AIMS | African Institute for Mathematical Sciences CAMEROON

Comparative Study on Sentiment Analysis Using RNN, LSTM, and GRU Models

Group 1

Supervised by :Paulin Melatagia Yonta, Ph.D.

Group Members

1. Abibou Moustapha Motapon Ndam
2. Elizabeth Mwania
3. Mona Hussein Ali Edris
4. Oumar Moussa
5. Fred Okondza

Table of contents

1. Introduction
   - Context
   - Objective
2. Methodology
   - Dataset generation with LLM
   - Dataset Structure
   - Text cleaning and preprocessing
   - Model Layers
3. Results
   - Model Performance
   - Model Selection
4. Conclusion
5. Future Work
6. References

Introduction

Sentiment analysis with natural language processing allows monitoring public opinion from online texts. This study focuses on sentiment classification (positive, negative, neutral) using RNN, LSTM and GRU models. We have used tools like TensorFlow, Keras, Ollama, Scikit-Learn among other Python libraries.

Context

- Sentiment analysis is a growing discipline, crucial for understanding emotions and opinions expressed in textual data.
- Various applications:
  - strategic monitoring;
  - brand reputation monitoring;
  - customer service improvement;
  - market research, etc.

Objective

- Develop and compare different deep learning architectures for sentiment analysis on a textual dataset.
- Evaluate the impact of hyperparameter tuning and identify the best-performing model

Dataset generation with LLM

Dataset Generation Process with Ollama :

- **Ollama** : State-of-the-art language model that can generate coherent and relevant text.
- Advantage: Allows you to create a labeled dataset without having to manually collect and annotate data.

Dataset Structure

- We have created a balanced dataset with 486 positive, 489 negative and 483 neutral reviews (total of 1458 sentences).
- Data organization is saved in a CSV file with two columns: "sentence" and "sentiment" (the label).

Text cleaning and preprocessing

Why?
Improve data quality and facilitate model training.
Cleaning Steps:

- **Convert to lowercase:**To reduce vocabulary complexity
- **Remove punctuation:** To focus on keywords
- **Remove stop words:** To eliminate common words without semantic importance.(e.g. "the", "the", "the", "of", "to", "is", "are")
- **Tokenized text:** Split sentences into individual words (tokens) for processing and used NLTK/Spacy for efficient tokenization
- **Lemmatization:** Converted words to their base form e.g. running to run which reduces vocabulary size and improve model generalization
- **One-Hot Encoding of Labels :** Converted sentiment categories (Positive, Negative, Neutral) into numerical format for multi-class classification
- **Padding Sequences:** To standardize input. (ensured all text sequences were of equal length) for RNN/LSTM/GRU

Text Cleaning and Preprocessing

	Text_Final	tokens	sentiment	Pos	Neg	Neut
0	passion engagement demonstrated last nights de...	[passion, engagement, demonstrated, last, nigh...	positive	1	0	0
1	exploring song lyrics different genres opens w...	[exploring, song, lyrics, different, genres, o...	positive	1	0	0
2	im really impressed quickly able solve problem...	[im, really, impressed, quickly, able, solve, ...	positive	1	0	0
3	exploring song lyrics reveals rich tapestry em...	[exploring, song, lyrics, reveals, rich, tapes...	positive	1	0	0
4	hear new solar farm built town great step towa...	[hear, new, solar, farm, built, town, great, s...	positive	1	0	0

Text after Preprocessing

Representing Words with Word2Vec

Why Word2Vec ?

- Word2Vec transforms words into numerical vectors, capturing their meaning and context.
- We used a pre-trained Word2Vec model (GoogleNews-vectors-negative300) to benefit from pre-existing linguistic knowledge.

Model Layers

Input layer
The input layer receives the tokenized text data, where each word is represented by an integer index.
Input Shape (MAX_SEQUENCE_LENGTH,) is the input sequence of integers, where MAX_SEQUENCE_LENGTH is the maximum length of a sentence after padding.

Model Layers

Embedding layer

The embedding layer will convert the integer-encoded words into dense vectors of fixed size (embeddings). It has three arguments.

- The input dimension in our case is 300 words
- The Weights, in this case Pre-trained Word2Vec embeddings are used
- The Trainable layer is set to trainable=False to avoid updating the embeddings during training.

Model Layers

RNN/GRU/LSTM Layer

- Recurrent Neural Network (RNN)processes text as a sequence and updates a hidden state h_t at each time step t, allowing it to leverage memory of previous words.
- Long Short-Term Memory (LSTM) improve RNNs by incorporating long-term memory through gating mechanism using Forget gate, Input gate and Output gate
- Gated Recurrent Unit (GRU): GRU is a simplified version of LSTM, requiring fewer parameters while achieving similar performance. GRUs rely on two key gates: update gate and reset gate.

Model layers

RNN/GRU/LSTM Parameters

- Units: Number of hidden units in the layer (256 before tuning, 128 for RNN, and 256 for LSTM/GRU after tuning).
- Dropout: Fraction of units to drop during training (0.2 before tuning, 0.4 for LSTM/GRU after tuning).
- Recurrent Dropout: Fraction of recurrent connections to drop during training (0.2).

