Classification of Internal Mailing Lists

Erin Knight
School of Information
University of California - Berkeley

Ryan Greenberg
School of Information
University of California - Berkeley

Abstract

The School of Information at UC Berkeley has two main internal mailing lists, Fun and Noise, for informal communication between students, faculty and alumni. Fun is meant for time-sensitive, actionable information such as events, parties and concerts. Noise is intended to cover everything else, including technology reviews, job opportunities, political opinions, news, and knowledge sharing. This project seeks to address two problems with these lists. First, although people may have a sense of which list messages belong to, people often send content to one list when it should be sent to the other, or they duplicate content on both lists. Second, the Noise list has so much traffic that it is difficult for readers to prioritize their attention. We trained a classifier to distinguish Noise messages from Fun messages, and another classifier to organize Noise messages into discrete categories. These classifiers could be implemented as programs on a mail server to direct incoming messages and categorize them accordingly.

1 Introduction

Students, faculty, staff, and alumni at UC Berkeley's School of Information use two internal mailing lists extensively for communication, Noise (noise@ischool) and Fun (fun@ischool). Noise is dedicated to discussions of new technologies, politics, and general "noise," whereas Fun is meant for specific opportunities such as an upcoming movie showing or a party invitation.

Starting in 1999, the Fun mailing list was the main medium for informal exchange at the iSchool. In 2004, the Noise list and the distinction between the two lists emerged from students' desire to separate time-sensitive social activities and opportunities from general discussions and knowledge sharing. This distinction has been a contested topic over the years, one that was often strictly enforced by members of the iSchool community. On occasion, messages sent to the wrong list were met with a response to the original sender (and the whole list): "This is really more appropriate for Fun" or “Do not send emails like this to Fun. This is a Noise discussion”. It was an indirect slap on the wrist and branding that the sender was unconnected with or an outsider to the iSchool community.

Lately, however, the distinction between these two mailing lists has been muddied by misuse. Many students end up send messages to both lists to be sure their message is received. This means many Fun emails end up in Noise and vice-versa. Whether the issue stems from an inability to determine the distinction, or an unwillingness to decide between the two lists, the result dilutes the value of the distinction, which helps students, faculty, and staff filter information and prioritize their attention.

Information overload is also an issue within the Noise list itself. Because it is seen as the catchall for information exchange outside of events or activities, it is understandably flooded with a variety of messages. This corpus, while informal, represents a valuable record of student knowledge, trends, expertise and social interaction. It is a running record of the ideas, topics and opinions that are important to the community at any given

time. One may go so far as to assert that most of the iSchool implicit knowledge, at some point, is expressed or built upon through Noise. That said, there is also a lot of "noise", with random messages about an apartment for rent or a lost power cord. Weeding out the "noise" from Noise is difficult. Faced with increasing traffic, many subscribers cannot manage the flow and as a result filter out the list completely. These students and faculty potentially miss out on some critical knowledge sharing and discussions that are relevant to them, because there is not way to easily evaluate the relevance and priority of Noise messages at a glance. One solution would be to classify the Noise messages into several top-level categories to enable people to follow specific categories, or prioritize certain messages and their time.

Our work on these problems consisted of two parts. First, we trained a classifier to label messages as either Noise or Fun, to help separate traffic to the two lists. Second, we trained a separate classifier to subcategorize messages sent to Noise with labels including technology, politics, job opportunities, news and miscellaneous.

2 Related Work

Email classification is one of the most common uses of statistical NLP. Since Paul Graham wrote "A Plan for Spam", statistical language techniques have become the foundation of spam detection. [1] Using Bayesian filtering with the frequencies of tokens in email message bodies and headers has yielded a high level of accuracy with a minimal number of false positives. Researchers have done significant work in both NLP and machine learning related to spam classification of email.

There are fewer examples of classifying email in domains beyond spam/non-spam. One example is Linger, a system that allows users to train a classifier by placing email messages into folders. The system learns the characteristics of messages in those folders and seeks to automatically file messages into folders in the future. [2] Although the classification techniques used by Linger are similar to the ones we used, this system works on the end user's machine whereas we would like to label messages at the point where the mail server receives them.

