AOEC: A Tool for Automated Online Email Categorization

Zhe Yu, Di Chen
Amritanshu Agrawal, Shijie Li
Com Sci, NC State, USA
{zyu9, dchen20, aagrawa8, sli41}@ncsu.edu

1. INTRODUCTION

More and more people are using email as a daily communication tool for their work and daily lives. Its growing importance has inspired many attempts to develop intelligent tools for classifying, organizing, and presenting email messages (e.g., Bellotti etal., 2003 [2]; Murakoshi et al., 1999 [8]; Sproat et al., 1998 [11]). Among these tools, Email automatic categorization has became a basic and important part for email applications. That is because, for most people, there will be hundreds of emails flowing into the mailboxes everyday, while people usually don't have the patience to read such a large amount of emails one by one. So automatic categorizations and classifications for email are crucial to improve the efficiency of dealing with the flow of emails.

1.1 Problems

There are two main problems in existing email applications. First of all, for most popular Email applications like Outlook, Gmail and Yahoo Mail, they only automatically categorize emails into several default folders, such as Promotions, Updates, Social, Forums and so on. While users do not have the option to create their own folders that have the automatic categorization feature. It is true that these default folders have covered the need for common email users, but it is far away from "enough" for heavy email users who have to categorize their emails in a more detailed way.

Another problem we notice is that the existing automatic categorization systems only ask for users' feedback in a quite passive way, which means they seldom ask feedback from the user for their auto-categorization result, or they put their feedback buttons in the second or third level of the user interface. Even worse, some email applications do not have a feedback mechanism at all for their automatic categorization.

1.2 User

Based on the problems above, our target users are those who gave to deal with emails everyday, who need more flexibility to manage their email folders, and who also need their emails be automatically categorized into the folders by their own.

1.3 Tool

Email client receiving real-time email streams, and monitor user behaviors to provide non-intrusive notification asking for user feedback.

Besides the basic structure of Gmail, we will add the following features: a) user can create new folders with any arbitrary name; b) incoming emails will be automatically classified into each folders; c) one emails may appear in multiple folders d) a folder named "Confusing" will store the most confusing emails that the classifiers are confused about and wait for the user's judgement. e) implicit feedback: if user opened one email under a folder without moving it to other folders, we will treat this as an implicit feedback.

1.4 Goal

For this project, our goal is to offer users a more flexible, active and automatic email categorization. To achieve this, a new categorization logic is needed. Like other email applications, we will first classify emails into several default folders, which is based on massive email training set. Then, we will actively ask the user for feedback in a non-disturbing way. User can choose the right folder to put the new incoming mail or create new folder if they like. Finally, after collecting the new categorization record, we will re-train our classifiers to adapt the new user-specific features.

All the materials for this project can be found at our Github repository[1].

2. CHALLENGES

Literature and several user experiences are studied to identify the challenges we may face to achieve our goal. Start with one seed paper [1], each member of the team reviewed three papers, either highly cited or most recent. In addition to literature review, several users are interviewed to express their difficulties using existing tools. The following problems are summarized to provide a guideline for the next step of our project.

2.1 Lack of Training Examples

2.1.1 Challenge

Most of the text mining classifiers requires thousands of training examples to build a reliable model. However, more than half the users we interviewed expressed their unwillingness to provide such a great amount of training examples. The number of training examples acceptable is dozens for each folder and hundreds in total according to our interview. This leads to the lack of training examples in the initial step. For a text categorization task, the dimensionality for feature is usually thousands, which is a lot higher than the available training examples. The classifier trained on such a feature matrix will have a huge bias.

[1]https://github.com/azhe825/CSC510.git

2.1.2 Solution

There are three methods we proposed to solve the problem:

a) try different types of classifiers to see if anyone works better in this scenario;

b) apply LDA first to get a topic model before training, so that the dimension of features can be reduced to less than the number of training examples;

c) implement incremental learning, we may have bad performance at beginning but will keep learning until saturation.

The result of these three solutions are presented in Section 3.3.

