BRAC
UNIVERSITY

Inspiring Excellence

CSE 440: Natural Language Processing II

Group Project

Section: 02 | Group: 07

1. Task Description

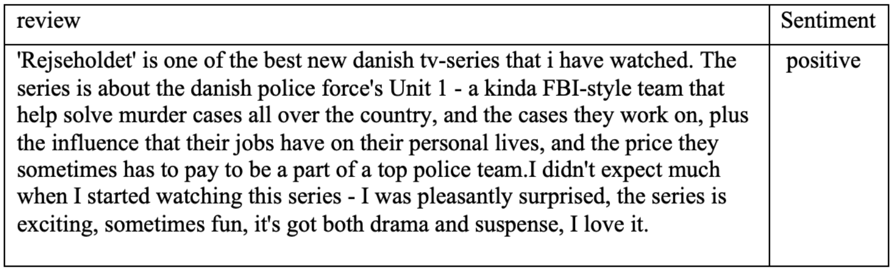
Our task is to build a shallow model, LSTM Model and Bi-LSTM to classify movie reviews.

2. Objectives

1. Clean the dataset
2. Split the dataset into 80% train and 20% test.
3. Apply Glove Embedding on reviews
4. Build models

3. Dataset Analysis

The dataset is an IMDB movie review dataset that contains 50000 movie reviews with positive and negative sentiments as classes. The dataset has an equal distribution of samples in each class, resulting in a balanced 50-50 dataset.



review	Sentiment
'Rejseholdet' is one of the best new danish tv-series that i have watched. The series is about the danish police force's Unit 1 - a kinda FBI-style team that help solve murder cases all over the country, and the cases they work on, plus the influence that their jobs have on their personal lives, and the price they sometimes has to pay to be a part of a top police team.I didn't expect much when I started watching this series - I was pleasantly surprised, the series is exciting, sometimes fun, it's got both drama and suspense, I love it.	positive

Fig 1: Example of Dataset

Total vocabulary size of the dataset is 167313 words and maximum sequence length is 1437 words. The most frequent word in the dataset is “Movie”.

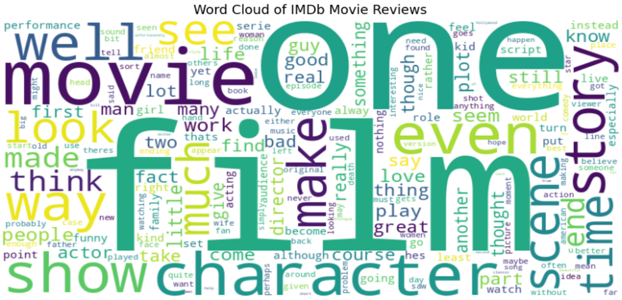


Fig 2: Word cloud of imdb Movie review dataset

4. Cleaning Dataset

To clean the dataset, we remove punctuation marks, html tags and also we remove stop words. To remove stop words, we use “nltk stopwords english”.

5. Word Embedding

To apply word embedding we did tokenization then we applied GloVe embedding on our dataset. We use 100D GloVe embedding. GloVe is a word embedding technique that represents words as 100-dimensional vectors, capturing semantic relationships through co-occurrence statistics in large text corpora.

6. Models

6.1 Shallow Model

In the shallow model, after the embedding layed there is only a Dense layer with output of 10 and activation function “tanh” then a Dense layer output of 1 and activation function sigmoid function.

6.2 LSTM

The LSTM model features a layer of 64 units with a 'tanh' activation and an embedding layer with pre-trained GloVe vectors for the representation of contextual text in vector form. In the final stage of binary classification, a 'sigmoid' activated dense layer is used. It's compiled with the 'adam' optimizer and 'binary_crossentropy' loss.

6.3 Bi-LSTM

The Bi-LSTM model utilizes a 100-dimensional embedding layer with pre-trained GloVe vectors to represent word meaning, while a layer with 64 units evaluates long-term dependencies in review sequences. The final thick layer with sigmoid activation predicts positive or negative emotion. The model was trained for 20 epochs with a batch size of 32 using the Adam optimizer and binary cross-entropy loss.

