1 Introduction

Nowadays, a typical user receives about 40-50 email messages every day. For some people hundreds of messages are usual. Thus, users spend a significant part of their working time on processing email. As the popularity of this mean of communication is growing, the time spent on reading and answering emails will only increase. At the same time, a large part of email traffic consists of non-personal, non-time critical information that should be filtered. As a result, there has recently been a growing interest in creating automatic systems to help users managing an extensive email flow.

Generally, the main tool for email management is text classification. A classifier is a system that automatically classifies texts into one (or more) of a discrete set of predefined categories. For example, for email management one could benefit from a system that classifies incoming messages as junk and non-junk or as important and unimportant.

Text classification means reading the contents of the text and then identifying the context in which it is written and classifying it into pre-defined categories. For example, text can be a paragraph from a newspaper article or data from business documents. Such a text can be categorized into labels like sports news, entertainments news or customer reviews, marketing, etc. This ability of text classification can be applied to email classification to categorize the email messages that are sent to users in large numbers into folders for ease of segregation. The algorithm will be given a large dataset of emails and the folders of categorization, and the output will be class labels for each of the emails. There are various text classification algorithms used currently that can be broadly classified into two categories: Eager Learning Algorithms and Lazy Learning Algorithms. Both the types of algorithms have two phases: Training Phase and Classification Phase. In the training phase, the algorithm is given a set of training data that has email messages along with their class labels (folder to which email belongs). Since the training phase requires already classified emails, such an algorithm is called Supervised Learning Algorithm, as it is supervised with known emails. In the classification phase, the learned algorithm is used to classify the unread emails into folders. Eager Learning is a learning method in which the system tries to construct a general, input independent target function during training of the system, as opposed to in lazy learning, where generalization beyond the training data is delayed until a query is made to the system. Eager learning is an example of offline learning, in which post-training queries to the system have no effect on the system itself, and thus the same query to the system will always produce the same result. Lazy Learning is a learning method in which generalization beyond the training data is delayed until a query is made to the system, as opposed to in eager learning, where the system tries to generalize the training data before receiving queries.

Most text classification approaches use supervised learning for building a classification system. In a supervised learning setting, we are given examples that belong to a two-class concept (e.g. interesting and uninteresting email). All examples are labeled with respect to their membership in one of the two classes. A machine learning system induces, from these examples (referred to as a training set), a general description of both classes. It is important that such a description has predictive power, i.e. it predicts with high degree of success the class of unseen

examples. In our example, we want a machine learning system to come up with a definition of what makes an interesting email message, so that this definition will successfully pick interesting email from the user's inbox in the future (e.g. it will work well on examples different from the ones in the training set).

The email classification problem can be solved by using both these approaches. The Naïve Bayes Algorithm is an eager learning algorithm based on probability functions while k-Nearest Neighbors Algorithm is a lazy learning algorithm based on similarity measure and distance metrics. These text classification algorithms can be modified to classify emails into folders. The goal of this project is exploring the possibility of applying an efficient algorithm to email classification i.e. the ability to classify emails dynamically based on the category labels generated by user to email classification.

1.1 Motivation

Text classification in Data Mining and Machine Learning is a growing field of research. New algorithms are being developed day by day to optimize the old algorithms and overcome their shortcomings. On a similar ground, as lot of data is being communicated using emails, email classification is the need of the hour. Thus, to contribute to the field of text classification and solve the problem of email classification by improving the algorithms and classifying data in a more efficient manner, we chose to address this problem. The existing system of email classification was not sufficient for the current volume as well as diversity of data.

1.1.1 Disadvantages of the Current System

Presently, well known email clients such as Yahoo!, Gmail and Outlook do provide categorization of emails into tabs like Inbox, Social and Promotions. However, such classification has many drawbacks.

- The classification is based only on the sender's email address and sometimes on the specific keywords contained in the email. This does not provide any customization option and the user has to manually change the folder to which an email should belong.
- All the labels under which emails are classified are predefined and the user cannot change them or provide a new custom label for classification.
- To provide customization, these emails clients allow users to create custom labels. However, these labels are also based on sender's email address or certain keywords in the email message. Thus it does not provide proper content classification.
- Web clients also allow users to create custom folders to organize all the emails. But the user has to manually select a folder for each email which is a cumbersome task if there are a lot of pending email messages.

1.1.2 Advantages of the Proposed System

The email classification system that we propose overcomes all the above problems.

- ➢ The classification labels are very much user personalized so that users' emails are classified into folders of relevant topics.
- ➢ A smart email classification algorithm automatically assigns labels to emails based on the message content using supervised learning algorithms that learn from the user behaviour.
- ➢ Users can also create custom labels and train the algorithm with email examples belonging to that category, and the system, after learning from this training set, can classify future unread emails efficiently based on this training set.

1.1.3 Challenges

The characteristics of emails differ significantly and as a consequence email categorization poses certain challenges, not often encountered in text or document categorization. Some of the differences and challenges are:

- ➢ Each user's mailbox is different and is constantly increasing. Email contents vary from time to time as new messages are added and old messages are deleted. A categorization scheme that can adapt to varying email characteristics is important.
- ➢ Manual categorization of emails is based on personal preferences and hence the criteria used may not be as simple as those used for text categorization. This distinction needs to be taken into account by any technique proposed for email categorization.
- ➢ The information content of emails vary significantly, and other factors, such as the subject field, sender, CC field, BCC field, the person email is addressed to, play an important role in categorization. This is in contrast to documents, which are richer in content resulting in easier identification of topics or context.
- ➢ Emails can be categorized into folders and could also be categorized into subfolders within a folder.

