK & spaCy is a free open-source library for Natural Language Processing (NLP) in Python to support teaching, research, and development. Which are:-

- Free and Open source
- Easy to use
- Modular
- Well documented
- Simple and extensible

We have used NLTK for the data preprocessing tasks.

NLTK

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum.

Spacy

SpaCy is an open-source software library for advanced NLP, written in the programming languages Python and Cython. Unlike NLTK, which is widely used for teaching and research, spaCy focuses on providing software for production usage.[9][10] spaCy also supports DL workflows that allow connecting statistical models trained by popular ML libraries like Tensorflow, PyTorch or MXnet through its own machine learning library named Thinc.

Tokenization

Token means each "entity" that is a part of whatever was split up based on rules. For example, each word is a token when a sentence is "tokenized" into words. Each sentence can also be a token, if we tokenized the sentences out of a paragraph. Tokenization is the process of breaking a stream of text up into **sentences, words, phrases, symbols, or other meaningful elements called tokens**.

Stop Words Removal

Stopwords are common words that **generally** do not contribute to the meaning of a sentence. Most search engines will filter stopwords out of search queries and documents in order to **save space and time** in their index.

- Removing stopwords is not a hard and fast rule in NLP. It depends upon the task that we are working on.
- Having individual dictionary entries per version which includes every word would be highly redundant and inefficient.
- For tasks like text classification, where the text is to be classified into different categories, stopwords are removed or excluded from the given text so that **more focus can be given to those words which define the meaning of the text**.

Normalizing Text

The goal of both stemming and lemmatization is to **"normalize"** words to their **common base form**, which is useful for many text-processing applications.

Lemmatization

The Lemmatization process involves first determining the part of speech of a word, and applying different normalization rules for each part of speech.

Stemming

Stemming means heuristically removing the affixes of a word, to get its **stem (root)**. It is a rule-based process of stripping the suffixes (“ing”, “ly”, “es”, “s” etc) from a word.

N-Gram method

N-gram represents the count of a sequence of n words. The value of n can be one-word chunk (unigram), two-word chunk (bigram), or a three-word chunk (trigram). In this project, we used the default n-gram range for the count vectorizer (Bag of Word model) which is (1,1) that means only unigrams were selected.

Bag of Words

Bag of Words (BOW) is a text representation method that indicates the occurrence of elements of a vocabulary for a set of documents . A vocabulary is a set of unique words for a set of documents. The presence of words is measured using either count based or one-hot encoded vectors.

TF-IDF

TF (Term Frequency) means the number of times a particular word appears in a document. IDF (Inverse Document Frequency) is the frequency of the word occurring in different documents . In other words, IDF is the measurement of the importance of a specific term. The TF-IDF method is widely favoured because it tries to measure the most relevant words in a set of documents. Thus, depending on the word counts is no longer needed. The equations for this method is shown below:

$$tf_{t,d} = \frac{n_{t,d}}{\text{Number of terms in the document}} \quad (1)$$

Here, the numerator is the number of times the word ‘t’ is present in the document ‘d’.

$$idf_t = \log \frac{\text{number of documents}}{\text{number of documents with term 't'}} \quad (2)$$

TF-IDF is simply the multiplication of TF and IDF.

If both IDF and TF values are high for a particular term, then the value of TF-IDF is high. In that case, it is rare in the set of documents but frequent in a single document.

Word Embedding

Semantic information not being extracted from the text representation is a drawback in case of both BOW and TF-IDF. Moreover, for BOW, the matrix becomes very sparse for a big vocabulary which means each vector contains too many zeroes. Also, the vector size is equal to the vocabulary size. This hampers the computational efficiency to a great extent. For example, if a BOW output is fed to a Neural Network, the size of the input layer would be equal to the size of the vocabulary which is huge. No pair of one hot encoded or count based vector can represent any semantic similarity.

To overcome these problems, there is a concept named Word Embedding which is basically a feature representation of terms. Each vector of this representation has a size equal to the number of predefined features. So, there is no possibility of the matrix being sparse anymore. The word embedding matrix contains dense vectors (vectors with real values). Word Embedding is capable of extracting semantic information as two similar terms have similarity in their feature vectors. Thus, analogy between different terms can be found using Word Embedding.

Each one hot encoded vector of a particular vocabulary gets multiplied with the feature matrix of a pre-trained Word Embedding model. In this way the feature representation for that vocabulary is extracted.

The algorithms that are used to learn word embeddings can examine billions of words of unlabeled text - for example, 100 billion words and learn the representation from them which is available for free on the internet.

Embedding matrix:

- When we implement an algorithm to learn a word embedding, what we end up learning is an embedding matrix.
- Suppose we are using 10,000 words as our vocabulary (including token like <UNK>)
- The algorithm should create a matrix E of the size $300 \times 10,000$ if we are extracting 300
- features which means 300 dimensional embeddings.

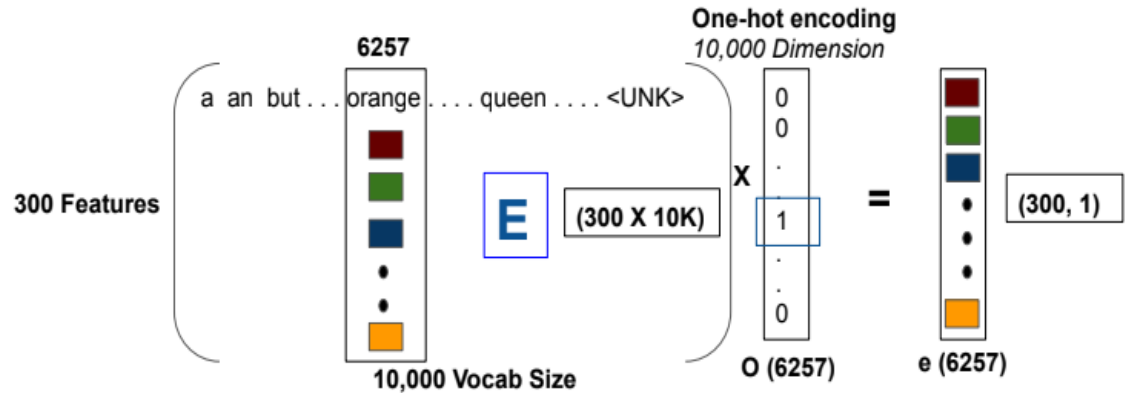


Fig: Embedding matrix

To find the embeddings of the word 'orange' which is at the 6257th position, we multiply the above embedding matrix with the one-hot vector of orange:

$$E \times O(6257) = e(6257)$$

The shape of E is (300, 10k), and O is (10k, 1). The embedding vector $e(6257)$ will be of the shape (300, 1).

Generally,

$$E \cdot O_j = e_j$$

- E = Embedding Matrix.
- O_j = One-hot vector of word j in the vocabulary.
- e_j = embedding for word j in the vocabulary.
- We initialize the matrix E (300 x 10K) randomly and use gradient descent to learn the parameters of the matrix.
- In practice, it is not efficient to use vector multiplication to extract the word embedding, but rather a specialized function to look up the embeddings.
- Because, O vector is a high dimensional one-hot vector and most of these elements will be zero.

Some renowned word embedding techniques are GloVe, Word2Vec, FastText etc. In this experiment, we have used GloVe and FastText. These techniques are briefly described below:

GloVe (Global Vectors for Word Representation)

The simplest GloVe vectors (6B) were trained with a corpus of 6 billion words. There are GloVe vectors with 50,100,200 or 300 dimensions. The training of the vectors was done with an unsupervised learning algorithm. The feature

representation of GloVe shows remarkable linear substructures on two dimensional word vector space.

FastText

This pre-trained feature representation is available for 157 languages. It is trained on common crawl and wikipedia using Continuous Bag of Words (CBOW) with a dimension of 300 and n-gram length of 5. FastText provides better performance compared to Word2Vec.[13],[14]

Baseline Models

We have decided to train two Deep Learning models for this binary classification problem of NLP. Some state of the art Deep Learning models in the field of sentiment analysis are LSTM, BERT, Transformer etc. But these models are comparatively advanced compared to our course objective. So, we have chosen simple ANN and RNN to proceed with the experiments.

Artificial Neural Network (ANN)

Neural Network is basically a stack of units that perform logistic regression and passes the output to the next layers. These outputs are the output of activation functions.

There are at least one input, one hidden (fully connected) and one output layer in a Neural Network. The internal layers are called hidden layers because the true values of those layers for the training set can't be observed.

Activation functions can be linear or nonlinear. Linear functions are not eligible for backpropagation and linear operation of a collection of layers or neurons produces a linear decision boundary. This is similar to linear regression. For generating complex non-linear decision boundaries, non-linear activations are used. They have non-zero derivatives which makes backpropagation possible.

Equations for multiple logistic regression units are converted to a vectorized equation for backpropagation.

Recurrent Neural Network (RNN)

Recurrent Neural Network is a variant of a feed forward Neural Network having an internal memory. It has recurrent characteristics as it repeats the same function for every chunk of input while the output of the current input depends on the computation of the previous one. The output copy is sent back into the network. It considers both the current input and the stored output from the previous input for making a decision.

RNN can work with sequences of input using their memory which makes them capable of performing tasks such as continuous handwriting recognition, speech recognition etc. In all other neural network variants, the inputs are independent of each other.

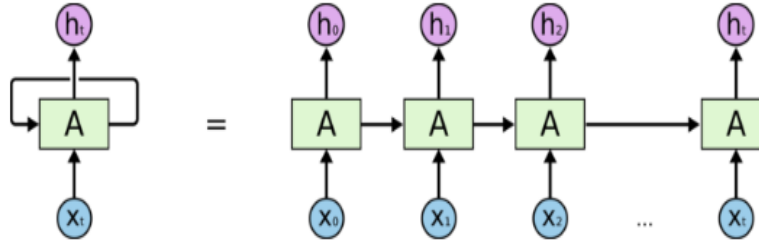


Fig: Basic component of an RNN

Here, A is the hidden layer and x(0) is the input at time=0.

Firstly, A takes x(0) from the input sequence and outputs h(0). This h(0) and x(1) is the input for the same hidden layer A at time t=1. Similarly, h(1) and x(2) is the input for time=2. This cycle continues until the loss is minimized during backpropagation.

The necessary equations for forward propagation over time are,

$$h_t = f(h_{t-1}, x_t) \quad (3)$$

$$h_t = \tanh(W_{hh}h_{t-1} + W_{hx}x_t) \quad (4)$$

$$y_t = W_{hy}h_t \quad (5)$$

$W(hh)$ is the weight assigned with the connection from the previous hidden state, $W(hx)$ is the weight at the connection from current input. $W(hy)$ is the weight from the last state to output state $y(t)$.

The mechanism of backpropagation for RNN is similar to the Neural Network.

The limitations of RNN are:

- The possibility of vanishing gradient problem as text sequences get larger.
- It can not work with too long text sequences using tanh or ReLU.

To overcome the vanishing gradient problem, LSTM (Long Short Term Memory) is used which is a modification of RNN.

The process of data acquisition and annotation are described below:

Data Acquisition

For acquisition we first copied the video id from the Url section .Then we pasted it in Google Sheet. In the sheet there is an option called tools .We selected the tool option and clicked on Script Editor . It is directly linked with this link [19]. Then we used a code which we collected from github [17] and it worked with only the initial comments not with replies, to extract the comment,name,likes,time and reply count. In google script editor services we used ‘Youtube Data API V3’ and ran the code . Finally, all those comments are stored in corresponding cells in google sheets. We only used the comment column from the sheets.We did the process for 7 times for collecting datas. So, we had 7 csv files from which we merged all the files using pandas concat function and made our raw dataset.

Preprocessing before Annotation

Pre-processing was a challenge for us in such a short period .We read some online Blogs [18] and tried to understand how to proceed further . Firstly, we removed emojis using a function where we put common unicode values of emojis. For unicode values we took help from the website [20] and using the ‘re’ function we subtracted those unicodes from the text column of our dataset. We found there are some Arabic, Chinese , Indian etc languages in our dataset. We did not want to work with those languages. So we removed non English text using the python library function. For doing this we again took help of the ‘re’ function and subtracted text of those languages . We also removed username, html tag url, # symbol etc with the help of python library.

Annotator Agreement

All 3 members of our project team participated in labelling the dataset. Before starting the procedure, we studied many relevant articles and papers related to corpus annotation

for sentiment analysis. We acted as independent annotators which means we weren't influenced by each other. The dataset was labelled with 2 mutually exclusive classes: 0 for negative and 1 for positive sentiment. We tried to avoid labelling any sample which represents inconsistent sentiment. Yet, the task was very difficult as for some videos the context of a particular comment conflicts with the context of the video. It wasn't even clear whether those comments were sarcastic or not. An MS Excel sheet was used for voting labels for the samples. To minimize the possibility of being influenced, we initially filled up 3 different CSV files and then merged them together. Then the final labels were decided according to majority votes.

