UNIVERSITY PARTNER

UNIVERSITY OF WOLVERHAMPTON

HERALD COLLEGE KATHMANDU

Artificial Intelligence and Machine Learning

(6CS012)

Task2: Text Classification Report

Name: Aayush Bahadur Shahi

Uni-id: 2329565

Group: L6CG9

Module Leader: Mr Siman Giri

1. Title

Sentiment Analysis of Book Review using RNN, LSTM, and Word2Vec Embeddings.

2. Abstract

This project addresses the challenge of classifying book reviews into different sentiment levels based on ratings from 1 to 5. The reviews were preprocessed through cleaning, tokenization, and lemmatization. We applied deep learning models like RNN and LSTM, using both Word2Vec and radnom embeddings for text representation. The LSTM model performed better that RNN in terms of accuracy. This demonstrates the effectiveness of deep learning in understanding book reviews and its potential in building improved recommendation systems.

Table of Contents

1. Title *2*

2. Abstract *2*

3. Introduction *1*

4. Dataset *1*

4.1 Source 1

4.2 Target 1

4.3 Preprocessing Steps 2

5. Methodology *3*

5.1 Text Processing 3

5.2 Model Architecture 4

5.3 .Loss Function 4

5.4 Optimizer 4

5.5 Hyperparameters 5

6. Results and Experiments *5*

6.1 RNN vs LSTM Performance 6

6.2 Computational Efficiency 8

6.3 Training with Different Embedding 8

6.4 Model Evaluation 8

7. GUI output *10*

8. Conclusion and Future Work *11*

Table of Figures:

1. Seeing the data using head() 2

2.Text Preprocessing 3

3. Tokenizer..3

4. RNN Model loss and accuracy 7

5. LSTM Loss and Accuracy 7

6. Confusion Matrix RNN 9

7. LSTM Confusion Matrix..9

8. LSTM + Word2vec 10

9. Positive rating 10

10. Negative rating..10

3. Introduction

This project focuses on predicting book review ratings based on the text written by users. The dataset contains reviews from May 1996 to July 2014. Each reviewer has written at least five reviews, and each book has at least five reviews. The rating is a number from 1 to 5, which is the taget we want to predict. Text classification is a useful task in many real-life application, such as understanding customer feedback. In this case, we use it to find out what rating a user might give, just by looking at their review. To, solve this, we use deep learning models like RNN (Recurrent Neural Network) and LSTM (Long Short -Term Memory). These models are good at working with text because they can understand the meaning and order of words in a sentences. The dataset is divided into three parts: training, validation, and test sets. We train our models on the trainning set and check how well they perform on the validation and test sets. In the past, many researchers have used both simple and advanced methods for text classification and sentiment analysis. Today, deep learning methods like LSTM are more popular because they give better results.

4. Dataset

4.1 Source

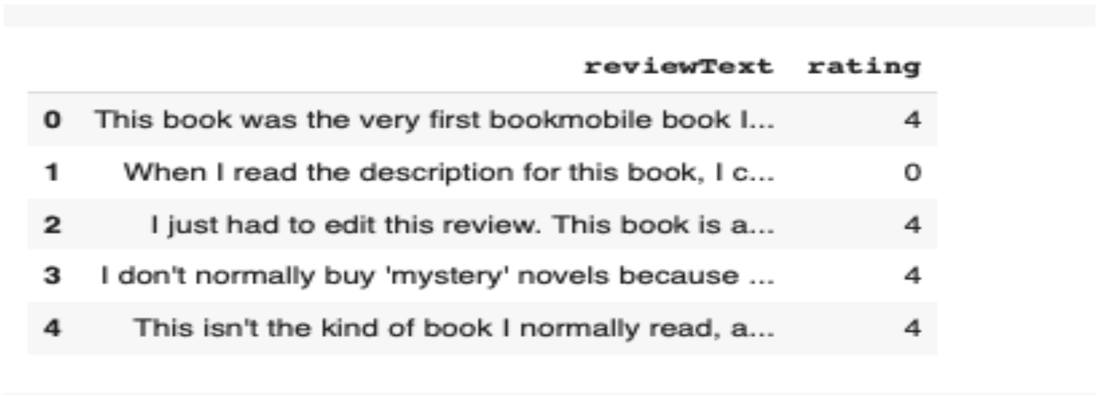
The dataset contains book reviews collected from May 1996 to July 2014. Each reviewer has written at least 5 reviews, and each book has at least 5 reviews.

