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 [7]; Sproat et al., 1998 [10]). 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. Plug-ins to monitor user behaviors in the email client to provide non-intrusive notification asking for user feedback.

The GUI demo when a new email arrives is presented in Figure 1. Upon 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) a folder named "Confusing" will store the most confusing emails for the classifier and wait for the user's judgement.

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. SPECIFIC PROBLEMS

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 Less 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.

2.1.2 Solution

The incremental learning ability can be the key solution of this problem. Start with hundreds of training examples and a simple model, feedback from user can be collected for the further training of the model.

Three types of user activity can be collected as feedback to the system:

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

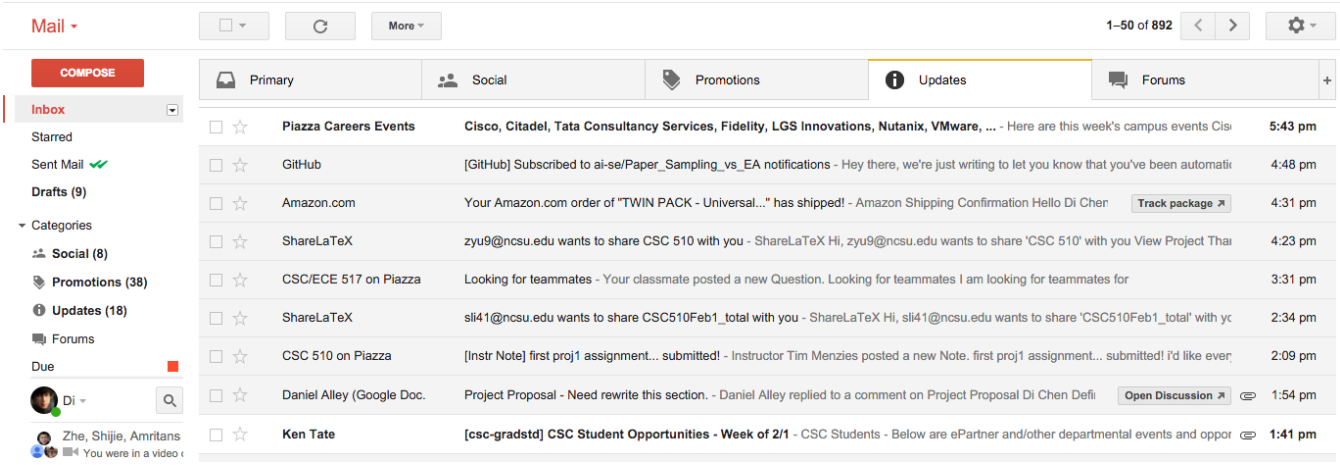

Figure 1: GUI Demo

a) user read, replied, or forwarded a certain email without moving it into another folder. This email will be treated as correctly classified.

b) user manually moved an email from the predicted folder to another folder. This email will be treated as misclassified and the latter folder becomes the true label of it in the next round of training process.

c) inspired by active learning, the most confusing emails can be put into a specific folder called "Confusing" that requires the user to manually choose the correct folder.

The first two types of user activity are referred to as implicit events. Section 2.2 will discuss more details about how to use implicit feedback to improve the system. With the help of user feedback, the system can grow stronger and more adaptive to this certain user and produce less errors as time passes by.

2.2 Alternative to Explicit Feedback

2.2.1 Challenge

When the classifier is not confident then positive feedback confirming the correctness of those predictions would be very helpful. The implicit feedback events are underutilized. It is a good alternative to explicit feedback. Users occasionally fail to provide corrective feedback, especially if they are working quickly or are deeply engaged in a task. Under such conditions, self training is risky [9]. This was the motivation to explore *implicit feedback*. By the implicit events/interactions, we mean these events :

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 tags are corrected almost immediately within a few interactions [1]. One way to use these events for implicit feedback events. Keep a threshold on these events, when it exceeds a total number of events then the tags are assumed to be correct and the classifier trains on that message. We can set less threshold for good confident tags and more threshold for bad confident tags. This made us to explore [9] and found 4 methods to tackle implicit feedback events:

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

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

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 tag, 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 [4]. 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 [5].