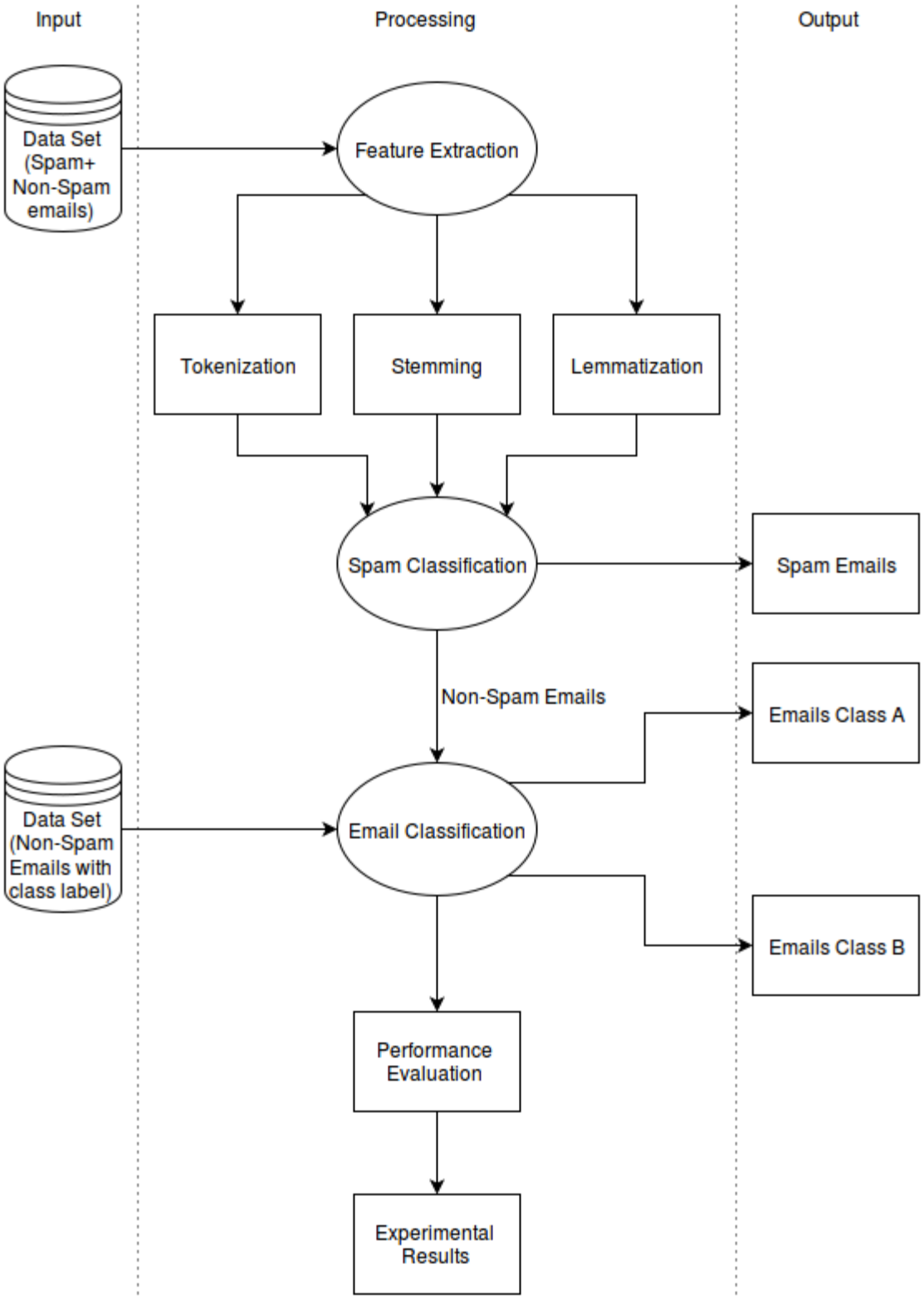
The differences in the emails categorized to subfolders may be purely semantic (e.g., meeting with travelling folder, conference expenses within travelling folder and many more).

1.2 Problem Definition

The problem is to classify a large volume of email data containing a variety of messages into pre-defined folders, that will help the users to organize their emails systematically and in a more convenient manner. The approach is to use text classification algorithms of Data Mining, Artificial Intelligence and Machine Learning, modify and optimize them according to the email classification problem and create an efficient system with high accuracy.

1.3 Methodology

The methodology for the email classification problem is shown in the following figure:



Feature Extraction:

The input consisting of a dataset containing both spam and non-spam emails is first preprocessed to obtain the various features. This is an extremely important step as the features extracted are used to classify the emails in the training set (supervised learning) and later used for classification of emails in the testing set (unsupervised). These features thus directly have implications on the how an algorithm behaves as well as on the efficiency and accuracy of the algorithm.

Spam Classification:

Spam is the unwanted part of email which is something that is not desired. It is of utmost importance that these mails be eliminated in order to classify emails into different categories more efficiently.