4.2 Target

The goal is to predict the rating given in each review. The ratings range from 1 to 5.

Features:

- Review text (what the user wrote)
- Rating (The number from 1 to 5)



	reviewText	rating
0	This book was the very first bookmobile book I...	4
1	When I read the description for this book, I c...	0
2	I just had to edit this review. This book is a...	4
3	I don't normally buy 'mystery' novels because ...	4
4	This isn't the kind of book I normally read, a...	4

1. Seeing the data using head()

4.3 Preprocessing Steps

To prepare the data, the following steps were done:

- Converted all text to lowercase
- Remove punctuation, numbers, and extra spaces
- Removed common words (like “the”, “is”, “and”)
- Applied lemmatization (Changed words to their base form)

Data Split- The dataset was divided into:

- Training set – to train the model
- Validation set – to check model performance during training
- Test set – to evaluate the final model

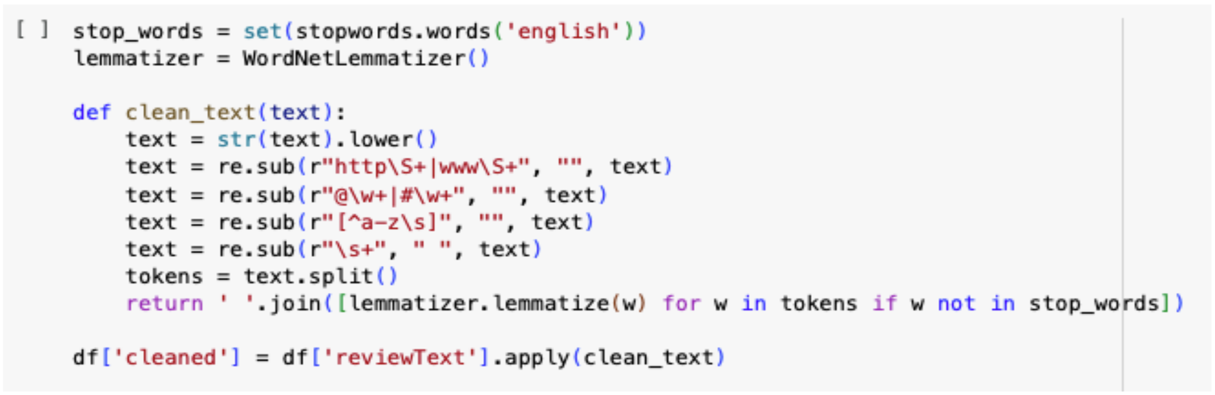
5. Methodology

This section explains the steps taken to build and train the deep learning model for predicting the review ratings.

5.1 Text Processing

Before using the text for model training, we applied the following steps:

- Lowercasing: All text was converted to lowercase to maintain consistency.
- Tokenization: The review text was split into individual words (tokens).
- Stopword Removal: Common words like "the", "is", and "and", that do not carry much meaning were removed.
- Lemmatization: Words were reduced to their base form (e.g., "reading" – "read").

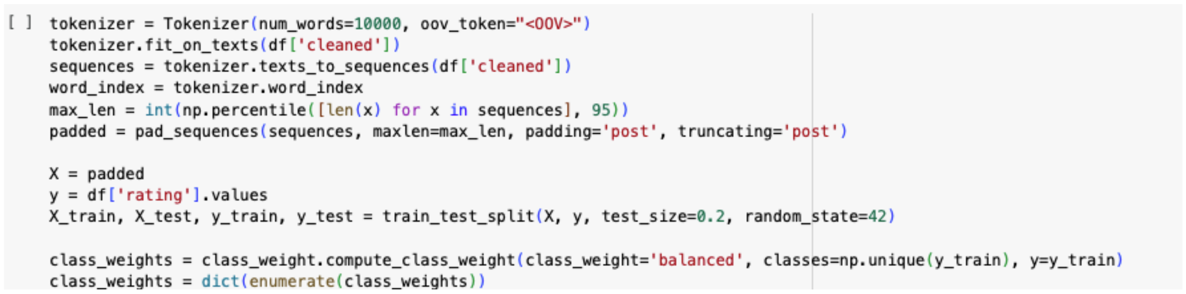


```
[ ]  stop_words = set(stopwords.words('english'))
     lemmatizer = WordNetLemmatizer()

     def clean_text(text):
         text = str(text).lower()
         text = re.sub(r"http\S+|www\S+", "", text)
         text = re.sub(r"@\w+|#\w+", "", text)
         text = re.sub(r"[^a-z\s]", "", text)
         text = re.sub(r"\s+", " ", text)
         tokens = text.split()
         return ' '.join([lemmatizer.lemmatize(w) for w in tokens if w not in stop_words])

     df['cleaned'] = df['reviewText'].apply(clean_text)
```

