

Indian Institute of Technology, Jodhpur

Natural Language Understanding

Assignment 1 Report

Submitted by:

1. Debonil Ghosh (M21AIE225)
2. Ravi Shankar Kumar(M21AIE247)
3. Saurav Chowdhury(M21AIE256)

Debonil Ghosh

Roll No: M21AIE225

Executive MTech Artificial Intelligence

Indian Institute of Technology, Jodhpur

Dataset used: IMDB Dataset

1. *Define your own train-val-test split. [Report the split chosen.]*

Train: Test: Val set : 18:5:2

Train set :36000

Test Set: 10000

Val Set: 4000

1. *Define a text pre-processing pipeline, i.e., stopword removal, lower casing, punctuation removal etc.*

|  |
| --- |
| The movie is good , I loved the movie. Will audience like it ?!! |
| Pipeline 1:Remove ('[/(){}\[\]\|@,;]') |
| The movie is good I loved the movie. Will audience like it |
| Pipeline 2: Remove('\_') |
| ThemovieisgoodIlovedthemovieWillaudiencelikeit |
| Pipeline 3: Stopwords |
| moviegoodlovedmoviewillaudiencelike |

1. *Developing ML methods:*
2. *Model a Naive Bayes classifier.*

*i.Count vectorizer features.*

*ii. TF-IDF features.*

Multinomial Naïve Bayes Classifier used.

Count Vectorizer:

IT will create a matrix of count of each word in each of the reviews data

Accuracy : 86.14 %

TF-IDF:

TF= Number of time word repeated in a sentence/ Number of words in sentence

IDF = Log(Number of sentences/Number of sentences containing the word)

TF\*IDF = Dependent Feature

Accuracy: 86.91 %

*II. Model a* ***decision tree*** *with TF-IDF features:*

To minimise the dimensionality of the feature space, we can preprocess the text data by tokenizing it into words, deleting stop words and punctuation, and using stemming or lemmatization. The TF-IDF transformation can then be used to generate a matrix of TF-IDF scores for each word in each review. To classify incoming reviews as positive or negative based on the TF-IDF features that are most discriminative for each class, we can lastly train a decision tree on the TF-IDF matrix. The decision tree that is produced can be used to examine the most significant words and phrases that affect the reviews' sentiment.

Accuracy: 71.74 %

Comparing TF\*IDF and Decision tree with TF\_IDf : we see that the

Accuracy of TF\*IDF is more.

1. *Developing Deep neural networks:*

***a.RNN model.***

* + 1. *64 hidden-vector dimension.*

Simple RNN model with 64 hidden layers.

Model: "SimpleRNNModel64"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

embedding\_6 (Embedding) (None, 64, 64) 8900032

simple\_rnn\_6 (SimpleRNN) (None, 64) 8256

dense\_6 (Dense) (None, 64) 4160

dropout (Dropout) (None, 64) 0

dense\_7 (Dense) (None, 1) 65

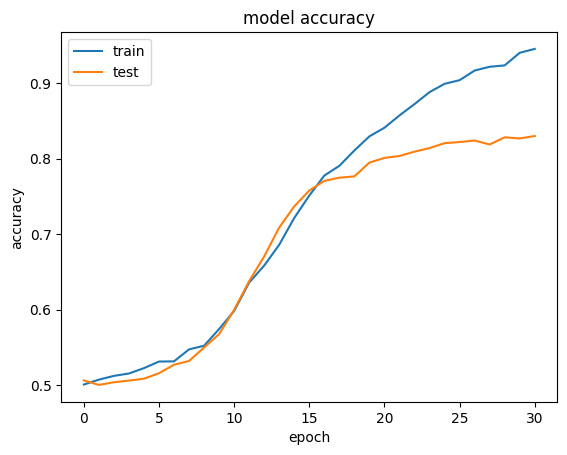
=================================================================