Preprocessing After Annotation (Text Normalizing)

After labelling the data we removed the special characters from the comment column of the dataset and also turned all the documents to lowercase. Then we tokenized the data in the comment column using **wordpunct_tokenize** function. **Wordpunct_tokenize** function is a function defined in NLTK library. For lemmatization of tokens we used the **WordNetLemmatizer** which is also defined in NLTK. We took noun and verb forms. Also for stemming we took help of NLTK library and used **PorterStemmer** module. We also replaced all digits with <NUM> and removed stop words with the help of NLTK. After doing all these we used those tokens in BOW and TF-IDF models. We also did a different try with the help of Spacy library and tokenized them and used them in the RNN models.

B. Methodology Diagrams:

The diagrams of our methodology are shown below:

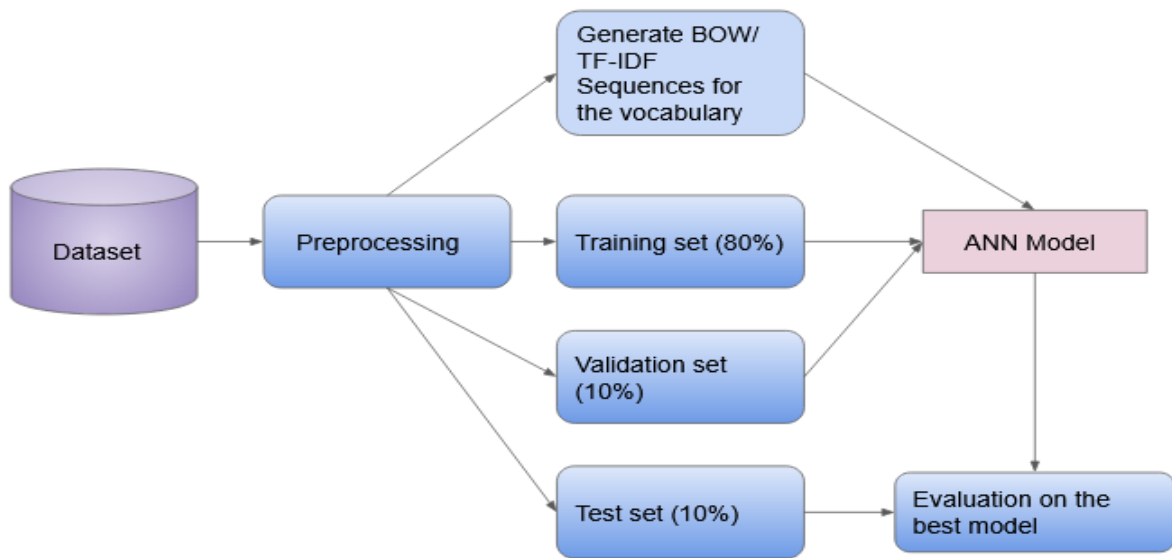


Fig: Bag of Words/ TF-IDF classifier

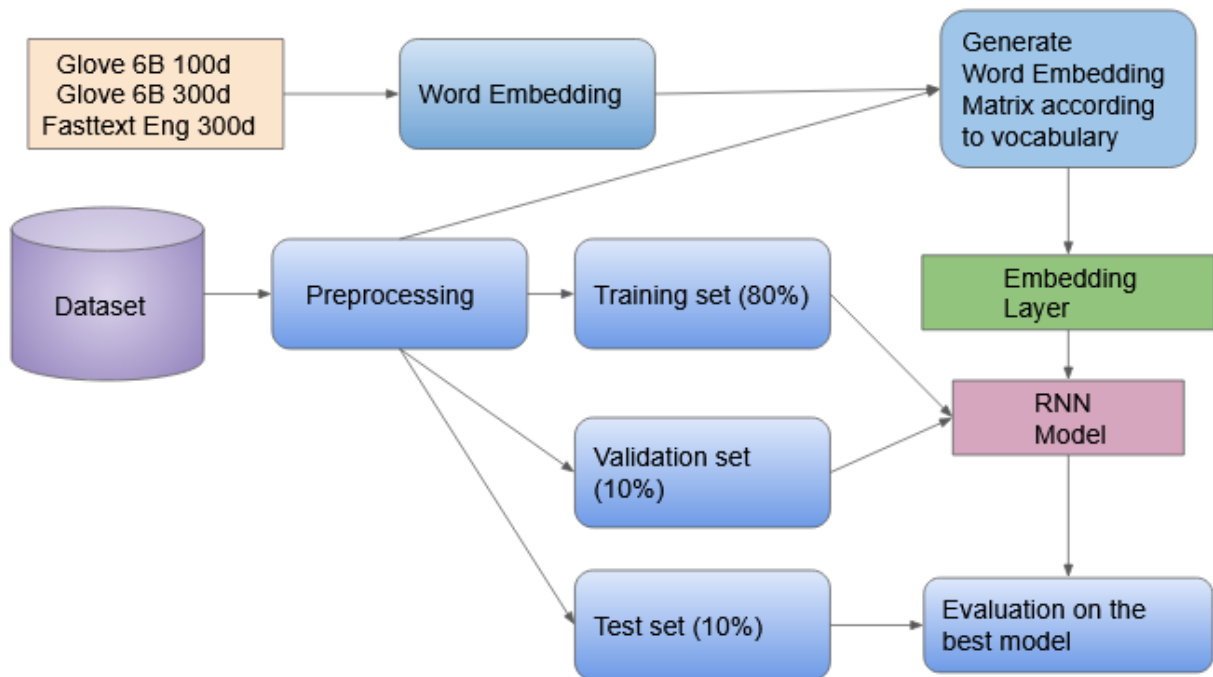


Fig: WordEmbedding-RNN classifier


Experiments

A. Dataset

For data acquisition, we selected 7 different videos. All of our target videos are in English but there are still some anomalies which we removed in the pre-processing part. The videos are about COVID-9 vaccines, gaming consoles (PS5), football tournament, war related documentary (D-Day), tech products (new iPhones), comedy and psychology. We avoided videos with political contents due to the perspective of comments being very complex to judge.

- The dataset consists of a total 1304 comments.
- Label 0 is for negative and 1 for positive polarity.
- Comments that seemed to be neutral are annotated as positive.
- There are 720 comments with positive polarity and 584 comments with negative polarity.
- The dataset has an 80-10-10 train-validation-test split.

The comments belong to 7 videos of different categories from 7 different channels:

1. **Vaccine side effects are actually a good thing:**
<https://www.youtube.com/watch?v=F6NKbQzo4aE>
2. **PS5 Honest Review - is it worth it? | The Tech Chap:**
<https://www.youtube.com/watch?v=9I6ww0wrF5g>
3. **LIVERPOOL CROWNED EUROPEAN CHAMPIONS! | Tottenham 0-2 LFC | Champions League Highlights:**
<https://www.youtube.com/watch?v=zIlwMVkeJ5E>
4. **D-Day (1944):**
<https://www.youtube.com/watch?v=4cGuB-OWR0g>
5. **iPhone 11: top 25+ features:**
<https://www.youtube.com/watch?v=RtsZGtW0Ljw>
6. **BOOMBASTIC Bean  | Mr Bean's Holiday | Funny Clips | Mr Bean Official:**
<https://www.youtube.com/watch?v=kSGrk5gVkmM>
7. **What is depression? - Helen M. Farrell:**
<https://www.youtube.com/watch?v=z-IR48Mb3W0>

Almost all the videos contain comments which are very confusing to annotate. Videos no. 3,5,6 contains mostly positive comments, whereas, video 1,7 contains mostly negative and video 4 contains comments from mixed polarity. 1 and 7 have many comments filled with disappointment, dark humor etc. Videos including number 4 have optimistic, pessimistic and sarcastic comments. Comments from video 3,6 are full of celebration and

joy. Comments in the tech related videos consist of criticism, show-off, recommendation etc. Some comments in these videos are just neutral or informative. Some screenshots of the comments and the dataset annotation process are shown below:

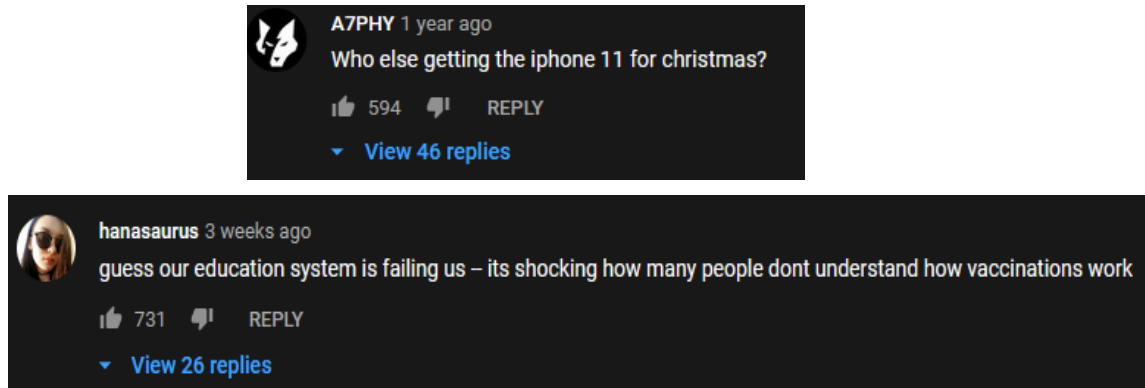


Fig: Screenshots of comments

text	Anas	Billah	Rupu	label
I have 8 of those symptoms	0	0	0	0
90 % of my friends used me :)	0	0	0	0
but you should also add that an overly simulated immune system is also a bad thing	0	0	0	0
Liverpool FC need a 100,000 capacity stadium.. This club has so much loyal fans	1	1	1	1
I have all of them is that normal ?	0	0	0	0
At least China is using proven technology to make their vaccines but it & 39 ; s less effective but it & 39 ; s	1	1	0	1
I miss being healthy . I â€™ m alway in pain , broke and unemployed .	0	0	0	0
Can you please do a remastered version of this	1	1	1	1
Thanks Vox , this content has certainly helped me in spreading the word to my vaccine-hesitant frie	1	1	1	1
Congratulations !! What a journey 2018/19 season !!	1	1	1	1

Fig: Annotations based on voting in MS Excel sheet

The count of comments in each polarity class is shown in the bar chart below:

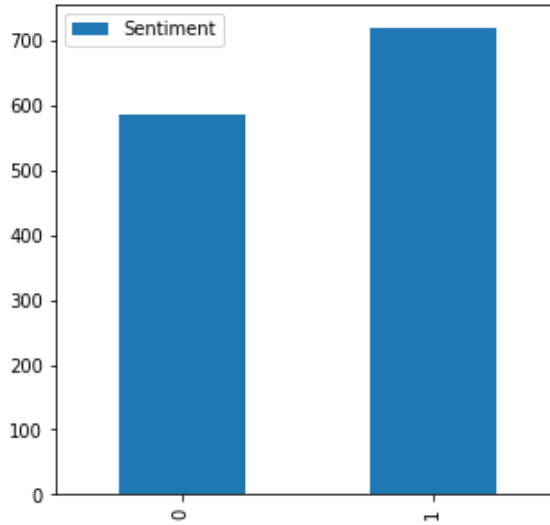


Fig: Frequency distribution of positive and negative polarity

B. Evaluation Metric

- ❑ Loss Graph : A loss is a number indicating how bad the model prediction on a sample dataset is. The loss is calculated on training and validation and its interpretation is how well the model is doing in these two sets.
- ❑ Precision : The precision is calculated as the ratio between the number of positive samples correctly classified to the total number of samples predicted as positive (either correctly or incorrectly).

$$Precision = \frac{T_p}{T_p + F_p}$$

- ❑ Recall : The ratio of correct positive predictions to the total positives examples

$$Recall = \frac{T_p}{T_p + F_n}$$

- ❑ F1 -Score : The F1-score combines the precision and recall of a classifier into a single metric by taking their harmonic mean. F1-score for a classification problem calculated as :

$$F1 - Score = \frac{2(Precision * Recall)}{Precision + Recall}$$

- ❑ Confusion matrix : A Confusion matrix is an N x N matrix used for evaluating the performance of a classification model, where N is the number of target classes. The matrix compares the actual target values with those predicted by the machine

learning or deep learning model. This gives us a holistic view of how well the classification model is performing and what kind of errors it is making.

- ❑ Accuracy : Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations.

$$Accuracy = \frac{T_p + T_n}{T_p + T_n + F_p + F_n}$$

- ❑ Macro avg : Macro-avg is mean average precision/recall/F1 of all classes.
- ❑ Weighted avg : Weighted average is the precision of all classes merged together.

In this experiment we are using accuracy, loss, confusion matrix, precision, recall, f1-score, support for evaluating the models.