2.Text Preprocessing



```
[ ]  tokenizer = Tokenizer(num_words=10000, oov_token="<OOV>")
     tokenizer.fit_on_texts(df['cleaned'])
     sequences = tokenizer.texts_to_sequences(df['cleaned'])
     word_index = tokenizer.word_index
     max_len = int(np.percentile([len(x) for x in sequences], 95))
     padded = pad_sequences(sequences, maxlen=max_len, padding='post', truncating='post')

     X = padded
     y = df['rating'].values
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

     class_weights = class_weight.compute_class_weight(class_weight='balanced', classes=np.unique(y_train), y=y_train)
     class_weights = dict(enumerate(class_weights))
```

3. Tokenizer

5.2 Model Architecture

1. Simple RNN: A Recurrent Neural Network (RNN) was used to process the sequence of words in each review. RNNs can capture the order of words but may struggle with longs sequences due to the vanishing gradient problem.

2. LSTM: To improve performance, we used Long Short-Term Memory (LSTM) networks. LSTMs are type of RNN that can remember long-term dependencies, making them better at understanding the full context of a review.

3. Word2Vec Embedding: We use pretrained Word2Vec embeddings to convert words into numerical vectors. These embeddings help the model understand word meanings and relationship more effectively that using random or on-hot encodings.

5.3 .Loss Function

Since we are predicting ratings from 1 to 5, this is a multi-class classification problem.
We used categorical cross-entropy as the loss function, which is suitable for such tasks.

5.4 Optimizer

We used the Adam optimizer, which is widely used for training deep learning models due to its fast and effective performance.

5.5 Hyperparameters

Number of epochs: 30 (with early stopping)

Callbacks = Early Stopping

Validation split = 0.2

6. Results and Experiments

To evaluate the performance of different deep learning models for text classification, three models were trained and implemented:

1. Simple RNN Model:

- Layers: Embedding -> SimpleRNN -> Dropout -> Dense (Reu) -> Dense (Softmax)
- Total Parameters: 1,294,597
- Best Validation Accuracy: 25%
- Observation: Performed poorly; failed to learn effectively from the data.

2. Bidirectional LSTM Model

- Layers: Embedding -> Bidirectional LSTM -> Dense (ReLU) -> Dropout -> Dense (Softmax)
- Total Parameters: 1,383,109
- Best Validation Accuracy: 44%
- Observation: Showed significant improvement over Simple RNN, but started overfitting after a few epochs.