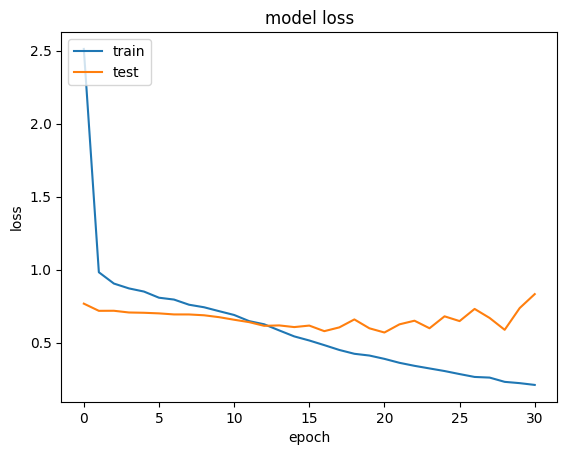
Total params: 8,912,513

Trainable params: 8,912,513

Non-trainable params: 0

The layers of the RNN are shown above





Classification Report

precision recall f1-score support

0 0.85 0.83 0.84 5044

1 0.83 0.85 0.84 4956

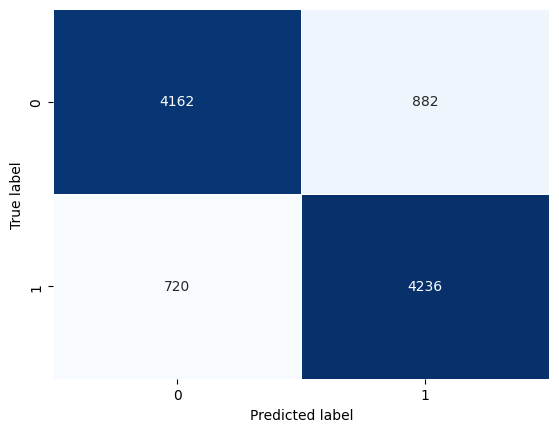
accuracy 0.84 10000

macro avg 0.84 0.84 0.84 10000

weighted avg 0.84 0.84 0.84 10000

Accuracy : 0.8398

Accuracy :83.98 %



Confusion Matrix

* + 1. *256 hidden-vector dimension.*

Simple RNN model with 256 hidden layers.

Model: "SimpleRNNModel256"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

embedding\_10 (Embedding) (None, 64, 256) 35600128

simple\_rnn\_10 (SimpleRNN) (None, 256) 131328

dense\_14 (Dense) (None, 256) 65792

dropout\_4 (Dropout) (None, 256) 0

dense\_15 (Dense) (None, 1) 257

=================================================================

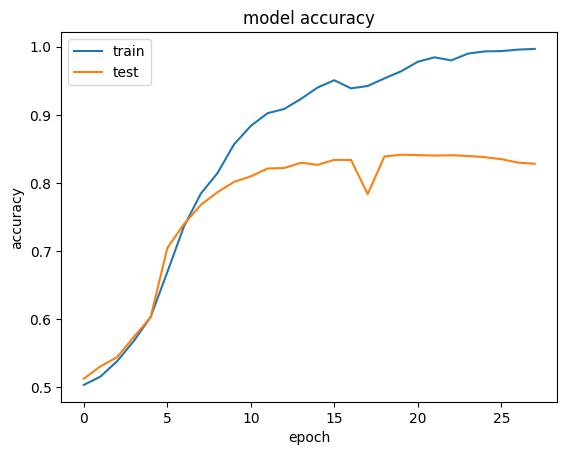
Total params: 35,797,505

Trainable params: 35,797,505

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layers of the RNN are shown above





Classification Report

precision recall f1-score support

0 0.84 0.82 0.83 5044

1 0.82 0.85 0.83 4956

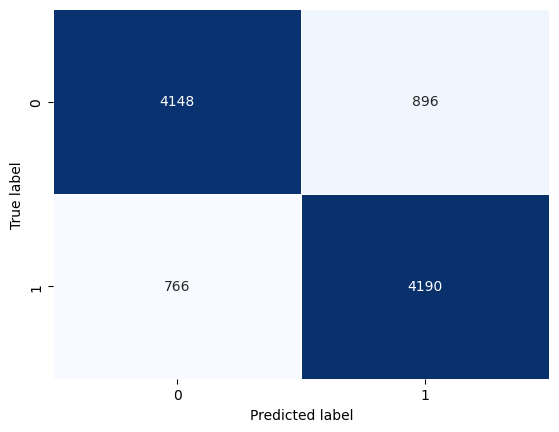
accuracy 0.83 10000

macro avg 0.83 0.83 0.83 10000

weighted avg 0.83 0.83 0.83 10000

Accuracy : 0.8338

Accuracy : 83.38 %



Confusion Matrix

***4.b 1-layer LSTM model.*** *[choose 64 or 256 as hidden-vector representation based on the results from 4.a. Report the choice and its justification.]*

We have chosen 64 layer LSTM because our dataset (IMDB)dataset has only 2 classes and accuracy of performance for both 64 and 256 will be similar. Moreover the training duration of 64 Layer LSTM is less compared to 256 Layer LSTM. Over that it also solves the problem of overfitting which was evident in this case.

