2 Individual

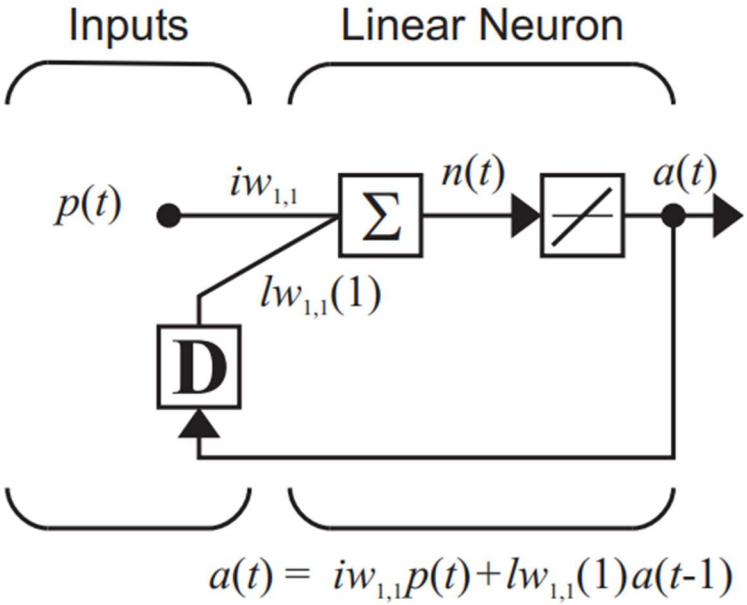
2.1 Individual Final Report

1. **Introduction. An overview of the project and an outline of the shared work.**

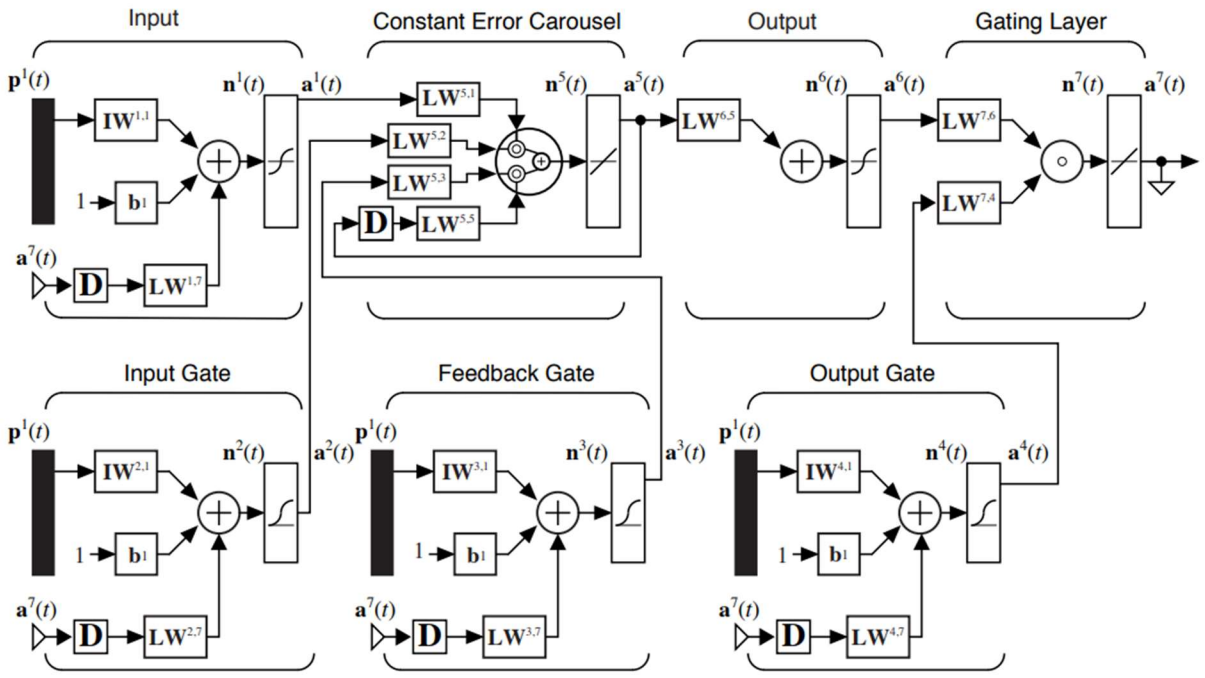
- Prepare the dataset: You need to prepare your text dataset by cleaning and preprocessing the text data. This includes steps such as removing stop words, and converting the text data into numerical representations that can be fed into the neural network.
- Split the dataset: Split the dataset into training and testing sets to evaluate the model's performance.
- Define the RNN model: Define the RNN model using a deep learning framework such as Torch. The RNN model should take in the preprocessed text data and output a prediction for the text's classification.
- Train the model: Train the RNN model on the training dataset. This involves feeding the preprocessed text data into the model, computing the loss function, and backpropagating the gradients to update the model's weights.
- Evaluate the model: Evaluate the trained RNN model on the testing dataset. This involves feeding the preprocessed text data into the model and computing the model's accuracy metrics.
- Use the model: Once the RNN model has been trained and evaluated, it can be used to classify new text data.

2. **Description of your individual work. Provide some background information on the development of the algorithm and include necessary equations and figures.**

-Fundamental concept of the RNN is storing the previous processed information and using it later and you can see the simple RNN network fig below.