Email Classification:

This is the step where the actual classification of non-spam emails takes place using supervised learning algorithms. The features extracted are used in classifying emails in the training set and then based on these results emails are classified in the testing set.

1.3.1 Algorithm for Email Classification

The steps involved in the process to achieve these objectives are:

Input: Dataset of unclassified emails.

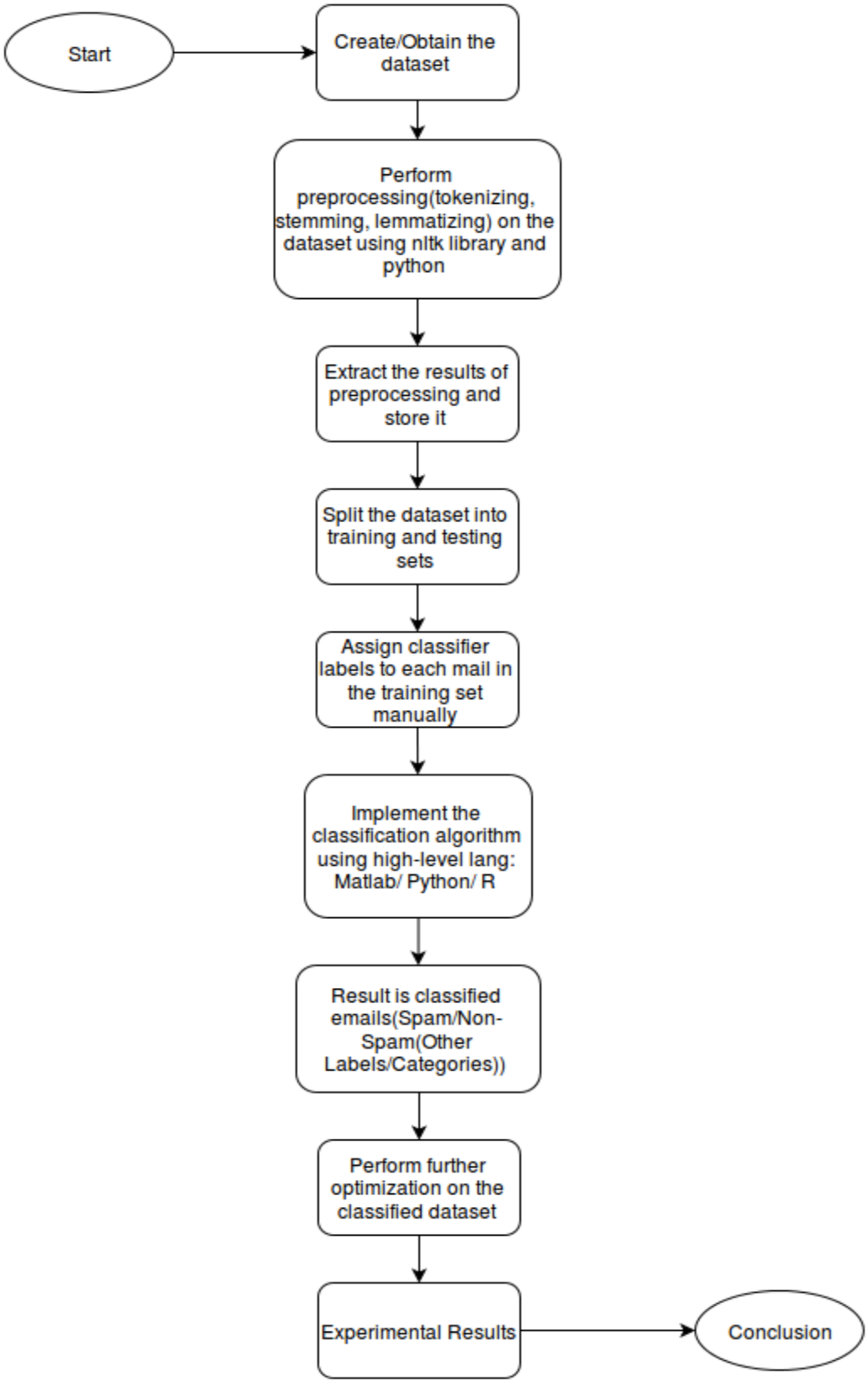
Output: Different categories of classified emails

Algorithm:

1. Extract the features relevant to the categories from the dataset.
   1.1. Perform pre-processing on the dataset to identify keywords and tags.
   1.2. Split the pre-processed dataset into training and testing sets.
   1.3. Use these tags to build classifier labels for the training and testing sets.
2. Implement the supervised learning classification algorithm on the training set and testing set and check its accuracy.
   2.1. If accuracy less than minimum
   2.2. Go back to step 1 to extract features again.
   2.3. Else
   2.4. Optimize the algorithm to improve accuracy.
      2.4.1. If accuracy not improved
      2.4.2. Go back to step 2.
      2.4.3. Else
      2.4.4. Compare the accuracy with that of other algorithms.
         2.4.4.1. If accuracy less than other algorithms
         2.4.4.2. Switch to other algorithm.

