Detecting Spam

CSEC620 - Project 4 - Mehul Sen

Overview

In this report, we present the implementation and evaluation of various algorithms and transformers on the SMSSpamCollection dataset. The dataset comprises unstructured text data, which we preprocess and convert into numerical vectors to build a classifier. Our primary objective is to identify whether an SMS message is spam or not. We have evaluated the performance of KMeans, KNN, and Transformer algorithms and compared our results with the paper's findings that used the same dataset.

Quick Start

This section will outline the necessary steps to use the K-Means, K-Nearest Neighbour and the Transformer algorithms. This project was completed using Visual Studio Code with Jupyter Notebook. If you require assistance, we recommend referring to the following resource for setting up the notebook: [Running Jupyter Notebook on Visual Studio Code | Medium]

1. *Setup Dataset Folder*: To set up the dataset folder, start by unzipping and extracting the smsspamcollection.zip files and placing them in a Dataset folder in the root directory. The dataset should be correctly located using the provided zip file, `project_4_sen.zip`. The desired folder structure should be as follows with the required files:

```
- requirements.txt
- project_4_detecting_spam.ipynb
- Dataset/
    - readme
    - SMSSpamCollection
```

2. *Import Libraries*: Next, import the necessary libraries from requirements.txt using the following command: `pip install -r requirements.txt`.
3. *Run Notebook*: Finally, run the Jupyter Notebook (`project_4_detecting_spam.ipynb`) and execute each cell block sequence from top to bottom.

Dataset

The project uses a single dataset known as the SMSSpamCollection dataset, which is described in Almeida, T.A., Gomez Hidalgo, J.M., Yamakami, A. Contributions to the study of SMS Spam Filtering: New Collection and Results. This dataset is a collection of SMS messages that have been tagged for SMS Spam research. It comprises a total of 5,574 SMS messages in English, with each message categorized as either ham (legitimate) or spam. Of the 5,574 messages, 4,827 (86.6%) are legitimate, while 747 (13.4%) are spam. Each message is contained in a single line in the dataset files and consists of two columns; one with the label (ham or spam) and the other with the raw text. For more details on the dataset, please refer to the readme file.

Term-Document Matrix

A term-document matrix is created using TF-IDF values. Each column is a document vector used in algorithms as a datapoint.

K-Means Algorithm

The K-Means algorithm was implemented from the Project 2, it was modified to fit an uneven matrix.

The key steps of KMeans are:

- Initialize K random centroids
- Calculate the distance of each point to each centroid
- Assign points to the nearest centroid
- Recompute centroids as cluster means
- Repeat steps 2-4 until convergence

The KMeans class implements this in Python, supporting the customization of distance metrics and maximum iterations. It is used on the term-document matrix to classify spam from ham.

K-Nearest Neighbor Algorithm

It is a simple and widely used algorithm for classification and regression tasks in machine learning. It works based on the principle that similar things exist in close proximity. In classification tasks, when given a new, unknown data point, kNN identifies its category by analyzing the categories of the 'k' nearest data points in the feature space. The majority class among these neighbors determines the classification of the new point.

Transformer Algorithm

Transformers are a type of model architecture introduced in the paper "Attention is All You Need" by Vaswani et al. They are particularly effective for many Natural Language Processing (NLP) tasks. It is comprised of the following six key components:

- Self-Attention Mechanism
- Encoder-Decoder Structure
- Positional Encoding
- Multi-Head Attention
- Feed-Forward Neural Networks
- Layer Normalization and Resodual Connections

They are used to take in the tokenized sequence from the dataset and passing them through the embedding layer, then the transformer block and finally through a global average pooling layer and two dense layers. The output is passed through a softmax activation to produce the probability of whether it is spam or ham.

Evaluation

The following were the performance metrics of each of the three algorithms:

KMeans Algorithm

Obtained Metrics

- **Precision** : 0.4356211865639221
- **Recall** : 0.48071692642764496
- **F1 Score** : 0.4558170408516287
- **Accuracy** : 0.8305814788226848

Calculated Metrics

- **FPR** = 0.343
- **MCC** = 0.166

KNN Algorithm

Obtained Metrics

- **Precision** : 1.0
- **Recall** : 0.6358381502890174
- **F1 Score** : 0.7773851590106007
- **Accuracy** : 0.9486133768352365

Calculated Metrics

- **FPR** = 0
- **MCC** = 0.623

Transformer Algorithm

Obtained Metrics

- **Precision** : 0.993006993006993
- **Recall** : 0.9403973509933775
- **F1 Score** : 0.9659863945578232
- **Accuracy** : 0.9910314083099365

Calculated Metrics

- **FPR** = 0.007
- **MCC** = 0.934

Results

In the paper titled "Contributions to the Study of SMS Spam Filtering: New Collection and Results" authored by Tiago A. Almeida, Jose Maria Gomez Hidalgo, and Akebo Yamakami, the findings are compared with my own. The evaluation of classifiers is based on Accuracy, Spam Caught (SC%), which is Recall, Blocked Hams (BH%), which is False Positive Rate (FPR), and MCC. After analyzing the results, the following trends have been observed between their findings and mine:

KMeans Algorithm

In the paper, the only clustering algorithm mentioned was the Expectation-Maximization (EM) clustering algorithm. This algorithm estimates cluster densities through an iterative soft clustering process. According to the paper, the EM algorithm achieved an accuracy of 85.40%, recall of 17.09%, FPR of 4.18%, and MCC of 0.185. However, my KMeans algorithm resulted in an accuracy of 83.05%, recall of 48.07%, FPR of 34.30%, and MCC of 0.166, which is lower than the EM result reported in the paper. This difference in performance could be due to variations in implementation details or parameters in comparison to the EM algorithm. Additionally, my algorithm only used 150 iterations, and increasing the number of iterations could potentially result in better performance. Nevertheless, these results suggest that clustering algorithms have relatively weak performance in SMS spam detection tasks.

KNN Algorithm

In the report, the performance of a 1-nearest neighbor (1NN) algorithm was analyzed. The accuracy was reported to be 92.7%, the recall was 43.81%, the false positive rate (FPR) was 0.00%, and the Matthew's correlation coefficient (MCC) was 0.636. On the other hand, my k-nearest neighbor (KNN) algorithm with k=1 achieved an accuracy of 94.86%, a recall of 63.58%, an FPR of 0.00%, and an MCC of 0.623. Compared to the results reported in the paper, my 1NN algorithm has a higher accuracy and recall, but a slightly lower MCC value.
These higher metrics indicate that my algorithm can more effectively identify spam compared to the 1NN algorithm reported in the paper. However, the slightly lower MCC suggests that my algorithm is worse at predicting the non-spam class, creating an imbalance.

Transformer Algorithm

In the paper, the transformer algorithm is not evaluated. Therefore, I compared its results with the best performing classifier, which happened to be SVM. The SVM classifier was reported to have an accuracy of 97.64%, recall of 83.10%, FPR of 0.18%, and MCC of 0.893. However, my transformer algorithm achieved better results with an accuracy of 99.10%, recall of 94.03%, FPR of 0.007%, and MCC of 0.934. These results demonstrate that the transformer's ability to learn contextual relationships between words is very useful in detecting spam. Additionally, the neural network architecture of the transformer algorithm provides much better performance compared to simpler linear models such as KMeans and KNN. Overall, the transformer algorithm significantly outperforms all the algorithms tested in the paper.

Conclusion

The Transformer algorithm dramatically exceeds the performance of algorithms in the paper, KNN is comparable, and KMeans is weaker. The transformer's more advanced deep learning approach is much better suited for this problem than linear/clustering models.