Output Shape

- (None, 256): The output is a single vector of size 256 (number of units) for each sequence

Model layers

Fully Connected layer with dropouts

- Units: 128 (number of neurons in the layer).
- Activation: ReLU (Rectified Linear Unit) for introducing non-linearity.
- Dropout: Fraction of units to drop during training (e.g., 0.2 before tuning, 0.4 for RNN/LSTM, and 0.2 for GRU after tuning)
- Output Shape: (None, 128): The output is a vector of size 128

Activation Layer

Softmax is used to decides, whether a neuron should be activated or not by calculating weighted sum and further adding bias with it by introducing non-linearity into the output of a neuron.

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}, \quad i \in \{1, 2, 3, \ldots, K\}$$

Model Layers

Output layer Produce final prediction of the three classes

- Has 3 unit for each class
- Use softmax to output probabilities of each class

Compilation:

- Adam (default), tuned to RMSprop for RNN
- Categorical Cross Entropy loss for multiclass classification

Model Layers

Hyperparameter tuning

Using Keras Tuner the best hyperparameters were as follows:

- RNN: 128 units, dropout = 0.2, optimizer = RMSprop.
- LSTM: 256 units, dropout = 0.4, optimizer = Adam.
- GRU: 256 units, dropout = 0.4, optimizer = Adam.
- Batch size 32
- Learning rate 0.001

Model performance

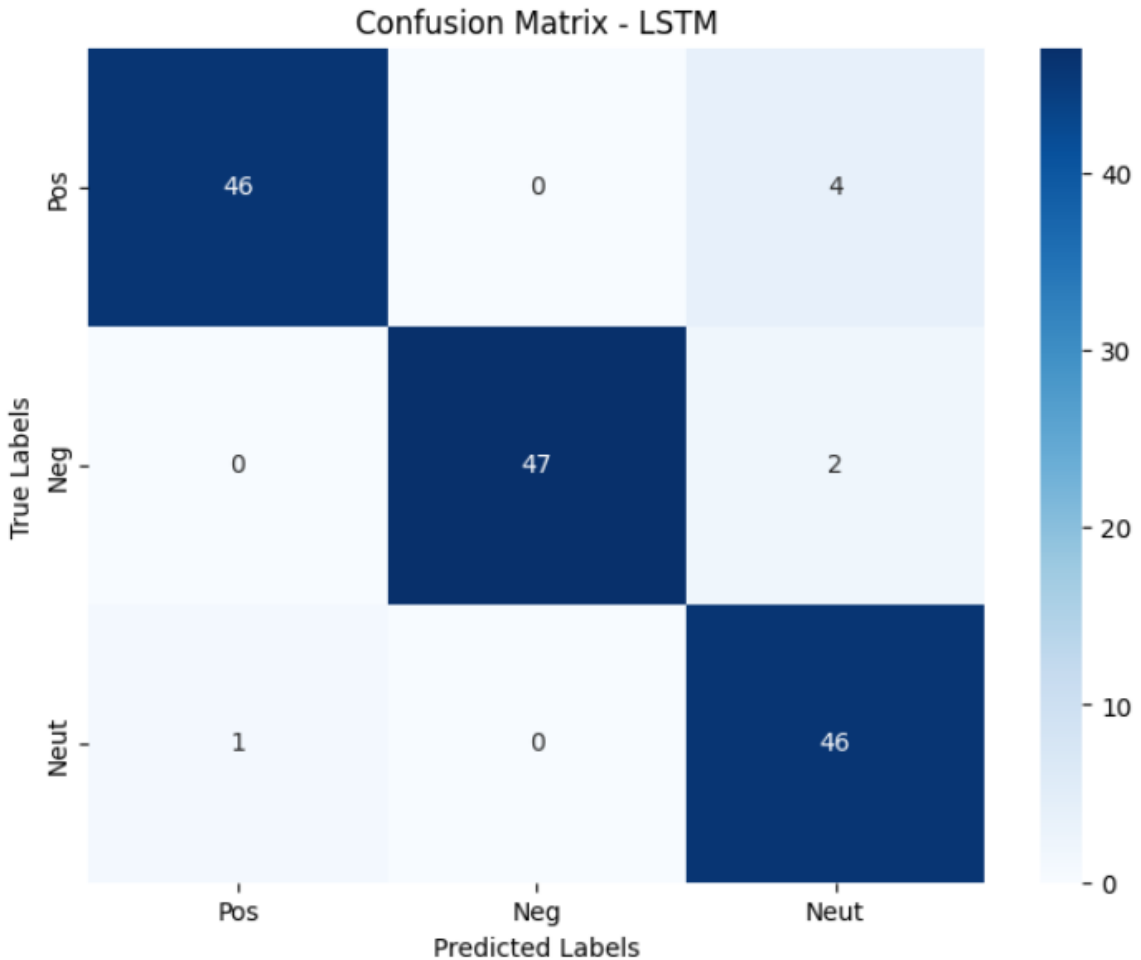
Model	Accuracy	Precision	Recall	F1-Score
RNN	0.894	0.901	0.89	0.896
LSTM	0.945	0.944	0.949	0.948
GRU	0.932	0.932	0.924	0.934

