Identifying Important Communications

Aaron Jaffey
ajaffey@stanford.edu

Akifumi Kobashi
akobashi@stanford.edu

Abstract

As we move towards a society increasingly dependent on electronic communication, our inbox sizes have become so large that many of us cannot keep up with the deluge of information. To that end, we wanted to know if it would be possible to reduce some aspect of this social noise by using machine learning to filter out what users consider to be unimportant. Although this sounds similar to a spam filtering problem, we wanted to consider all major forms of electronic communication: phone, email, and Facebook. Furthermore, this type of classification is highly individual, compared to spam filtering, so one's own communications provide the best source for training data. Using Naive Bayes and SVM classifiers trained on a sizeable set of communication metadata, cross-validation showed that the algorithms performed decently on these different sources, achieving a best accuracy around 84% for Facebook data, 89% for cellular data, and 98.5% for email data. These data sources appear to be weakly correlated, and so it is difficult to improve classification accuracy by linking various communications using an address book.

1 Methodology and Data

In order to train classification algorithms, it was first necessary to collect personal communications data. For each of the three means of communication we utilized, we wrote Python scripts to scrape the data. We obtained email data using IMAP, Facebook data using the Facebook developer API, and cell phone data through an export process from our cell phone carriers.

For this problem, we chose a simple binary classification scheme to divide communications into important and unimportant. While this is a rather granular classification scheme, it makes automated and manual classification of training data easier. For example, for email, we are able to classify training data using different methods based on one's usage patterns. In one case, we can consider read messages in the the user's inbox as important, and unread ones as unimportant. Alternatively, we can consider messages in one's inbox as important, and archived messages as unimportant. Similarly, we can assume that emails one replied to were important. Facebook message data is rather difficult to classify automatically by examining metadata. While the Facebook message center is constantly changing, users do not typically receive sufficient unimportant mail that is discarded to use this as training data. Furthermore, almost all personal messages appear in the user's inbox. Determining if a Facebook message was replied to is problematic, because messages can be divided into short snippets from chat sessions, and clustering them to identify conversations is an undertaking in itself. Due to these difficulties, we chose to classify Facebook messages manually. Classifying phone call data posed a similar problem, so we also chose to classify calls manually.

With this project, we hoped not only to attempt to classify important communications within each platform, but also to link our classification results together in a logical and useful way. A natural way to do this is to connect data based on the other party in the communication. We began by exporting our personal address book data into a form that could be integrated with machine learning algorithms, and queried to derive a unique identifier for a person based on their name, email address, or phone number. If the third party was not in the address book, we generated a new unique identifier for them. By including features that link a communication to its sender/recipient, we attempted to

derive an importance measure for individuals one communicates with. This data can be plugged back into the algorithms and be used to classify new test data with better theoretical accuracy. For example, if users were to make a number of phone calls to their friend, we hoped the algorithm might increase the predicted importance of emails received from their friend.

Since we chose to make this a binary classification problem, natural choices of algorithms are supervised learning algorithms such as Naive Bayes, and Support Vector Machines. We wrote an implementation of Naive Bayes in MATLAB, and used LIBSVM [1].

2 Choice of Features

We chose to include as many features as possible from metadata that had non-zero data, and data that was easily convertible to a numeric value. In cases where there was redundant or irrelevant data (i.e. cell phone call records contain a rate code responding to time of day, in addition to a timestamp, as well as a billing category that depends on the other partys carrier), we left out the redundant features to avoid adding extra noise into our classification problems. For cell phone data, we have seven per-call features–day of week, time of day, duration, frequency of calle(r/ee), time since last call with this person, whether or not a call was incoming, and if the calle(r/ee) is in the address book. Facebook metadata provides six features: whether the message is read, the fraction of unread messages in a thread, number of messages in a thread, frequency of messages from the sender/recipient, length of the message, and number of words in the message. Email included these six, in addition to whether the users email was located on the To, the CC, or neither.

In order to integrate the different forms of electronic communication, we added an identifier for the other party in the communication to each data vector. We experimented with different ways of integrating this information, including using one binary feature for each person the user has communicated with, using a single identifier for the other party involved, and directly including features detailing the frequency of communication with the party. We settled on the first method since it was simplest way to handle communications with multiple recipients.

3 Combining Data Sources

Since we classified three different types of communications data separately, we hoped to combine the results of training on these data sets in a useful way. That is, we hoped to increase our accuracy of classification by associating communication across different modes. We tried two different ways of integrating data from the different sets. First, we tried direct insertion of features. In other words, for each of the three modes we explored, we injected features associated with the sender/initiator, some derived from other modes. For example, features used in email classification included the frequency of phone calls sent and received with the sender of the email, in addition to the frequency of email communication. In total, four features were added to each of the three mode of communication, corresponding to the frequency of sent/received messages from the other two modes.

