CS7.401. Introduction to NLP ✱

Textual Coherence

Team 10: ByteSpeak
Smruti Biswal - 2020112011
Pranali Jagdale - 202010198
Shiva Shankar - 2023202005

Objectives

1. Learn about Discourse Coherence
2. Use LSTM to measure coherence
3. Implement GRU to measure coherence
4. Implement RNNs to measure coherence

Introduction

- Textual coherence is a crucial aspect of natural language processing that measures the **readability, clarity**, and **consistency** of ideas expressed in a passage.
- Within a discourse, coherence is exhibited at both **local** and **global** levels. Local coherence refers to the coherence between **adjacent sentences** or clauses, while global coherence refers to the **overall coherence** of the entire text.
- Modeling textual coherence is a challenging task due to the complexities involved in understanding the semantic relationships and logical flow of ideas within a text.
- This project aims to explore and experiment with neural models for measuring textual coherence.

Datasets

1. Grammarly Corpus of Discourse Coherence:

   - The dataset is annotated into classes 1,2 and 3 where 3 denotes the most coherent paragraph.
   - It consists of four training datasets (1000 paragraphs each) and four testing datasets (200 paragraphs each).
   - We have merged the four training datasets along with three testing datasets to train our model (4600 paragraphs) and used the remaining test data to test the accuracy of our model.
   - Some datasets being more open-domained than the others, we decided to vary the test dataset chosen in order to compare the results. Additional details regarding the dataset can be found here.

2. Wikipedia-CNN Dataset:

   - The dataset consists of coherent text from Wikipedia and the CNN/Daily news sets.
   - The text dataset also contains respective replacements to make every paragraph incoherent. These replaced sets are used as negative samples in our model.
   - The dataset can be found here and this paper can be referred for the same.

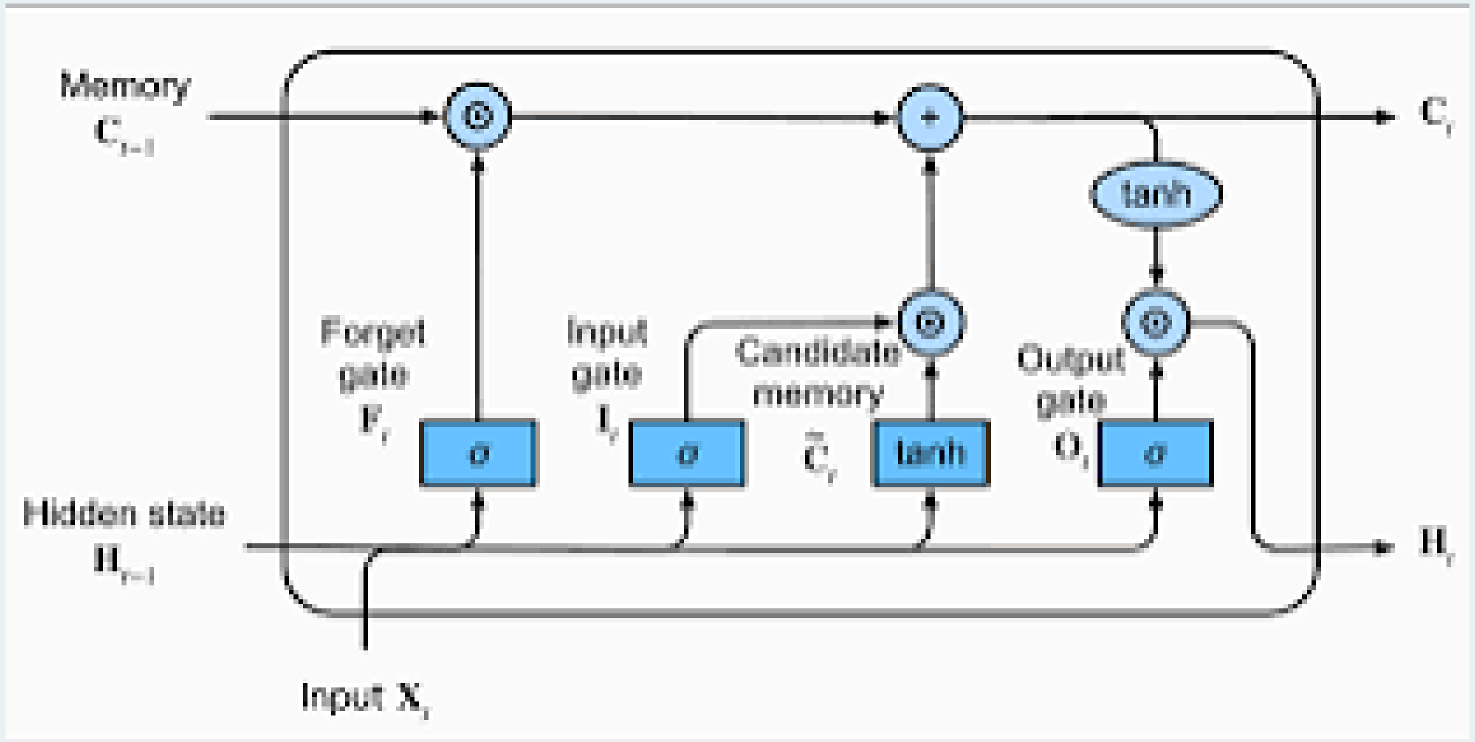
What is Discourse Coherence?