C. Results

Bag of Words

Settings 1

- Batch Size :100
- Number of Hidden Layer : 2 (Hidden 1 = 100 , Hidden 2 = 50)
- Model : Shallow Neural Network
- Activation Function : ReLu
- Learning Rate : 5e - 4
- Optimizer: Adam
- Validation Sample : 130

In this setting we achieved a **validation accuracy of 73 %** with having 57 samples as negative and 73 samples as positive sentiment. From the confusion matrix the summation of true positive and true negative are (52 + 43) = **95** which are **correctly classified**. So (130-95) = 35 samples are miss-classified. We found more false negative samples than false positive. So our model has faced the challenge to classify positive sentiment more than negative. From the loss curve it is visible that there is healthy correlation in between training and validation loss. Both losses seemed to reduce and the model trained well .For testing the dataset we separated 130 samples and did not put them in train and valid sets. From our visualization four out of five test samples are correctly classified i.e. Sometimes three are correctly classified.

	precision	recall	f1-score	support
0	0.67	0.75	0.71	57
1	0.79	0.71	0.75	73
accuracy			0.73	130
macro avg	0.73	0.73	0.73	130
weighted avg	0.74	0.73	0.73	130

Figure: Validation Evaluation in Setting 1 of BOW

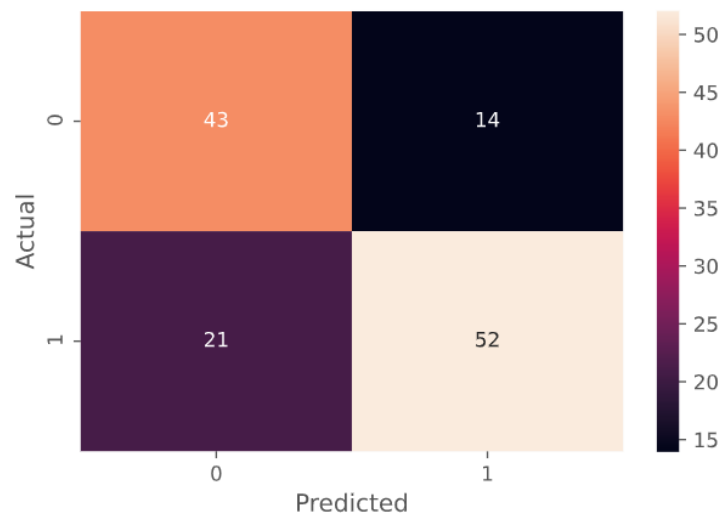


Figure: Confusion matrix in Setting 1 of BOW

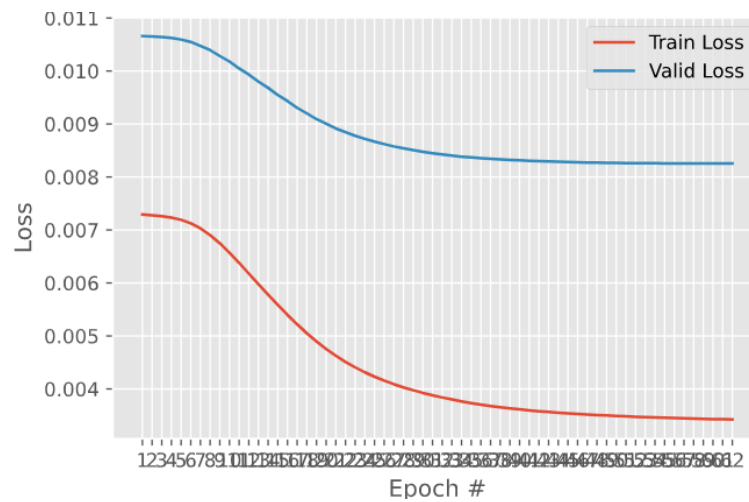


Figure: Loss Graph in Setting 1 of BOW

Settings 2

- Batch Size :100
- Number of Hidden Layer : 2 (Hidden 1 = 100 , Hidden 2 = 50)
- Model : Shallow Neural Network
- Activation Function : LeakyReLU
- Learning Rate : $5e - 4$
- Optimizer: Adam
- Validation Sample : 130

In this setting we changed the activation function from ReLU to LeakyReLU. We achieved a **validation accuracy of 74 %** with having 52 samples as negative and 78 samples as positive sentiment. From the confusion matrix the summation of true positive and true negative are (54 + 42) = **96** which are **correctly classified**. So (130-96) = 34 samples are miss-classified. We found more false negative samples than false positive, the same as setting 1. So our model again has faced the challenge to classify positive sentiment more than negative. From the loss curve it is visible that there is healthy correlation in between training and validation loss. Both losses seemed to reduce and the model trained well .For testing the dataset we separated 130 samples and did not put them in train and valid sets. From our visualization four out of five test samples are correctly classified i.e. Sometimes three are correctly classified.The change in activation function slidely increased the overall validation accuracy.

	precision	recall	f1-score	support
0	0.64	0.81	0.71	52
1	0.84	0.69	0.76	78
accuracy			0.74	130
macro avg	0.74	0.75	0.74	130
weighted avg	0.76	0.74	0.74	130

Figure: Validation Evaluation in Setting 2 of BOW

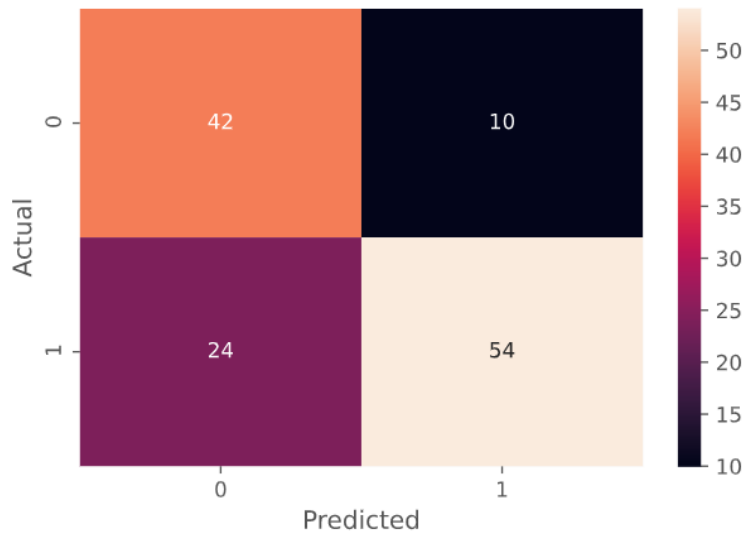


Figure: Confusion in Setting 2 of BOW

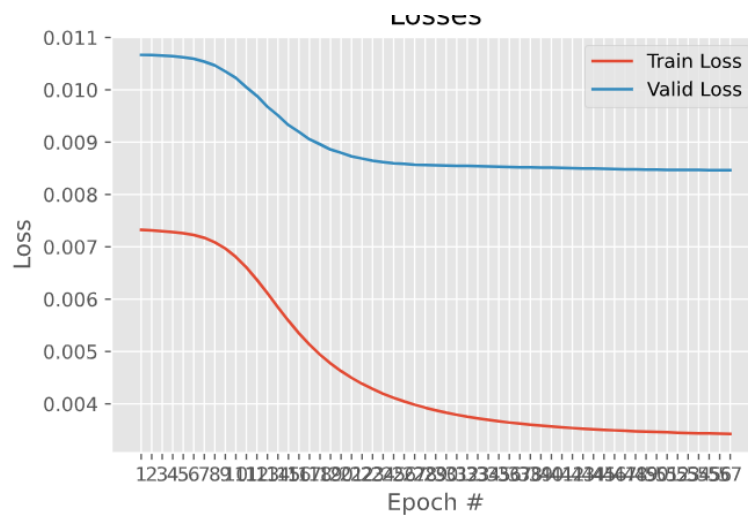


Figure: Loss Graph in Setting 2 of BOW

Settings 3

- Batch Size :100
- Number of Hidden Layer : 2 (Hidden 1 = 100 , Hidden 2 = 50)
- Model : Shallow Neural Network
- Activation Function : SiLU ,ReLU
- Learning Rate : $5e - 4$
- Optimizer: Adam
- Validation Sample : 130

In this setting we changed the both activation functions and took two different activation function. We achieved a **validation accuracy of 73 %** with 53 samples as negative and 77 samples as positive sentiment. From the confusion matrix the summation of true positive and true negative are $(53 + 42) = 95$ which are **correctly classified**. So $(130-95) = 35$ samples are miss-classified. We found 24 false negatives which was more than twice of false positives. So our model has faced the challenge to classify more positive sentiment more than negative again. From the loss curve it is visible that there is healthy correlation at beginning in between training and validation loss. Valid loss seemed to be constant at some point and the model was not trained well, but instead overfitting the training data. For testing the dataset we separated 130 samples and did not put them in train and valid sets. From our visualization four out of five test samples are correctly classified i.e. Sometimes three are correctly classified.

	precision	recall	f1-score	support
0	0.64	0.79	0.71	53
1	0.83	0.69	0.75	77
accuracy			0.73	130
macro avg	0.73	0.74	0.73	130
weighted avg	0.75	0.73	0.73	130

Figure: Validation Evaluation in Setting 3 of BOW

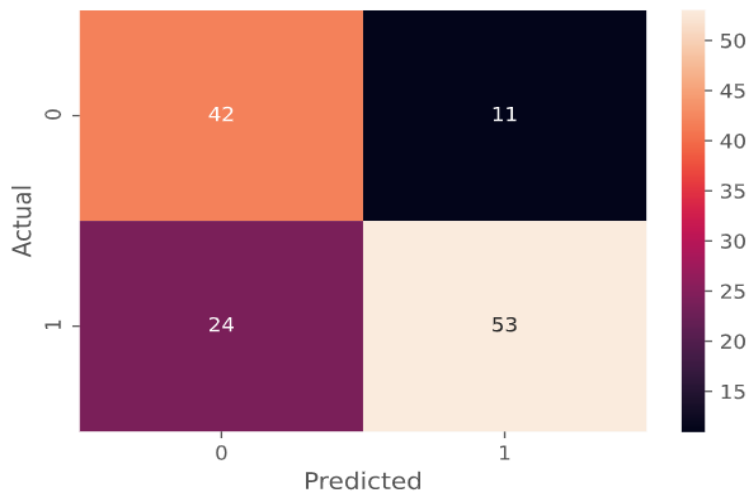


Figure: Confusion Matrix in Setting 3 of BOW

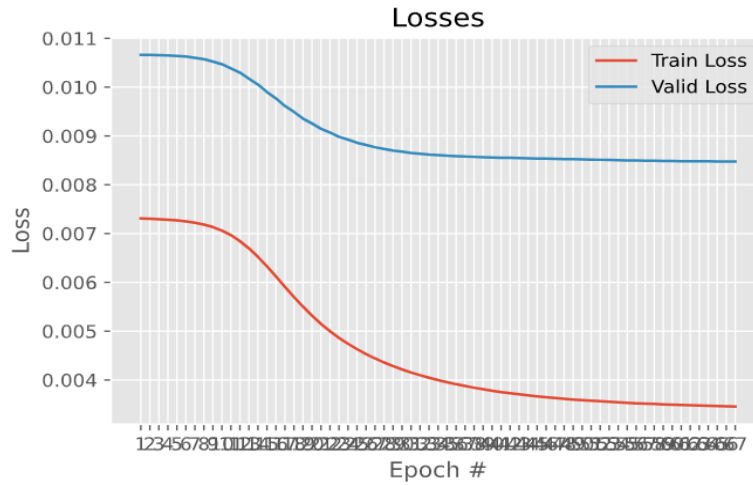


Figure: Loss Graph in Setting 3 of BOW

Settings 4

- Batch Size :128
- Number of Hidden Layer : 1 (Hidden 1 = 100)
- Model : Shallow Neural Network
- Activation Function : SiLU
- Learning Rate : 5e - 4
- Optimizer: Adam
- Validation Sample : 130

In this setting we removed the ReLU activation functions and took one activation function SiLU. Also for this setup one hidden layer shallow Neural Network used. We achieved a **validation accuracy of 74 %** with 50 samples as negative and 80 samples as positive sentiment. From the confusion matrix the summation of true positive and true negative are $(55 + 41) = 96$ which are **correctly classified**. So $(130 - 96) = 34$ samples are miss-classified. We found 25 false negatives which was more than twice of false positives. So our model has faced the challenge to classify more positive sentiment more than negative again. In short changing parameters does not improve the overall accuracy of the model. From the loss curve it is visible that there is healthy correlation at beginning in between training and validation loss. Valid loss seemed to be constant at some point and the model was not trained well, but instead overfitting the training data. For testing the dataset we separated 130 samples and did not put them in train and valid sets. From our visualization four out of five test samples are correctly classified i.e. Sometimes three are correctly classified.