Additionally, in general, much work has been done in NLP on topic classification. In fact, NLTK has many corpora that are built around these types of NLP techniques like the Reuters topic collection. [3] Real world applications such as Google News and advanced search engines do topic classification to carve up a vast information space into smaller, targeted categories to help people find relevant information more easily.

3 Data and Features

3.1 Corpus Construction and Preprocessing

We initially used one year of data for each list, framed around the school year calendar, which was August 2008 to August 2009. This timeframe yielded far fewer messages from Fun, so we expanded the timeframe to include two years of messages from Fun, from August 2007 to August 2009. We retrieved archives of all the messages from these timeframes using the email interface provided by the Majordomo software that maintains the mailing lists.

Once we had archive files containing all of the messages sent to each list, we split these files into individual messages. We used the Python Email module to process each message, removing binary file attachments, multiple message payloads, HTML formatting, extraneous headers, and signatures. Because our goal was to classify incoming messages, we only used messages that started threads and ignored any replies. This was based on the assumption that replies would be posted to the list where the initial message was sent. After preprocessing we had 1485 messages in Noise and 789 in Fun.

We used the Natural Language Toolkit (NLTK) [4] to load these messages as a corpus. By subclassing NLTK's CategorizedPlainTextCorpusReader we were able to isolate the header, subject and body of the message for more fine-tuned analysis. Messages were labeled according to the mailing list they were originally sent to. Cross-posted messages were labeled Fun.

3.2 Features

Our initial assumption was that the words for each message alone would be enough to classify messages. Based on our analysis of the messages themselves, we used the following features:

- *Words.* Do people use different words in the material they send to Fun compared with messages sent to Noise? We used the 2500 most common words from each category in the corpus as features, after excluding words longer than 14 characters or shorter than 4 characters. We also used modal verbs as features.
- *Links.* Does the email contain a link? We assumed that the Noise messages would be more likely to contain links since much of the message traffic is sharing links.
- *Forwarded messages.* Was the email a forwarded message? Many people forward other emails to Noise, although we were uncertain if this feature would be significant when compared across lists.
- *Dates, Times, and Ordinals.* Many posts to Fun involve events with include a date or time, which we thought would be a distinguishing feature. We checked for this by looking for time strings like "6pm", "5:30", month names, and other date indicators like "5th".

We experimented with using popular bigrams from the corpus as a feature, but this increased processing time considerably with negligible benefit. For the second task involving classifying messages as parts of Noise, we added the length of the message itself as a feature.

3.3 Labeling

Both of our analyses required supervised training, so we needed to label the corpora.

Inclusion in one list or the other sufficed for labeling for the first analysis. As mentioned above, loading the corpus using NLTK's CategorizedPlainTextCorpusReader enabled easy and automatic labeling as either "fun" or "noise".

For the second part of our work we used the same previously processed Noise corpus mentioned above, which included 1485 messages (excluding replies). We reviewed these messages and hand-labeled them into five categories:

- *tech* - Technology reviews and discussions.
- *news* - Knowledge-sharing, typically forwarding a link and the associated discussion.
- *politics* - Opinions and debate around political issues, mostly the 2008 election given the time frame we used.
- *jobs* - Job or grant opportunities, contests.
- *misc* - Everything else.

This was a slightly time-intensive exercise since we manually sorted the list into five folders representing the five categories. We ended up with 82 messages in 'jobs', 461 in 'misc', 622 in 'news', 106 in 'politics' and 198 'tech'. After sorting, we loaded these messages using our subclass of NLTK's CategorizedPlainTextCorpusReader, which again automatically applied the labels based on the folder the messages were in.

4 Models and Results

4.1 Noise vs. Fun Classification

Our first analysis involved a binary classification task, since all messages were either labeled 'noise' or 'fun'. We used a Naive Bayes classifier, which achieved an 80% accuracy rate. Many of the most distinguishing features were words obviously related to events, which are the types of messages that we want classified as “fun”. For example, the following words occurred 8 to 10 times more often in messages from Fun than Noise: *musicians*, *nearby, musical, concert, celebrate, prize, dance, audiences, admission* and *doors*. The features we defined besides words had less discriminating power. In one pass the only other features that appeared in the top 1000 features were whether the message had "AM" or "PM" (4.1 times more likely to be Fun) or whether the message had an associated time (2.7 times more likely to be Fun).

