# Deep Learning Models for Sentiment Analysis

- Ashutosh Agrawala

***Abstract*- Sentiment Analysis is one of the major areas of focus where deep learning has found an application. Sentiment Analysis has been applied to analyze movie reviews, tweets, newspaper articles etc. to gain insightful information about people's opinions and views on topics of national and international importance. In this report we will look at various ways of analyzing movie reviews from the famous IMDB movie review dataset along with pretrained word embeddings used to train different deep learning architectures discussed in the paper.**

## INTRODUCTION

Deep Learning and neural networks have become very popular in the recent past owing to the availability of high compute power in the form of GPU's and TPU's as well as their ability of being universal approximators giving state of the art results for almost all kinds of applications and use cases. Sentiment analysis is one such use case which provides important and insightful information about people's opinions and views on topics of high importance. In this exercise we use the IMDB movie review dataset to showcase the results obtained from various neural network architectures. In this project we used the wiki-news 300d pretrained word embedding file for converting each word form of our movie reviews into a tensor of length 300.

There are various neural network architectures available for sentiment Analysis which can be used to obtain state of the art results for this use case. The simplest model is the one that we call the **Baseline** Model which consists of an embedding layer which converts each word of the input reviews into a tensor of 300 elements using the pretrained word embedding file and a fully connected layer followed by a sigmoid for converting our outputs into probabilities. This model is the base for any other complex architecture and we keep adding architectures between these to obtain better results.

The best form of architecture in general for NLP applications are Recurrent neural network architectures. All the RNN architectures have some form of memory which can be used to detect sequences in texts. The different kinds of RNN that have been explored in the project include Simple Vanilla RNN's, LSTM networks and Gated recurrent units. All these layers of mentioned architectures were added between the embedding layer and the FCN of the Baseline Model and experimented with. Another architecture which is the most recent one is the self - attention Layer which is a great solution for sequence to sequence models and using key and query to highlight the most important observations for an input following the query. All the mentioned architectures were experimented with by tuning various hyperparameters in the model and observations were noted.

## PREPROCSSING STEPS FOR THE INPUT

### 1-loading the Dataset:

This step involved loading each of the movie reviews from the given tar file of movie reviews into a list of reviews and a list of their corresponding labels without actually untarring the file of the movie reviews.

### 2-Preprocessing the text in the movie review:

This step involved extracting all the unique words found in the list of movie reviews obtained and creating a dictionary out of it and associating each word with a unique integer and thus generating a token dictionary for our lookup.

### 3-Encoding reviews:

In this step we encoded the reviews into a list of integers using the lookup dictionary we created in the previous step.

### 4-Encoding labels:

We also encoded the corresponding labels obtained in the first step into 0's and 1's. 0 for negative and 1 for positive.

### 5-Padding Sequences:

Based on a fixed sequence length we zero padded the encoded movie reviews if they were less than the sequence length else ignored the part after the sequence length to make all the movie reviews of same length.

### 6-Building word embedding dictionary:

This step involved the conversion of all pretrained words in the token dictionary obtained in step 2 into a tensor of dimension 300 using the pretrained word embedding dictionary.

### 7-Creating a Data Loader:

In this step I created a tensor dataset of corresponding movie reviews and labels and then patched them together into a data loader object with a batch size of 50 as the batch size needs to be an integral multiple of the number of the number of training examples which are 1900 reviews in our case.

**MODEL IMPLEMENTATION AND RESULTS**

**Model 1-BaseLine Model:**

Output size = 1
Learning rate = 0.1
epochs = 100
neurons in FCN = 16

BaseSentiment(

(embedding): Embedding(54720, 300)
(fcn): Linear(in_features=32, out_features=1, bias=True)
(sigmoid): Sigmoid()

)

**Accuracy = 53%**

**Model 2-RNN using LSTM:**

Output size = 1
Learning rate = 0.01
epochs = 10
number of lstm cells = 16
neurons in FCN = 16

**Unidirectional**:
Layers = 1
RNNSentiment(

(embedding): Embedding(54720, 300)
(lstm): LSTM(300, 16, batch_first=True)
(dropout): Dropout(p=0.5, inplace=False)
(fcn): Linear(in_features=16, out_features=1, bias=True)
(sigmoid): Sigmoid()

)

**Accuracy = 61%**

Layers=2
RNNSentiment(
(embedding): Embedding(54720, 300)
(lstm): LSTM(300, 16, num_layers=2, batch_first=True)
(dropout): Dropout(p=0.5, inplace=False)
(fcn): Linear(in_features=16, out_features=1, bias=True)
(sigmoid): Sigmoid()
)