	precision	recall	f1-score	support
0	0.62	0.82	0.71	50
1	0.86	0.69	0.76	80
accuracy			0.74	130
macro avg	0.74	0.75	0.74	130
weighted avg	0.77	0.74	0.74	130

Figure: Validation Evaluation in Setting 4 of BOW

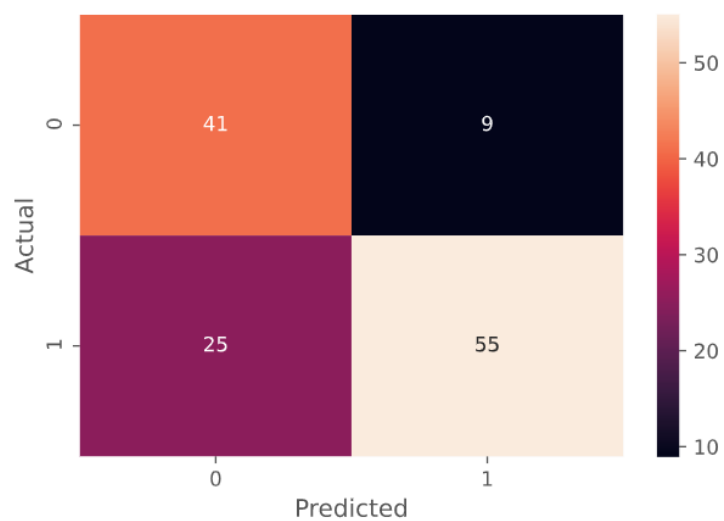


Figure: Confusion Matrix in Setting 4 of BOW

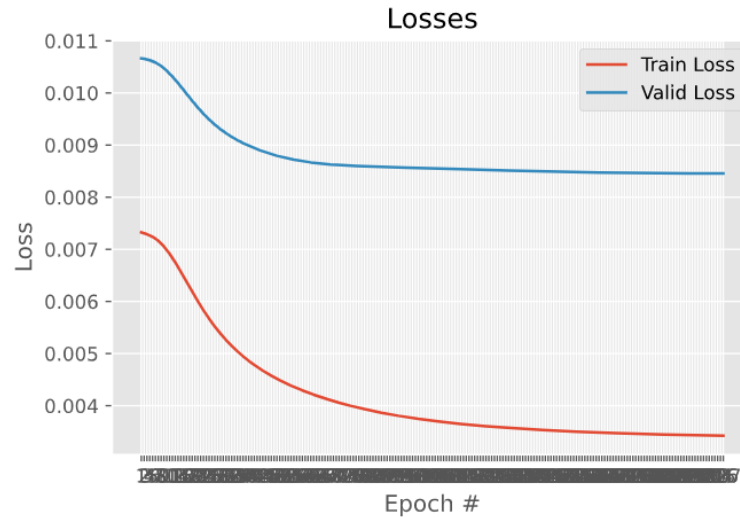


Figure: Loss Graph in Setting 4 of BOW

Settings 5

- Batch Size :128
- Number of Hidden Layer : 1 (Hidden 1 = 100)
- Model : Shallow Neural Network
- Activation Function : ReLU
- Learning Rate : 5e - 4
- Optimizer: Adam
- Validation Sample : 130

In this setting we removed the SiLU activation functions and took one activation function named ReLU..Also for this setup same as the previous one hidden layer shallow Neural Network used. We achieved a **validation accuracy of 75 %** with 53 samples as negative and 77 samples as positive sentiment. From the confusion matrix the summation of true positive and true negative are (54 + 43) = **97** which are **correctly classified**. So (130-97) = 33 samples are miss-classified. We found 23 false negatives which was way ahead of false positives. So our model has again faced the challenge to classify more positive sentiment more than negative again . Also changing parameters does not improve much the overall accuracy of the model .From the loss curve it is visible that there is healthy correlation at beginning in between training and validation loss. Valid loss seemed to be again constant at some point and the model was not trained well ,but instead overfitting the training data .For testing the dataset we separated 130 samples and did not put them in train and valid sets. From our visualization four out of five test samples are correctly classified i.e. Sometimes three are correctly classified.

	precision	recall	f1-score	support
0	0.65	0.81	0.72	53
1	0.84	0.70	0.77	77
accuracy			0.75	130
macro avg	0.75	0.76	0.74	130
weighted avg	0.77	0.75	0.75	130

Figure: Validation Evaluation in Setting 5 of BOW

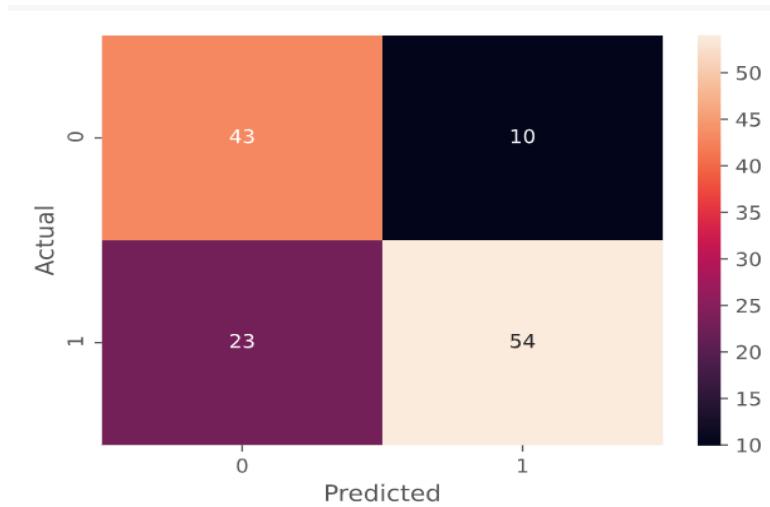


Figure: Confusion Matrix in Setting 5 of BOW

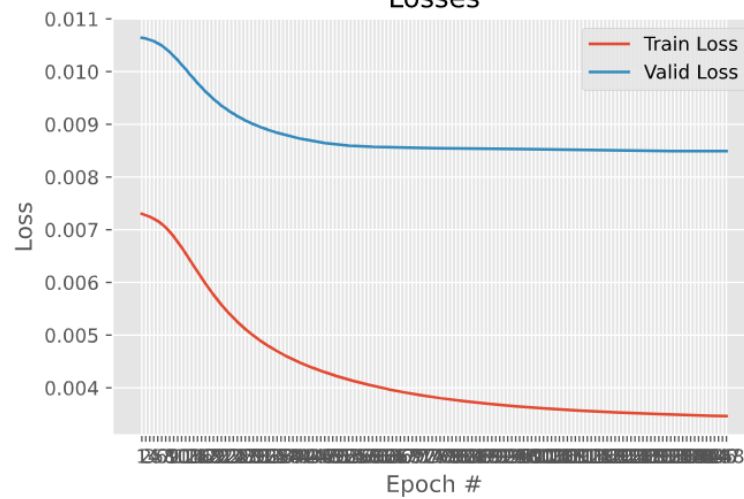


Figure: Loss Graph in Setting 5 of BOW

Summary of All 5 Settings

- Faced more miss classification while classifying positive sentiments. Probably one reason, due to more positive sentiments than negative in the dataset .Second, the train set is so small to learn for this model.
- Model overfit at certain points as the learning stops for the validation set.

TF-IDF

Settings 1

- Batch Size :100
- Number of Hidden Layer : 2 (Hidden 1 = 200 , Hidden 2 = 100)
- Model : Shallow Neural Network
- Activation Function : ReLU
- Learning Rate : $5e - 5$
- Optimizer: Adam
- Validation Sample : 130

In this setting we used ReLU as an activation function and two hidden layers in a shallow neural network. we achieved a **validation accuracy of 75 %** with having 50 samples as negative and 80 samples as positive sentiment. From the confusion matrix the summation of true positive and true negative are (63 + 34) = **97** which are **correctly classified**. So (130-97) = 33 samples are miss-classified. We found almost equal false negative samples than false positive. From the loss curve it is visible that there is healthy correlation in between training and validation loss. Both losses seemed to reduce and the model trained well .For testing the dataset we separated 130 samples and did not put them in train and valid sets. From our visualization four out of five test samples are correctly classified i.e. sometimes three are correctly classified.

	precision	recall	f1-score	support
0	0.67	0.68	0.67	50
1	0.80	0.79	0.79	80
accuracy			0.75	130
macro avg	0.73	0.73	0.73	130
weighted avg	0.75	0.75	0.75	130

Figure: Validation Evaluation in Setting 1 of TF-IDF

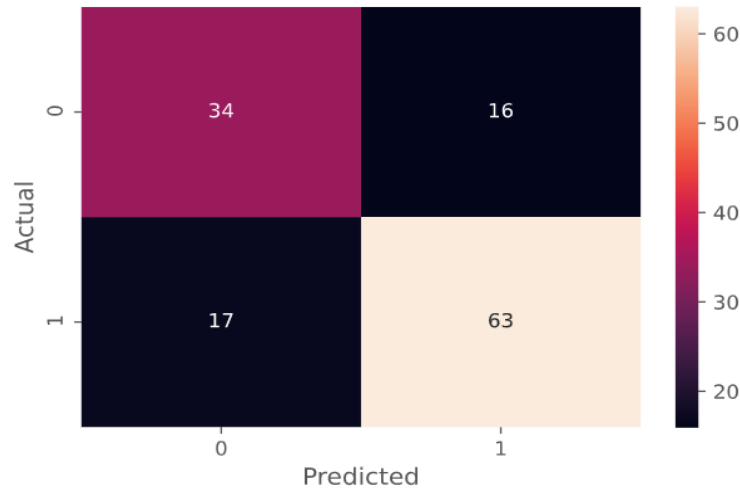


Figure: Confusion Matrix in Setting 1 of TF-IDF

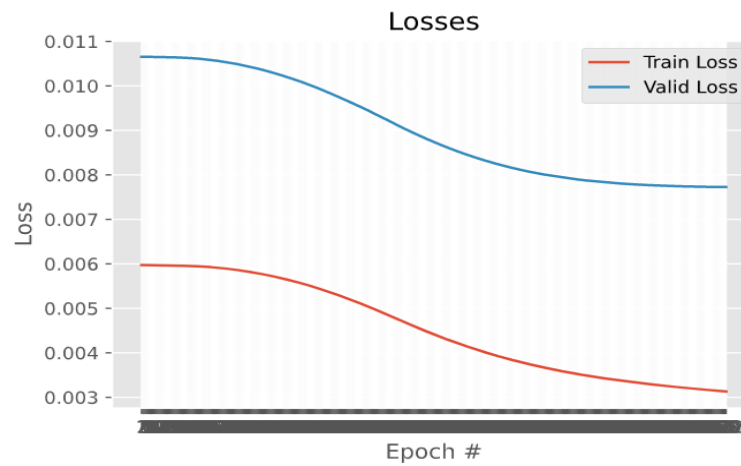


Figure: Loss Graph in Setting 1 of TF-IDF

Settings 2

- Batch Size :100
- Number of Hidden Layer : 1 (Hidden 1 = 200 ,Hidden = 100)
- Model : Shallow Neural Network
- Activation Function : LeakyReLU , ReLU
- Learning Rate : $5e - 5$
- Optimizer: Adam
- Validation Sample : 130

In this setting we changed one of the activation functions from ReLU to LeakyReLU. We achieved a **validation accuracy of 78 %** with having 51 samples as negative and 79 samples as positive sentiment. From the confusion matrix the summation of true positive

and true negative are $(62 + 40) = 102$ which are **correctly classified**. So $(130-102) = 28$ samples are miss-classified. We found more false negative samples than false positive . So our model has faced the challenge to classify positive sentiment more than negative. From the loss curve it is visible that there is healthy correlation in between training and validation loss. Both losses seemed to reduce and the model trained well .But at some point the validation loss was static and did not change .So the model was not improving instead overfit training datas.For testing the dataset we separated 130 samples and did not put them in train and valid sets. From our visualization four out of five test samples are correctly classified .The change of one of activation function increased the overall validation accuracy.

	precision	recall	f1-score	support
0	0.70	0.78	0.74	51
1	0.85	0.78	0.82	79
accuracy			0.78	130
macro avg	0.78	0.78	0.78	130
weighted avg	0.79	0.78	0.79	130