Model: "SingleLSTMLayer64"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

embedding\_14 (Embedding) (None, 64, 256) 35600128

lstm\_3 (LSTM) (None, 64) 82176

dense\_20 (Dense) (None, 64) 4160

dropout\_7 (Dropout) (None, 64) 0

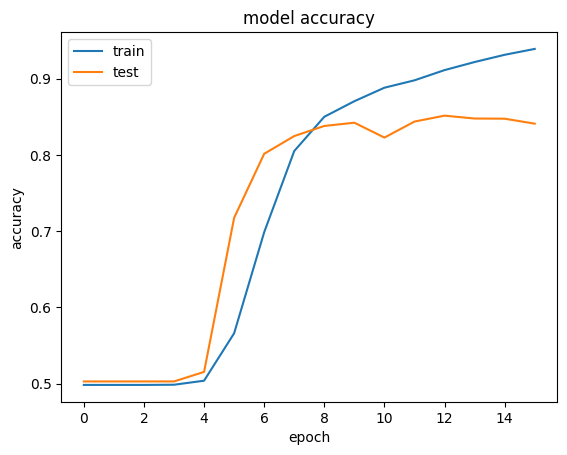
dense\_21 (Dense) (None, 1) 65

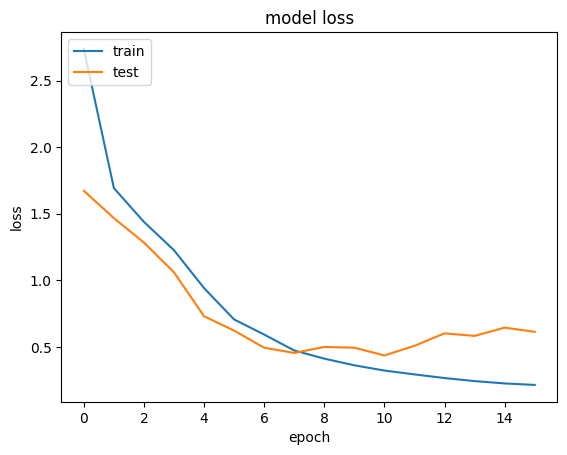
=================================================================