- Discourse coherence refers to the smooth and logical progression of ideas within a piece of writing or conversation.
- It entails ensuring that each sentence, paragraph, or section connects logically to the preceding and succeeding parts, maintaining a cohesive narrative or argument.
- Coherence is achieved through the use of transitional phrases which signal shifts in thought or indicate relationships between ideas.
- Coherent discourse also considers the context in which it occurs, tailoring the language and content to suit the needs and expectations of the listeners or readers.

Models Used

LSTM

- Long Short-Term Memory (LSTM) is a type of Recurrent Neural Network (RNN) architecture used in the field of deep learning.
- LSTMs are designed to solve the problem of vanishing and exploding gradients that can occur in traditional RNNs, making them effective for learning dependencies in sequence data.
- Each LSTM unit contains a cell, an input gate, an output gate, and a forget gate. These gates control the flow of information into and out of the cell, allowing the network to retain or forget information dynamically over time.



Approach

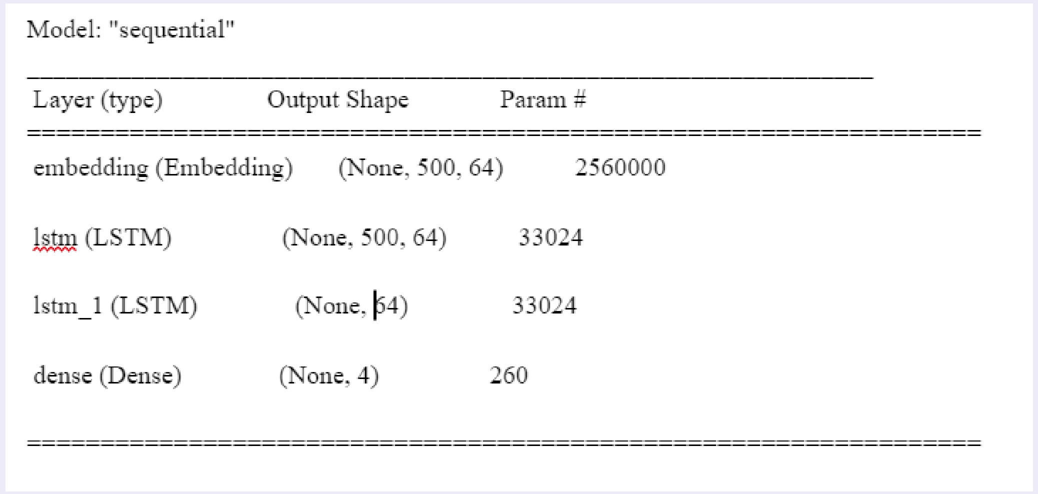
We trained our model on the GCDC corpus using the default annotation to make a three-way multi-classifier. Later, we switched this approach, remodeled our data to binary labels- coherent and incoherent- and used this data on a binary classifier to observe better results