We also tried using the decision values from training each data set to produce an importance metric for each sample. This approach made sense, particularly for Naive Bayes and Linear SVM, since in the first case, the decision value was a probability the message was important, and in the second case, the value was the distance from the margin. Using the address book, we mapped these decision values into a matrix by individual and communication type. We repeated the following algorithm until cross-validation accuracy stopped changing:

1. For each communication type j, train the algorithm and then classify the training data using the model.
2. Compute the following
   (a) For each sample $x_j^{(i)}$, extract the contacts vector $c_j^{(i)}$, where each element in $c_j^{(i)}$ is a binary value corresponding to whether or not a particular contact was involved in the message. All together, the $c_j^{(i)}$ values produce a matrix C.
   (b) Find the matrix $D = v^T C$, where v is a vector containing the decision values for each sample.

(c) To compute a per-contact value, we averaged all of the non-zero columns in D to get the vector c'_j.

3. Stack the c'_j vectors vertically to create a new matrix C', and generate a new vector c'' by taking the geometric means column by column. The choice of geometric mean was motivated by the fact that the margins (or probability values) from different data sets are not comparable in absolute values, but make more sense when compared relatively. In doing so, we chose to weight each of the three forms of communication equally, but introducing weighting coefficients for each of the communication methods may be useful if there is sufficient overlap in the data for this technique to be effective.

4. For each communication type, multiply the vectors c'' and $c_j^{(i)}$ element by element to modify the original data, producing new training data X_j.

4 Results

4.1 Comparison of different algorithms and kernels

We compared three algorithms for classification; a support vector machine with linear and gaussian kernels, in addition to a Naive Bayes classification. Using 10-fold cross validation, we generated statistics for the accuracy of different data set sizes. For data sets smaller than the full data set, we performed the cross-validation test 30 times and averaged the results to better represent the true accuracy of these smaller data sets. SVM (Figure 1) with both kernels performed extremely well for email data with accuracies close to 100%, much better than the Naive Bayes which still achieved a respectable 90% accuracy. Naive Bayes accuracy (Figure 2) was slightly better than SVM accuracy for a very small data set, as expected, due to smaller sample complexity. Accuracy using a linear kernel was very close to accuracy using a Gaussian kernel.

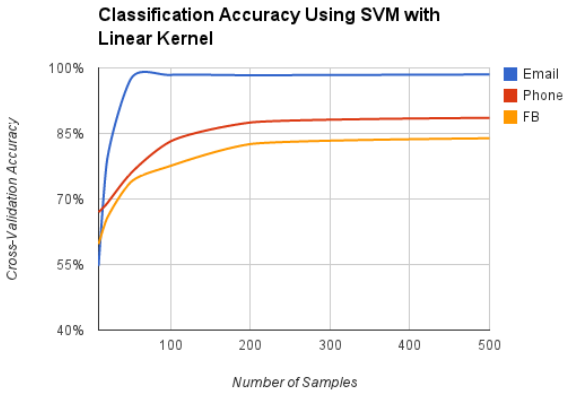

Figure 1: Cross-Validation accuracy of SVM using Linear kernel.

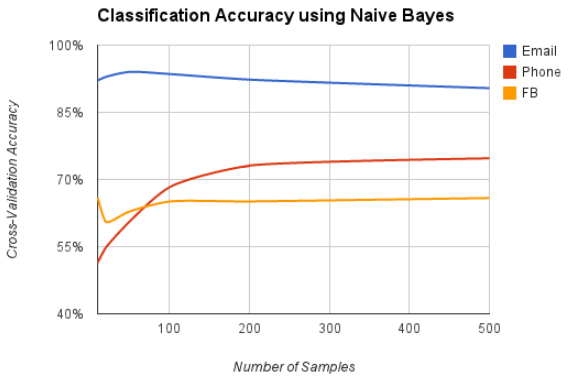

Figure 2: Cross-validation accuracy of Naive Bayes.

4.2 Combining modes of communication

Our attempt to combine the data from different sources was not very beneficial for large sample sizes when applied to our data sets in either of our strategies. With our first tactic of feature injection, the cross-validation accuracy increased slightly when combined with phone data, but with the full email data set, this benefit shrank to less than 0.2% across the board. The learning curve in figure 3 illustrates the improvements seen. In our

second strategy, we saw a limited improvement in cross-validation accuracy for Naive Bayes of 0-0.5%, and an improvement for SVM kernels of 0-0.2%.

4.3 Feature selection

By running a feature selection algorithm based on F-scores [2], we found that in most cases the frequency of contact with the sender of the message was the most heavily weighted feature. This is in line with our hypothesis that it is the person that matters, and not so much the content of the message. Interestingly, the accuracy of classification dropped, then plateaued after just five features, indicating that high accuracy can be attained with very few features. The five, in order of weighting are 1) the number of emails sent to the sender 2) number of emails received from the sender 3) number of emails in the thread 4) whether the email was directed to us on To or CC and 5) the number of words in the subject.