Total params: 35,686,529

Trainable params: 35,686,529

Non-trainable params: 0





Classification Report

precision recall f1-score support

0 0.80 0.91 0.85 5044

1 0.89 0.77 0.83 4956

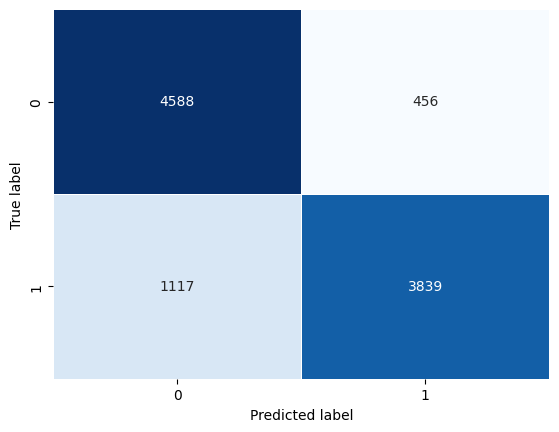
accuracy 0.84 10000

macro avg 0.85 0.84 0.84 10000

weighted avg 0.85 0.84 0.84 10000

Accuracy : 0.8427

Accuracy 84.27 %



***Developing Deep neural networks:***

***c. 2-layer LSTM model***

Layer (type) Output Shape Param #

=================================================================

embedding\_15 (Embedding) (None, 64, 256) 35600128

lstm\_4 (LSTM) (None, 64, 64) 82176

lstm\_5 (LSTM) (None, 32) 12416

dense\_22 (Dense) (None, 32) 1056

dropout\_8 (Dropout) (None, 32) 0

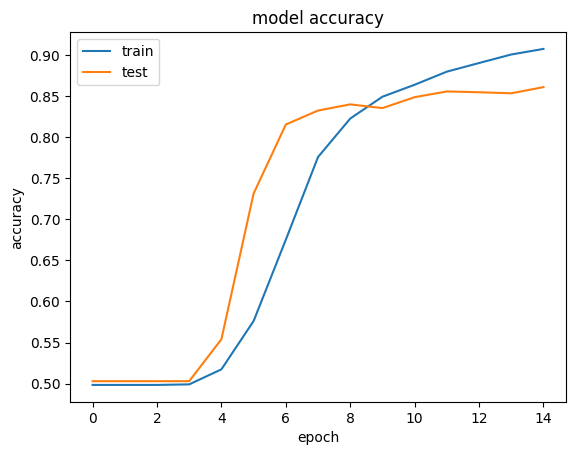
dense\_23 (Dense) (None, 1) 33

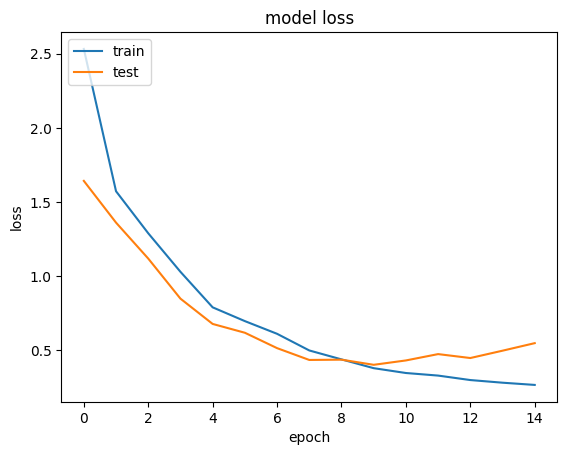
=================================================================

Total params: 35,695,809

Trainable params: 35,695,809

Non-trainable params: 0





Classification Report

precision recall f1-score support

0 0.86 0.86 0.86 5044

1 0.85 0.86 0.86 4956

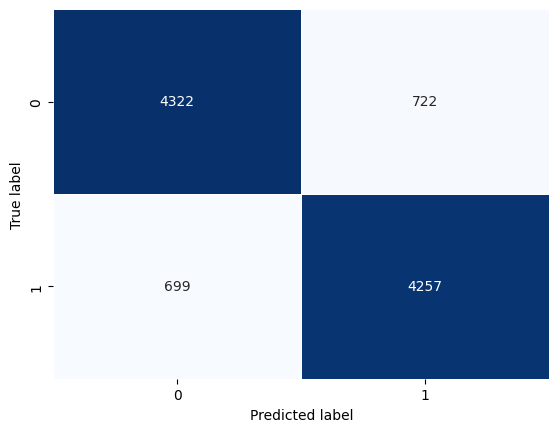
accuracy 0.86 10000

macro avg 0.86 0.86 0.86 10000

weighted avg 0.86 0.86 0.86 10000

Accuracy : 0.8579

Accuracy : 85.79 %



Confusion Matrix

**4.d*. 1-layer Bi-LSTM model***

Model: "SingleBiLSTMModel"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

embedding\_17 (Embedding) (None, 64, 256) 35600128

bidirectional\_1 (Bidirectio (None, 128) 164352

nal)