2.2 Compensation to Explicit Feedback

2.2.1 Challenge

In most of the email categorization tools, the implicit feedback events are underutilized. When the classifier is not confident, positive feedback confirming the correctness of predictions would be helpful. It is a good compensation to explicit feedback. Users do not like to be asked to provide corrections frequently, especially if they are working quickly or are deeply engaged in a task. Under such conditions, self training is risky [10]. This was the motivation to explore *implicit feedback*. The implicit events/interactions are defined as follows:

a) user add or remove a tag on the message;

b) user add or remove a flag from the message;

c) user move the message to a folder;

d) user copy, reply, forward, or print a message;

e) user save an attachment from the message.

2.2.2 Solution

The study shows most of the bad labels are corrected almost immediately within a few interactions [10]. One way to use these events for implicit feedback events is to keep a threshold on these events. When it exceeds a total number of events, the labels are assumed to be correctly predicted and the classifier trains on that message. We can set less threshold for good confident labels and more threshold for bad confident labels. This made us to explore [10] and found 4 methods to tackle implicit feedback events:

a) No Implicit Feedback (NoIF): This creates a training example for a label if the user adds or removes that label from the email message. This doesn't change the confidence level of the other labels.

b) Simple Implicit Feedback (SIF): When the user changes any label, immediately treats all remaining labels as correct and creates implicit feedback training examples. This changes the confidence level of the other labels.

c) Implicit Feedback without SIF (IFwoSIF): This maintains a count of the total number of implicit feedback events. When this count exceeds a specified threshold, then it creates the implicit feedback training examples.

d) Implicit Feedback with SIF (IFwSIF): If the user changes a label, then implicit feedback examples are immediately created. Otherwise, continue to count up implicit events to reach a specified threshold.

With the addition of implicit feedback on top of explicit feedback, the system is more adaptive and produce less errors as time passes by.

2.3 Importance of Folders may Vary

2.3.1 Challenge

Users definitely do not want to miss a single important email. Some of the folders can be of high importance that False Negative costs much more than False Positive. This phenomenon is commonly described in the binary classification case of email categorization– spam filtering [5]. Failure to identify a spam is always less important to failure to identify non-spam.

2.3.2 Solution

One possible solution for this problem would be to allow each incoming email belongs to several folders. This will change the problem to multi-label, multi-classification problem.

In addition, different weight can be addressed on the labels. User can be asked to put an importance rank when creating folders. For the folders with high rank, the threshold can be reduced to allow emails to go into that folder more easily.

2.4 Not Every Folder is Content-aware

2.4.1 Challenge

There are three types of email folders– content-aware, time-aware, and participants-aware– while most of the existing methods can only correctly classify emails with content-aware folders [6].

Content-aware: Folders contain emails of the same topic, e.g. "sports", "music".

Time-aware: Emails in this type of folders are categorized regarding the time they are received, e.g. "2012 Summer".

Participants-aware: This type of folders contain emails with the sender or recipients from a particular group of users, e.g. "Supervisor", "PhD Council".

2.4.2 Solution

Suggested by [6], three separate models can be built. Each one of the models focuses on one type of folders by constructing the feature set and classifiers specifically according to the characteristic of the target folder type. In the prediction step, the predicted probability of the three models will be used to make a decision fusion and thus decide which folder the incoming email belongs to.

In addition, users will be asked to select the type when they are creating new folders. This simple action of users can greatly facilitate the system in training.

2.5 Non-text Content can be Useful

2.5.1 Challenge

The expressions of email nowadays are not just confined to the text content. Instead, with the convenience of GUI and embedded HTML as well as the support of MIME (multipurpose Internet mail extension), lots of emails can carry graphical attachments, linkages to on-line information and non-text characters [4]. For example, many people use email to share and discuss photos taken together with their friends or families. And they also suggest interesting on-line videos to friends by providing website linkages. Also, people today prefer to use non-text characters such as Emoji to convey