1.3.2 Flowchart

The flowchart depicting the steps involved in email classification is as follows:


Start
Create/Obtain the dataset
Perform preprocessing(tokenizing, stemming, lemmatizing) on the dataset using nltk library and python
Extract the results of preprocessing and store it
Split the dataset into training and testing sets
Assign classifier labels to each mail in the training set manually
Implement the classification algorithm using high-level lang: Matlab/ Python/ R
Result is classified emails(Spam/Non-Spam(Other Labels/Categories))
Perform further optimization on the classified dataset
Experimental Results
Conclusion

1.4 Objectives

The project aims to achieve the following objectives:

- Categorization of email according to the content (mailer, recipient, subject, message, cc list, bcc list, attachments).
- Categorization of email according to activities (organizing conference, reviewing papers, purchasing equipment, making design decisions, managing candidate interviews, etc.).

1.5 Summary

In this chapter, we gave a brief introduction about the project specifying the nature of the project, explaining specific aspects of the project and giving an overview of what is to be done next. First, we explained the motivation behind choosing the topic, the need for a better system and why the current system fails, and finally the challenges that are faced due to this problem. Next, we defined the email classification problem and the approach to be used in addressing this problem. A methodology was also introduced regarding what is necessary for classifying emails in an appropriate manner and an algorithm and flowchart was proposed that is to be used in implementing that methodology. Finally, we also presented various objectives that need to be fulfilled as a part of the problem.

2 Literature Review

Before proceeding with the project, we went on to read a few research papers on text and email classification by various researchers, studied their algorithm and reviewed their approach. There are many approaches to classification and the research papers showed a variety of implementation techniques. The papers that we selected were on topics like spam & non-spam email classification, text classification using k-NN algorithm and text classification using Naïve Bayes algorithm.

2.1 Spam & Non-spam email classification

One of the major goals of email classification is spam detection. This sub section describes some research papers related to spam email classification.

We selected some papers, based on citation, related to spam detection or filtering. Those papers are: Blanzieri and Bryl 2008, Zhou et al. 2010, Harisinghaney, A. et al. Different papers discussed the use of different algorithms and also application of the algorithms in different places between email senders and receivers.

Blanzieri and Bryl (2008) presented a technical report in 2008 to survey learning algorithms for spam filtering. The paper discussed several aspects related to spam filtering such as the proposals to change or modify email transmission protocols to include techniques to eliminate or reduce spams. Some methods focused only on content while others combined header or subject with content. Some other email characters such as size, attachments, to, from, etc. were also considered in some cases. Feature extraction methods were also used for both email content, attached and embedded images.

Zhou et al. (2010) proposed a spam-based classification scheme of three categories. In addition to typical spam and not spam categories, a third undetermined category is provided to give more flexibility to the prediction algorithm. Undecided emails must be re-examined and collect further information to be able then to judge whether they are spam or not.

Harisinghaney, A. et al. presented a research paper with the aim to detect text as well as image based spam emails. To achieve the objective they applied three algorithms namely: KNN algorithm, Naï ve Bayes algorithm and reverse DBSCAN algorithm. Pre-processing of email text before executing the algorithms is used to make them predict better. The paper uses Enron corpus's dataset of spam and ham emails. They provide comparison performance of all three algorithms based on four measuring factors namely: precision, sensitivity, specificity and accuracy. They are able to attain good accuracy by all the three algorithms.

2.2 K-Nearest Neighbors Algorithm

In this section, we will describe some papers related to the analysis of email messages for purposes other than spam detection. This section primarily describes research papers that used k-NN algorithm for email classification. Two IEEE papers on the same were found namely: Lijuan Zhou et al. 2010, Chakrabarty et al. 2014.

k-NN has been used in research paper by Lijuan Zhou et al. 2010, to increase the precision of classification because of the uneven density of training data with respect to text classification. It involved preprocessing of training data by clustering and then classify using a new KNN algorithm which adopts a dynamic adjustment in each iteration for the neighborhood number k. This resulted in good performance in text classification.

Chakrabarty et al. 2014 presented a frame work for Email classification based on an improved k-NN classification using a minimum spanning tree algorithm considering the case where the number of clusters are unknown initially. Experimental results showed that the proposed algorithm outperformed state of art classification algorithms like standard k-NN.

We will be using k-NN algorithm for email classification beyond spam filtering which has not been done till now. The use of k-NN was much focused towards binary classification. At first k-Means algorithm will be used to create clusters from large email data representing different user email folders. Then for each new mail introduced in inbox nearest cluster will be found. Now k-NN algorithm will be applied to the new mail to find k-Neighbors within the identified cluster. Thus the class with major votes will be assigned to new mail. This is beneficial when there are overlapping clusters. Also the dataset to be examined in k-NN will be proportional to the number of clusters. Thus speed and accuracy will increase.

2.3 Naïve Bayes Algorithm

This section describes the research papers that specifically focus on Naïve Bayes algorithm for text and email classification. Three papers have been described here namely: A. R. Kulkarni et al. 2012, Noaman et al. 2010, Han Joon Kim et al.

A. R. Kulkarni et al. 2012 presented a research paper that did classification of abstracts considering their contexts using Naïve Bayes classifier and Apriori association rule mining. Here, the context of an abstract is found by looking for associated terms that help focus on abstract and interpret the information beyond simple keywords.