dense\_26 (Dense) (None, 32) 4128

dropout\_10 (Dropout) (None, 32) 0

dense\_27 (Dense) (None, 1) 33

=================================================================

Total params: 35,768,641

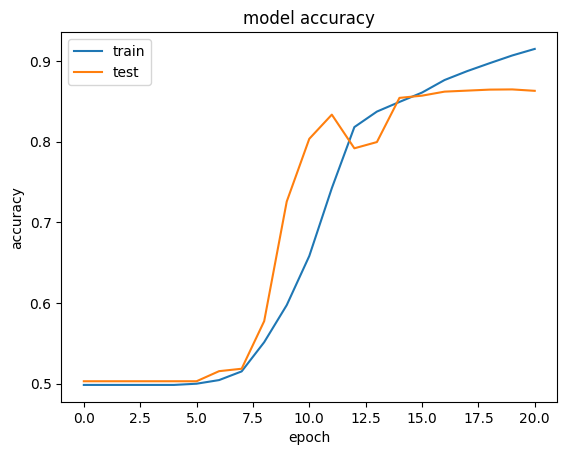
Trainable params: 35,768,641

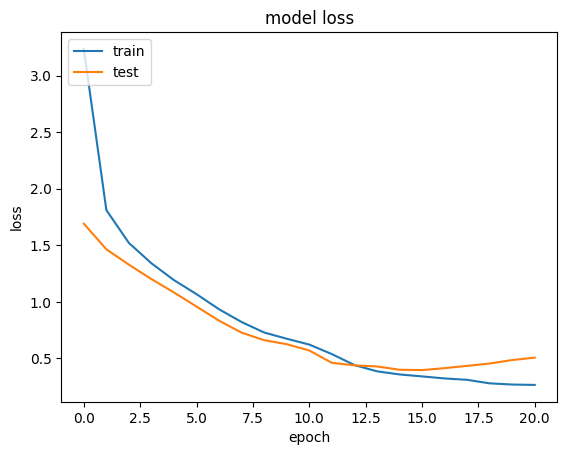
Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

None

Hidden Layers of Bi LSTM





Classification Report

precision recall f1-score support

0 0.86 0.87 0.87 5044

1 0.87 0.86 0.86 4956

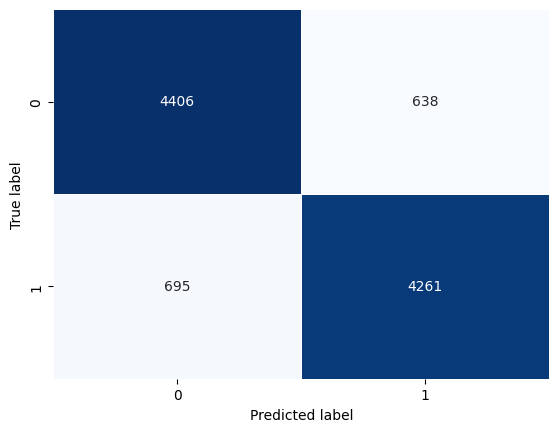
accuracy 0.87 10000

macro avg 0.87 0.87 0.87 10000

weighted avg 0.87 0.87 0.87 10000

Accuracy : 0.8667

Accuracy 86.67 %



Confusion Matrix

4b **1-layer LSTM model.** vs 4d **1-layer Bi-LSTM model.**

1-layer LSTM model Accuracy : 84.27 %

1-layer Bi-LSTM model Accuracy 86.67 %

The performance of Bi\_LSTM is better than 1 Layer LSTM because of thefollowing reasons:

1. Bi directional LSTM processes the input sequence in twodirections both forward and backward.
2. Bi-directional LSTM solves vanishing gradient problem in 1 layer LSTM.
3. Bi-directional LSTM has multiple parameters .

***4.e Use Google word2vec embeddings as input embedding to model in 4.d.***

Model: "SingleBiLSTMModel"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

embedding\_3 (Embedding) (None, 64, 256) 1792

bidirectional\_3 (Bidirectio (None, 128) 164352

nal)

dense\_3 (Dense) (None, 32) 4128

dropout (Dropout) (None, 32) 0

dense\_4 (Dense) (None, 1) 33

=================================================================

Total params: 170,305

Trainable params: 170,305

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Epoch 1/100

1125/1125 [==============================] - 162s 140ms/step - loss: 3.8885 - accuracy: 0.4994 - val\_loss: 2.1759 - val\_accuracy: 0.5005

Epoch 2/100

1125/1125 [==============================] - 140s 125ms/step - loss: 2.3539 - accuracy: 0.4985 - val\_loss: 1.3062 - val\_accuracy: 0.4975

Epoch 3/100

1125/1125 [==============================] - 141s 125ms/step - loss: 1.7592 - accuracy: 0.4958 - val\_loss: 1.0192 - val\_accuracy: 0.4990

...

Epoch 60/100

1125/1125 [==============================] - 309s 274ms/step - loss: 0.4742 - accuracy: 0.8022 - val\_loss: 0.4066 - val\_accuracy: 0.8487

