PROJECT REPORT

Introduction

The development of sentiment analysis models plays a crucial role in the field of natural language processing and machine learning. In this project, our goal is to build a sentiment analysis model using supervised learning, focusing on two specific approaches: basic Recurrent Neural Networks (RNN) and Long Short-Term Memory Neural Networks (LSTM).
Tasks Completed:

Data Collection:
We have successfully obtained the dataset required for sentiment analysis from the UC Irvine Machine Learning Repository. This dataset, known as the "Sentiment Labelled Sentences Data Set," serves as the foundation for our project.

Text Preprocessing:
Using tools like NLTK in Google Colab, we performed text preprocessing. This included tasks such as tokenization, converting to lowercase, and removing stopwords to prepare the data appropriately for supervised learning.

Dummy Classifier as Baseline:
We implemented a Dummy Classifier as a baseline to assess the performance of our subsequent models. This simple classifier will provide key metrics such as accuracy, precision, recall, and F1-score, serving as an initial reference.

In the upcoming stages, we will explore more advanced models, including RNN and LSTM, and evaluate their performances using specific metrics. We will also carry out hyperparameter tuning to optimize the performance of these models.

This project represents a significant step toward understanding and applying advanced sentiment analysis techniques to real-world data, aiming to achieve accurate and efficient models in the classification of opinions.

Objectives

1. Building Sentiment Analysis Models:

   Develop and train sentiment analysis models using supervised approaches, specifically focusing on Recurrent Neural Networks (RNN) and Long Short-Term Memory Neural Networks (LSTM).

2. Utilization of the Selected Dataset:

   Apply the "Sentiment Labelled Sentences Dataset" from the UC Irvine Machine Learning Repository as a foundation for the construction and evaluation of models.

3. Effective Dataset Compilation:

   Effectively obtain the necessary dataset for sentiment analysis from the UC Irvine Machine Learning Repository, ensuring the quality and representativeness of the collected opinions.

4. Efficient Text Preprocessing:

   Conduct efficient text preprocessing using tools such as NLTK in Google Colab, including tasks like tokenization, conversion to lowercase, and removal of stopwords, to prepare the data adequately for supervised learning.

5. Establishment of a Baseline with Dummy Classifier:

   Implement a Dummy Classifier as an initial reference to assess the performance of subsequent models, providing key metrics such as accuracy, precision, recall, and F1-score.

6. Try to level up implementing a vanilla RNN
7. Try to enhance performance using a LSTM RNN
8. find the best hyperparameters for our models.

Details of the sentiment dataset

The sentiment dataset used in this project originates from the "Sentiment Labelled Sentences Dataset" of the UC Irvine Machine Learning Repository. Here are key details about this dataset:

Dataset Origin:
This dataset has been compiled from opinions expressed on three different websites: amazon.com, imdb.com, and yelp.com. Each opinion has been labeled as positive (1) or negative (0), providing a balance of 500 instances for each category on each website.

Dataset Structure:
The dataset consists of 1,500 positive opinions and 1,500 negative opinions, evenly distributed across the three websites. Each opinion is labeled with its respective sentiment, facilitating the training and evaluation of sentiment analysis models.

Dataset Purpose:

The dataset was created in the context of the article 'From Group to Individual Labels using Deep Features,' Kotzias et al., KDD 2015. Its purpose is to serve as a fundamental tool for research and development in sentiment analysis models.

Dataset Availability:
You can access the "Sentiment Labelled Sentences Dataset" through the link provided in the UC Irvine Machine Learning Repository:
https://archive.ics.uci.edu/dataset/331/sentiment+labelled+sentences

This dataset serves as the cornerstone of our project, providing essential raw material for the construction, training, and evaluation of sentiment analysis models using advanced supervised learning approaches.

Preprocessing steps

Before training sentiment analysis models, it's crucial to preprocess the text data to enhance its suitability for machine learning algorithms. The following preprocessing steps were applied to the dataset:

Tokenization:
Break the sentences into individual words or tokens. This step aids in understanding the structure of the text and facilitates further analysis.

