Content-aware Email Multiclass Classification
Categorize Emails According to Senders

Liwei Wang, Li Du

s

Abstract

People nowadays are overwhelmed by tons of coming emails everyday at work or in their daily life. The large quantities of emails keep causing confusions. Not only spam emails are considered to be 'junk', but also unwanted emails (e.g. advertisements) cause people to waste time on reading them. Therefore, it becomes urgent to develop reliable automatic categorization of emails to save the trouble. This project aims at researching on ways doing supervised and unsupervised classification of emails according to email content, in particular, putting emails into folders in terms of role of email senders.

1. Introduction

Email classification falls into the text categorization reign of machine learning. Example like spam email filtering has been the one of the most popular topics in text categorization over the years. There are several problems to be recognized and tackled in the process of coming up the solution:

- Signature of emails

For text categorization, the most common signature (or representation) of emails can be words (Dumais et al., 1998), sequences of words (Caropreso et al., 2001), part-of-speech tags (Finn et al., 2002), word clusters (Bekkerman et al., 2003). [2]

- Feature selection

By choosing words, possibly counts of words on the token list in the emails, i.e. bag of words, as feature vector, it's readily seen that dimension of feature vector and what features to choose affect classification in one way or another. An unacceptably high dimension of feature vector may make classifier, e.g. SVM less reliable when there are not enough training examples. On the other hand, features play little role on distinguishing emails from one class to another should be discarded.

- Classifier selection

There are several existing supervised and unsupervised classifiers out there to be selected. The problem is to choose the one fits our problem best and the one gives the best performance.

2. Design Methodology

2.1 Dataset

We use a large corpus of real world email messages from Enron employees. [1] The Enron corpus was made public during the legal investigation concerning the Enron Corporation. This dataset, along with a thorough explanation of its origin, is available at http://www-2.cs.cmu.edu/~enron/. In the raw Enron corpus, there are a total of 619,446 messages belonging to 158 users. Since email classification is highly dependent on identity of individuals, different people from different backgrounds may have different folders for emails. Therefore, we choose emails from employees in the same company, so composition of their emails is similar. Among all the messages in this dataset, we randomly picked over 2500 emails as our own dataset.

2.2 Workflow Overview

The following is a general workflow for the implementation of classifiers from raw data to final results. We only show the supervised classification aspect. Unsupervised part can be viewed as an extension, which will be discussed in the following sections.

1) Labeling:

We put all the messages into 5 categories:

- Friends and families (18%)
- Business, e.g. announcements, information from own company (18%)
- Colleagues, e.g. co-workers at same company or alliance partners (36%)
- Advertisements publishers (8%)

- Organizations, e.g. organizations other than own company. This may include other organizations the subject participates or activities they are involved in. (20%)

2) Stemming:

We used an open source python code to do the stemming. The stemmer stemmed the words in the email documents so that words with the same meaning will have the same form as our data.

3) Get words statistics and token list:

The data examples are then feed into a program extracting statistics of the dataset. The result of this step gives a sorted list of all tokens appeared in the dataset and along with their count (i.e. number of times of presence in all the email documents). In summary, there are 2740 emails in all, and 20000+ tokens.

4) Modify token list:

Based on the preliminary token list we get from the last step, we can have a general knowledge of the construction of the list. There is a whole large number of tokens having only a small number of count, so that we can discard them as they rarely appear in other email documents. On the other hand, there are also some tokens having too many counts, which are not informative in terms of the classification task. Eventually, the range of count we define is from 200 to 1800. We also added 4 other special token: NUMBER, DOLLAR, HTTP and EMAIL. As a result the token list is reduced to 588+4 = 592 long.

5) Convert examples into matrix:

The main task of this step is to turn each example into a feature vector and then put them together to form an entire matrix. The following is the list of things done in order:

- In a large loop, read the email one by one, for each email:
- First, store its label. Then for each other word in this email, take down the number of its appearance, and store the result accordingly in its feature vector (the feature vector is a vector of all tokens, the value of each entry of the vector is the count of the token corresponding to that entry)
- Special words are replaced by tokens NUMBER, DOLLAR, HTTP and EMAIL.
- Put all the vectors into one matrix and rearrange them into sparsed form.
- Shuffle the examples (put them into random order)
- Divide the examples into 10 groups. Every time, put 9 of them into training examples, and the remaining one as testing examples. As a result, we have 10 training – testing pairs.
- Write the training and testing examples into files

6) Classification:

We did the classification using Naïve Bayes and SVM, to compare the result of the two methods. To train and test one class as positive examples (labeled as 1), we treat all other classes as negative examples (labeled as 0 for NB or – 1 for SVM) and do this for every class.

2.3 Supervised Classification using Na ïve Bayes and SVM

Na ïve Bayes and Support Vector Machine are applied to this project implementation. On the one hand, Na ïve Bayes are quite common for email classification, which can be considered as baseline. Support vector machine, on the other hand, proves to have good performance on text categorization.