Figure: Validation Evaluation in Setting 2 of TF-IDF

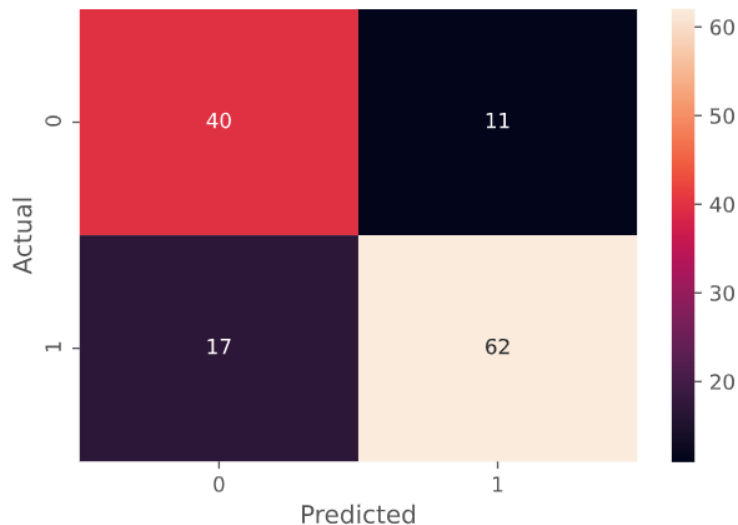


Figure: Confusion Matrix in Setting 2 of TF-IDF

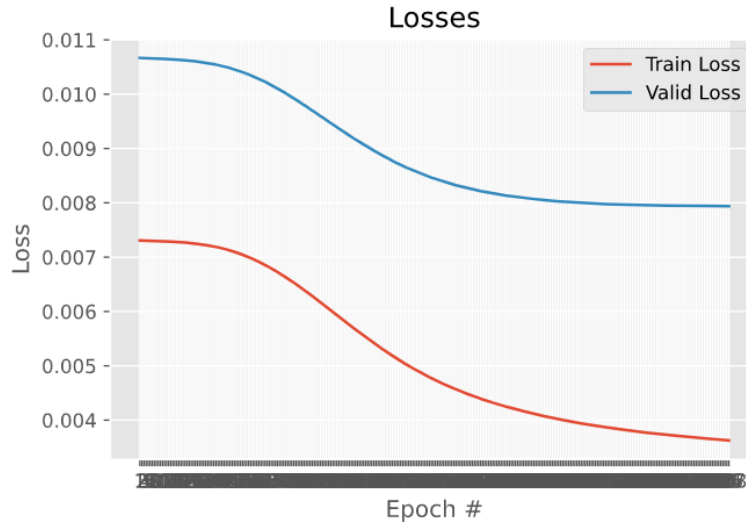


Figure: Loss Graph in Setting 2 of TF-IDF

Settings 3

- Batch Size :100
- Number of Hidden Layer : 1 (Hidden 1 = 100)
- Model : Shallow Neural Network
- Activation Function : TanH
- Learning Rate : $5e - 5$
- Optimizer: Adam
- Validation Sample : 130

In this setting we removed one of the activation functions and changed the other one to TanH. Instead of two hidden layers we used one hidden layer in this setting. We achieved a **validation accuracy of 75 %** with having 65 samples as negative and 65 samples as positive sentiment. So accuracy decreased compared to the previous setting. From the confusion matrix the summation of true positive and true negative are $(49 + 48) = 97$ which are **correctly classified**. So $(130-97) = 33$ samples are miss-classified. We found almost equal false negative samples and false positive. From the loss curve it is visible that there is healthy correlation in between training and validation loss. Both losses seemed to reduce and the model trained well. But at some point the validation loss was static and did not change. So the model was not improving instead overfit training datas. For testing the dataset we separated 130 samples and did not put them in train and valid sets. From our visualization four out of five test samples are correctly classified, sometimes three. The changed in activation function decreased the overall validation accuracy.

	precision	recall	f1-score	support
0	0.75	0.74	0.74	65
1	0.74	0.75	0.75	65
accuracy			0.75	130
macro avg	0.75	0.75	0.75	130
weighted avg	0.75	0.75	0.75	130

Figure: Validation Evaluation in Setting 3 of TF-IDF

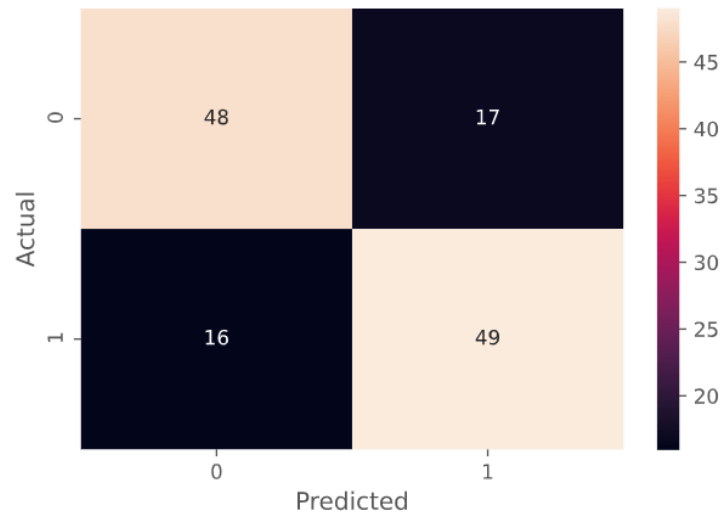


Figure: Confusion Matrix in Setting 3 of TF-IDF

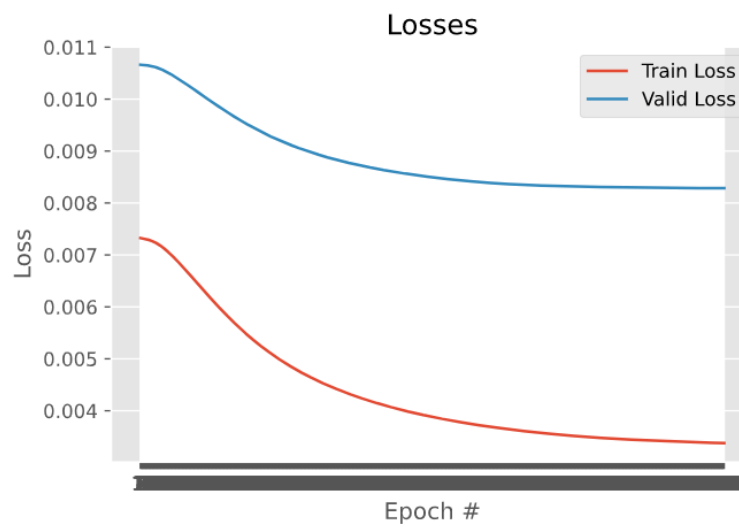


Figure: Loss Graph in Setting 3 of TF-IDF

Settings 4

- Batch Size :128
- Number of Hidden Layer : 1 (Hidden 1 = 100)
- Model : Shallow Neural Network
- Activation Function : SiLU
- Learning Rate : $5e - 5$
- Optimizer: Adam
- Validation Sample : 130

In this setting we removed the activation TanH and changed it with SiLU activation function . We achieved a **validation accuracy of 75 %** with having 49 samples as negative and 81 samples as positive sentiment. Our accuracy still remained the same compared to the previous setting. From the confusion matrix the summation of true positive and true negative are (64 + 34) = **98** which are **correctly classified**. So (130-98) = 32 samples are miss-classified. We found almost equal false negative samples and false positive. From the loss curve it is visible that there is healthy correlation in between training and validation loss at beginning . Both losses seemed to reduce and the model trained well .But at some point the validation loss was static and did not change .So the model was not improving instead overfit training datas . For testing the dataset we separated 130 samples and did not put them in train and valid sets. From our visualization four out of five test samples are correctly classified,sometimes three .The change in activation function did not improve the overall validation accuracy.

	precision	recall	f1-score	support
0	0.67	0.69	0.68	49
1	0.81	0.79	0.80	81
accuracy			0.75	130
macro avg	0.74	0.74	0.74	130
weighted avg	0.76	0.75	0.75	130

Figure: Validation Evaluation in Setting 4 of TF-IDF

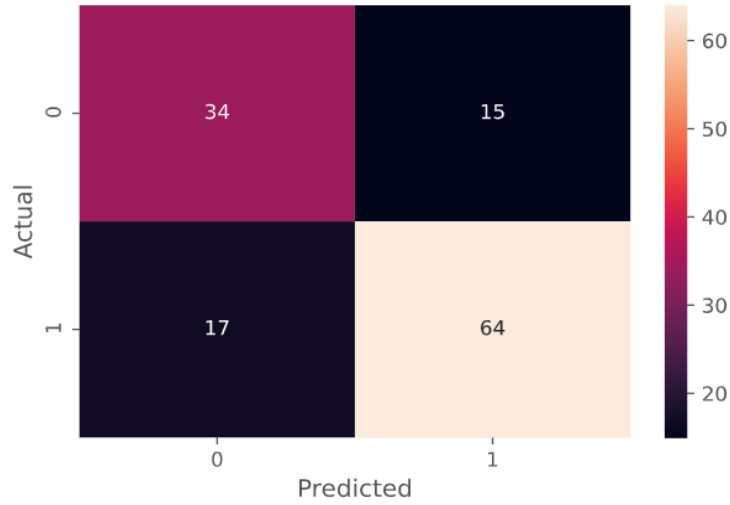


Figure: Confusion Matrix in Setting 4 of TF-IDF

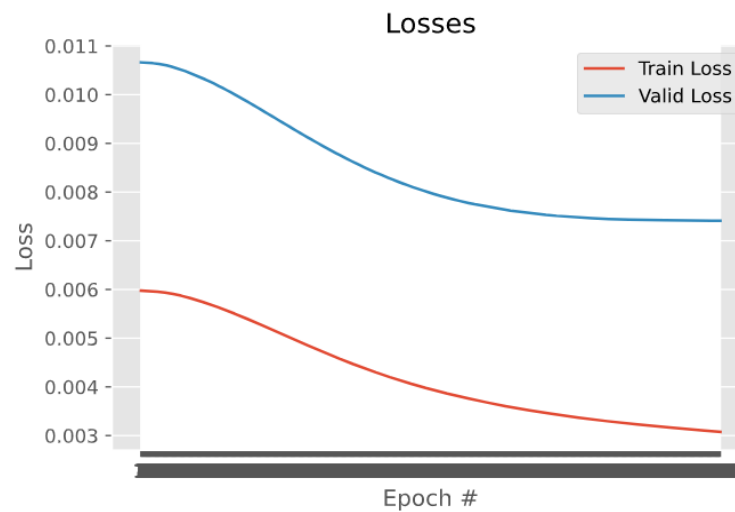


Figure: Loss Graph in Setting 4 of TF-IDF

Settings 5

- Batch Size :128
- Number of Hidden Layer : 1 (Hidden 1 = 100)
- Model : Shallow Neural Network
- Activation Function : ReLU
- Learning Rate : $5e - 5$
- Optimizer: Adam
- Validation Sample : 130

In this setting we removed the activation SiLU and changed it with ReLU activation function . We achieved a **validation accuracy of 80 %** with having 51 samples as negative and 79 samples as positive sentiment. Our accuracy jumped with a **5% hike** compared to the previous setting. From the confusion matrix the summation of true positive and true negative are (63 + 41) = **104** which are **correctly classified**. So (130-104) = 26 samples are miss-classified. We found more false negative samples than false positive. From the loss curve it is visible that there is healthy correlation in between training and validation loss at beginning . Both losses seemed to reduce and the model trained well .But at some point the validation loss was static and did not change .So the model was not improving at that point instead overfit training datas . For testing the dataset we separated 130 samples and did not put them in train and valid sets. From our visualization four out of five test samples are correctly classified .The change in activation function actually improved marginally well in the overall validation accuracy.

	precision	recall	f1-score	support
0	0.72	0.80	0.76	51
1	0.86	0.80	0.83	79
accuracy			0.80	130
macro avg	0.79	0.80	0.79	130
weighted avg	0.81	0.80	0.80	130

Figure: Validation Evaluation in Setting 5 of TF-IDF

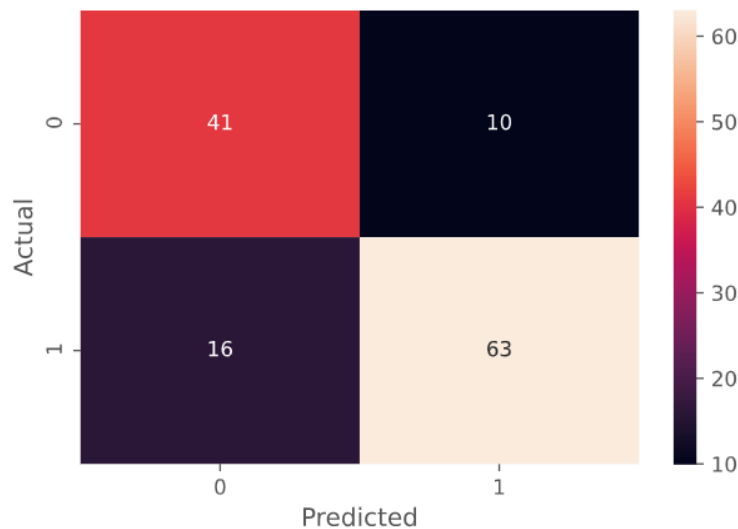


Figure: Confusion Matrix in Setting 5 of TF-IDF

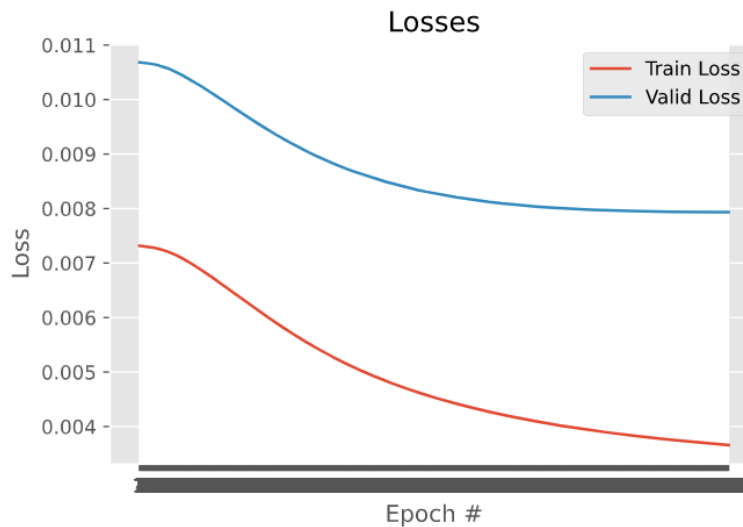


Figure: Loss Graph in Setting 5 of TF-IDF

Summary of All 5 Settings

- Faced more miss classification while classifying positive sentiments. Probably one reason, due to more positive sentiments than negative in the dataset. Second, the train set is so small to learn for this model. Although, the problem arises less compared to BOW.
- Model overfits at certain points as the learning stops for the validation set at some point like BOW.
- Overall miss-classification decreased, validation accuracy had increased from the previous Bow Model.