Lowercasing:
Convert all text to lowercase. This ensures uniformity and prevents the model from treating words with different cases as distinct entities.

Stopword Removal:
Eliminate common stopwords (e.g., "the," "and," "is") from the text. Stopwords are often irrelevant for sentiment analysis and can be removed to focus on more meaningful content. In order to get those words out of the data, we used nltk to have acces to a set of these words so we can filter them out.

Alphanumeric Filtering:
Remove non-alphanumeric characters from the text. This step helps in cleaning the data and eliminating unnecessary symbols or characters. This has been done using the library ”string” which provides a punctuation set.

Text Lemmatization or Stemming (Optional):
Reduce words to their base or root form. While this step is optional, it can be beneficial for reducing dimensionality and improving the efficiency of the model.
By implementing these preprocessing steps, we ensure that the text data is in a suitable format for training machine learning models. This clean and standardized data will contribute to the effectiveness and accuracy of the sentiment analysis models developed in this project.

Description of the Dummy Classifier

In the initial stages of our sentiment analysis project, we employed a Dummy Classifier as a baseline model for performance evaluation. The Dummy Classifier serves as a simple and rudimentary benchmark, providing a reference point against which the performance of more sophisticated models can be compared. Here are key characteristics of the Dummy Classifier:

Random Classification:

The Dummy Classifier randomly assigns labels to instances, mimicking a scenario where predictions are made without any real understanding of the data.
Baseline Metrics:

The primary purpose of the Dummy Classifier is to establish baseline metrics, including accuracy, precision, recall, and F1-score. These metrics help us gauge the performance that more advanced models should surpass.
Comparative Benchmark:

By evaluating the performance of subsequent models against the Dummy Classifier, we can assess the added value and effectiveness of our more complex sentiment analysis models.
Insights into Model Complexity:

The Dummy Classifier provides insights into the level of complexity required to effectively analyze sentiment in the dataset. More advanced models should outperform the Dummy Classifier, demonstrating the value of their sophistication.
While the Dummy Classifier itself may not provide accurate sentiment predictions, it plays a crucial role in establishing a starting point for our project, guiding the development and assessment of more advanced models. The baseline metrics generated by the Dummy Classifier serve as a reference for evaluating the success of our sentiment analysis models.

Description of the Vanilla RNN

In the initial stages of our recurrent neural network (RNN) sentiment analysis project, we utilized a basic RNN model as a foundational benchmark for performance evaluation. The RNN serves as a starting point, allowing us to establish key characteristics and metrics:

Tokenization and Padding:
The reviews were tokenized and transformed into sequences, with a vocabulary size of 1000. Padding was applied to ensure uniform length (maxlen=100) for input sequences.

Categorical Labels:
The dependent variable was converted into categorical labels (binary classification) using one-hot encoding.

Model Architecture:
The RNN model consists of an embedding layer with a vocabulary size of 1000 and an embedding dimension of 32. A SimpleRNN layer with 8 neurons captures sequential patterns, followed by a dense layer producing a binary output through sigmoid activation. This model will then vary through the gridsearch for finding the best hyperparameters.

Training and Compilation:
The base model was compiled with the Adam optimizer and binary crossentropy loss. Training was conducted for 30 epochs.

Grid Search for Optimization:
To optimize the model, a grid search was performed, varying parameters such as embedding dimension, number of RNN units, and optimizer (Adam or RMSprop). Cross-validation with stratified k-fold (k=3) was employed for robust parameter tuning.

Evaluation Metrics:
The best parameters and accuracy were determined through grid search. The model's performance was assessed on the test set using metrics like accuracy, precision, recall, and F1-score. The predicted values were compared against the actual values for comprehensive evaluation.

This RNN baseline model, though simplistic, lays the groundwork for assessing more sophisticated sentiment analysis models. The grid search and comprehensive evaluation metrics provide insights into model optimization and effectiveness in capturing sentiment patterns within the dataset, showing that an RNN is clearly not the best option for performing a sentiment analysis.

