Brandon Stanyer

June 11, 2024

IST 664 Final Project

**NLP Models for Email Spam Detection**

**Introduction:**

Email spam detection is one of the many areas where NLP models have become highly effective and widely used. It is relatively infrequent that spam emails get past the modern algorithms used by Google and others. In this project, various NLP processes are used to train models for detecting spam emails. Many of these models show great promise, especially if they can be combined. This analysis involved a total of 8 different experiments and models using Naïve Bayes machine learning. In conclusion of this analysis, alternative models or adaptations are recommended.

**About the Dataset:**

This dataset used for this analysis is from the Enron Public Email Corpus. The dataset was released in 2006 and contains 3,672 normal emails (which are labeled “ham”) and 1,500 spam emails. Each email contains information about the date, sender, recipient, subject, and contents of each email. For this project, we will only be analyzing the writing contents of each email. For efficient analysis, only 1000 spam and 1000 ham emails were used.

**Data Pre-Processing:**

As mentioned prior, models in this analysis only used the writing contents of each email. Information about the date, sender, recipient, etc. were all ignored for this analysis. The first part of data preprocessing involved importing the contents of each email into a list. Each entry in the list is a tuple containing the email contents and a label of “spam” or “ham”. Furthermore, the email contents were tokenized using the NLTK word tokenizer function. As such, each email is a tuple that looks like the example below. Last, the emails were shuffled using random.seed(9) and random.shuffle. It should be noted that the seed of 9 is used as the seed for all random functions in this analysis.

*Example Email in the data: ( [ 'Hi' , 'Tom' , ',' , 'How' , 'are' , 'you' , '?' ] , ham )*

**Experiments:**

**Overview:**

There was a total of 8 different experiments done and models created for this analysis. Some of these models stand alone, but others are adaptations of previous models. Each machine learning model had the goal of correctly classifying the data as “spam” or “ham”. Each machine learning model was given a feature set which provided various features of each email. Each experiment had a different feature set or parameters for those features. All models were trained using the NLTK Naïve Bayes Classifier so that their results could be compared. For all these experiments, the same models and assessments were done to ensure that differences in accuracy could only be attributed to the differences in feature sets and parameters.

Since there were many experiments for this analysis, each experiment and model will be summarized. Each summary will contain an overview of the NLP process, the feature set, the accuracy measures, and a performance evaluation. For the training and testing of each model, a 90/10 split is done. As such, 1800 samples were used for training and 200 were used for testing. Below is a summary of each experiment.

**Experiment 1**

NLP Task: Bag of Words

Feature set Description:

The most common words of the emails are the features for each email. The most common 1,000 words found in the emails are identified and used to describe each email. Every email will have 1,000 features labeled as V\_word, where “word” is replaced with one of the most common words. Each feature is either true or false depending on whether or not that word is found in the dataset. As such, this is a sparse dataset, and each email only has a few features which are labeled as True.

|  |  |  |
| --- | --- | --- |
|  | Pred spam | Pred ham |
| True spam | 105 | 7 |
| True ham | 25 | 63 |

Accuracy: 84% Confusion Matrix:

Precision for “Ham”: 0.81 Precision for Spam: 0.91

Recall for “Ham”: 0.94 Recall for Spam: 0.72

F1-Score for “Ham”: 0.87 F1-Sccore for Spam: 0.80

Summary:

For a simple model such as this, the effectiveness of this model is relatively good. With an 84% accuracy, it appears to be an effective model. However, there were 25 out of 200 emails which were authentic emails but classified as spam. The precision for ham emails of 0.81 is lower than expected. This model could not be implemented because it would classify too many emails as spam. Important emails would regularly not make it to a user’s inbox, which would present problems.

**Experiment 2**

NLP Task: Bag of Words - Filtered

Feature set Description:

This has almost the exact same feature set as experiment 1. The two differences from experiment 1 are that stop words are removed and the number of common words increased to 2000.