Comparison Table

All our previous two models we used a stop condition that if the current validation loss is greater than the last three validation losses we called early stop.

Using BOW Technique In different Settings

Settings	Batch Size	Iteration Number	Learning rate	Accuracy	Dataset Length
1	100	63	5e-4	73	1304
2	100	58	5e-4	74	
3	100	50	5e-4	73	
4	128	208	5e-4	74	
5	128	149	5e-4	75	

Using TF - IDF Technique In different Settings

Settings	Batch Size	Iteration Number	Learning rate	Accuracy	Dataset Length
1	100	473	5e-5	75	1304
2	100	289	5e-5	78	
3	100	1875	5e-5	75	
4	128	1457	5e-5	75	
5	128	1139	5e-5	80	

(GloVe100d/GloVe300d/FastText) & RNN

For the following models, Spacy was used as tokenizer instead of NLTK. Three pre-trained word embedding models were used for feature extraction. Then the vector was fed into the RNN embedding layer as input. All the RNN models use categorical cross-entropy loss with softmax activation with two output units.

GloVe100d & RNN

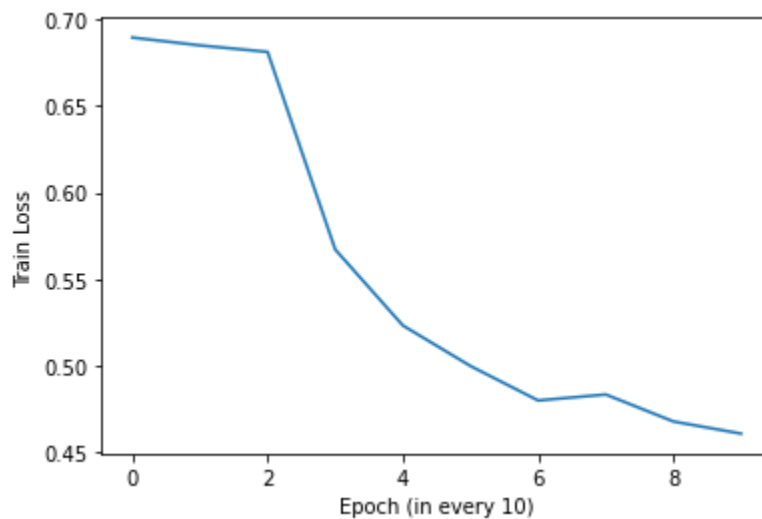
Settings 1

- Batch Size: 400
- Epoch: 100
- Embedding Dimension: 100
- Hidden Units: 500
- Activation Function: Tanh
- Optimizer: Adam
- Learning Rate: $1e-4$

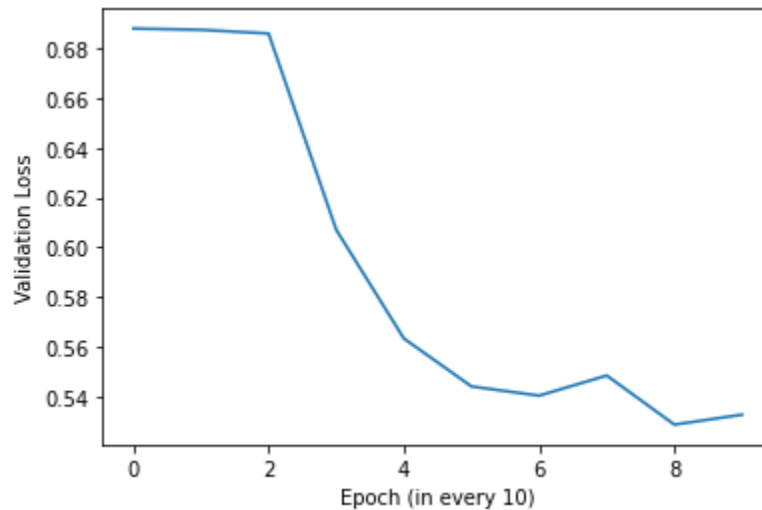
This configuration has a batch size of 400. Most of the other batch size values more or less than 400 performed worse. Our learning rate choices were $1e-4$ or values close to this i.e. $1e-3$ and the number of hidden units was 500 for the same reason. We started with Tanh activation because it is the default one in pytorch but later noticed that models with ReLU activation show high bias or variance.

- ❑ **Val Acc:** 75.38
- ❑ **Val Loss:** 0.535
- ❑ **Training Acc:** 82.04

Though there is a significant gap between training and validation accuracy, 75.38 is a satisfactory validation accuracy score for this dataset.



From the curve of training loss for model-1 it can be observed that initially the training loss was decreasing linearly with a very small slope but after that the loss decreases at a logarithmic rate. The loss has a slight increase during 60-80 epochs.



The validation loss curve has almost the same properties as the one for training but the initial linear decrease part is less steep and the logarithmic decrease part is more steep. Also the loss is more after 60 epochs. After 100 epochs the validation loss becomes 0.535 which is clearly more than training loss at 100 epochs.

Settings 2

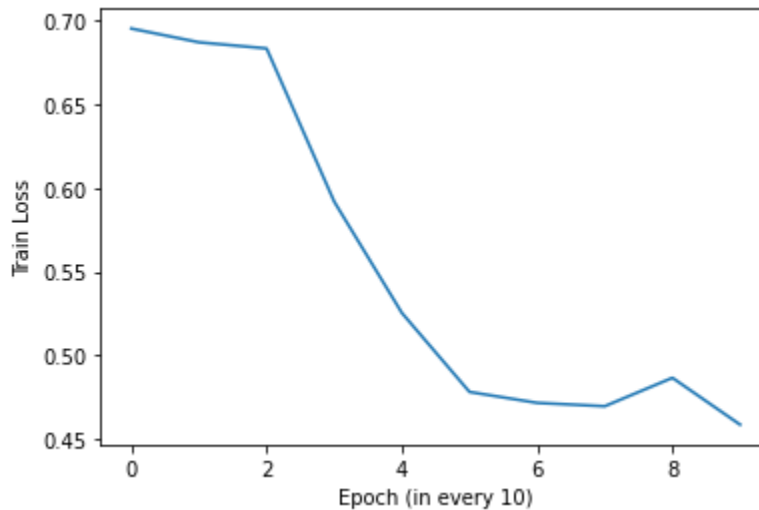
- Batch Size: 128
- Epoch: 100
- Embedding Dimension: 100
- Hidden Units: 500
- Activation Function: Tanh
- Optimizer: Adam
- Learning Rate: $1e - 4$

In this model we tried a different batch size and the rest of the hyperparameters are the same as the previous model.

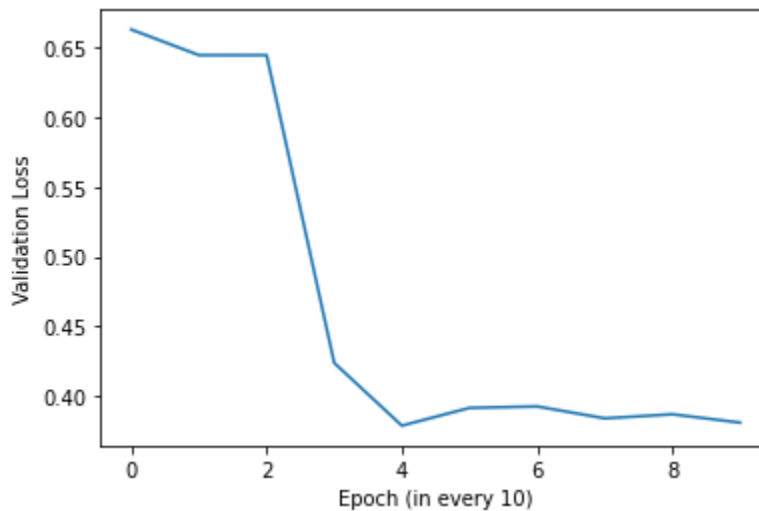
- ❑ **Val Acc:** 85.16
- ❑ **Val Loss:** 0.379
- ❑ **Training Acc:** 81.17

Though performance with 128 batch size is remarkable and the overfitting is reduced, most of the models with batch size choices around 128 (64 or 256) surprisingly lagged far

behind. This setting shows 85.16 validation accuracy which is the second highest for GloVe 100d models.



The training loss curve is almost similar to the previous model. The loss decreases at a logarithmic rate after around 20 epochs. After around 70 epochs, the loss increases and then decreases linearly.



Firstly the validation loss linearly decreases, then it is constant till around 20 epochs. After that the loss decreases linearly with two different slopes, the first one of which is pretty steep. The validation loss after 100 epoch is close to the loss at 40 epochs. In this model the validation loss is less than the training loss.

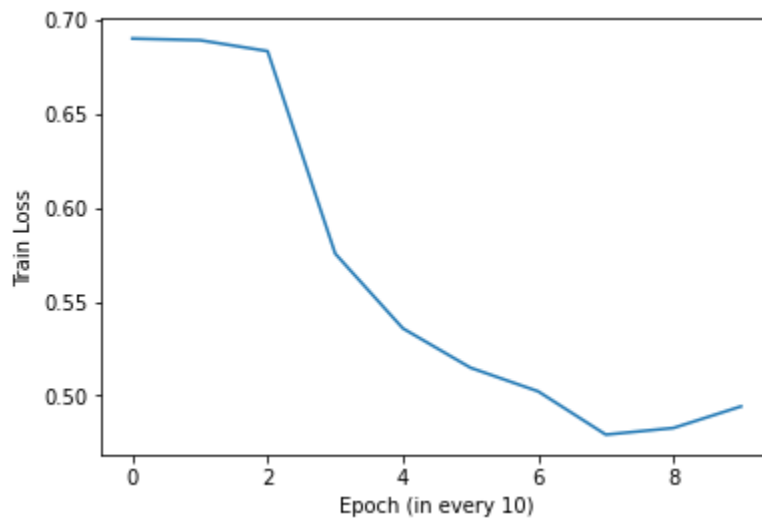
Settings 3

- Batch Size : 128
- Epoch: 100
- Embedding Dimension: 100
- Hidden Units: 500
- Activation Function : Tanh
- Optimizer: RMSprop
- Learning Rate : $1e - 4$

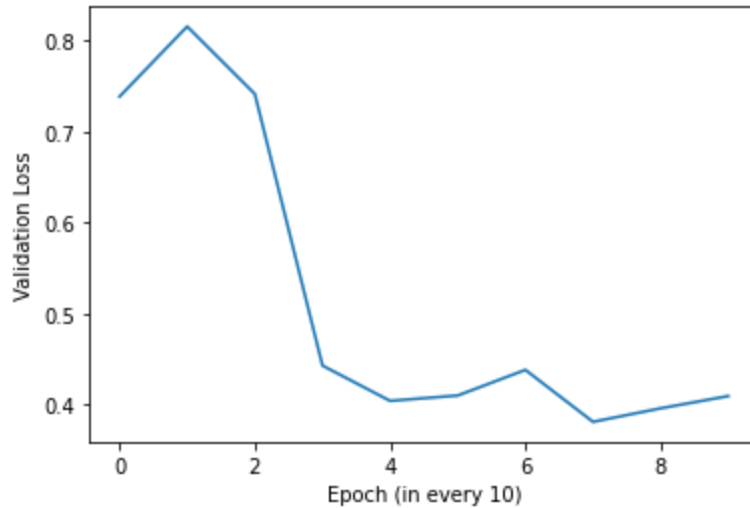
In this configuration we have tested the RMSprop optimizer with the so far best hyperparameter setting. This model is not only the best among the GloVe 100d RNN models but also all the models discussed in this report in terms of accuracy.