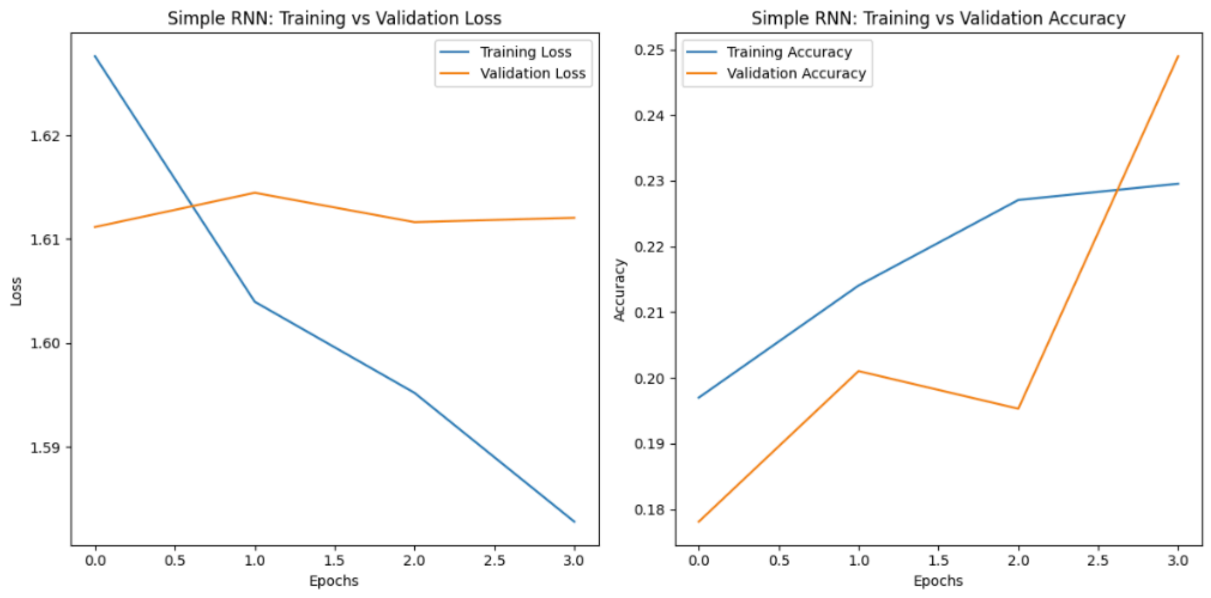
3. LSTM with Word2Vec Embeddings
    - Layers: Pre-trained Word2Vec Embedding -> LSTM -> Dense (ReLU) -> Dropout -> Dense (Softmax)
    - Total Parameters: 2,088,485
    - Best Validation Accuracy: 0.1703%
    - Observation: Performed poorly; failed to learn effectively from the data.

All models used early stopping to prevent overfitting. The Bidirectional LSTM model performed the best among the three, showing the highest accuracy on the validation set.

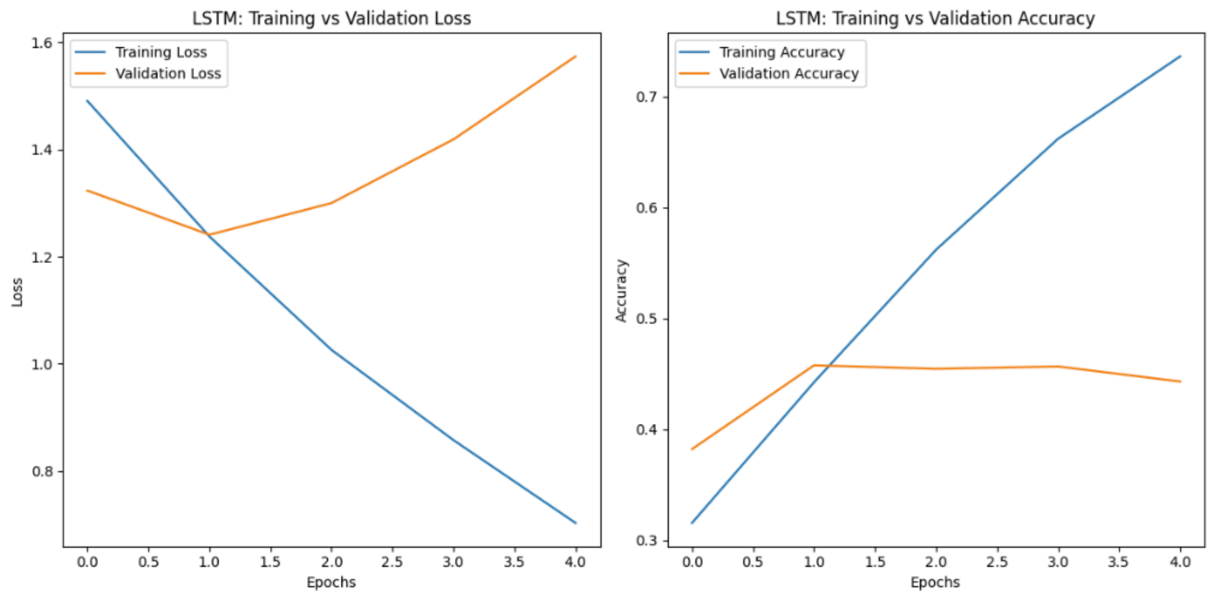
6.1 RNN vs LSTM Performance

The RNN and LSTM models based on accuracy, loss and training time.
Accuracy of RNN 25% loss is higher and training time is faster. Accuracy of LSTM is 45% loss is lower and training time is slower.