Accuracy: 92.5% Confusion Matrix:

|  |  |  |
| --- | --- | --- |
|  | Pred spam | Pred ham |
| True spam | 101 | 11 |
| True ham | 4 | 84 |

Precision for “Ham”: 0.96 Precision for Spam: 0.88

Recall for “Ham”: 0.90 Recall for Spam: 0.95

F1-Score for “Ham”: 0.93 F1-Sccore for Spam: 0.92

Summary:

This model had some notable improvements from the first model. With an overall accuracy of 92.5%, we can classify this as an effective classification model for spam detection. This shows an improvement in performance at predicting ham emails, with precision of 0.96. However, the performance for spam emails could be improved.

**Experiment 3**

NLP Task: Cross Validation

Feature set Description:

This experiment uses the same feature set as experiment 2, but it applies cross validation. Changing the number of folds for cross-validation drastically changed the performance. 4 folds generated the best results.

Fold 1 Accuracy: 95.6%

Fold 2 Accuracy: 94.6%

Fold 3 Accuracy: 95.2%

Fold 4 Accuracy: 92.4%

Mean Accuracy: 94.45%

Summary:

The cross-validation performance was very similar to the previous experiment. Due to the relatively small size of the dataset, it can be speculated that more data could increase the performance with cross-validation. For future analysis, I would import more email data and attempt cross-validation again. All things considered, the performance of this model with cross validation is the best so far.

**Experiment 4**

NLP Task: Bag of Words – Filtered with Less Words

Feature set Description:

This is another modification of experiment 1, using common words as word features. This model filters out stop words, like experiment 2, but uses only the 1000 most common words as word features.

Accuracy: 93% Confusion Matrix:

|  |  |  |
| --- | --- | --- |
|  | Pred spam | Pred ham |
| True spam | 99 | 14 |
| True ham | 0 | 88 |

Precision for “Ham”: 1.00 Precision for Spam: 0.86

Recall for “Ham”: 0.88 Recall for Spam: 1.00

F1-Score for “Ham”: 0.93 F1-Sccore for Spam: 0.93

Summary:

Small changes to the bag of words model shows some improvement over previous models. With an accuracy of 93%, this can be considered an effective model for this task. This model also has no errors with predicting authentic emails as spam emails, which is more important than its converse. The performance of predicting spam emails is still lower than ideal, but perhaps with more changes to parameters this can improve.

**Experiment 5**

NLP Task: POS Tags

Feature set Description:

This feature set uses the parts of speech tags in each email as word features. Along with classifying each token as noun, verb, etc., the feature set contains features for the previous word, the next word, and the suffix of each word. This feature set required more complex programming to obtain these word features and flatten the feature set for analysis.

Accuracy: 85% Confusion Matrix:

|  |  |  |
| --- | --- | --- |
|  | Pred spam | Pred ham |
| True spam | 93 | 19 |
| True ham | 11 | 77 |

Precision for “Ham”: 0.89 Precision for Spam: 0.80

Recall for “Ham”: 0.83 Recall for Spam: 0.88

F1-Score for “Ham”: 0.86 F1-Sccore for Spam: 0.84

Summary:

Using Parts of Speech Tags appeared to be moderately effective at classifying spam emails. It was the worst performing model so far with an accuracy of 85%, which indicates that the parts of speech labels of the words in an email may not be the best predictors of spam emails. It may be worth considering combining POS tags and other Bag of Words features into one feature set.

**Experiment 6**

NLP Task: Word Frequency

Feature set Description:

This feature set contains features which are the count of the common words in each email. Although this is similar to experiment 1, each feature is an integer and not a True/False value. Stop words are once again filtered out and only the most common 1000 words are used for features.