Noaman et al. 2010 have used the Naïve Bayes approach to assign online Arabic documents, the information libraries and Arabic document corpus to various categories. They were able to use rooting algorithms in combination with Naïve Bayes and obtained an average accuracy of ~62.23%.

Han Joon Kim et al. proposed a novel idea of incremental learning with naïve bayes classifier since it is easy to incrementally update its pre-learned classification model and feature

space. Our work mainly focuses on improving naive Bayes classifier through feature weighting strategy. Their basic idea is that parameter estimation of naive Bayes can consider the degree of feature importance as well as feature distribution. In addition, they have extended a conventional algorithm for incremental feature update for developing a dynamic feature space in operational environment.

3 Preprocessing

Preprocessing is the process of converting data into computer understandable format. Filtering out useless data is the primary step of pre-processing. The basic idea of preprocessing is to combine the various programming techniques with large quantities of text that will automatically extract key words and phrases that sum up the style and content of a text. Some of the terms and their meanings in preprocessing are as below:

- **Corpus** - Body of text, singular
- **Lexicon** - Words and their meanings
- **Token** - Each "entity" that is a part of whatever was split up based on rules

NLTK module has been used for performing pre-processing on the text.

3.1 Tokenizing

The first step of pre-processing is Tokenizing. There are two ways in which tokenizing can be performed – splitting across sentences or words. NLTK module which takes into account various factors while tokenizing. Consider the following example: "Hello Mr. Suraj, how are you doing today? The weather is pleasant, and Programming is awesome. The sky is pinkish-blue. You shouldn't eat eggs."

The above sentence, after tokenizing would be something like this: ['Hello', 'Mr.', 'Suraj', ',', 'how', 'are', 'you', 'doing', 'today', '?', 'The', 'weather', 'is', 'pleasant', ',', 'and', 'Programming', 'is', 'awesome', '.', 'The', 'sky', 'is', 'pinkish-blue', '.', 'You', 'should', "n't", 'eat', 'eggs', '.']

There are a few things to note here. First, notice that punctuation is treated as a separate token. Also, notice the separation of the word "shouldn't" into "should" and "n't." Finally, notice that "pinkish-blue" is indeed treated like the "one word" it was meant to be turned into. The next step would be to derive a meaning by looking at the words. We have to think of ways to add value to the words but also see that some of the words are worthless. They are a form of “stop words”, which will be the next step in preprocessing.

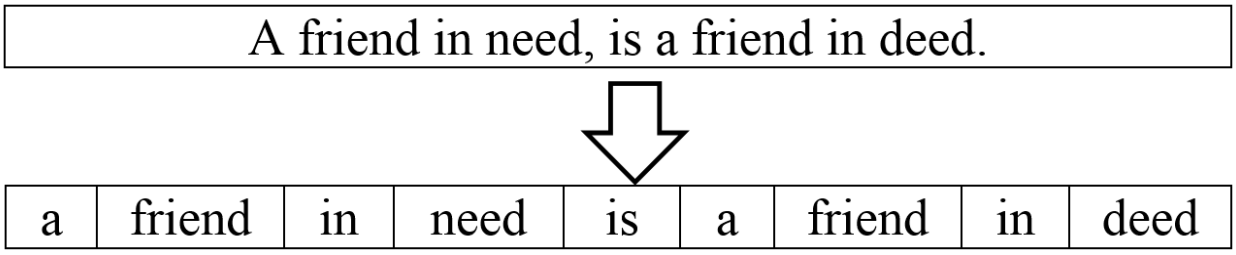


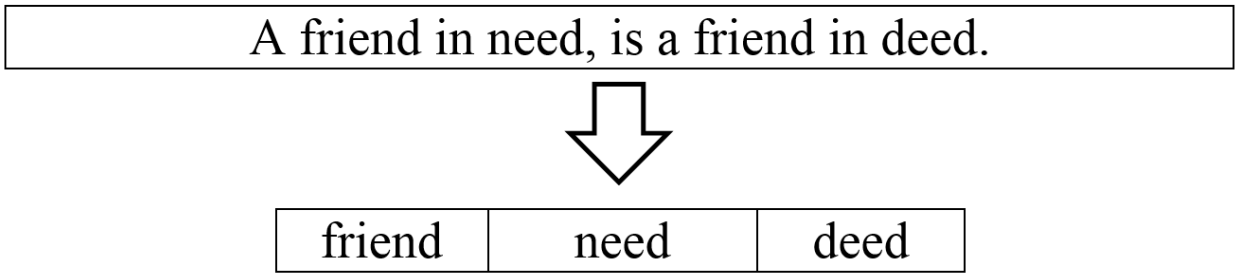
Fig. 1. Tokenization Example

3.2 Stop words removal

Natural language processing refers useless words (data), as stop words. It is observed that some words carry more meaning than other words. Also, there are words which are plain useless, and are filler words. An example of one of the most common, unofficial, useless words is the phrase "umm." People use "umm" frequently. The frequency of usage varies from person to person. For most analysis, these words are useless.

We would not want these words taking up space in our database, or taking up valuable processing time. As such, we call these words "stop words" because they are useless, and we wish to do nothing with them. Another version of the term "stop words" can be more literal: Words we stop on.