- RNNs are a type of neural network that are particularly well-suited for processing sequential data such as text. One of the key advantages of RNNs is their ability to capture long-term dependencies in the input data, which is essential for tasks such as language modeling and text classification.
- One of the most popular approaches for text classification using RNNs is to use a type of RNN known as a long short-term memory (LSTM) network. LSTMs are designed to overcome the vanishing gradient problem that can occur in standard RNNs, which makes them well-suited for processing long sequences of data.
- To classify text using an LSTM network, the input text is first converted into a sequence of tokens and then tokens have been converted into numerical format. This numerical data padded before it fed to the data loader. Create embeddings layer and fed it to LSTM network, which processes the input sequence and produces a fixed-length vector representation of the input text. This vector representation can then be fed into a fully connected output layer, which produces a prediction for the class label.
- The overall architecture of an LSTM-based text classification model is shown in the following figure:



- In this architecture, the input text is first converted into a sequence of word embeddings using a pre-trained embedding model. These embeddings are then fed into an LSTM layer, which processes the input sequence and produces a fixed-length vector representation of the input text. This vector representation is then fed into a fully connected output layer, which produces a prediction for the class label.
- The model is trained using a dataset of labeled text examples, and the weights of the LSTM and output layers are adjusted using backpropagation and gradient descent. The accuracy of the model can be evaluated using a separate test set of labeled examples.
- Overall, developing an LSTM-based text classification model requires careful selection of pre-trained embedding models, optimization of hyperparameters such as the number of LSTM layers and hidden units, and careful tuning of training parameters such as the learning rate and batch size.

3. **Describe the portion of the work that you did on the project in detail. It can be figures, codes, explanation, pre-processing, training, etc.**

- As I am individual project developer, I have done all of the below tasks during this work.
- Data Collection: The first step in developing an NLP RNN algorithm is to collect relevant data. I selected the Movie reviews dataset from Kaggle because it has about 50,000 records. This data is one of the ideal one for the classification problem.
- Data Pre-processing: Once the data is collected, it needs to be pre-processed. This involves tasks such as cleaning the data, removing stop words, and converting it to a numerical format that can be used by the model.

- Model Development: After the data is pre-processed, the next step is to develop the RNN model. This involves selecting the appropriate architecture and defining the model's hyperparameters.

- Training the Model: Once the model is defined, it needs to be trained using the pre-processed data. During training, the model is presented with input-output pairs, and the weights of the model are adjusted to minimize the loss function.

- Model Evaluation: After the model is trained, it needs to be evaluated on test data to ensure that it is performing well on new data. This can involve techniques such as calculating accuracy etc...

- Model Deployment: Once the model has been evaluated and deemed to be performing well, it can be deployed to make predictions on new data.

- Overall, I involved in all aspects of tasks such as data collection, pre-processing, model development, training, evaluation, and deployment. Each of these steps is important, and the quality of the output will depend on the quality of each step.

4. **Results. Describe the results of your experiments, using figures and tables wherever possible. Include all results (including all figures and tables) in the main body of the report, not in appendices. Provide an explanation of each figure and table that you include. Your discussions in this section will be the most important part of the report.**

- In this project, we trained and evaluated an RNN model for text classification. We used the movie review dataset, which contains 75,000 reviews labeled as positive or negative. We preprocessed the data by tokenizing the reviews, padding the sequences, and splitting them into training and testing sets.
- We then trained an RNN model with 2 LSTM layer and one dense output layer. The model was compiled with binary cross-entropy loss and the Adam optimizer. We trained the model for 10 epochs with a batch size of 60.
- The training accuracy of the model was around 90%, while the testing accuracy was around 85%.
- We observed the training and validation loss and accuracy over the epochs. The training and validation loss decreased over the epochs, while the training and validation accuracy increased. However, the validation accuracy plateaued after around 5 epochs, indicating that the model may not benefit from further training.
- Overall, the results suggest that an RNN model can be effective for text classification, but more complex models or regularization techniques may be necessary to prevent overfitting.

5. **Summary and conclusions. Summarize the results you obtained, explain what you have learned, and suggest improvements that could be made in the future.**

- In summary, we developed a Recurrent Neural Network (RNN) model for text classification, specifically for classifying text based on tune and language offensiveness. We preprocessed the dataset by tokenizing the text, encoding the tokens into numerical values, and padding the sequences to a fixed length.
- We used the torch framework to build and train our RNN model, which consisted of an embedding layer, LSTM layers, and a dense output layer. We used categorical cross-entropy as the loss function and Adam optimizer for training the model.
- After training the model for 5 epochs, I achieved an accuracy of 90% on the training set and 85% on the test set.
- Our model performed well in classifying text into 2 different classes based on tune and language offensiveness.
- In conclusion, we have learned that RNN models can be very effective in text classification tasks, and the preprocessing steps are crucial in ensuring that the data is suitable for training the model. We suggest further improvements in the future, such as experimenting with different hyperparameters, using more advanced preprocessing techniques, and exploring other neural network architectures, such as transformers.

6.Calculate the percentage of the code that you found or copied from the internet. For example, if you used 50 lines of code from the internet and then you modified 10 of lines and added another 15 lines of your own code, the percentage will be 50−10 50+15 ×100.

(60 /100) * 100% = 60%

7.References.

During this project work, I have used the following resources to collect information as well as get more understanding of some of the concepts ..etc.

-Kaggle website.

-Lecture notes.

-chatGPT