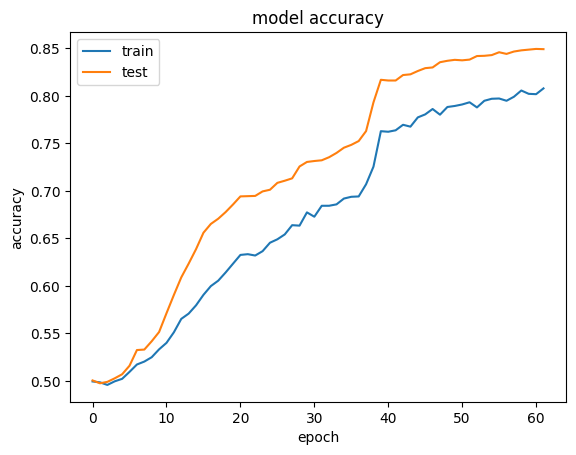
Epoch 61/100

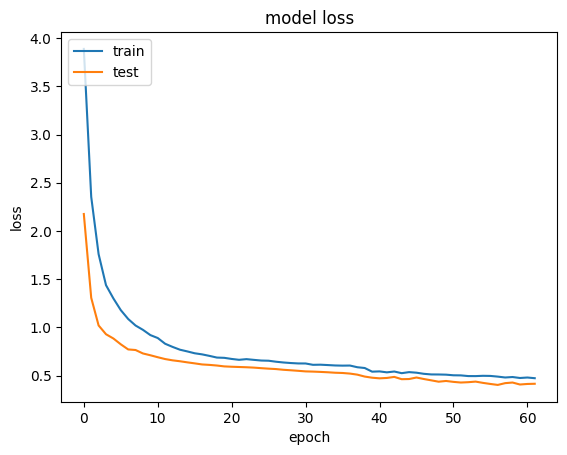
1125/1125 [==============================] - 308s 274ms/step - loss: 0.4792 - accuracy: 0.8019 - val\_loss: 0.4123 - val\_accuracy: 0.8495

Epoch 62/100

1125/1125 [==============================] - 311s 276ms/step - loss: 0.4721 - accuracy: 0.8079 - val\_loss: 0.4143 - val\_accuracy: 0.8493

Epoch 62: early stopping





Classification Report

precision recall f1-score support

0 0.84 0.85 0.85 5044

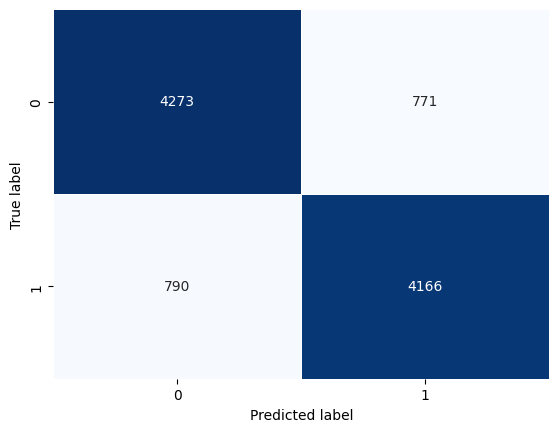
1 0.84 0.84 0.84 4956

accuracy 0.84 10000

macro avg 0.84 0.84 0.84 10000

weighted avg 0.84 0.84 0.84 10000

Accuracy : 0.8439 (84.39 %)



**word2vec embeddings vs 1 Layer Bi LSTM**

word2vec accuracy 84.39%

1 layer Bi LSTM 86.6 %

It could be preferable to use a 1-layer Bi-LSTM model rather than word2vec embeddings. This may occur if the task for the model to recognise intricate connections and patterns in the data that are challenging to represent using embeddings alone. By processing the input sequence in both forward and backward directions and updating its hidden states correspondingly, a Bi-LSTM model can learn these patterns and dependencies.

***4.f. Use Glove embeddings as input embedding to model in 4.d. [Compare the performance 4.f vs 4.d]***

Epoch 1/10 1125/1125 [==============================] - 62s 53ms/step - loss: 0.5142 - accuracy: 0.7524

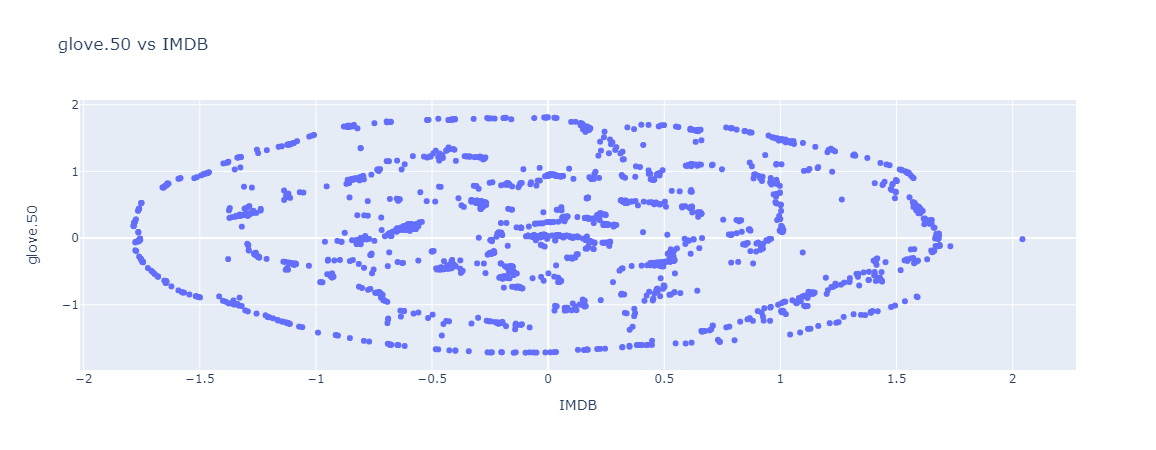
Epoch 2/10 1125/1125 [==============================] - 63s 56ms/step - loss: 0.4158 - accuracy: 0.8123