LSTM model split into training and testing sets (4400/400) for Yelp and Enron data

Approaches:

1. Multi-class Classification LSTM

- **Embedding Layer:** Converts word indices to 64-dimensional vectors, handling up to 40,000 words.
- **LSTM Layers**: Two layers with 64 units each, using a dropout of 0.3 to reduce overfitting. The first layer returns sequences; the second does not.
- **Output Layer**: A dense layer with softmax activation, targeting four classes.
- **Training**: The model is compiled with categorical cross-entropy loss and trained for 10 epochs.
- Accuracy: 37.74999976158142



```
Model: "sequential"
_________________________________________________________________
 Layer (type)            Output Shape            Param #
=================================================================
 embedding (Embedding)      (None, 500, 64)         2560000

 lstm (LSTM)             (None, 500, 64)         33024

 lstm_1 (LSTM)             (None, 64)            33024

 dense (Dense)           (None, 4)            260

=================================================================
```

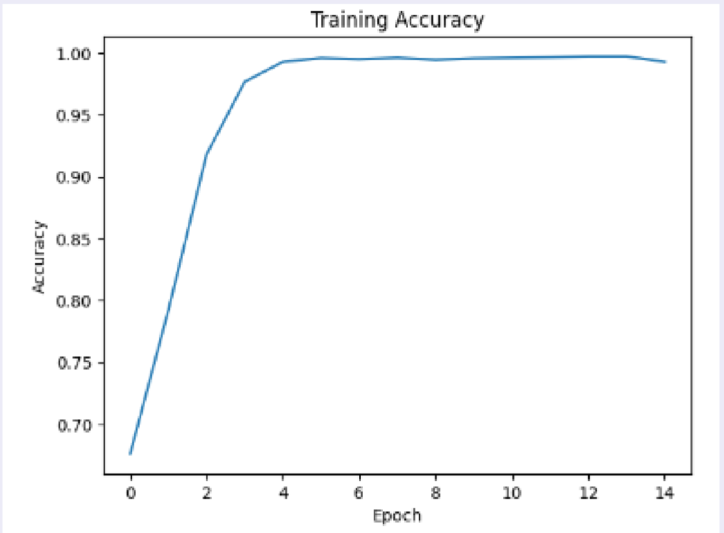
Approach

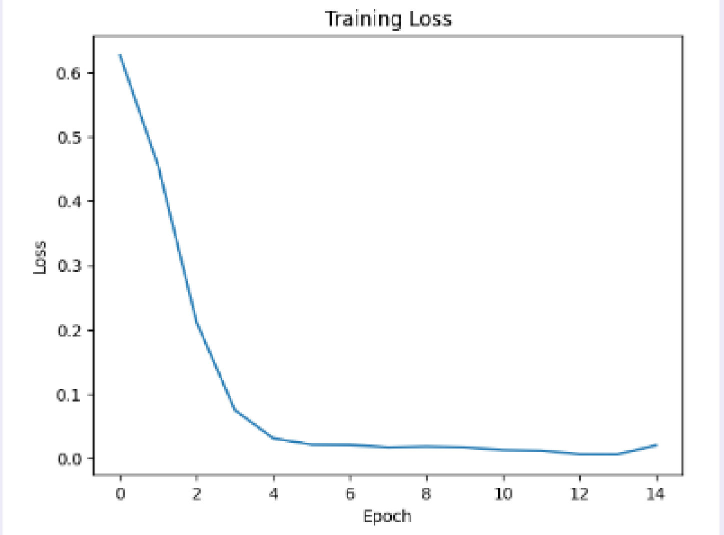
2. Binary Classification LSTM

- **Input Length**: Adjusted to 501 to include an additional feature.
- **Dropout Rate**: Slightly reduced to 0.2.
- **Output Layer**: Modified for binary output using softmax.
- **Loss Function**: Binary cross-entropy.
- **Training**: The model trains for 15 epochs, using the standard data augmented with binary labels.