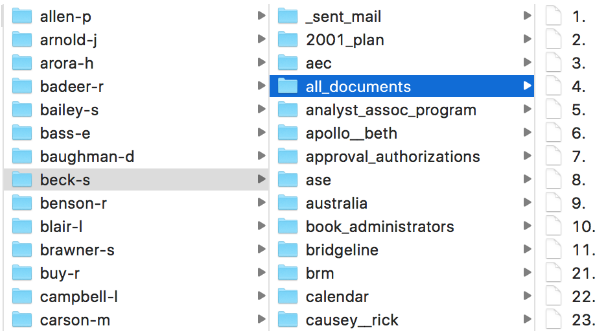
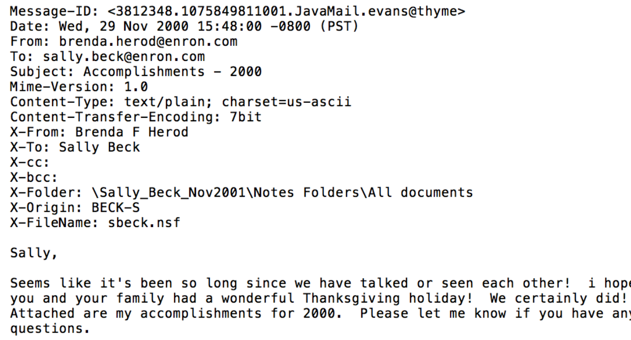


Figure 1: Structure of Enron Data Set



```
Message-ID: <3812348.1075849811001.JavaMail.evans@thyme>
Date: Wed, 29 Nov 2000 15:48:00 -0800 (PST)
From: brenda.herod@enron.com
To: sally.beck@enron.com
Subject: Accomplishments - 2000
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
X-From: Brenda F Herod
X-To: Sally Beck
X-cc:
X-bcc:
X-Folder: \Sally_Beck_Nov2001\Notes Folders\All documents
X-Origin: BECK-S
X-FileName: sbeck.nsf

Sally,

Seems like it's been so long since we have talked or seen each other!  i hope
you and your family had a wonderful Thanksgiving holiday!  We certainly did!
Attached are my accomplishments for 2000.  Please let me know if you have any
questions.
```

Figure 2: Email Content

emotions and ideas. In these situations, even the names of the attached files and domains of website links might be more useful than the whole text content to determine the email categorizations. And considering these files or websites are probably viewed by thousands of people who share the same interesting, they are more precise to describe the themes of the parent emails. In addition, some emails even contain only the non-text contents. Instead, they embed HTML codes inside email to present the information neatly.

2.5.2 *Solution*

A thorough analysis of the non-text content in the emails is not practical due to the computational speed requirements. Instead, we will extract partial information in those non-text parts and convert them into text labels. For example, instead of analyzing the attached graphs using computer visions, we only extract the text properties such as file names, authors, create dates, graph formats and so on.

3. EXPERIMENT

We divide our experiment into two phases:

a) In February, data is collected and preprocessed. Then we focus on the primary problem, lack of training examples, and try all three solutions. Statistical results are collected and compared to determine the best solution for this problem.

b) In March, we build a GUI to implement our best solution from February. Then we define our telemetry and repeatedly test on our GUI. Find small problems that matters most from user experience and build smallest solutions for them. We keep adding features and end up with a satisfiable product.

As a result, we solve the first problem described in Section 2.1 by the end of February and the second and third problem described in Section 2.2, 2.3 by the end of March. The last two problem described in Section 2.4, 2.5 remain unsolved because they are of least importance according to user experience.

3.1 Data Collection

The Enron Corpus is a large database of over 600,000 emails generated by 158 employees [7] of the Enron Corporation and acquired by the Federal Energy Regulatory Commission during its investigation after the company's collapse. It is sized to about 400Mb, tarred and gzipped. The May 7, 2015 Version of Enron Email Data will be used as our data set[2]. The dataset here does not include attachments. The Enron dataset which we will be using is preprocessed and provided after the classification of these emails into few categories [1]. The structure of the mail looks into various folders according to the user's name and each of these folders are further classified into folders of categories as shown in Figure 1. An example of the content of one email is presented in Figure 2.

We extract data from seven different users for the experiment. Each user is treated as an independent data set. Within each data set, folder names are used as the true label of emails inside. Within each folder, ten randomly picked emails are treated as initial training examples, all the other emails are treated as test examples with labels unknown.

All the emails was stored in the text file of the MIME standard of email format. To grab the email subject and body texts, we use the 'email' python package to separate the email subject and body from other components. This package could accurately parse the email of standard format into text segmentation. And then we apply the regular expressions to filter out digits and non-character tokens and also remove common words to reduce noise and computational consumptions. Finally, we save the processed email subject and body as list of words for each email document, and convert those words into a numeric word vector as features for later categorization using techniques such as tf-idf (term frequency - inverse document frequency) and Word2Vec.