We also used a maximum entropy classifier since we had multiple, potentially dependent features, and we were trying to determine the topic from the words, which is more in line with a discriminative approach. After four iterations of training, our classifier achieved only 65% accuracy. This could be because the features were not dependent, or due to some technical issues we had with the classifier, including several iterations with a “nan” result.

Given a balanced corpus with two categories, a classifier could achieved 50% accuracy by chance,

which means that 80% would be a significant improvement in accuracy. However, the Noise and Fun do not receive equal volumes of messages, and judging the significance of our results depends on the timeframe being evaluated. There has been a fall-off over the past year in traffic to Fun, which means that a classifier selecting the largest category by default could achieve nearly 95% accuracy. This recent phenomenon is part of the problem we are trying to solve, so we think a more fair comparison is the timeframe of our corpus when traffic was split more evenly across categories, with messages to Noise comprising about 72% of the total. Using this metric, an 80% accuracy rating is still an improvement, though not as dramatic as it might seem.

4.2 Noise Classification

The Noise classification was a multi-class classification problem since any given message could have been in one of five categories. We achieved practically the same accuracy with a Naive Bayes and Decision Tree classifier.

Naive Bayes Classifier - We achieved a 63% accuracy and the most informative features were:

```
contains(election) = True     politi : misc =56.6 : 1.0
contains(campaign) = True     politi : misc =48.3 : 1.0
contains(opportunity) = True  jobs : news =  43.6 : 1.0
contains(participants) = True jobs : news =  33.3 : 1.0
contains(learning) = True     jobs : news =  33.3 : 1.0
contains(complete) = True     jobs : misc =  25.5 : 1.0
contains(participate) = True  misc : news =  24.0 : 1.0
contains(article) = True      news : misc =  23.2 : 1.0
contains(appreciate) = True   jobs : news =  23.1 : 1.0
contains(submitted) = True    jobs : news =  23.1 : 1.0
contains(purpose) = True      jobs : news =  23.1 : 1.0
contains(schedule) = True     jobs : news =  23.1 : 1.0
contains(academic) = True     jobs : news =  23.1 : 1.0
contains(forwarded) = True    jobs : news =  23.1 : 1.0
contains(competition) = True  jobs : news =  23.1 : 1.0
contains(proposals) = True    jobs : news =  23.1 : 1.0
contains(advance) = True      jobs : news =  23.1 : 1.0
contains(valuable) = True     jobs : misc =  21.5 : 1.0
contains(creative) = True     jobs : misc =  21.5 : 1.0
has_link = False              tech : news =  21.5 : 1.0
```

Decision Tree Classifier - This classifier also had approximately 60% accuracy. The output pseudocode illustrated the significance of various features and combinations of features, such as “has link” and “craigslist”.

```
Pseudocode from Classifier

if has_link == False:
        if contains(software) == True:
                if contains(volunteers) == False:
                        if contains(management) == False: return 'tech'
                        if contains(management) == True: return 'misc'
                if contains(volunteers) == True: return 'jobs'
if has_link == True:
        if contains(event) == False:
                if contains(craigslist) == False: return 'news'
                if contains(craigslist) == True: return 'misc'
        if contains(event) == True:
                if length == 0.0: return 'politics'
                if length == 100.0: return 'news'
                if length == 200.0: return 'misc'
                if length == 700.0: return 'jobs'
                if length == 1500.0: return 'news'
```

While 63% accuracy is lower than that achieved in part one in absolute terms, it represents a greater improvement. For a classifier with five labels, chance selection can achieve 20% accuracy. By selecting the largest category (‘news’) as a default, our classifier could achieve 40% accuracy because 'news' in our corpus accounted for 40% of messages. Compared with this baseline, 63% is a significant improvement over chance and optimal selection.

5 Future Work

For this project, we were focusing solely on the classification and underlying NLP methods. We would like to see user interface components for each of these classifiers that would allow the community using these mailing lists to benefit from our work. Additionally, we would like to explore other approaches to the classification that would combine our analyses for potentially better results.

5.1 Interfaces

Based on our work in part one, separating messages sent to Fun and Noise, we envision a website or application interface where students and faculty could "check" their message before sending it, to find out which list would be most appropriate. Another possibility would be to send all messages to a single address and rely on our classifier installed on the mail server to redirect messages to the appropriate destination.