3. Binary Classification LSTM with Feature Augmentation

- **Feature Augmentation**: Cosine similarity scores are added as an extra input feature.
- **Training**: Uses the same LSTM configuration as the binary classification model but now includes the augmented feature.
- Implemented the compute_paragraph_similarity function to calculate similarity scores between consecutive sentences within paragraphs of the dataset. These scores were then added as a new column named 'similarity_scores' to the dataset.
- The compute_paragraph_similarity function processes a DataFrame containing text paragraphs by performing the following steps:
  - Iterates through each paragraph, computing cosine similarity between consecutive sentences.
  - Tokenizes sentences and removes stopwords to focus on keywords.
  - Constructs binary vectors for each sentence based on word presence, then calculates the dot product.
  - Determines the minimum cosine similarity score within each paragraph.
  - Appends the minimum similarity score for each paragraph to the dataset as a new column.

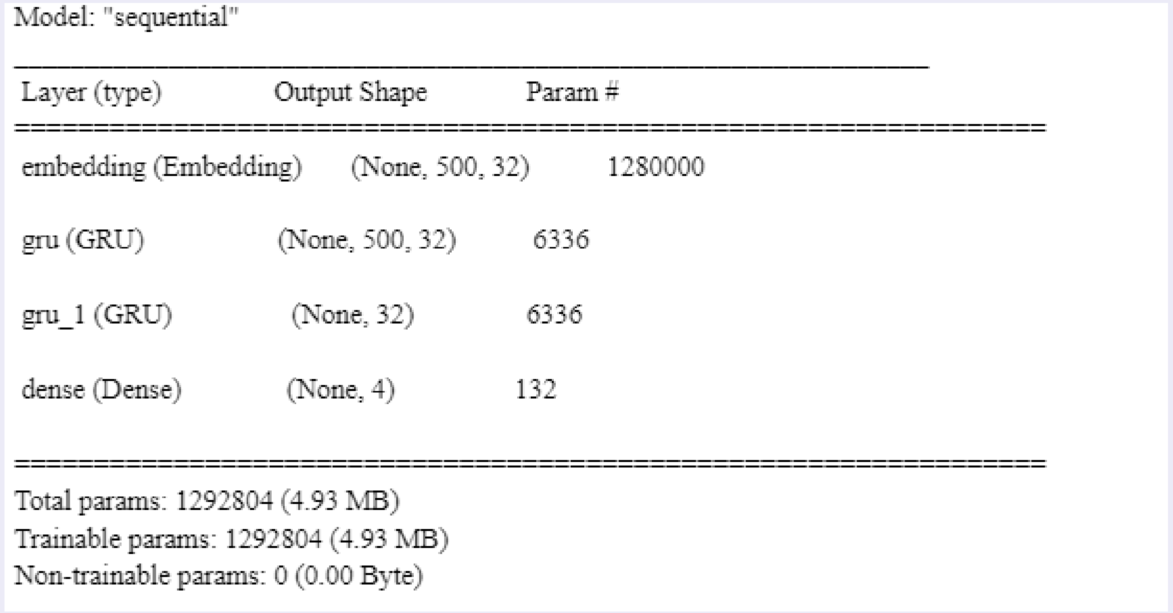




```
Accuracy:  57.499998807907104
```