Model Performance before tuning

Model	Accuracy	Precision	Recall	F1-Score
RNN	0.901	0.905	0.896	0.891
LSTM	0.955	0.959	0.956	0.959
GRU	0.942	0.945	0.931	0.936

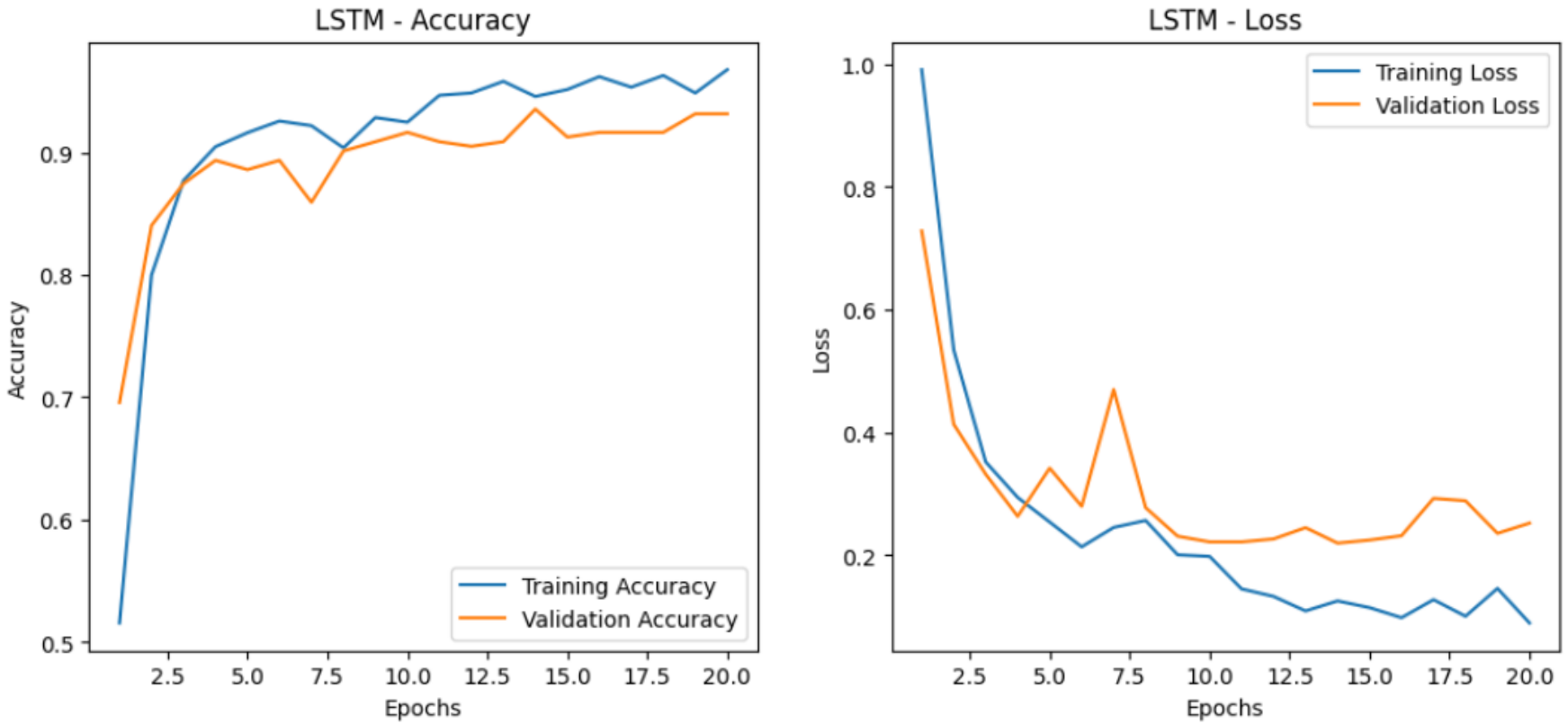
Model Performance after tuning

Model Selection



LSTM Confusion matrix

Model Selection



LSTM Plot of Accuracy and Loss

Conclusion

- Generally, after hyperparameter tuning both models slightly improved their performances.
- LSTM's memory cells and gating mechanisms (input, forget, and output gates) allow it to retain important information over long sequences, making it ideal for sentiment analysis
- GRU performance differ slightly with LSTM which make it suitable when computation considerations are required.
- RNNs struggle with long-term dependencies due to the vanishing gradient problem, which makes it difficult to learn relationships between distant words in a sequence.

Future Work

Use larger datasets with transformers (e.g., BERT,) can lead to improved accuracy. These models can be trusted in various system for analyzing user reviews and many more.

References

- Chicco, D., & Jurman, G. (2020). *Advances in Machine Learning* .
- UCI Machine Learning Repository. (2020). Text Analysis Records Dataset. University of California, Irvine, School of Information and Computer Sciences. Available at: `https://archive.ics.uci.edu/ml/datasets/LLM`

We appreciate your kind attention and feedback