- ❑ **Val Acc:** 85.55
- ❑ **Val Loss:** 0.404
- ❑ **Training Acc:** 76.53
- ❑ **Test Acc:** 87.89

It can be noticed that the validation accuracy is a lot more than the training accuracy. Since this is the best model, we evaluated this model with our test set. The test accuracy was 87.89% which is excellent.



The training loss graph is almost similar to the previous settings. After 65-70 epochs, the loss keeps increasing.



The validation loss graph is very different from the previous ones as initially the loss increases till around 0.80. After that the loss linearly keeps decreasing and increasing with different slopes. The final validation loss is 0.404.

GloVe300d & RNN

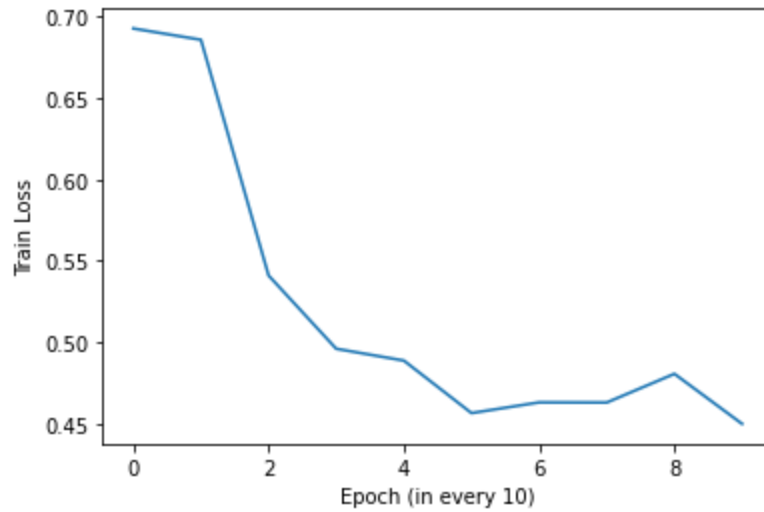
Settings 4

- Batch Size : 128
- Epoch: 100
- Embedding Dimension: 300
- Hidden Units: 500
- Activation Function : Tanh
- Optimizer: RMSprop
- Learning Rate : $1e - 4$

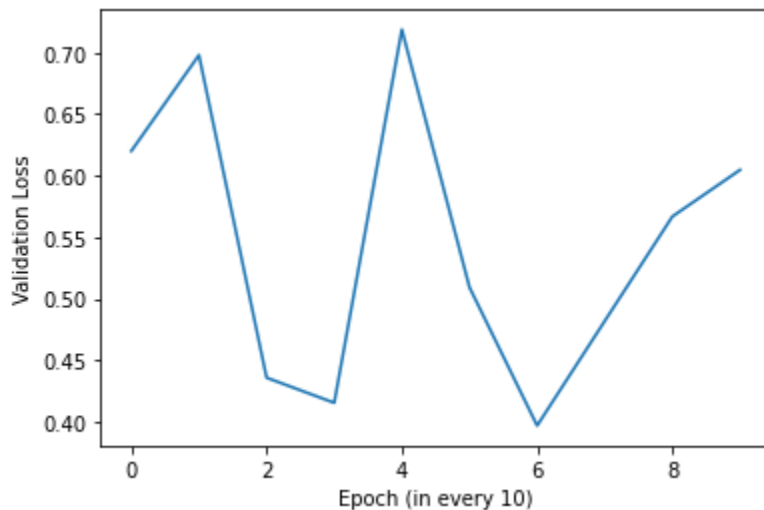
This configuration is the same as configuration 3 which is the best performing setting. Only the embedding dimension is different as the pre-trained model uses a dimension of 300 for word embedding.

- ❑ **Val Acc:** 61.72
- ❑ **Val Loss:** 0.669
- ❑ **Training Acc:** 81.78

From the training and validation accuracy it is obvious that the model is overfitting. There is a huge difference between training and validation accuracy.



The training loss curve is similar to the previous configurations but this time the logarithmic decreasing curve starts early at around 15 epochs.



The validation loss is very unstably increasing and decreasing. All these changes are linear for a long period. The loss reaches its peak at around 43 epochs. This graph shows the highest noise among all models. The final loss is 0.669. From the training and validation loss graphs it is clearly visible that this model has overfitting/ high variance.

Settings 5

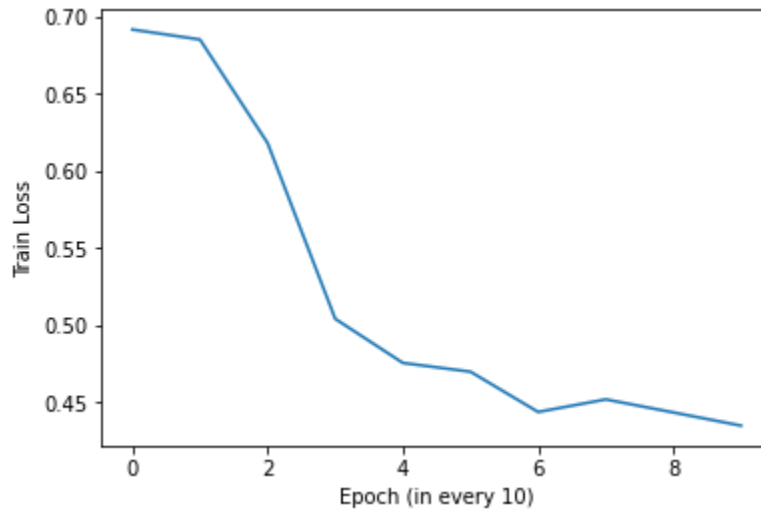
- Batch Size : 400
- Epoch: 100
- Embedding Dimension: 300
- Hidden Units: 1024
- Activation Function : Tanh
- Optimizer: Adam

- Learning Rate : $1e-4$

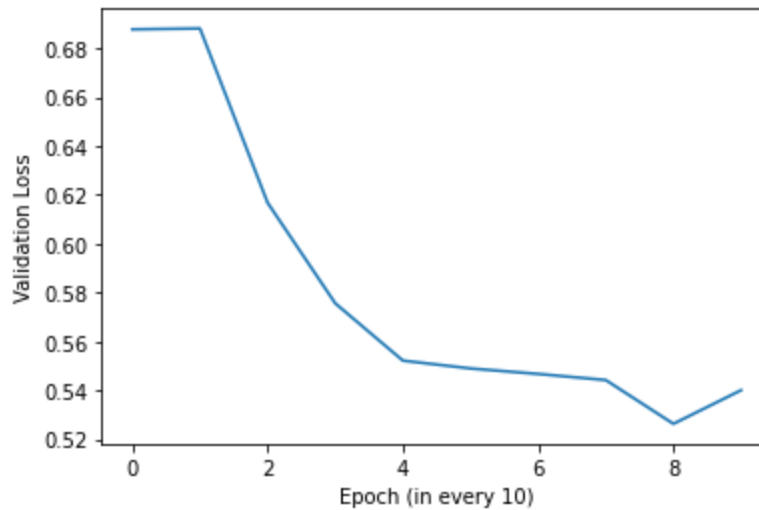
This configuration uses a batch size of 400. All other hyperparameters are similar to the previous configurations except the number of hidden units. In this setting, the hidden unit (1024) is double of all other settings.

- ❑ **Val Acc:** 73.85
- ❑ **Val Loss:** 0.534
- ❑ **Training Acc:** 83.92

The change in the number of hidden units resulted in a decreased validation accuracy.



The training loss graph is similar to the previous one but the loss doesn't increase much in the later stages of the training phase.



The validation loss is constant for the few initial epochs and then logarithmically decreases. The validation loss is 0.534 after 100 epochs.

FastText & RNN

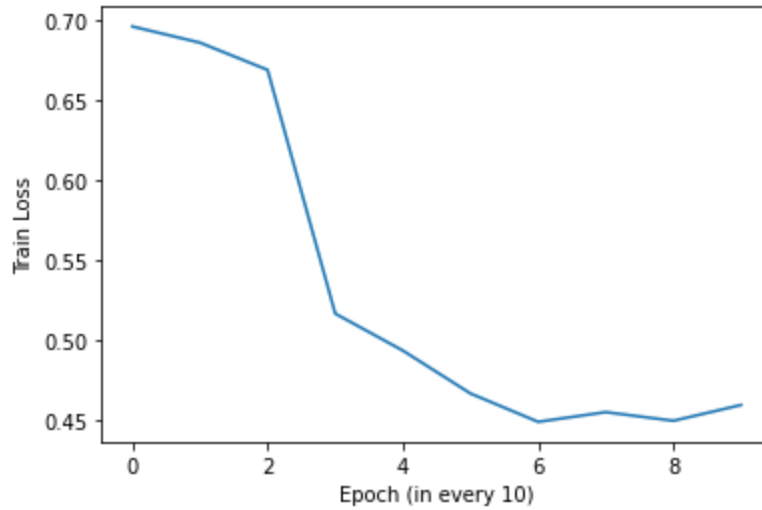
Settings 6

- Batch Size : 128
- Epoch: 100
- Embedding Dimension: 300
- Hidden Units: 500
- Activation Function : Tanh
- Optimizer: RMSprop
- Learning Rate : $1e - 4$

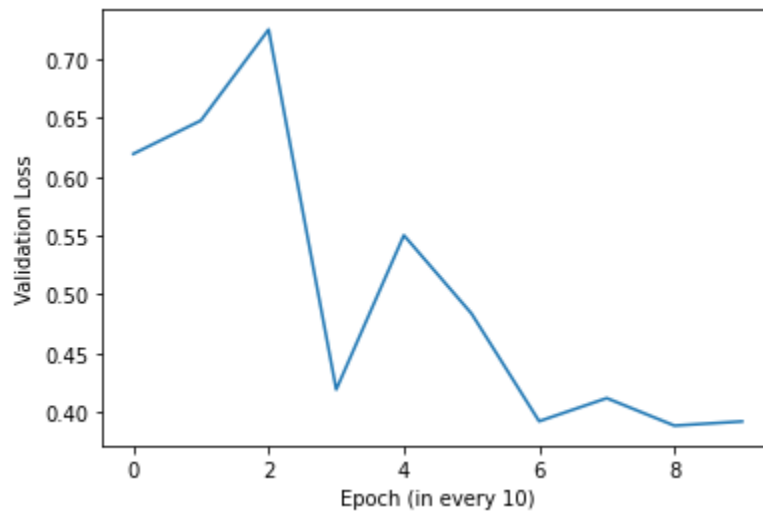
This configuration again uses 500 hidden units and RMSprop optimizer. This model uses FastText for word embedding.

- ❑ **Val Acc:** 84.38
- ❑ **Val Loss:** 0.401
- ❑ **Training Acc:** 82.47

The validation accuracy is 84.38 which is very close to the best model. Validation accuracy is higher than the training accuracy.



The training loss is linearly decreasing till 60 epochs with various slopes. After that it slightly keeps increasing and decreasing.