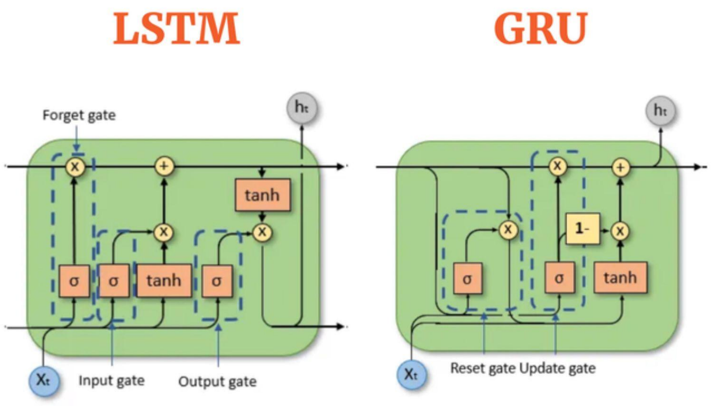


Fig 3: LSTM and GRU model internal structure

6.4 GRU

In the GRU model, 64 units are employed and trained over 20 epochs using the Adam optimizer and binary cross-entropy loss. The 'tanh' activation function is utilized in the GRU layer, and an embedding layer initialized with pre-trained GloVe vectors is employed for contextual text representation. The bidirectional nature of the GRU layer enables it to capture sequential dependencies in both forward and backward directions.

7. Result

We implemented our model in Google Colab using T4 GPU. We analyze our models with different combinations; we change the batch size, unit, and activation function. Firstly, we applied a shallow model using 64 batch sizes with two different activation functions: "relu" and "tanh". In this shallow model, after changing parameters, we got almost similar accuracy.

When we replaced the first dense layer of the model with the LSTM layer, we also tried different parameters and activation functions. First of all, we tried to tune units 32 and 64, and then we also explored different batch sizes. We tried various combinations of units, activation functions, and batch sizes. We closely observe the situation where using fewer units makes a difference in the training part, and we see that we got an improvement in using 64 units over 42 units. Also, we observe that there is no significant improvement if we try the same parameter with 128 units. We surprisingly observed that we got very low accuracy when we used 128 units and the activation function of "LeakyRelu" or "relu".

Then we replaced the LSTM layer with a bi-LSTM layer. In bi-LSTM, we got an improvement over LSTM accuracy. We tried the same hyperparameters in bi-LSTM as the LSTM layer and observed the improvement in accuracy. We also observed that in bi-LSTM, our loss was less than LSTM.

Here is the summary of our best results and hyperparameters of shallow model, LSTM and Bi-LSTM

Model	Unit	Batch size	Activation Function	Training Accuracy	Testing Accuracy
Shallow	-	64	tanh	0.5471	0.5359
Shallow	-	64	relu	0.5468	0.5353
LSTM	64	32	tanh	0.9702	0.8590
Bi-LSTM	64	32	tanh	0.9725	0.8685
GRU	64	32	tanh	0.9712	0.8714

Fig 4: Model Summary

The shallow model has comparatively low effectiveness, with training accuracy of 54.68 % and testing accuracy of 53.53 %. On the other hand, the more complex models (LSTM, Bi-LSTM, and GRU) show significant improvements. The LSTM model achieves an admirable 97.02 % in training accuracy and 85.90 % in testing accuracy. Based on this, the Bi-LSTM model makes the accuracy even better, achieving a training accuracy of 97.25 % and testing accuracy of 86.85 %

8. Improvements

After LSTM and bi-LSTM, we modeled GRU to improve our accuracy. We trained GRU with the same hyperparameters as LSTM and bi-LSTM and observed the accuracy, and we saw that the testing accuracy we got was slightly better in GRU than bi-LSTM. The GRU model comes out with a training accuracy of 97.12% and a testing accuracy of 87.14%. This accuracy can be improved if we use regularization techniques. For example, if we add dropout or early stopping to the GRU model, accuracy can improve.