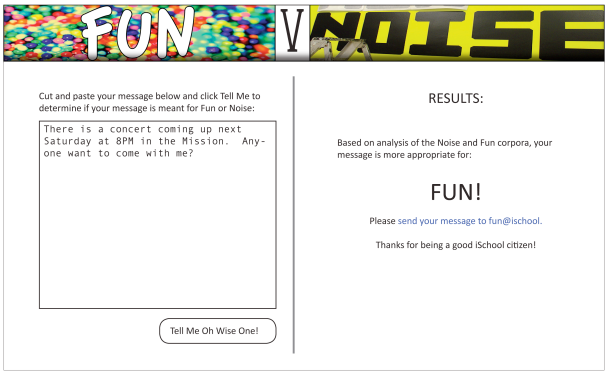

Figure 1. Interface mockup of website for users to determine which mailing list their message belongs on.

Our work on the classification of the Noise corpus would be most valuable if it facilitated organized or labeled messages in users' email clients. One simple way to achieve this would be to run our classifier on the mail server and to apply an X-header with the category of each incoming message. Then users could set up a rule to label and file the messages appropriately using Gmail or another email client.

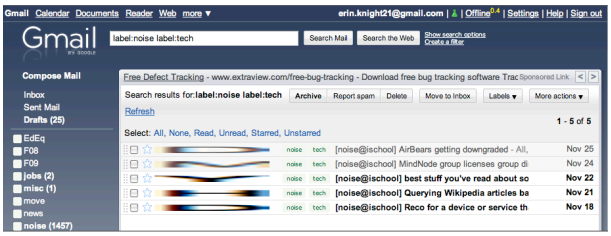

Figure 2. Email client view with Noise messages labeled appropriately.

5.2 Further Classification

Another avenue for future work would be to combine the two classification problems discussed in this paper. If we considered Fun as another one of the categories of content comprising Noise, we might achieve better results. Rather than comparing messages from Fun against the all of the different kind of messages in Noise, some of which may share characteristics with Fun, we could use 'fun' or 'events' as a sixth label in a unified analysis.

6 Conclusions

We achieved only moderate success in our first task of separating messages directed to Noise and Fun (80% compared with a 72% baseline). The words in the messages proved to be the strongest features for training, and many words associated with our characterization of Fun emerged as highly distinguishing features. Ultimately, messages sent to Noise and Fun were not as distinct as we had imagined. Part of this is due to the problem we were trying to solve, since people have directed messages inappropriately in some instances. As a result, messages that are more properly Fun may have appeared in Noise and vice versa. Another confounding factor is that messages about academic events and course offerings often looked very similar to Fun messages in that they have a time and use some of the same words as Fun. Additionally, evaluating the effectiveness of our classifier was difficult because of changes in traffic on Noise and Fun over time.

Classifying messages sent to Noise was simpler. Using a Naive Bayes classifier we achieved 63% accuracy (40% baseline). Although this is less accurate in absolute terms than our work in part one, it represents a greater improvement. A future version would combine the two analyses and add the Fun messages as another category in the classification, for potentially better results.

Overall we consider our classification efforts to be fairly successful, given the data we had to work with. We feel the separate mailing lists, and distinction between them, are important for information filtering. Additionally, carving up the Noise list is very valuable for prioritizing the iSchool community's information influx. Providing interfaces to these classifiers, either via a website or as an integrated part of a mail server, could allow users to consume information sent to these internal mailing lists more efficiently.

7 Resources

[1] Paul Graham, “A Plan for Spam,” Aug. 2002. Retrieved from http://www.paulgraham.com/spam.html.

[2] J. Clark, I. Koprinska, and J. Poon, “LINGER-a smart personal assistant for e-mail classification,” Proc. of the 13th Intern. Conference on Artificial Neural Networks (ICANN’03), 2003, pp. 274–277.

[3] Bird, S., Klein, E., Loper, E. (2009). Natural Language Processing with Python - Analyzing Text with the Natural Language Toolkit. O'Reilly Media. http://www.nltk.org/book

4] NLTK API. (2009). API documentation for NLTK. http://nltk.googlecode.com/svn/trunk/doc/api/index.html

[5] Manning, C. & Schütze, H., Foundations of Statistical Natural Language Processing, MIT Press. Cambridge, MA: May 1999.