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 [5], 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 [3]. 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 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.

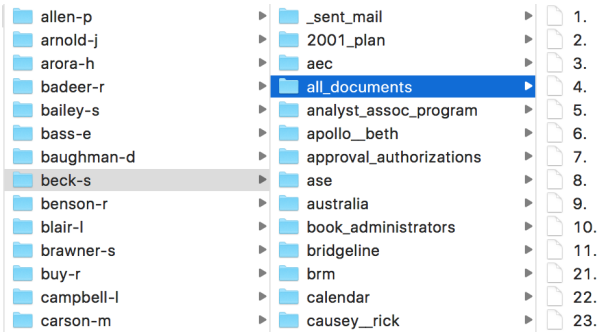

Figure 2: Structure of Enron Data Set

```
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 3: Email Content

3. PLAN OF WORK

The two major parts of this project are GUI and algorithm. Both of these are important to the success of a software product. Training and testing data set need to be collected and proper performance metrics be selected in order to correctly evaluate the performance of different algorithms. On the other hand, only real user experience can be used to properly evaluate the GUI design.

3.1 Data Collection

The Enron Corpus is a large database of over 600,000 emails generated by 158 employees [6] 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 2. An example of the content of one email is presented in Figure 3.

At this point, we will be using subset of this data set comprising about 5 users and 10 folders for each users. Timeline will be followed to simulate the process of receiving new emails.

3.2 Performance Metrics

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

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 [8].

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 [8].

3.3 Path to Success

3.3.1 Start with the Simplest

Without considering any challenge described in Section 2, the baseline result will be created as follows.

GUI: create a simple GUI framework that allow user to receive and send emails. Allow the user to create folders and automatically put incoming emails into one of the folders.

Algorithm: focus on text categorization. Use standard preprocessors (stemming and stop word removal, term frequency, hashing trick, L2 normalization) to exact features from email subject and body, compare different classifiers (SVM, Naive Bayes, Decision Tree).

Test: test without GUI to have accuracy, F_M. Decide which classifier is most suitable. Incorporate algorithm with GUI. Collect user experience.

3.3.2 Add User Feedback

As described in Section 2.1 and 2.2, user feedback will be introduced in our tool.

GUI: add event-driven functions to collect implicit user activities. Add a folder called "Confusing" to allow the most confusing emails to come in.

Algorithm: set the initial number of training example to be one hundred. Retrain the classifier once 50 more emails come in.

Test: test without GUI to have accuracy, F_M. Incorporate algorithm with GUI. Collect user experience. Find best way to facilitate user as well as collect new training examples.

3.3.3 Add Multi-label Structure

As described in Section 2.3, multi-label structure will be introduced.

GUI: allow one email to appear in multiple folders. Fix possible bugs when synchronizing. Ask the user to rank the importance when creating a new folder.

Algorithm: classifiers will now predict probabilities for each label. If the probability of a email belonging to one label is higher than its threshold, this email will go into the folder of that label. Different labels can have different threshold according to their importance.

Test: test without GUI to have accuracy, F_M. Incorporate algorithm with GUI. Collect user experience. Find best way to facilitate user.

3.3.4 Add Decision Fusion

As described in Section 2.4, decision fusion will be introduced.

GUI: ask user to select the type when creating folders.

Algorithm: have three independent models focusing on different types of folders, each has different feature set. Apply decision fusion to decide the belonging of incoming emails.

Test: test without GUI to have accuracy, F_M. Incorporate algorithm with GUI. Collect user experience.

3.3.5 Add Non-text Content as Features

As described in Section 2.5, non-text content will be included as features.

GUI: nothing will change to GUI.

Algorithm: extract non-text content as features.

Test: test without GUI to have accuracy, F_M. Compare with previous results to see whether this feature is beneficial.

4. SUMMARY

It can be very challenging for a team of four to finish everything described in Section 3.3. In the next step of our project, it is possible that we decide to either:

a) give up GUI design, since the purpose of this course is not designing a user interface. It is better to focus on the most import part.

b) give up the last half part of features described in Section 3.3. The last half part of features are less import comparing to the previous half.

5. 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] 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.

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

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

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

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

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

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

[10] 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.