3.2 Performance Metrics

Accuracy: accuracy is the most commonly applied performance metric when comparing multi-classification results. Its problem is that the performance on minority classes are barely reflected in accuracy [9].

F_M**:** F-score on each class can be calculated after the trained classifier is tested on test set. The mean of F-score on each class is then calculated to represent the overall performance of the classifier. Regardless of the population within each class, the F_M represents performance on each class equally [9].

We are using F_M in the comparison of performance.

3.3 Three Solutions for Lack of Training Examples

[2]https://www.cs.cmu.edu/ ./enron/

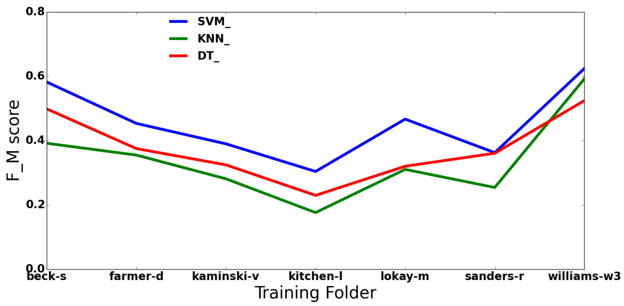


Figure 3: Performance among different users.

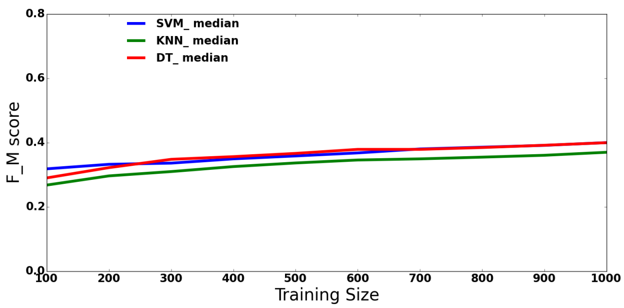


Figure 4: Performance for different classifiers when increasing the training dataset.

3.3.1 Supervised Learning

For supervised learning, we assume that we have gotten some emails categorized by uses, and predict the new coming emails based on the categorized one. In this solution, we totally test 3 classifiers incrementally among 7 different users.

Specifically, we first selected 7 users from Enron Email dataset that have the most sub-folders. Then, we removed the sub-folders that contain less than 10 emails. After that, we only select 10 emails as our training dataset for each sub-folder per user, so that we could simulate the real-life scenery of predicting email folders based on limited categorized emails. Finally, we use three different classifiers to train and predict the emails.

The results are shown in Figure below. As you can see in Figure 8, the prediction performance varies among different users. It is because some users may label them emails more generally, while the others more precisely. Also, the content of the email sub-folders also matters. In spite of that, we can know that SVM performs better than DT and KNN. For Figure 6, we can learn that, averagely, the more training data we have, the better our prediction will be. But, generally speaking, the improvements is not as big as we have expected.

3.3.2 Unsupervised Learning

Unsupervised learning is the machine learning task of inferring a function to describe hidden structure from unlabeled data. There are various tools of doing unsupervised learning, and we used Latent Dirichlet allocation [3]. We used LDA because of its useful importance and advantages for finding useful topics from an unlabeled data. LDA features were extracted and given as input to SVM learner. LDA is used as a feature extractor. We have already shown in Figure 4 that SVM performs better than the other classifiers.

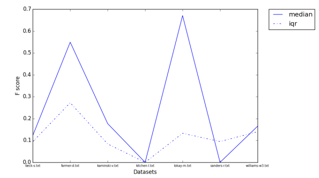


Figure 5: Performance of LDA with 12 clusters

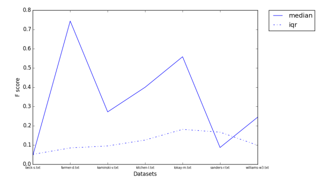


Figure 6: Performance of LDA with 50 clusters

The Famous Enron email dataset is used and have already specified about the pre-processing steps and its final structure above. We tested the data on 7 different users. There are different labels already defined for the mails. We selected 5 top labels for each user dataset, and converted it into a binary classifier where those 5 labels are given ”YES” label and others as ”NO” label. And we also selected different ranges of cluster size namely 20, 50 and 100. Respective results can be seen for all the different cluster size given below.

