Duvan Ricardo Cuero Colorado
Alejandra Díaz Parra

Sentiment Analysis with Recurrent Neural Networks (RNN) and LSTM

Introduction

This project aims to build a sentiment analysis model using supervised learning techniques with vanilla Recurrent Neural Networks (RNN) and Long Short-Term Memory networks (LSTM). The goal is to classify sentences as positive (labeled as 1) or negative (labeled as 0) based on their sentiment. The categorization of the sentiment of this sentences can be framed as a binary classification problem for the following reasons:

- Binary classification deals with the task of distinguishing between two mutually exclusive categories or classes. In this case, we're trying to answer the question "Does this sentence has a positive or negative sentiment?", defining these two sentiments as the target classes.
- The models that work with this type of problem require a labeled dataset where each example is associated with one of the two classes, which can be readily collected for positive and negative sentences.
- Binary classification is conceptually simpler than multi-class classification. It typically requires fewer parameters and can be easier to implement, which can be advantageous when designing and training machine learning models for genre classification.

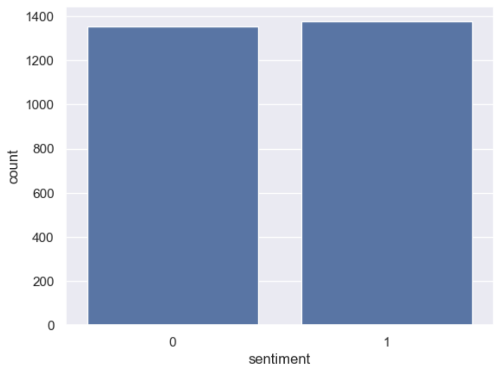
Objectives

- Preprocessing of text data by tokenizing sentences, lowercasing text and removing stop words using NLTK
- Implementation of DummyClassifier as a baseline model
- Implementation of Recurrent Neural Networks sentiment analysis model
- Implementation of LSTM sentiment analysis model
- Hyperparameter tuning using GridSearchCV
- Analysis of the resulting metrics that evaluate the performance of each model
- Comparison of the performance between models

Data description

The datasets used were a collection of sentences from the websites of *Amazon, IMDb* and *Yelp*. After merging the three of them, a new dataset with 2748 entries and 2 attributes was obtained, one of them containing the labeling of the sentence in form of numeric data, and the other one containing the text of the sentence.
The dataset contained 17 duplicated values, all of which have been removed.

The distribution of sentences labeled as positive or negative in the dataset is shown in the following figure:



Where the positive sentences are labeled as “1”, and the negative ones are labeled as “0”.

Theory

RNN Model:

Description:

- The Vanilla RNN model for sentiment analysis consists of an embedding layer, a SimpleRNN layer with 64 units, dropout for regularization, and a dense layer with a sigmoid activation function for binary classification.
- The model is trained using binary cross entropy loss and the Adam optimizer.

Architecture:

Embedding Layer: Converts words into vectors with a dimension of 50.
SimpleRNN Layer: Contains 64 units, capturing sequential patterns.
Dropout Layer: Applied with a dropout rate of 0.5 to prevent overfitting.
Dense Layer: Single dense layer with a sigmoid activation function for binary classification.

Training Procedure:

1. Tokenization and padding are applied to the input sequences.
2. Grid search is conducted to optimize hyperparameters like batch size, epochs, optimizer, and loss.
3. Training results, including accuracy and loss, are printed.
4. Evaluation metrics such as accuracy, precision, recall, and f1-score are computed on the testing set.

LSTM Model:

Description:

- The LSTM model for sentiment analysis is composed of an embedding layer, an LSTM layer with 64 units, and a dense layer with a sigmoid activation function for binary classification.
- The model is trained using binary cross entropy loss and the Adam optimizer. Grid search is employed to find the best hyperparameters.

Architecture:

Embedding Layer: Converts words into vectors with a dimension of 32.
LSTM Layer: Contains 64 units, allowing the model to capture long-term dependencies.
Dense Layer: Single dense layer with a sigmoid activation function for binary classification.

Training Procedure:

1. Tokenization and padding are applied to the input sequences.
2. Grid search is conducted to optimize hyperparameters like batch size, epochs, optimizer, and loss.
3. The final LSTM model is defined, compiled, and trained on the training set.
4. Training results, including accuracy and loss, are printed.
5. Evaluation metrics such as accuracy, precision, recall, and f1-score, along with the kappa score, are computed on the testing set.

Results

For the results, we focus on the results obtained in the different neural networks models we implemented. To reiterate, we are looking to determine the sentiment of a sentence. To do this, we propose a variety of models that are going to be trained and tested with our dataset. The distribution we considered to implement these models is 70% of the data is going to be part of the training dataset and the remaining 30% is the test dataset.

The hyperparameters obtained using GridSearchCV that were used for each model are presented in the following table:

	batch_size	**epochs**	**loss**	**optimizer**
RNN	128	20	binary_crossentropy	rmsprop
LSTM	128	20	binary_crossentropy	rmsprop

Evaluation

For the evaluation, we used accuracy, precision, recall, F1-score and kappa. Following, are the results obtained in each of the models:

	DummyClassifier		RNN		LSTM	
Metrics	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
Accuracy	0.5	0.51	0.95	0.79	0.93	0.79
Precision	0.25	0.25	0.94	0.76	0.94	0.80
Recall	0.5	0.5	0.96	0.85	0.92	0.80
F1	0.33	0.34	0.95	0.80	0.93	0.80
Kappa			0.89	0.57	0.86	0.59

Conclusions

RNN Model:

Training Performance:
- The Vanilla RNN model achieves a high training accuracy of approximately 95%, indicating effective learning from the training data.
- The training loss is relatively low, around 0.15, suggesting good convergence during training.

Testing Performance:
- The model's performance on the testing set is slightly lower, with an accuracy of approximately 79%.
- Precision, recall, and f1-score metrics are around 80%, suggesting a balanced performance between precision and recall.

Overfitting:
- There might be a slight overfitting issue as the model performs better on the training set compared to the testing set.

LSTM Model:

Training Performance:
- The LSTM model achieves a high training accuracy of approximately 93%, similar to the Vanilla RNN.
- The training loss is relatively low, indicating effective learning.

Testing Performance:
- The model's performance on the testing set is also around 79%, similar to the Vanilla RNN.
- Precision, recall, and f1-score metrics are consistent with the Vanilla RNN.

General Observations:

Model Comparison:

- Both models demonstrate similar performance on the testing set, indicating that the more complex LSTM model does not necessarily outperform the Vanilla RNN for this specific sentiment analysis task.

Overfitting Consideration:

- Monitoring overfitting is crucial, and techniques such as dropout layers[1] in the Vanilla RNN can help mitigate this issue.

Further Investigation:

- Further investigation into misclassifications and potentially exploring more sophisticated architectures or pre-trained embeddings could enhance model performance.

[1] "What is the Dropout Layer? | Data Basecamp." 5 abr. 2023, https://databasecamp.de/en/ml/dropout-layer-en. Fecha de acceso 21 nov. 2023.