The final validation loss is 0.401. This loss graph is very different from the other graphs as it initially increases linearly with two different slopes, then decreases and increases linearly and creates two peak loss points but those loss values are smaller. After 100 epochs the validation loss is 0.401

Settings 7

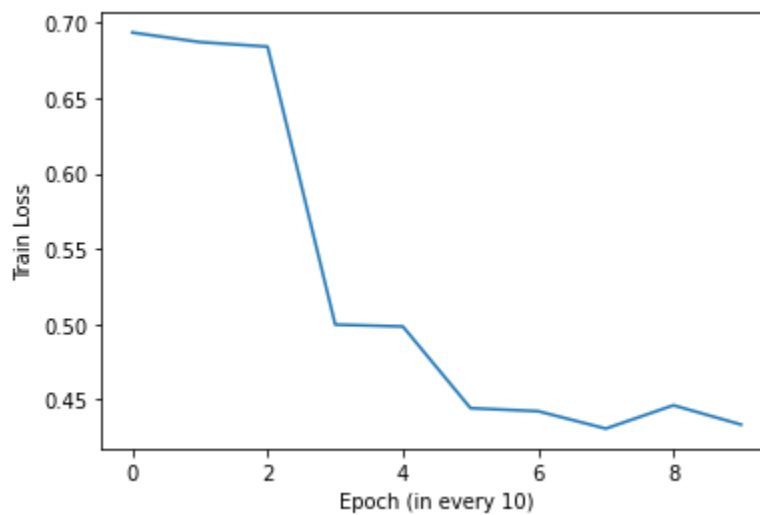
- Batch Size : 128
- Epoch: 100
- Embedding Dimension: 300
- Hidden Units: 500
- Activation Function : Tanh

- Optimizer: Adam
- Learning Rate : $1e - 4$

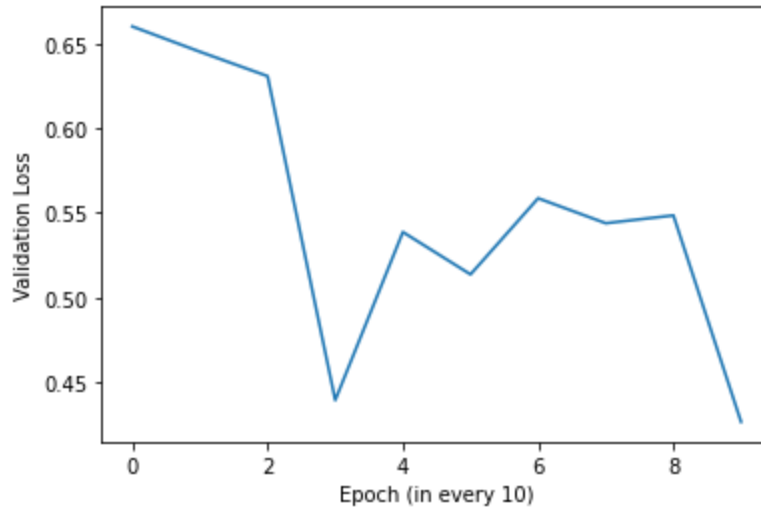
This configuration is the same as the previous setting but it is compiled with Adam optimizer.

- ❑ **Val Acc:** 84.77
- ❑ **Val Loss:** 0.386
- ❑ **Training Acc:** 83.92

The validation accuracy is the highest among all FastText configurations and it is close to the best models of this experiment.



The graph above indicates that the training loss has some almost constant portions throughout the training phase.



The validation loss keeps increasing more than decreasing after 30 epochs. This region is very different from all other models.

Overview of the settings:

- The best model has GloVe-6B-100d word embedding. The GloVe embedding with 300 dimension models lagged behind their 100 dimension counterpart. But we did not train too many configurations with 300 dimension GloVe. The best models with FastText embedding showed remarkable validation accuracy. Still their performance was not better than GloVe 100d models.
- The test accuracy of the best model was 87.89% which was overwhelming . Setting-1 is not the 3rd best model for GloVe 100d in terms of validation accuracy but still we've shown it in this report because it gained an accuracy of 76.34% in the test set which was very surprising.
- In this experiment, we have tried optimizers other than Adam and RMSprop and learning rate values other than $1e-4$ but for those choices, the models performed comparatively poorly. RMSprop optimizer was used to compile the best model.
- In the best models, the number of hidden units in the RNN layer was either 500 or 1024. But layer with 500 neurons was predominant in most of the models.
- RNN in pytorch has two activation function choices, one is Tanh and the other is ReLU. All of the best models have Tanh as activation as in the experiment, using ReLU activation resulted in very low accuracy and high loss.
- Either 128 or 400 batch size was the most frequent choice in most of the best models.
- Using a number of epochs more than 100 did not provide much improvement rather caused overfitting in most of the models.

Glove 100d & RNN

Settings	Batch Size	Epoch	Embedding Dim.	Hidden Units	Activation	Optimizer	Learning rate	Validation Accuracy
1	400	100	100	500	Tanh	Adam	1e-4	75.38
2	128	100	100	500	Tanh	Adam	1e-4	85.16
3	128	100	100	500	Tanh	RMSprop	1e-4	85.55

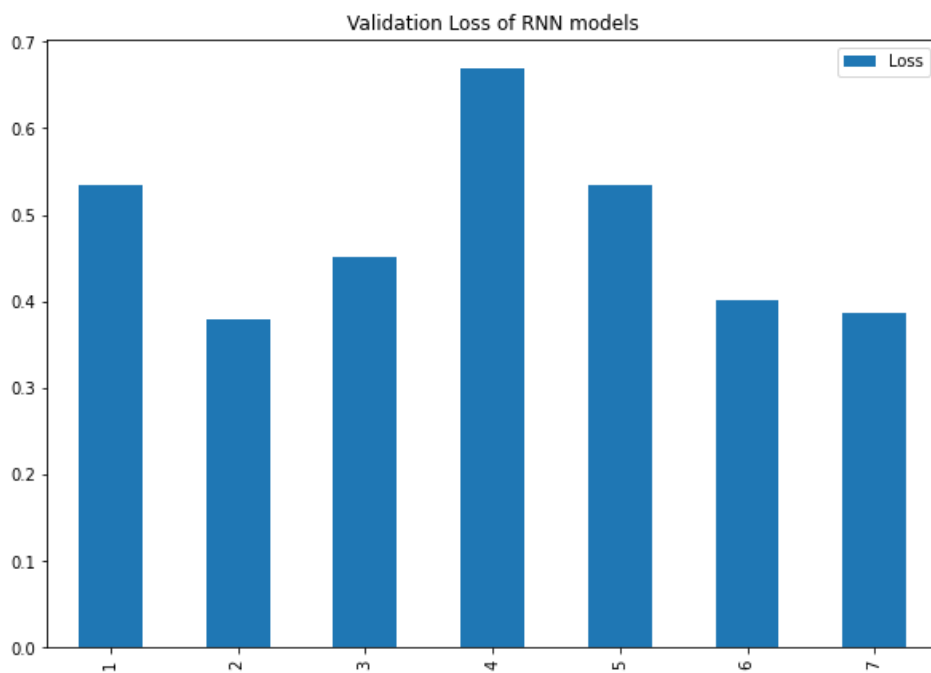
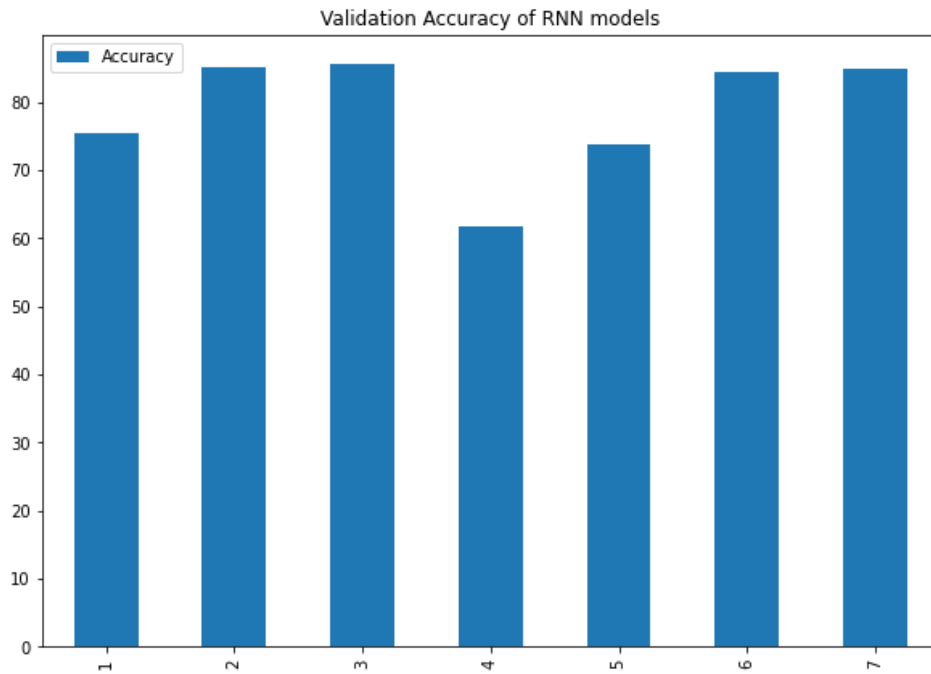
Glove 300d & RNN

Settings	Batch Size	Epoch	Embedding Dim.	Hidden Units	Activation	Optimizer	Learning rate	Validation Accuracy
4	128	100	300	500	Tanh	RMSprop	1e-4	61.72
5	400	100	300	1024	Tanh	Adam	1e-4	73.85

FastText 300d & RNN

Settings	Batch Size	Epoch	Embedding Dim.	Hidden Units	Activation	Optimizer	Learning rate	Validation Accuracy
6	128	100	300	500	Tanh	RMSprop	1e-4	84.38
7	128	100	300	500	Tanh	Adam	1e-4	84.77

The bar charts of accuracy and loss of the RNN models are shown below:



It can be noticed that model 2 has the lowest loss. Though model 3 has the highest accuracy, it doesn't have the lowest loss score.

Conclusion

Research based on detecting sentiment polarity from YouTube comments need a lot more progress. In this experiment, we have put emphasis on problems such as 1)existing sentiment datasets having limitations, 2)annotation of dataset, 3)building a properly annotated dataset, 4)proper preprocessing 5)gaining an above average classification performance etc.

From the result analysis it is evident that higher dimensional word embedding doesn't necessarily ensure higher model performance. 85.55% (GloVe 100d) and 80% (TF-IDF) are the highest validation accuracy for our word embedding+RNN and vectorizer+ANN models respectively. CNN and LSTM should be tried in future for further improving the performance. We would try bigrams and trigrams in the future. The dataset should be expanded for better results. Better preprocessing based on recent papers will be helpful. With such a small dataset, the mentioned configurations showed amazing results. Many configurations of RNN we tried show overfitting. Vanishing gradient of RNN can be a logical reason behind it. We tried with a tiny dataset, so it is reasonable that not all of the best configurations could achieve a decent test accuracy. But most of the trials with different user input texts were predicted correctly by the TF-IDF and BOW models. Using both FastText and GloVe resulted in high evaluation scores but which one is the best variant among these embedding methods for this dataset can not be confidently determined. Both 100d and 300d GloVe helped the RNN gain decent accuracy in the best settings. We explored a vast range of hyperparameter choices for the models.

References

1. Chu, T., Jue, K., & Wang, M. (2016). Comment abuse classification with deep learning. Von <https://web.stanford.edu/class/cs224n/reports/2762092.pdf> abgerufen.
2. Zaheri, S., Leath, J., & Stroud, D. (2020). Toxic comment classification. *SMU Data Science Review*, 3(1), 13.
3. Kazhuparambil, S., & Kaushik, A. (2020). Cooking is all about people: Comment classification on cookery channels using bert and classification models (malayalam-english mix-code). *arXiv preprint arXiv:2007.04249*.
4. Tripto, N. I., & Ali, M. E. (2018, September). Detecting multilabel sentiment and emotions from bangla youtube comments. In *2018 International Conference on Bangla Speech and Language Processing (ICBSLP)* (pp. 1-6). IEEE.
5. Novendri, R., Callista, A. S., Pratama, D. N., & Puspita, C. E. (2020). Sentiment Analysis of YouTube Movie Trailer Comments Using Naïve Bayes. *Bulletin of Computer Science and Electrical Engineering*, 1(1), 26-32.
6. Asghar, M. Z., Ahmad, S., Marwat, A., & Kundi, F. M. (2015). Sentiment analysis on youtube: A brief survey. *arXiv preprint arXiv:1511.09142*.
7. Ansari, U. M. M. Z. Sentiment Analysis on Youtube Comments: A brief study.

8. Uryupina, O., Plank, B., Severyn, A., Rotondi, A., & Moschitti, A. (2014, May). SenTube: A Corpus for Sentiment Analysis on YouTube Social Media. In *LREC* (pp. 4244-4249).
9. https://en.wikipedia.org/wiki/SpaCy#cite_note-5
10. https://en.wikipedia.org/wiki/SpaCy#cite_note-Bird-Klein-Loper-Baldrige-6
11. <https://www.kaggle.com/datasnaek/youtube-new>
12. <https://www.kaggle.com/datasnaek/youtube>
13. https://markroxor.github.io/gensim/static/notebooks/Word2Vec_FastText_Comparison.html
14. <https://towardsdatascience.com/short-technical-information-about-word2vec-glove-and-fasttext-d38e4f529ca8>
15. https://www.nltk.org/_modules/nltk/sentiment/vader.html
16. <https://textblob.readthedocs.io/en/dev/>
17. https://github.com/MAN1986/LearningOrbis/blob/master/scrapeCommentsWithoutReplies.gs?fbclid=IwAR17W_uZ1rFGtw5Wesxa_CpaX-UjLOnMqJO6ca28c667MYnfFUEqCHira4s
18. https://www.analyticssteps.com/blogs/extracting-pre-processing-youtube-comments?fbclid=IwAR0iOTlxqULkMFmB5rK1ziuE_JjT0TXaEZW0BQ5Jyszid7-wlJ8h-Z3CH6g
19. <https://script.google.com/>
20. <https://unicode.org/emoji/charts/full-emoji-list.html>