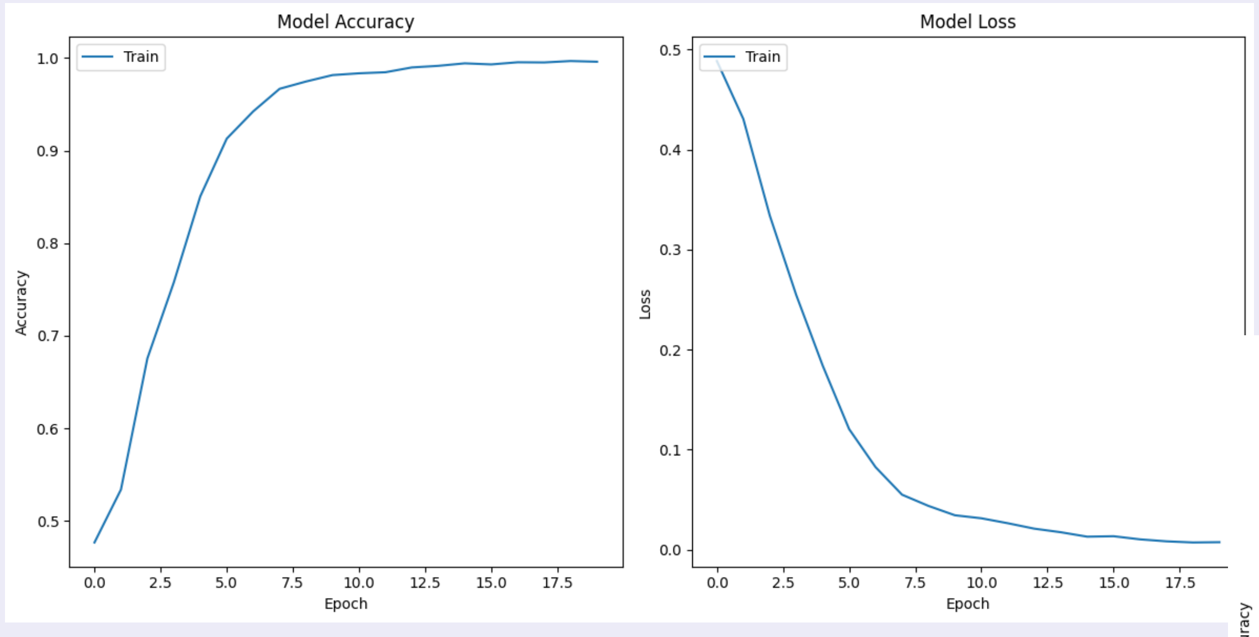
GRU

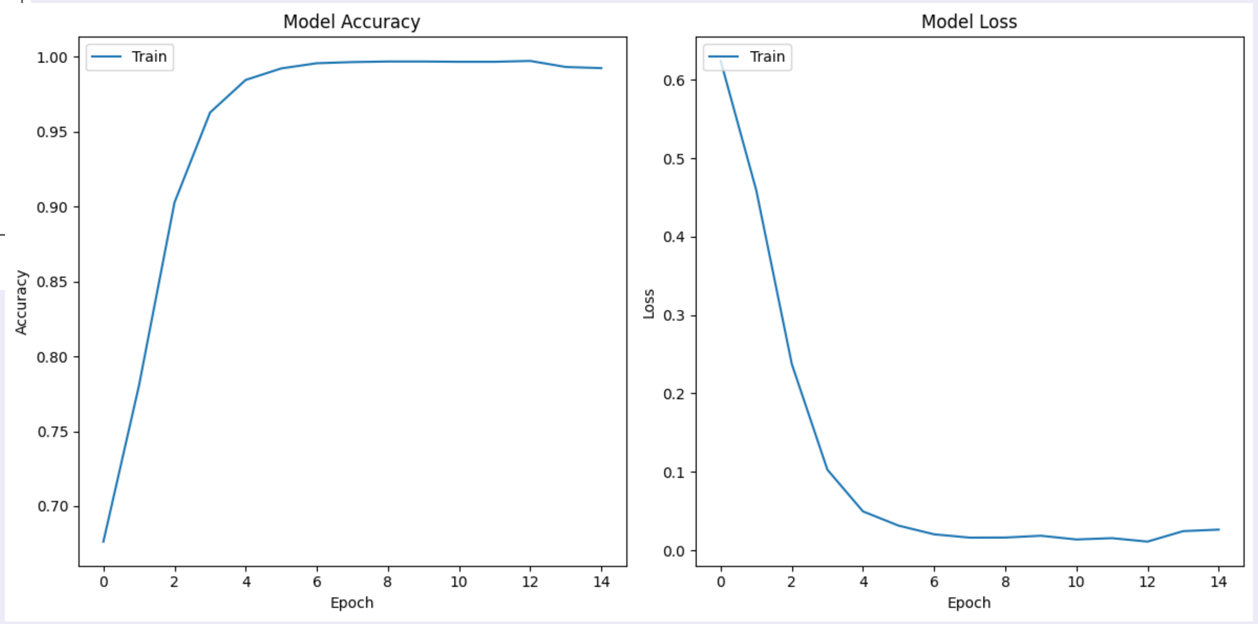
- Similar to Long Short-Term Memory (LSTM) units, Gated Recurrent Unit (GRU) models are used for learning from sequence data.
- **Three models are implemented and trained**: a standard GRU model, a binary-labeled GRU model, and a GRU model that incorporates similarity scores as features.
- The GRU (Gated Recurrent Unit) models are configured with embedding layers for input word vectorization, GRU layers for learning from sequence data, and dense layers for output classification. The binary-labeled model and the one incorporating similarity scores differ slightly in output layer configuration and data input structure.
- Models are trained with padded sequences of tokenized text, where padding ensures uniform input size. Training involves adjusting model weights to minimize a loss function (binary cross-entropy) over several epochs, and accuracy is monitored as a key performance metric.
- **Incorporation of Similarity Scores**:
- For the model using similarity scores, these scores are appended to the input feature matrix, increasing the input dimension by one. This allows the model to learn not only from the textual content but also from the structural similarity between sentences within the text.
- **standard GRU**