Consider for example the following splitting after the tokenizing of a sentence: ['This', 'is', 'a', 'sample', 'sentence', ',', 'showing', 'off', 'the', 'stop', 'words', 'filtration', '.'] The output after stop word removal would be ['This', 'sample', 'sentence', ',', 'showing', 'stop', 'words', 'filtration', '.']



3.3 Stemming

Stemming is basically a normalizing method. Many a times there are variations of words that carry the same meaning, other than the tense involved. The reason why we stem is to shorten the lookup and normalize sentences. Example:

I was taking a ride in the car.

I was riding the car.

This sentence means the same thing. “in the car”, is the same. “I was”, is the same. the ‘ing’ denotes a clear past-tense in both cases, it is not truly necessary to differentiate between ride and riding, in the case of just trying to figure out the meaning of what this past-tense activity was.

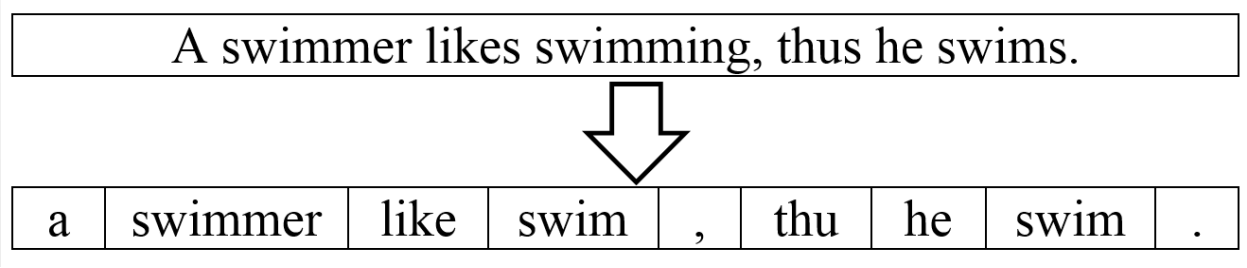


Fig. 3 Stemming example

3.3.1 Porter Stemmer

Porters stemming algorithm is as of now one of the most popular stemming methods proposed in 1980. Many modifications and enhancements have been done and suggested on the basic algorithm. It is based on the idea that the suffixes in the English language (approximately 1200) are mostly made up of a combination of smaller and simpler suffixes. It has five steps, and within each step, rules are applied until one of them passes the conditions. If a rule is accepted, the suffix is removed accordingly, and the next step is performed. The resultant stem at the end of the fifth step is returned. The rule looks like the following: → For example, a rule (m>0) EED → EE means "if the word has at least one vowel and consonant plus EED ending, change the ending to EE". So "agreed" becomes "agree" while "feed" remains unchanged. This algorithm has about 60 rules and is very easy to comprehend. Porter designed a detailed framework of stemming which is known as 'Snowball'. The main purpose of the framework is to allow programmers to develop their own stemmers for other character sets or languages. Currently there are implementations for many Romance, Germanic, Uralic and Scandinavian languages as well as English, Russian and Turkish languages. Based on the stemming errors, Paice reached to a conclusion that the Porter stemmer produces less error rate than the Lovins stemmer. However it was noted that Lovins stemmer is a heavier stemmer that produces a better data reduction [1]. The Lovins algorithm is noticeably bigger than the Porter algorithm, because of its very extensive endings list. But in one way that is used to advantage: it is faster. It has effectively traded space for time, and with its large suffix set it needs just two major steps to remove a suffix, compared with the five of the Porter algorithm.

Advantages	**Limitations**
Produces the best output as compared to other stemmers	The stems produced are not always real words
Less error rate	It has at least five steps and sixty rules and hence is time consuming

Compared to Lovins it's a light stemmer	
The Snowball stemmer framework designed by Porter is language independent approach to stemming.	

3.4 Tf-Idf approach

Tf-idf stands for term frequency-inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in information retrieval and text mining. The tf-idf value increases proportionally to the number of times a word appears in the document, but is offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general. So,

- If a word appears frequently in a document, it's important. Give the word a high score.
- But if a word appears in many documents, it's not a unique identifier. Give the word a low score.

Therefore, common words like "the" and "for", which appear in many documents, will be scaled down. Words that appear frequently in a single document will be scaled up. There are four functions involved in calculation of tf-idf score for the words in all emails in the corpus:

- tf(word, test) computes "term frequency" which is the number of times a word appears in an email test, normalized by dividing by the total number of words in test.
- n_containing(word, emaillist) returns the number of documents containing word.
- idf(word, emaillist) computes "inverse document frequency" which measures how common a word is among all documents in emaillist. The more common a word is, the lower its idf. We take the ratio of the total number of documents to the number of documents containing word, then take the log of that. Add 1 to the divisor to prevent division by zero.
- tfidf(word, test, emaillist) computes the TF-IDF score. It is simply the product of tf and idf calculated.

After calculation of tf-idf score for words in an email, we multiplied particular word frequency with its tf-idf score which acts as weight to the classifier model. Thus, it enhances the accuracy of results of classification by providing more weightage to the important words rather than useless words like 'the', 'is', etc.

3.5 Bigrams and Trigrams

Suppose we have a sentence with the words "mother dairy" which refers to the name of a company. If we consider unigrams "mother" and "dairy", then the algorithm will consider "mother" as a relationship and classify incorrectly. To solve this problem N-grams are used.

