ASM7 Report

Ece Sena Etoglu

June 22, 2024

1 Introduction

The chosen task is Sentiment Classification for movie reviews. The dataset is a variant of The Stanford Sentiment Treebank. This dataset is chosen as it is a well known dataset for Sentiment Analysis tasks.

2 Dataset Description

train: 6920 samples, validation: 872 samples, test 1821 samples. Each sample is a sentence for a moview review.

3 Data Preprocessing

3.1 General Cleaning Purposes

For general cleaning purposes, the following preprocessing steps were applied:

- Remove HTML tags to clean the text from any HTML artifacts.
- Remove URLs to eliminate web links from the text.

3.2 Preprocessings to Increase Performance

Since this is a binary sentiment classification task (positive or negative) rather than a density of emotion, we only need the presence of an emotion. The following preprocessings are done as their object can be an indication of density of an emotion. We don't need them:

- Convert Upper Case Letters to Lower Case Letters
- Remove Punctuations
- Remove Emojis (rather than decoding it into text)
- Lemmatization: Lemmatization over stemming is preferred in this task (sentiment classification) because it reduces the words to their base dictionary form, which I think can give a better insight for the context.

4 Dataset Statistics

Various statistics can be observed from the .ipynb file

5 Models and Feature Sets

For each model, 2 different vectorization techniques are used: TF-IDF, Count Vectorizer. Their performances are evaluated on the validation dataset. Best performing technique is selected for each model and evaluated again on the test set to obtain a final result to compare with the other model.

- Model 1: Naive Bayes Classifier
- Model 2: RNN:
  For simplicity of the NN, Simple RNN and a Dense layer is used. For the optimizer, Adam is used for it's reduced need for manual hand tuning.

6 Evaluation Metrics

As it can be seen from the Dataset Statistics, the classes are balanced.
Also for the chosen task, sentiment classification for movie reviews, we are not strictly interested in semantic importance of recall (like in a medical task) or semantic importance of precision (like in a spam detection).

Because of these reasons, accuracy serves as an appropriate metric for comparison. But other metrics are also visible in the tables.

7 Experimental Results

7.1 Naive Bayes Internal Comparison

Count Vectorizer vs. TF-IDF:
Count Vectorizer performed 0.38% better. Although the difference is minor, the result might seem unexpected since TF-IDF is considered as a more advanced technique: It not only focuses on the frequency of words present in the corpus but also provides the importance of the words.

This slight advantage of Count Vectorizer may be attributed to specific dataset characteristics:

- **Short Documents:** For datasets with shorter documents, the raw frequency counts from Count Vectorizer might be more effective than TF-IDF's weighted scores.
- **Rare Words as Noise:** TF-IDF tends to emphasize rare words, which can introduce noise rather than valuable information in some datasets, potentially impacting performance.

7.2 RNN Internal Comparison

Count Vectorizer vs. TF-IDF in RNN:
Count Vectorizer outperformed TF-IDF by 1.33% on RNN. However, both methods fell short compared to Naive Bayes.

This might seem unexpected as RNN's were popular for NLP tasks. The reasons for this unexpected performance are discussed in 7.3 section. In short, Word embedding techniques like Word2vec are more suitable for RNNs than TF-IDF or Count Vectorizers.

Epochs 7 vs Epochs 15:
Increasing the epoch size reduced the accuracy %1.8 probably due to overfitting. I was not expecting an increase because the theoratical problem was the vectorization technique. Different results could be obtained by playing with the RNN architecture. As it is not the main point, it is not included in the code.

Utilizing Word2VEC
I implemented the code to utilize Word2VEC to finalize my theoratical reasonings which I believe to be true. However I couldn't solve some errors that I included in the notebook. Also I included the whole code for this section in commented form.

7.3 RNN vs. Naive Bayes

- **Performance Comparison:** Naive Bayes (with Count Vectorization) achieved 7% higher accuracy than RNN (with Count Vectorization), using accuracy as the metric.

  The lower performance of RNNs compared to Naive Bayes might seem unexpected, considering RNNs are generally effective for NLP tasks. However, this can be attributed to the vectorization techniques used, which do not fully leverage RNN's strengths in processing sequential data. RNNs excel in capturing sequential dependencies, which are lost in the bag-of-words representation used by TF-IDF and Count Vectorizer.

  Word embeddings like Word2Vec are better suited for RNNs. Although Word2vec also does not preserve order, it preserves the contextual information and as a conclusion of it: sequential information within the text.

- **Computational Efficiency Comparison:** Although a benchmark test was not conducted, runtime observations suggest that Naive Bayes is simpler and more computationally efficient than RNNs, requiring less training data, which aligns with theoretical expectations.

- **Confusion Matrix:** Confusion matrixes are plotted in the .ipynb file. Since it is a binary classification task, I found it hard to make conclusions from confusion matrixes. If it were a multiclass classification It would be more practical to come up with a comparison.

8 Conclusion

Naive Bayes is simple to implement and computationally efficient compared to RNNs, making it a suitable baseline for many NLP tasks. While RNNs are more powerful in capturing patterns, leveraging their strengths requires the use of appropriate vectorization techniques.

Additionally, although TF-IDF is a more advanced technique compared to Count Vectorizer, its advantages may not be fully realized depending on the dataset, as in this task.