Epoch 3/10 1125/1125 [==============================] - 58s 52ms/step - loss: 0.3757 - accuracy: 0.8344

Epoch 8/10 1125/1125 [==============================] - 52s 46ms/step - loss: 0.2656 - accuracy: 0.8892

Epoch 9/10 1125/1125 [==============================] - 51s 46ms/step - loss: 0.2448 - accuracy: 0.8985

Epoch 10/10 1125/1125 [==============================] - 52s 46ms/step - loss: 0.2282 - accuracy: 0.9069



Classification Report

precision recall f1-score support

0 0.87 0.85 0.86 5044

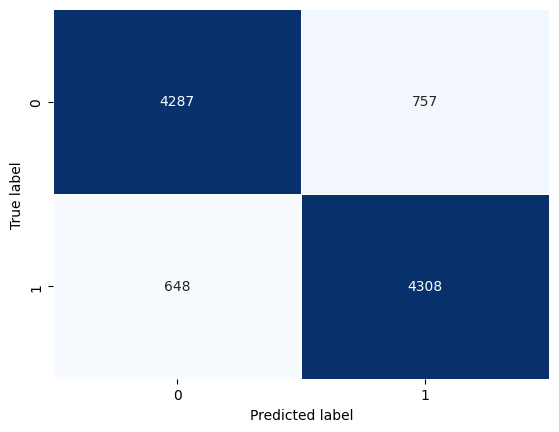
1 0.85 0.87 0.86 4956

accuracy 0.86 10000

macro avg 0.86 0.86 0.86 10000

weighted avg 0.86 0.86 0.86 10000

Accuracy : 0.8595 (85.95%)



**Comparing Glove Embeddings vs 1 Layer Bi-LSTM**

Glove Embeddings Accuracy : 0.8595 (85.95%)

1 Layer Bi-LSTM Accuracy : 0.8676 (86.76 %)

Bi directional LSTM processes the input sequence in two directions both forward and backward. Thus in certain cases Bi-LSTM performs better than Glove Embeddings.

**Comparing Glove Embeddings vs Word2vec**

Glove Embeddings Accuracy : 0.8595 (85.95%)

Word2vec Accuracy : 0.8439 (84.39 %)

Bi directional LSTM processes the input sequence in two directions both forward and backward. Thus in certain cases Bi-LSTM performs better than Glove Embeddings.

Contribution:

1. The dataset was mutually decided.
2. The coding of each question was discussed and built by all the members together. And the best performing code was chosen.
3. Knowledge sharing was done on regular basis .
4. Report was prepared part by part by all three members.

References:

1. <https://sabber.medium.com/classifying-yelp-review-comments-using-cnn-lstm-and-pre-trained-glove-word-embeddings-part-3-53fcea9a17fa>
2. <https://www.analyticsvidhya.com/blog/2020/05/what-is-tokenization-nlp/>
3. <https://medium.com/swlh/sentiment-classification-for-restaurant-reviews-using-tf-idf-42f707bfe44d>
4. <https://www.youtube.com/watch?v=D2V1okCEsiE&t=169s>
5. <https://www.tensorflow.org/guide/keras/rnn>
6. <https://reader.elsevier.com/reader/sd/pii/S187705091831439X?token=C80CEF15EFECC5B9FAE297D64CE50EFA7AAC055038951C8EB82D867A83B1AEF1CE0EE4E3535E057DE4DEA9B033AE1E8B&originRegion=eu-west-1&originCreation=20230301154620>