- LSTM gave better accuracy and lower loss than RNN
- RNN trained faster but did not perform well.
- LSTM is better at understanding long sequences, so it works better for text classifications.



4. RNN Model loss and accuracy



5. LSTM Loss and Accuracy

6.2 Computational Efficiency

All model were trained on Google Collab using a Tesla T4 GPU.

Simple RNN params 1.29M Train time 20sec

BilLSTM params 1.38M train time 55-80 sec

LSTM+Word2Vec params 2.09m train time 22 sec

6.3 Training with Different Embedding

Random Embeddings: Start from scratch, leading to slower training and lower accuracy.

Word2Vec Embeddings: Pretrained on large data, improving training speed and accuracy by capturing word relationship.

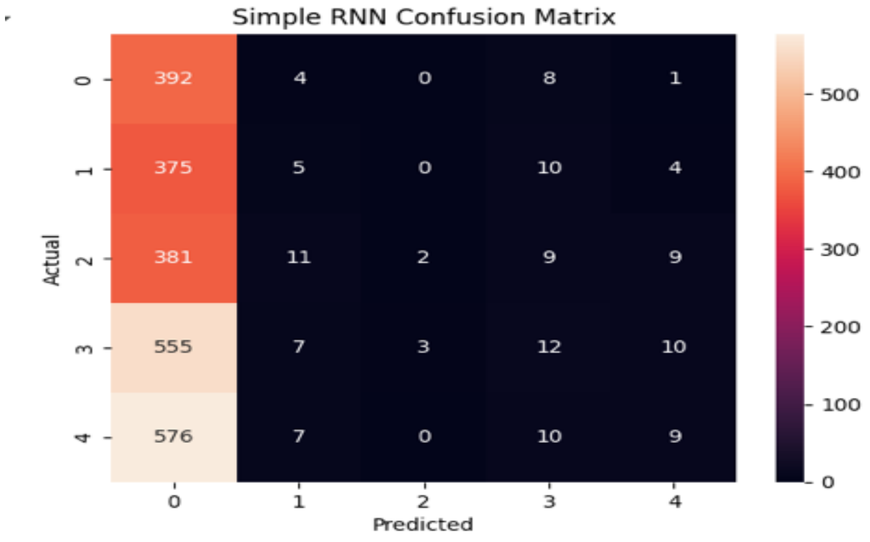
Improvement with word2Vec: Faster training, Higher accuracy, better generalization.

6.4 Model Evaluation

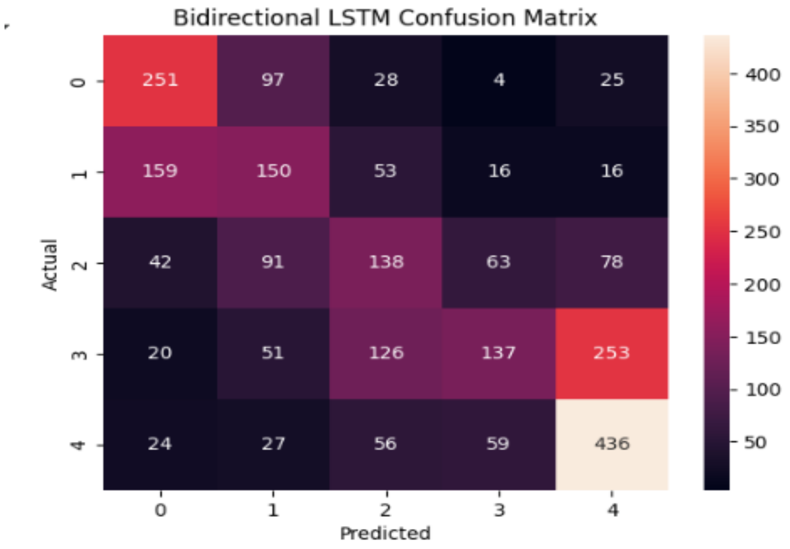
Accuracy: Measures the overall correct predictions.