N-grams of text are basically a set of co-occurring words within a given mail. For *N*=2 it is known as bigrams. For ex. "The cow jumps over the moon" then the bigrams would be: "The

cow", "cow jumps", "jumps over", "over the", "the moon". For *N*=3 it is called a trigrams. For the above example, the trigrams would be: "The cow jumps", "cow jumps over", "jumps over the", "over the moon".

Considering bigrams and trigrams tends to increase the performance of the classifier in classification of emails.

3.6 Additional Features

In addition to the above mentioned features we have also maintained a count of urls, email ids and numbers as new feature sets. This is because the probability of occurrence of urls and email ids in personal category of emails is less as compared to other categories.

4 Naive Bayes

The Naive Bayes algorithm is an intuitive method that uses the probabilities of each attribute belonging to each class to make a prediction. It is the supervised learning approach you would come up with if you wanted to model a predictive modeling problem probabilistically.

In the real world scenario, email classification systems should handle the problem of incomplete training set and no prior knowledge of feature space. Hence, Naïve Bayes proves to be better approach since it becomes easy to incrementally update its pre-learned classification model and feature space through a feature weighting strategy.

The probability of a class value given a value of an attribute is called the conditional probability. By multiplying the conditional probabilities together for each attribute for a given class value, we have a probability of a data instance belonging to that class. To make a prediction we can calculate probabilities of the instance belonging to each class and select the class value with the highest probability.

4.1 General Algorithm

1. Firstly, get the probability for a text to belong to each of the categories that is test against. The category with the highest probability for the given text wins:

$$classify(word_1, word_2 \ldots word_n) = \underset{cat}{argmax} \log_e(P(cat)) + \sum_{i=1}^{n} \log_e(P(word_i|cat))$$

2. Do note that above we also eliminated the *denominator* from our original formula, because it is a constant that we do not need (called *evidence*)
3. Instead of the probabilities of each word, store the (natural) logarithms of those probabilities so that instead of multiplying the numbers, we add them instead

4.2 Previous Research

In a research paper titled, "Identifying context of text documents using Naïve Bayes Classification and Apriori Association Rule Mining" by A. R. Kulkarni, V. Tolekar dated 5 September, 2012, classification of abstracts was done considering their contexts using Naïve Bayes classifier and Apriori association rule mining. Here, the context of an abstract is found by looking for associated terms that help focus on abstract and interpret the information beyond simple keywords.

In a research paper titled, "Naïve Bayes Classifier based Arabic document Classification" by Noaman, Elmougy, Ghoneim, Hamza dated March 2010 have used the Naïve Bayes approach to assign online Arabic documents, the information libraries and Arabic document corpus to

various categories. They were able to use rooting algorithms in combination with Naïve Bayes and obtained an average accuracy of ~62.23%.

Hence, on similar lines we can apply this algorithm to generate categories dynamically and assign emails to them accordingly.

4.3 Multinomial Naive Bayes Model

The Naïve Bayes classification technique is based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as 'Naive'.

Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

Bayes theorem provides a way of calculating posterior probability $P(c \mid x)$ from $P(c)$, $P(x)$ and $P(x \mid c)$. Look at the equation below:

$P(c \mid x)$ is the posterior probability of *class* (*c*, *target*) i.e. category for given test email (*x*, *attributes*).
$P(c)$ is the prior probability of *class i.e. category.*
$P(x \mid c)$ is the likelihood which is the probability of *test email* given *class.*
$P(x)$ is the prior probability of test email.

$$P(c \mid x) = \frac{P(x \mid c)P(c)}{P(x)} \tag{1}$$

$$P(c \mid x) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c) \tag{2}$$

Constructing classifier by (1) and (2):

$$\hat{y} = \underset{k \in \{1, \ldots, K\}}{\arg\max} p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k) \tag{3}$$

Here '*y*' is the assigned class value to the test mail depending on the class giving highest probability for the test mail. '*n*' indicates the no of features of email considered for building feature set for email dataset.

Multinomial model is one of the models generated from Naïve Bayes algorithm. With a multinomial event model, samples (feature vectors) represent the frequencies with which certain events have been generated by a multinomial where is the probability that event occurs (or such multinomial in the multiclass case). A feature vector is then a histogram, with counting the number of times event was observed in a particular instance. This is the event model

typically used for document classification, with events representing the occurrence of a word in a single mail. The likelihood of observing a histogram is given by,

$$p(x|C_k) = \frac{(\sum_i x_i)!}{\prod_i x_i!} \prod_i p_{ki}^{x_i} \tag{4}$$

Here,

$$p_{ki} = \frac{count(w_k, C_i) + 1}{count(w, C_i) + |V|} \tag{5}$$

where, $count(w_k, C_i)$ *is the count of occurrence of word* w_k *in category* C_i *and word* $w \in V$ *(Vocabulary list).*

It is often desirable to incorporate a small-sample correction, called pseudocount, in all probability estimates such that no probability is ever set to be exactly zero. This way of regularizing Naive Bayes is called Laplace smoothing when the pseudocount is one, and Lidstone smoothing in the general case. Thus for smoothing (5), 1 and size of vocabulary list $|V|$ is added to the numerator and denominator respectively.

