International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2017 IJSRCSEIT | Volume 2 | Issue 3 | ISSN : 2456-3307

# Smart Email Notifier Using Supervised Learning

**Amey Bahulkar, Denny Sam, S. S. Pawar**
Sinhgad Academy of Engineering, Kondhwa, Maharashtra, India

## ABSTRACT

Communication via email is surely the most common and important aspect of a professional life and oftentimes our inbox is inundated with worthless emails. Various studies have shown that interruptions due to email usage have a negative impact on productivity thus there is a strong need of an intelligent system that could notify the user only when an important email arrives so This paper focuses on the applicability of machine learning to answer a simple question: is an incoming email worthy enough of user's time? To answer this question we have used two ML models namely, Multinominal Naive Bayes and Support Vector Machines. In addition, this project aims to use the learned models to build a real-time email notifier software that will estimate whether a received email is worth notifying the user.

**Keywords:** Supervised Learning, Naïve Bayes, Support Vector Machines, TF-IDF, python, .NET.

## I. INTRODUCTION

Oftentimes, we are disturbed and interrupted by the notification of a new email arriving. Should we break away from our crucial task to read it or not? Ideally, we would only accept the disturbance to our work only if said email requires an urgent reply. Our goal is to model various features extracted from email data set using multinomial naive Bayes and SVM classifiers to determine which feature is most effective at predicting whether an email requires a response. This project explores the details of which email features provide the most useful information in terms of predicting whether a received email is worthy of user's time This paper aims at developing a mail client that works on the learned model and intelligently notifies the user depending on the nature of the incoming email. In section 2 we discuss in detail about the data collection process that resulted in the formation of training and test data. In section 3 we discuss the ML models that we used. In section 4 we train our models on the data and present the results. In Section 5 the results are discussed and conclusions are presented. In section 6 we describe how the learned models were used to develop a real time software. Finally, in section 7 we discuss further directions that the project could take.

## II. MATERIALS AND METHODS

A significant amount of time was spent on data preprocessing and manipulation as the raw data was not of any use. We used a Python script that is capable of accessing email. The program iterates through all the received email in the local Gmail account within a specified time period and populates various data structures that allow easy access to the source data for the features with which we are interested in experimenting[2]. The emails that we extracted from our inbox were of little use as it contained way more social networking emails than professional and important emails. To overcome this problem we also used the Enron email dataset to supplement our dataset[6]. Preparing the dataset resulted into a feature matrix containing a row for each email, describing its features like subject, body, to, etc.

| Raw Features | Ref. Name |
|---|---|
| 'Body' term frequency | *body* |
| 'Subject' term frequency | *subject* |
| 'To' names | *to* |
| 'From' names | *from* |
| 'CC' names | *cc* |
| | |

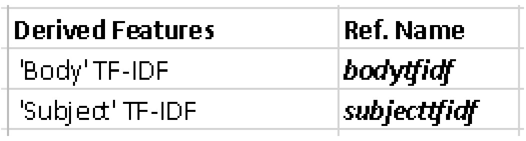
**Figure 1:** Features and their associated reference names.

### 2.1 Raw Features

The body feature was extracted from the BODY and the subject feature was extracted from the SUBJECT field. These fields were processed to make them punctuation-free. The last three raw features extracted were sourced from the TO, CC and FROM fields. Column i for each email in these feature matrices was a binary feature corresponding to whether or not the person at index i in the associated person dictionary was present in the respective field of the email i.e. a feature vector.

### 2.2 Derived Features

Even if the features are cleaned and properly preprocessed, they are often inefficient at the task that we want to do. The reason behind this is that the words like "is", "the", "and" etc. Are more common than more important and rare words like names or places. This leads to inefficient results. To overcome this we developed TF-IDF of the features[5]. In addition, we also consider the exclusivity of the email, i.e. if a user is one of the two recipients of the email then the email is worthy of a reply and an email which is sent to 50 recipients may not require a reply. This information about the recipients is collected from the To and CC fields. As we will see, these derived features tend to be efficient for the computations and give more reliable results. The derived features are shown in figure 2.