**Accuracy = 60%**

**Bidirectional**:
Layers = 1
RNNSentiment(
(embedding): Embedding(54720, 300)
(lstm): LSTM(300, 16, batch_first=True, bidirectional=True)
(dropout): Dropout(p=0.5, inplace=False)
(fcn): Linear(in_features=16, out_features=1, bias=True)
(sigmoid): Sigmoid()
)

**Accuracy = 65%**

Layers =2
RNNSentiment(
(embedding): Embedding(54720, 300)
(lstm): LSTM(300, 16, num_layers=2, batch_first=True, bidirectional=True)
(dropout): Dropout(p=0.5, inplace=False)
(fcn): Linear(in_features=16, out_features=1, bias=True)
(sigmoid): Sigmoid()
)

**Accuracy = 62%**

**Model 3- RNN Simple Vanilla:**

Output size = 1
Learning rate = 0.05
epochs = 20
number of rnn cells = 32
neurons in FCN = 32

**Unidirectional:**
Layers = 1
SimpleRNNSentiment(
(embedding): Embedding(54720, 300)
(rnn_cell): RNN(300, 32, batch_first=True)
(dropout): Dropout(p=0.5, inplace=False)
(fcn): Linear(in_features=32, out_features=1, bias=True)
(sigmoid): Sigmoid()
)

**Accuracy = 52%**

Layers = 2
SimpleRNNSentiment(
(embedding): Embedding(54720, 300)
(rnn_cell): RNN(300, 32, num_layers=2, batch_first=True)
(dropout): Dropout(p=0.5, inplace=False)
(fcn): Linear(in_features=32, out_features=1, bias=True)
(sigmoid): Sigmoid()
)

**Accuracy = 53%**

**Unidirectional:**
Layers = 1
SimpleRNNSentiment(
(embedding): Embedding(54720, 300)
(rnn_cell): RNN(300, 32, batch_first=True, bidirectional=True)
(dropout): Dropout(p=0.5, inplace=False)
(fcn): Linear(in_features=32, out_features=1, bias=True)
(sigmoid): Sigmoid()
)

**Accuracy = 56%**

Layers = 2
SimpleRNNSentiment(
(embedding): Embedding(54720, 300)
(rnn_cell): RNN(300, 32, num_layers=2, batch_first=True, bidirectional=True)
(dropout): Dropout(p=0.5, inplace=False)
(fcn): Linear(in_features=32, out_features=1, bias=True)
(sigmoid): Sigmoid()
)

**Accuracy = 51%**

**Model 4- RNN GRU:** **Accuracy = 89%**

output_size = 1
learning_rate = 0.01
epochs = 20
number of gru units = 16

**Unidirectional:**
Layers = 1
GRURNNSentiment(
  (embedding): Embedding(54720, 300)
  (gru): GRU(300, 16, batch_first=True)
  (dropout): Dropout(p=0.5, inplace=False)
  (fcn): Linear(in_features=16, out_features=1, bias=True)
  (sigmoid): Sigmoid()
)

**Accuracy = 77%**

Layers = 2
GRURNNSentiment(
  (embedding): Embedding(54720, 300)
  (gru): GRU(300, 16, num_layers=2, batch_first=True)
  (dropout): Dropout(p=0.5, inplace=False)
  (fcn): Linear(in_features=16, out_features=1, bias=True)
  (sigmoid): Sigmoid()
)

**Accuracy = 73%**

**Bidirectional**:
Layers = 1
GRURNNSentiment(
  (embedding): Embedding(54720, 300)
  (gru): GRU(300, 16, batch_first=True, bidirectional=True)
  (dropout): Dropout(p=0.5, inplace=False)
  (fcn): Linear(in_features=16, out_features=1, bias=True)
  (sigmoid): Sigmoid()
)

**Accuracy = 70%**

Layers = 2
GRURNNSentiment(
  (embedding): Embedding(54720, 300)
  (gru): GRU(300, 16, num_layers=2, batch_first=True,
bidirectional=True)
  (dropout): Dropout(p=0.5, inplace=False)
  (fcn): Linear(in_features=16, out_features=1, bias=True)
  (sigmoid): Sigmoid()
  )

**Accuracy = 87%**

**Model 5- Self Attention:**

output_size = 1
learning_rate = 0.01
epochs = 4
bilstm units = 16

AttentionSentiment(
  (word_embeddings): Embedding(54720, 300)
  (bilstm): LSTM(300, 16, dropout=0.5, bidirectional=True)
  (W_s1): Linear(in_features=32, out_features=350, bias=True)
  (W_s2): Linear(in_features=350, out_features=30, bias=True)
  (fc_layer): Linear(in_features=960, out_features=2000, bias=True)
  (label): Linear(in_features=2000, out_features=1, bias=True)
  (sigmoid): Sigmoid()
)