- Na ïve Bayes

Given m examples, and each example is represented by a vector $x^{(i)} \in \mathbb{R}^n$, to simplify just denote it as x. Assuming elements in the feature vector are independent. We can build our model by solving ML function to get parameters:

$$\mathcal{L}(\phi_y, \phi_{i|y=0}, \phi_{i|y=1}) = \prod_{i=1}^{m} p(x^{(i)}, y^{(i)}).$$

Here we can use Bayes rule to decompose the right hand side of the equation. The parameters allow us to make predictions on new data. We also used Laplacian smoothing to make sure that we do not encounter 0 occurrence of counts.

- Support vector machine

Support vector machine is achieved by building a hyper plane dividing positive and negative classes. The optimization problem is:

$$\begin{aligned}\min_{\gamma,w,b} \quad & \frac{1}{2}||w||^2 \\ \text{s.t.} \quad & y^{(i)}(w^T x^{(i)} + b) \geq 1, \quad i = 1, \ldots, m\end{aligned}$$

where *w* and and *b* are the parameters for the model. A revised version of the optimization problem takes into account data points that are allowed to be misclassified (non-separable cases) by adding another parameter *C* controlling the weighting of such 'error' allowed. The kernel we chose for this project is a linear kernel, because it makes no sense using higher dimension feature vector when the dimension is already very large for SVM and a linear kernel serves right for this project.

2.4 Unsupervised classification using hierarchical clustering

After getting the result from supervised classification, we found that there is an inevitable problem: labeling can be misleading. Since we do not know the ground truth of label of the emails data we have, the label are done manually, which gives rise to a highly unreliable label of the examples because on the one hand, it's difficult to stick to some criteria in the judgment on what the label of every document is (especially there is vague or subtle difference between one class from another for some cases); on the other hand, it's hard to see the intrinsic characteristic of some documents by just reading the email. Therefore, it will be helpful to come up with some way that allows the emails to automatically cluster themselves first and then, by assigning each cluster a label, we can do supervised learning on them. In particular, for application we do not know the label (this is true for most cases), we can use unsupervised learning to categorize the documents. And hopefully the result of clustering can give some insights on the characteristic of the documents and the label for it.

The algorithms for unsupervised classification available includes k-means, EM algorithm (which is 'softer' then k-means because it depends on some probability model in making estimation), factor analysis and etc. To implement this algorithm, a high dimension feature vector makes it extremely inefficient or unreliable. Therefore, to solve this problem, we either need to decrease the dimension of feature vector, or choose some algorithm that fits well for high dimension feature vector and text categorization. We first tried PCA to decrease the dimension and implement mixture of Gaussian, only to find that mixture of Gaussian fails to fit our problem because the number of parameter scales quadratic with length of feature vector (think about the covariance matrix).

Another approach to take is to use hierarchical clustering, and specifically, agglomerative clustering. This is a "bottom up" approach: each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy. [3] The metric can be used as closeness measurement between pair of examples includes Euclidean distance (l2) and squared ED, Manhattan distance, cosine similarity and etc. On the other hand, the linkage criterion specifying dissimilarity of sets includes maximum or complete linkage clustering, minimum or single-linkage clustering, Minimum energy clustering. We can try each of these metric and criteria and select the one resulting in best performance. (A review of cluster analysis in health psychology research found that the most common distance measure in published studies in that research area is the Euclidean distance or the squared Euclidean distance.) [3]

The implementation is quite straightforward, we only need to construct a matrix keeping distance of each pair in the examples and do the merging stage by stage and finally get a hierarchy of the clustering.

3. Results and Evaluation

Here we're giving the final results of our implementation of the proposed algorithm. By comparing different methods and different feature selection, hopefully one can have a general view on how to choose appropriate method and feature vector.

3.1 SVM vs. Naïve Bayes

As we are using 10-fold cross validation, the evaluation metric shows the average of 10 sets. We give the evaluation results for SVM and Naïve Bayes approaches respectively. For each approach, we give accuracy table and precision-recall curve (Table 1 and Figure 1).

Method / Label	SVM accuracy	NB accuracy
1	0.952	0.941
2	0.871	0.883
3	0.964	0.953

5	0.893	0.868
7	0.87	0.824

Table 1: Accuracy of SVM and NB

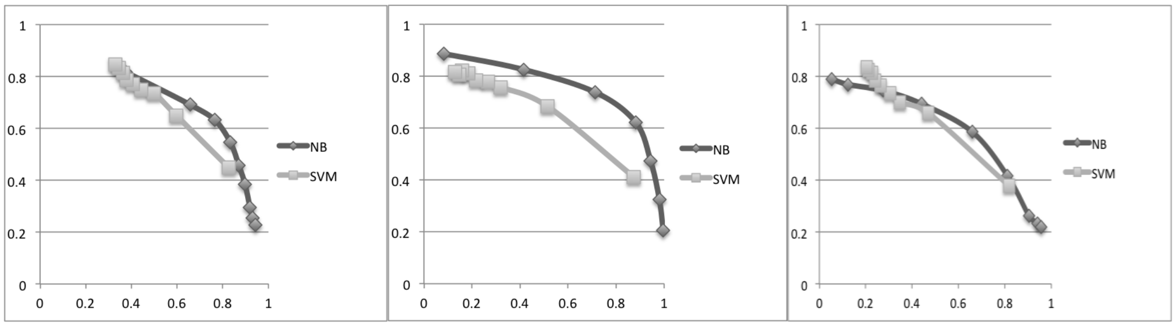

Figure1: Precision/recall curve for class5, 1 and 2

The accuracy table shows that SVM performs better than NB in terms of accuracy. But generally speaking, the accuracy is not high as expected. By reading precision/recall curve, SVM actually has smaller area-under-curve than Na ïve Bayes. Particularly, the curve of SVM is somehow parallel to diagonal of the graph which makes it far from a good classifier.