| **Derived Features** | **Ref. Name** |
|---|---|
| 'Body' TF-IDF | ***bodytfidf*** |
| 'Subject' TF-IDF | ***subjecttfidf*** |

**Figure 2 :** Derived features and their associated reference names.

### 2.3 Models

Multinomial naive Bayes was selected as the initial model due to its very simple implementation, lack of complex, empirically determined model parameters, and reasonable performance on text classification problems. SVM was selected so as to possibly fit a far more complex decision boundary (when using a non-linear kernels), thus giving a richer model. Additionally, a large amount of email history is available and SVM is better able to take the advantage of the additional data points in terms of providing better predictive accuracy. The metrics used to explore the performance of these various features are the classification errors and the F1 score. The F1 score attempts to provide a good measure of the models' accuracy by taking into account the models precision and recall. A F1 score of 1 indicates the model perfectly classifies the given data. The F1 score is very sensitive to the number of false positives and false negatives relative to the true number of positives and negatives, respectively, giving a powerful indicator of model success even in imbalanced data sets[4]. Formula for the F1 score is given in figure 3.

$$F1 = 2\frac{(precision \cdot recall)}{precision + recall}$$

**Figure 3:** The F1 score equation.

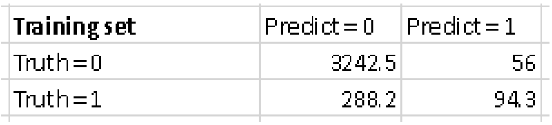
F1 score is a good indicator of the efficiency of the model when the dataset is imbalanced.

## III. RESULTS

In this section, we discuss the two models trained and their results in detail. We first trained the MNB and ran a cross-validation. Then we did the same on SVM model. For both of the below models, we ran k-fold cross validation with k = 10 on the train data and test data. The error metrics reported for are averaged over all 10 folds.

### 3.1 Naive Bayes

Multinomial naive Bayes was run against each feature to determine each feature's value. The accuracy of this model fluctuated with different feature set and with the most important features the accuracy came out to be 90%. For the rest of this paper, we will be focused on the F1 score. The confusion matrix of MNB is shown below.



| **Training set** | Predict = 0 | Predict = 1 |
|---|---|---|
| Truth = 0 | 3242.5 | 56 |
| Truth = 1 | 288.2 | 94.3 |

**Figure 4:** Confusion matrix on training set.

MNB behaves similarly on the body, subject, and subjecttfidf features with the most valuable model obtained coming from either the subjecttfidf or bodytfidf features. The to, cc, and from features were practically useless as they resulted in a model with training and test F1 scores below ~0.2 for all data sizes. Interestingly, the only feature on which MNB was able to decently model the training set was bodytfidf. Confusion matrix on the test data set is shown below.

| Test set | Predict=0 | Predict=1 |
|---|---|---|
| Truth=0 | 347.5 | 19 |
| Truth=1 | 38.1 | 4.4 |

**Figure 5:** Confusion matrix on Test set.

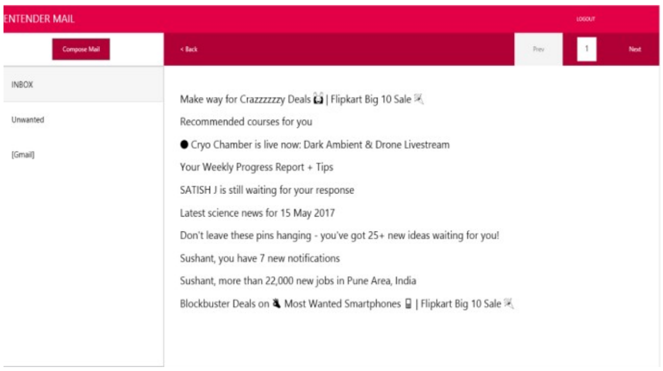
It was observed that other than body and bodytfidf, all other features were fruitless as they contained very little information to predict the output.