Confusion Matrix: Break down predictions into TP, FP, TN, and FN for detailed analysis.

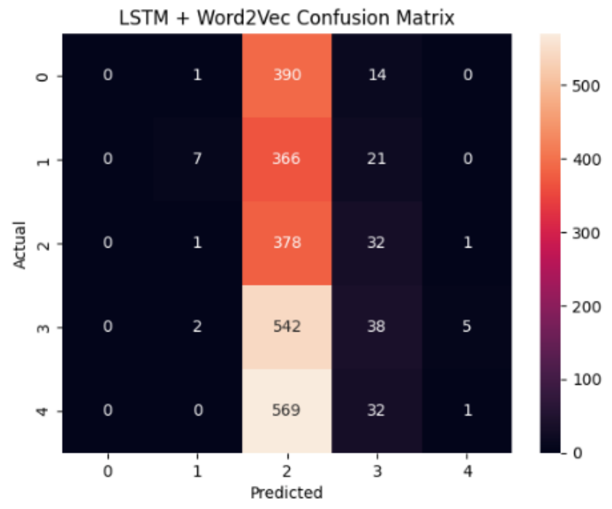
Precision: correct positive predictions, Recall: Correct identification of actual positives. F1-Score: Balances precision and recall.



6. Confusion Matrix RNN



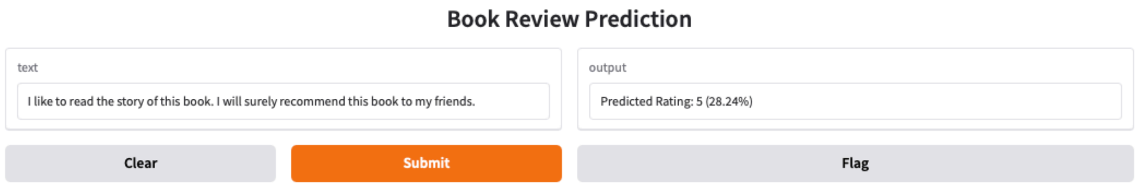
7. LSTM Confusion Matrix



Actual \ Predicted	0	1	2	3	4
0	0	1	390	14	0
1	0	7	366	21	0
2	0	1	378	32	1
3	0	2	542	38	5
4	0	0	569	32	1

8. LSTM + Word2vec

7. GUI output

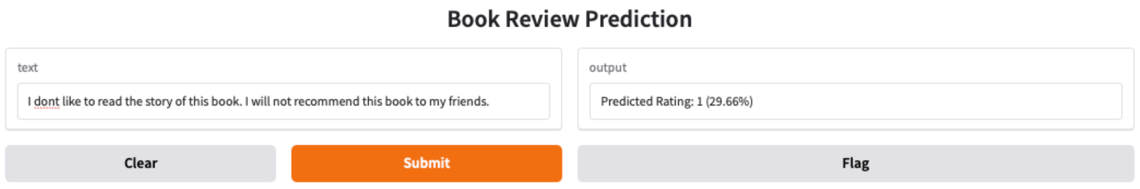


Book Review Prediction

text: I like to read the story of this book. I will surely recommend this book to my friends.

output: Predicted Rating: 5 (28.24%)

9. Positive rating



Book Review Prediction

text: I dont like to read the story of this book. I will not recommend this book to my friends.

output: Predicted Rating: 1 (29.66%)

10. Negative rating

8. Conclusion and Future Work

This project compared RNN and LSTM models from which the LSTM model outer performed the RNN in accuracy, showing better performance on text classification. Word2Vec embeddings didn’t provide significant improvement in this case.

Limitations, Improvements and Future work:
Overfitting and convergence issues were observed, with lower validation accuracy accuracy compared to training accuracy. Future work could involve hyperparameter tuning, using larger datasets, and exploring other embeddings. Exploring models like BERT and applying the model to real world tasks like sentiment analysis could be valuable next steps.