Description LSTM

We implemented an LSTM model, renowned for its prowess in capturing long-term dependencies within sequential data, a critical characteristic in text analysis. The model architecture comprised an initial embedding layer followed by a bidirectional LSTM layer, and additional dense layers tailored for classification tasks.

Through an exhaustive exploration of parameter combinations using GridSearchCV, we arrived at the conclusion that the optimal configuration for our model includes the following specifications:

- Embedding Dimensions: 24
- LSTM Layer Neurons: 16 (utilizing bidirectional processing)
- Dense Layer Neurons: 16, with Rectified Linear Unit (ReLU) activation functions
- Output Layer: A single neuron employing a sigmoid activation function for binary classification.

This refined configuration, achieved through meticulous parameter tuning, showcases the model's ability to effectively understand and analyze sentiment within textual data. The combination of bidirectional LSTM processing and carefully chosen hyperparameters contributes to the model's robust performance in capturing nuanced relationships within the input sequences.

Comparative and Analysis

The Dummy Classifier, serving as the baseline model, produced the following evaluation metrics:

Accuracy: 0.492
Precision: 0.492
Recall: 1
F1 Score: 0.659
Accuracy:

The achieved accuracy of 49.2% indicates the proportion of correctly classified instances. This value serves as a reference point for assessing the performance of more sophisticated models.
Precision:

With a precision score of 49.2%, the Dummy Classifier demonstrated the ability to correctly identify positive instances among those predicted as positive. Precision is particularly relevant in scenarios where false positives carry significant consequences.
Recall:

A recall score of 100% implies that the Dummy Classifier correctly identified all positive instances present in the dataset. While this might seem impressive, it's essential to consider the balance with precision and other metrics.
F1 Score:

The F1 score, a harmonic mean of precision and recall, is a balanced metric that considers both false positives and false negatives. The obtained F1 score of 0.659 reflects the overall effectiveness of the Dummy Classifier in a binary classification setting.

For the Neural Networks we got:

RNN:
Accuracy: 0.484, Precision: 0.480, Recall: 0.571, F1-score: 0.521

Accuracy:
The RNN achieved an accuracy of 48.4%, after gridsearch indicating the proportion of correctly classified instances. This metric serves as a benchmark for assessing the RNN's performance compared to the baseline Dummy Classifier.

Precision:
With a precision score of 48.0%, the RNN demonstrated its capabilities to correctly identify positive instances among those predicted as positive. Precision is crucial in situations where false positives have notable consequences, and this score provides insights into the model's precision-oriented performance.

Recall:
A recall score of 57.1% implies that the RNN correctly identified 57.1% of the positive instances present in the dataset. Unlike the Dummy Classifier, which had a perfect recall, the RNN strikes a balance between precision and recall, highlighting its ability to capture positive instances while considering false negatives.

F1 Score:
The F1 score, representing the harmonic mean of precision and recall, is a balanced metric that considers both false positives and false negatives. The obtained F1 score of 0.521 reflects the overall effectiveness of the RNN in a binary classification setting, showcasing a trade-off between precision and recall.

It's important to take into account that tests were executed on the percentage meant for tests, not on the train cases.
But during training goes up to 89% of accuracy.
LSTM:
Accuracy : 0,777, Precision : 0,726, Recall : 0.83, F1 Score : 0,774

Conclusion

As we can see, the performance of the models improves significantly as they become more complex. The dummy classifier is not expected to do so well, but the vanilla RNN is a big step forward. The use of shortTerm memory carried along the sequence of layers, works well in terms of accuracy but the backpropagation method is very unstable in these models. That's why the LSTM, with its complex design with its forget-gate, input-gate and output-gate, separating long-term and short-term memory, does an even better job. It allows the gradient descent to work well and prevents it from exploding or vanishing, allowing an efficient use of backpropagation.