3.2 Hierarchical clustering

Among the hierarchy of clusters we generate, we are giving the fraction of examples with same label in each cluster (Table 2). We also tried SVM on the clustered data (examples in the same cluster have the same label). Hopefully the results (Figure 2) may give us some insight on the strange behavior of the SVM precision/recall curve shown above.

For best performance, several methods and metrics are tried:

Method (dissimilarity criteria)	Unweighted average distance (UPGMA)	Centroid distance (UPGMC)	Furthest distance	Shortest distance	Inner squared distance
Metric (distance)	Euclidean (l2)	Cosine	Correlation	Hamming	Spearman

Results show that furthest distance --- cosine/correlation/spearman combination gives best performance.

Label / Cluster / Examples #.	1	2	3	5	7	Max rate
155	0	0	154	1		99.35%
326	29	58	15	205	17	62.88%
351	3	206	4	87	51	58.68%
886	100	142	4	136	504	56.88%
653	340	101	7	65	140	52.06%
289	6	32	0	13	238	82.35%

Table 2: Fraction of same label examples in each cluster

It's readily seen that label 3 is almost clustered into one cluster with percentage of over 99%. Other label documents are clustered into more than one cluster. Label 7 documents are the majority in two clusters. This makes sense since we have more label 7 documents in the datasets.

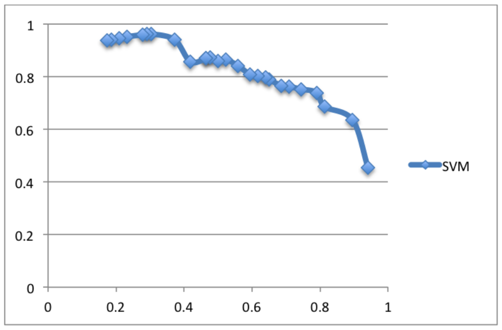

Figure 2: SVM on clustered data

This precision/recall curve looks better. It is highly probable that labeling the documents by hand is not reliable. Unsupervised learning can capture the nature of the documents and therefore gives clearer division between documents cluster. Another reason may be the representation method is not proper. Bag-of-words representation may not be the best fit for our project problem.

4. Future work

As the evaluation results show, there is still large margin for us to improve performance of the algorithms. This can be done in the following aspect:

- Choose different document representation. Other than bag-of-words representation, there are other ways representing our examples. One of the drawbacks of bag-of-words is that it ignores the semantic relationship between words in the sentence. Using sequences of words allows one to explore the underneath semantic relationship. This may require some natural language analysis. On the other hand, we can always weight different part of the document, e.g. gives higher weight to subject or attachment.
- Try other classification approaches. We've only tried SVM and Naïve Bayes while there're many other algorithms proved to be feasible for document categorization. For example, maximum entropy (which models the class-conditional distribution with the most uniform one that satisfies constraints given by the training set. [2]), wide-margin winnow which is an on-line learning algorithm and random forest which is a decision tree approach.
- Build more balanced datasets. The number of each class in the datasets is not balanced resulting in less reliable training model as well as evaluation results. If the testing set examples are not balanced, just by looking at accuracy may be misleading. To tackle this problem, we can either weight each class according to the documents number in each class (and it's not necessarily linear), or make copies of documents which belong to the minority class.

5. Conclusion

In summary, by comparing different supervised and unsupervised learning methods in text classification problems, we find that SVM performs better than Na ïve Bayes in terms of accuracy but actually has smaller area-under-curve. Possible reasons include: bad document representation choice, unreliable label by hand and improper classification approaches. To find the reason for that, unsupervised learning methods are taken and hierarchical clustering is selected. As is shown by the results, using hierarchical clustering to do the labeling automatically according to the interior correlation properties of the feature vectors tells that our subjective labeling is one of the factors that area-under-curve is small using SVM, and the classifications by using these new labels are better. The accuracy also could be improved by developing new feature vectors, classifiers and more balanced dataset.

Reference

[1] Bryan Klimt, Yiming Yang. *Introducing the Enron Corpus*.
[2] Ron Bekkerman. *Automatic Categorization of Email into Folders: Benchmark Experiments on Enron and SRI Corpora*. 2004.
[3] Wikipedia. *Hierarchical clustering*. http://en.wikipedia.org/wiki/Hierarchical_clustering