Although Laplace method is simple as it assumes there is at least one sample for each word-class pair but this addition to every word is not considered as the effective smoothing method. However, this problem has been widely studied in language models.

All the smoothing methods introduced in language models differ in likelihood estimation of probability $p(w_k \mid C_i)$. In this paper we also used an approach of absolute discount for smoothing. In Absolute Discount (AD) method, a small amount "*delta*" is subtracted from every positive word count for seen words in order to redistribute the discounted probability mass on unseen words.

In Absolute discount, $p(w_k \mid C_i)$ is calculated as:

$$p(w_k|C_i) = \left(\frac{\max(count(w_k, C_i) - delta, 0) + delta(Nuc_i) P(w_k)}{\sum_{w \in V} count(w, C_i)} \right) \tag{6}$$

where, $delta \in [0,1]$

Nuc_i is the number of unique words in Class C_i.

$P(w_k)$ is the probability of word w_k in collection model and is given by :

$$P(w_k) = \frac{\sum_{j=1}^{m} count(w_k, C_j)}{\sum_k \sum_j count(w_k, C_j)} \tag{7}$$

4.3.1 Results

We have considered a dataset of ~1400 emails and have classified them in 3 categories. We have also analyzed the working of Multinomial Naïve Bayes using Laplace Smoothing as stated in (3),(4) and (5) for avoiding zero probabilities which results in bad accuracy. Varying the ratio of emails considered for building training set so as to classify the test set data, we have built a feature set which is composed words with a frequency greater than 5 in the whole dataset after removing useless words from the set. Bigrams, trigrams and some additional features are also

considered for classification. The graph of the result with MNB-Laplace Smoothing is shown in Fig. 5.

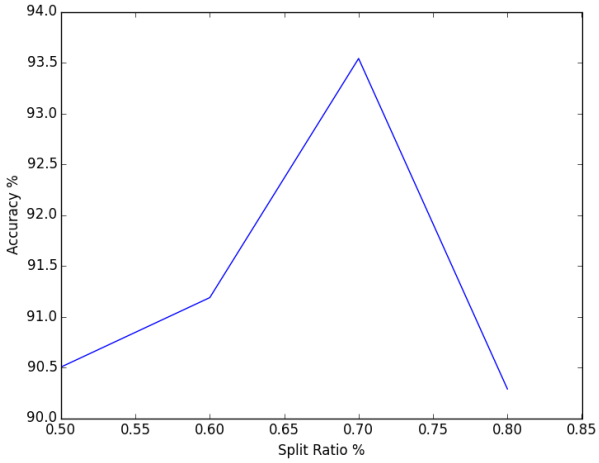


Fig. 5. The graph shows that 0.7 split ratio is the cut-off for best accuracy (approx. 93.5%) result with Naïve Bayes and after that point graph fluctuates.

We have also studied the working of Absolute Discount (AD) method for providing better smoothing to the Multinomial Naïve Bayes Model as stated in (6) and (7). For this smoothing technique only single words are considered as features because it totally depends on distribution of single words among all emails. The graph of the result with MNB-AD is shown in Fig 6.

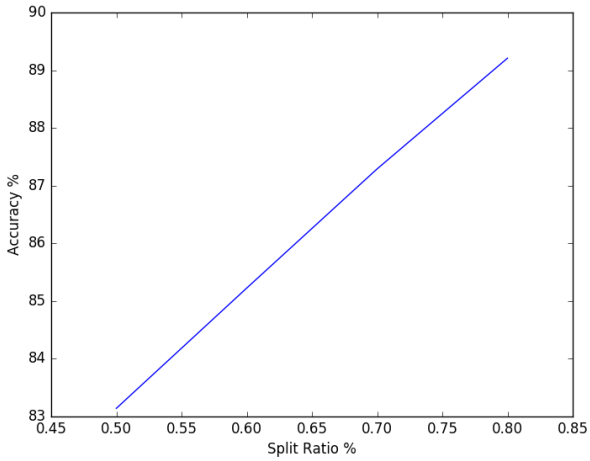


Fig. 6. The graph shows an increasing performance with increasing split ratio indicating the bigger is the size of training set, the better is the accuracy.

4.3.2 Analysis

4.3.2.1 Time Complexity

- Training Time: O(|D|*L + |C|*|V|)) where L is the average length of an email in D.

- Assumes all counts are pre-computed in O(|D|*L) time during one pass through all of the data.
- Generally, just O(|D|*L) since usually |C||V| < |D|L.
- Test Time: O(|C| L_t) where L_t is the average length of a test email.

4.3.2.2 Pros

- It is easy and fast to predict class of test email data set. It takes approx. 10 seconds for 1500 mails.
- Taking assumption of independence between features of mail, a Naive Bayes classifier performs better compare to other models like logistic regression and we need less training data. Accuracy is good even if training set is small compared to train set as we can see in the above graph for 0.1 split ratio accuracy comes to be around 75%.

4.3.2.3 Cons

- It perform well in case of categorical input variables compared to numerical variable(s). But features considered in emails are mostly numerical variables so category advantage is not present for email classification.
- Limitation of Naive Bayes is the assumption of independent predictors. But in reality, it is almost impossible that we get a set of features in email which are completely independent as a pair of words can help in prediction of class compared to a single word as a feature.