### 3.2 SVM

Next, we implemented SVM against each feature. We used the C-SVM implementation of SVM present LIBSVM for Matlab[1]. The SVM model didn't perform exceptionally better than MNB. The accuracy of the SVM was almost same as of MNB. SVM, as discussed above performed better on derived features than the raw features.[3] Overall, SVM didn't have any significant change in the F1 score and performed equivalently to MNB and the features performed comparatively well with subject, to, and subjecttfidf being the best with obtaining stable F1 scores on the test set between 0.3 and 0.4 for the data sets. Surprisingly, SVM didn't perform exceptionally better than MNB as the F1 score wasn't improved much.

## IV. DISCUSSION

Altogether, the ML models performed a good job in predicting the nature of email recieved. The overall problem that significantly affected this project was the high bias error associated with the majority of the model feature pairings and lack of diversity in the derived dataset which lead to significant under fitting and inability to adequately fit even the training data. Even though the models did perform well on some features, they failed to perform efficiently on other features. We believe that the model will perform better if we use the ensemble techniques and diverse data. Furthermore, the lack of complete dataset also affected the performance of the model. Specifically, the following aspects of the problem were not captured appropriately in the features: (1) Only 1 person within a role or team needs to provide a response. Responses from those "equivalent" people should be included in the positive class. (2) Responses can be sent over different media. In our day-to-day life, it is common to interact with someone via instant message after receiving a critical email rather than emailing a reply. Alternatively, one might just go to their office rather than sending a reply. (3) Multiple emails can be sent in one thread and if an email is replied not immediately prior, but several emails ago, which is not a scenario that can be captured with our current approach. Additionally, it's important to note that as people switch projects and change what they are working on over time, a model trained on various features 4-5 months ago might do a very poor job generalizing to the current email sent now. So, a periodic learning of the model is required for a proper working of the software. This can be seen even in our data sets where some features improved the performance of the model over certain email inputs while others deteriorated it. We believe that boosting techniques, ensemble techniques will significantly improve the performance of the model. In addition, a so called complete dataset will also improve the accuracy of the models.



Make way for Crazzzzzzy Deals | Flipkart Big 10 Sale
Recommended courses for you
● Cryo Chamber is live now: Dark Ambient & Drone Livestream
Your Weekly Progress Report + Tips
SATISH J is still waiting for your response
Latest science news for 15 May 2017
Don't leave these pins hanging - you've got 25+ new ideas waiting for you!
Sushant, you have 7 new notifications
Sushant, more than 22,000 new jobs in Pune Area, India
Blockbuster Deals on Most Wanted Smartphones | Flipkart Big 10 Sale

**Figure 6:** User interface(snapshot) of the software.

## V. CONCLUSION

Considering real-world application point of view, we developed a real-time software based on our learned models on Windows WPF platform. The software has two main elements: ML models and imapx[7] package. The imapx package provided most of the required functionality like real-time mail update, email retrieval, etc,. In addition, IronPython[8] module was used to facilitate the integration of python and c#. The working of the developed software is simple, when an email is received it is first sent to the learned models which predicts whether it is an important email. If the prediction is positive, the user is notified by a balloon message which when clicked redirects to the newly received email. GUI of the software can be seen in figure 6.

## VI. REFERENCES

[1]. Chang, C.-C., and Lin, C.-J. LIB-SVM: A Library for Support Vector Machines. Department of Computer Science, National Taiwan University, 2001.

[2]. Chaput, M. stemming 1.0. https://pypi.python.org/pypi/stemming/1.0Feb. 2010. Python implementation of theporter2 stemming algorithm.

[3]. Hsu, C.-W., Chang, C.-C., and Lin, C.-J. A Practical Guide to Support Vector Classsification. Department of Computer Science, National Taiwan University, Taipei, Taiwan,2003.

[4]. Wikipedia. F1 score. http://en.wikipedia.org/wiki/F1_score. Wikipedia page for the F1 score.

[5]. Wikipedia. tf-idf. http://en.wikipedia.org/wiki/Tf%E2%80%93idf. Wikipedia page for the TF-IDF metric.

[6]. https://www.kaggle.com/wcukierski/enron-email-dataset

[7]. A cross-platform library for .NET http://imapx.org

[8]. The IronPython package http://ironpython.net