Accuracy: 86.5% Confusion Matrix:

|  |  |  |
| --- | --- | --- |
|  | Pred spam | Pred ham |
| True spam | 85 | 17 |
| True ham | 10 | 88 |

Precision for “Ham”: 0.89 Precision for Spam: 0.84

Recall for “Ham”: 0.83 Recall for Spam: 0.90

F1-Score for “Ham”: 0.86 F1-Sccore for Spam: 0.87

Summary:

This model does not appear to be very effective compared to the first and second experiments. From this experiment we can learn that the count of each common word is not as valuable of a predictor as simply knowing if a word is present. This is surprising because the model has similar data and therefore should perform similarly. Perhaps this model would be more effective with a machine learning algorithm other than Naïve Bayes, such as SVM.

**Experiment 7**

NLP Task: TF-IDF Scores

Feature set Description:

This feature set utilized term frequencies (TF scores) as well as Inverse Document Frequencies (IDF scores). In short, these features define how often particular words appear within each email. Commonly used words have higher scores.

Accuracy: 91% Confusion Matrix:

|  |  |  |
| --- | --- | --- |
|  | Pred spam | Pred ham |
| True spam | 85 | 10 |
| True ham | 8 | 97 |

Precision for “Ham”: 0.91 Precision for Spam: 0.91

Recall for “Ham”: 0.89 Recall for Spam: 0.92

F1-Score for “Ham”: 0.90 F1-Sccore for Spam: 0.92

Summary:

Using Term Frequency-Inverse Document Frequency scores yielded good results. This more advanced algorithm achieved a 91% accuracy on the test data. Although it is not the highest accuracy of all the experiments so far, it does have more balanced results. The errors are evenly distributed between ham and spam. With the more data or a different machine learning algorithm, this has the potential to scale up best due to the seeming lack of bias.

**Experiment 8**

NLP Task: Bag of Words + Custom Features

Feature set Description:

This feature set combines the most accurate model feature set with additional features. The original bag of words feature set used previously filters out stop words and uses only the presence of the 1000 most common words for features. I have added additional features of email length and average word length as additional features for this feature set.

Accuracy: 93.5% Confusion Matrix:

|  |  |  |
| --- | --- | --- |
|  | Pred spam | Pred ham |
| True spam | 99 | 13 |
| True ham | 0 | 88 |

Precision for “Ham”: 1.00 Precision for Spam: 0.87

Recall for “Ham”: 0.88 Recall for Spam: 1.00

F1-Score for “Ham”: 0.94 F1-Sccore for Spam: 0.93

Summary:

This relatively simple model takes the best features of the experiments so far and utilizes them. Adding the additional features contributed to a slight increase in model performance. It should be noted that this same model generated accuracy measures as high as 98.5% depending on the train/test split. This showcases that it is an effective classification model for email spam. However, it may also indicate that the model has been overtrained on this dataset. It should also be noted that this model is very good at classifying authentic emails but does struggle more with classifying spam emails.

**Conclusion:**

These experiments showcase that NLP processes can be used to generate highly effective models for email spam detection. Each experiment helped revealed aspects of authentic and spam emails which were most effective in classification. It is likely that with a combination of these NLP processes, or utilizing other NLP tasks, a model with upwards of 98% accuracy can be achieved.

Surprisingly, one of the most effective predictors for spam emails was the presence of particular words. By using the bag of words process, models were able to be trained efficiently and generated effective models. It is clear that adjustments to the parameters of common words is required depending on the dataset. The best results came from properly filtering out words and combining other features in. Adding other simple features like email length and average word length made small improvements to these models.

As stated prior, a more effective model can be made using a combination of the features in each of these experiments. For a combination of these, it appears that incorporating TD-IDF scores into the bag of words feature set could generate better results. POS tags also could be useful, but there is not enough evidence for that from this analysis.

Given these results, it should be noted that the data and preprocessing methods may not translate to other datasets. Given the relatively small size of the dataset, it is quite possible that many of the models have been overtrained to the data from this corpus. Further analysis should be done to see if similar results can be obtained on larger or newer email collections.