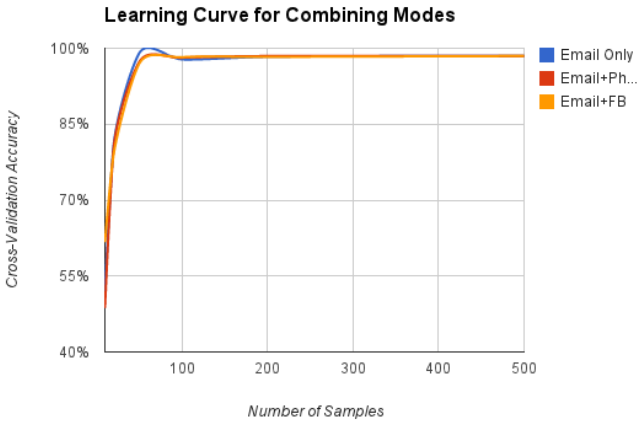

Figure 3: Cross-validation accuracy of email data, when using features inserted from data obtained through other communication types.

For cellular data, the most heavily weighted features were whether the calle(r/e) was in the address book, followed by the call duration. This is logical, since most important phone calls are from somebody one knows. This data depended much more on the presence of the contact identifiers than email in order to obtain high accuracy.

For FB data, message length was the most indicative feature, followed by the fraction of unread messages in a thread. This makes sense as well, as short chat messages are likely less important than longer ones.

4.4 Error analysis

Our initial efforts were focused on breaking past this 90% accuracy barrier, but a manual examination of all of the miscategorized emails indicated that all of these were emails that were read by mistake, or read based on the words of the subject. By manually classifying the misclassified emails into these two categories, we found that 11% of the miscategorized emails were read based on the words in the subject, but would have otherwise not been read (e.g. interesting job postings on a noisy mailing list). Thus, even introducing a textual feature recognition algorithm would have only reduced the incorrectly classified emails by about 11%, with the remaining 89% a systematic unavoidable error inherent in our metric of importance being measured by message read-unread status.

We also generated statistics on the precision and recall of our classifiers (Table 1) to see if they were more biased toward choosing messages as important or unimportant. Additionally, we obtained information on how well the data was balanced between the two classes. For phone, Facebook, and email data respectively, 47%, 55%, and 10% of the messages in the training data set were considered important. Fortunately, for email and phone data, the precision statistic, which represents the fraction $\frac{\text{true unimportant}}{\text{true unimportant+false unimportant}}$, was larger than the recall statistic, $\frac{\text{true unimportant}}{\text{true unimportant+false important}}$. This means those classifiers were less likely to miss an important message than they were to mark an unimportant message as important. For this problem, this was the preferred bias since misclassified unimportant messages can easily be discarded or looked at later. For the email data, the only unbalanced data set, the recall statistic was not too low, indicating that the classifier was good at not classifying too many unimportant messages as important.

Table 1: Linear SVM Classifier precision and recall using full data set.

	Email	**Phone**	**FB**
Precision	90.11%	84.03%	90.88%
Recall	88.14%	93.89%	85.76%

5 Conclusion

For many people, email, phone, and Facebook communication data sets seem to be used for distinct and mutually exclusive purposes. In other words, communication with a particular person was primarily restricted to one mode. For example, grandparents are always contacted by phone, parents by email, and friends by Facebook. In fact, our data showed that only 1.06% of contacts were common among more than one of the communication schemes. This corroborates our examination of why email classification accuracy barely improves with the addition of both Facebook and phone data. Because the strongest features were the number of emails sent and received, a boost in accuracy was only seen where an oft-contacted friend on one mode switched over to another mode temporarily. Continuing the example above, this would be an email from grandparents.

Accuracy obtained using the SVM classifier was decent enough to be used as an aid, but insufficient to use as a screening mechanism as is often used with spam filters. Since unimportant communications are still likely to be attended to by the recipient, having perfect accuracy is not crucial. One possible implementation for email or Facebook would be a filter based on importance, where the user would choose to read important messages immediately, and the others at a regular, but less frequent interval. A configurable threshold could allow the user to set a bias based on their level of concern about the chance of missing an important message until later.

6 Future plans

For this project, we chose to utilize large data sets from recent communications, we realized that our approach could also be applied to live data using an online learning approach. This would be a more practical implementation that would enable the algorithm to adapt to new contacts and importance preferences. Online learning could be particularly interesting for cellular communications–your phone might predict whether an incoming call is important, and then be trained based on whether or not you answered it.

Also, addition of textual features from both the subject and the body of messages would likely improve the accuracy of classifying Facebook data, since Facebook metadata is sparse and it is difficult to distinguish between different messages in a thread and their importance using exclusively metadata.

Another potential interesting extension would be the application of hierarchical machine learning principles, where many child classifiers could be trained for each different contact the user has, and a parent classifier could be used where the child classifiers are not accurate enough or have no information about a user.

References

[1] Chang, C., & Lin, C. (2011). LIBSVM: a library for support vector machines. ACM Transactions on Intelligent Systems and Technology (TIST), 2(3), 27.

[2] Chen, Y., & Lin, C. (2006). Combining SVMs with various feature selection strategies. Feature Extraction, 315-324.

[3] Hsu, C., Chang, C., & Lin, C. (2003). A practical guide to support vector classification. Retrieved from http://www.csie.ntu.edu.tw/ cjlin/papers/guide/guide.pdf.