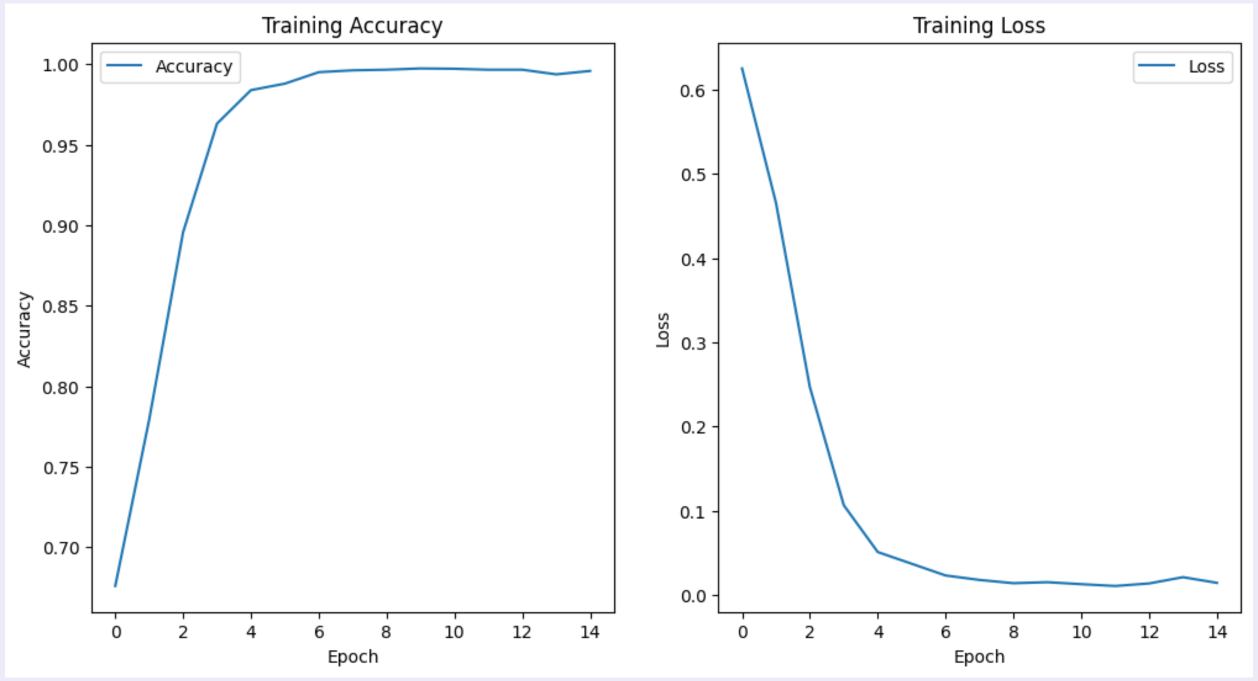
```
Model: "sequential"
_________________________________________________________________
 Layer (type)            Output Shape           Param #
=================================================================
 embedding (Embedding)      (None, 500, 32)         1280000

 gru (GRU)               (None, 500, 32)         6336

 gru_1 (GRU)              (None, 32)            6336

 dense (Dense)            (None, 4)             132

=================================================================
Total params: 1292804 (4.93 MB)
Trainable params: 1292804 (4.93 MB)
Non-trainable params: 0 (0.00 Byte)
```