Following results show highly instability of F-Score across different datasets. The reason for that is, if we have large data for positive class then it gets you good results but if number of negative labels dominant over positive labels, it performs poorly. The other problem which we saw, the instability of these clusters even if there was no changes in the data or the parameters of the algorithm. These results made us to not use unsupervised features to train a classifier.

3.3.3 Incremental Learning

3.4 GUI Development

The GUI is developed using Python Tkinter package. We took Gmail as our sample for making the Mailbox GUI. See Figure 9 for our mailbox interface.

We have provided users with read mails, move mails to different folders, creation of user defined folders. We have also provided the user with different features to select on demand. These features are Implicit User Feedback, Explicit User Feedback, Multi-folders. The GUI is designed to not only load local email files for training classifier, but also read email from the online email account for different users. Under the hood, it will connect the server at preset frequency, fetch emails from the online account, categorize them using local classifier and refresh the emails saved on local folders. Currently, it support Gmail accounts only for security settings.

3.5 Implicit User Feedback

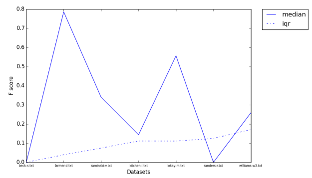


Figure 7: Performance of LDA with 100 clusters

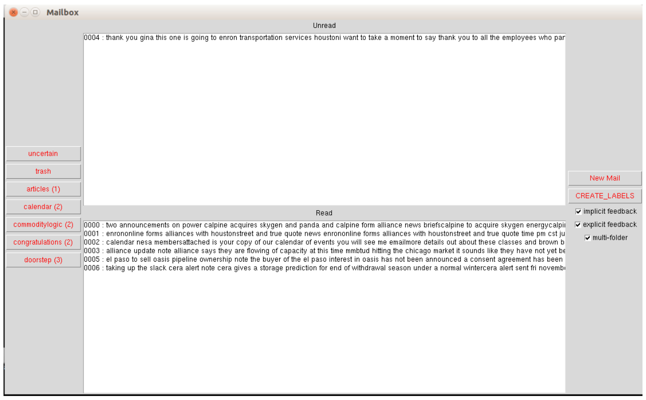


Figure 8: Graphical User Interface

3.6 Explicit User Feedback

3.7 Multi-folder

4. DISCUSSION

4.1 Validity Threads

5. CONCLUSIONS

6. REFERENCES

[1] R. Bekkerman. Automatic categorization of email into folders: Benchmark experiments on enron and sri corpora. 2004.

[2] V. Bellotti, N. Ducheneaut, M. Howard, and I. Smith. Taking email to task: the design and evaluation of a task management centered email tool. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 345–352. ACM, 2003.

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[4] V. R. Carvalho and W. W. Cohen. Learning to extract signature and reply lines from email. In *Proceedings of the Conference on Email and Anti-Spam*, volume 2004, 2004.

[5] G. V. Cormack. Email spam filtering: A systematic review. *Foundations and Trends in Information Retrieval*, 1(4):335–455, 2007.

[6] M. Dehghani, A. Shakery, and M. S. Mirian. Alecsa: Attentive learning for email categorization using structural aspects. *Knowledge-Based Systems*, 2016.

[7] B. Klimt and Y. Yang. The enron corpus: A new dataset for email classification research. pages 217–226, 2004.

[8] H. Murakoshi, A. Shimazu, and K. Ochimizu. Construction of deliberation structure in e-mail communication. *Computational Intelligence*, 16(4):570–577, 2000.

[9] M. Sokolova and G. Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437, 2009.

[10] M. S. Sorower, M. Slater, and T. G. Dietterich. Improving automated email tagging with implicit feedback. pages 201–211, 2015.

[11] R. Sproat, J. Hu, and H. Chen. Emu: An e-mail preprocessor for text-to-speech. In *Multimedia Signal Processing, 1998 IEEE Second Workshop on*, pages 239–244. IEEE, 1998.