GRU


Model Accuracy
Train
Accuracy
Epoch
Model Loss
Train
Loss
Epoch


Model Accuracy
Train
Accuracy
Epoch
Model Loss
Train
Loss
Epoch

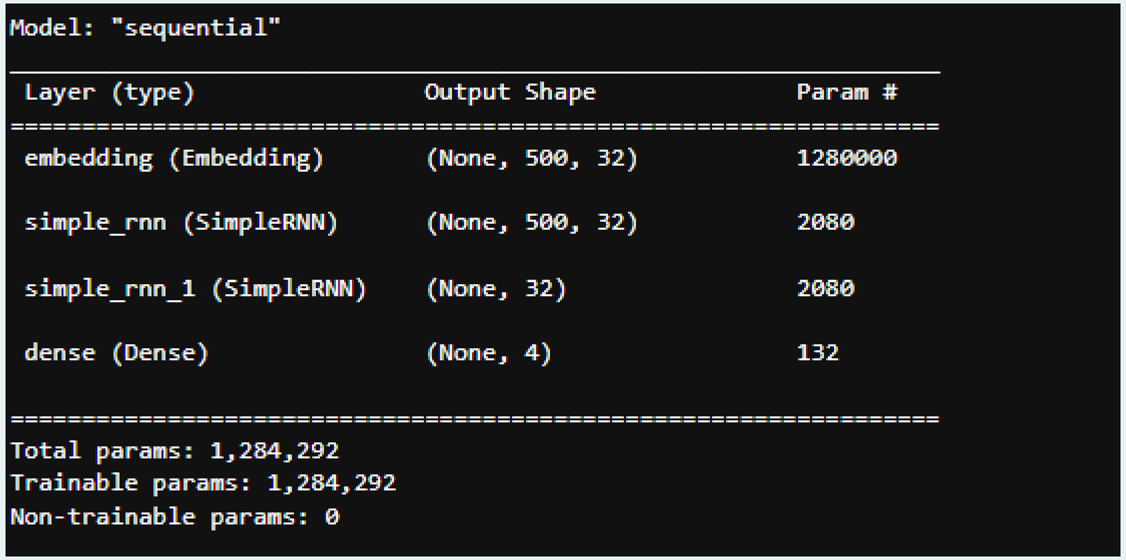
GRU



Accuracy: 66.25%

RNN

- Recurrent Neural Networks (RNNs) are a class of neural networks designed to handle sequential data. Unlike traditional neural networks that process inputs in isolation, RNNs have internal loops allowing them to maintain information in 'hidden' layers across inputs. This makes them particularly suited for tasks where context or the temporal sequence of data is important, such as language modeling, speech recognition, and time series prediction.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 500, 32)           1280000

 simple_rnn (SimpleRNN)      (None, 500, 32)           2080

 simple_rnn_1 (SimpleRNN)    (None, 32)                2080

 dense (Dense)               (None, 4)                 132

=================================================================
Total params: 1,284,292
Trainable params: 1,284,292
Non-trainable params: 0
```

Accuracy:
53.50000262260437

thank you