# Constructing Personal Networks Through Communication History

Ryan Houlihan* and Hayk Martirosyan†
*Stanford University*

(CS 229 Final Project)
(Dated: December 12, 2014)

This study aims to predict a user's relationships with his or her contacts based solely on the words used in electronic communications between them. Software was created that reads in bulk exported email data and builds a comprehensive graph database of people, the words communicated between them, and their relationships to each other. Several stages of preprocessing and feature selection were applied, and then various classifiers were shown to effectively predict relationships using 10-fold cross validation. Logistic regression showed 5.9% testing error on a sample set of 3000 emails and 309 contacts. The study shows promising results and suggests future work that incorporates message meta-data, other mediums of communication like text messaging, and larger data sets would only improve upon our results.

## I. INTRODUCTION

In the modern technological age communications with friends and acquaintances are no longer transient. Instead, each of us amasses a large collection of electronic communications whether through email, cell or video recordings. In everyday life ones speech and behavior around others directly reflect the kind of relationship one has with another; whether they are friends, lovers, family, strangers, superiors, inferiors, etc. As a team we questioned whether this circumstantial speech and behavior was also present when communicating electronically. We hypothesized that the content of electronic communications alone would allow us to classify inter-personal relationships – something humans can easily do when observing two strangers or old friends interacting with one another.

## II. PLATFORM

To test this hypothesis we decided to use the mass of a person's email as our data set, and classify their email contacts into friend and acquaintance categories. Email is the most widely used form of electronic communication and we believe one is more likely to communicate with all types of people, whether friends or strangers, via email then they are via mobile mediums like text messaging or Snapchat. As features we used individual words and their respective frequency counts in all emails to a given contact. This was chosen over phrases as it greatly simplified our process. For our models we decided to focus on supervised learning approaches after briefly exploring unsupervised approaches such as K-means. The details are outlined in the following sections.

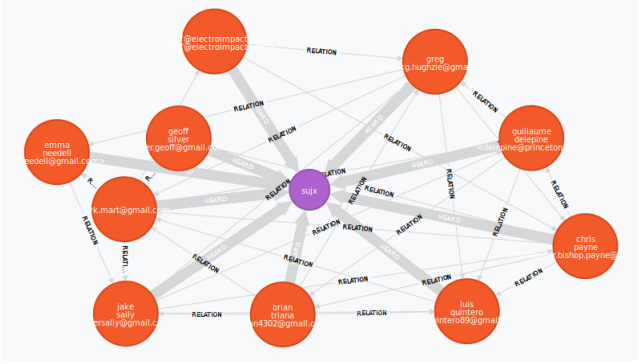


FIG. 1. Example subgraph for a single Word node, people who have heard this word, and their relationships.

### A. Data

Our data set consists of bulk exported emails from GMail which are parsed and curated through software we developed. We first parse the email data in MBOX format into a standardized YAML file containing message entries with fields for date, to, from, cc, bcc, and text fields. During this process, we only consider plain-text email sections. A thorough set of regular expressions are used to lowercase, eliminate links, numbers, punctuation, and all non-word tokens. We decode all encoded data and output in full Unicode. We further process the words by selecting 500 random words from each email and applying a stemming algorithm. When extracted for classification, the term-frequency inverse-document-frequency transformation is also applied to normalize the data.

We then load this data into the Neo4J graph database, creating Person, Message, and Word nodes, and Role, Relation, Alias, and Heard edges. Heard edges keep a count of the frequency of stemmed words spoken between people, and eventually become our feature vectors. Relation edges are used to describe person-to-person relationships, and correspond to our classification labels. Role edges are used to label a person as either the sender or receiver of

* rhouliha@stanford.edu
† hayk@stanford.edu

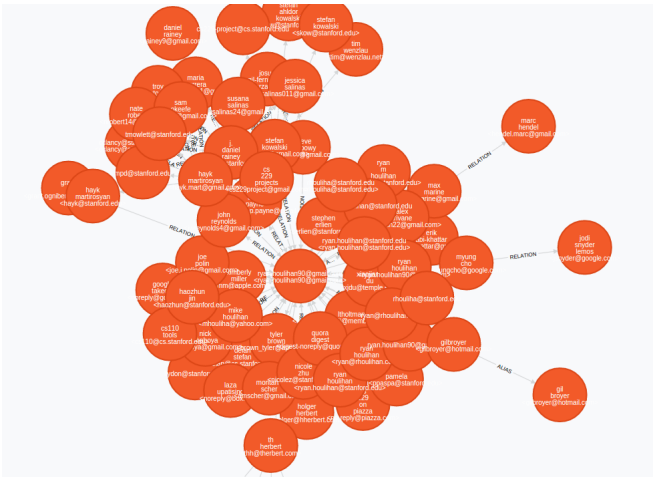

FIG. 2. An example subgraph of Alias edges that combine Person nodes corresponding to the same human but with different email headers.

a given message (to/from/cc/bcc).

When we first developed our graph, we noticed that for a given contact, there might be 5-10 distinct email headers, where either the name or email address was changed. This noisy data meant we were classifying the same person many times. To alleviate this issue, we built a heuristic module which parses through Person nodes, applies fuzzy string matching, and with some user assistance, connects all People nodes that belong to a single person using an Alias edge. The result is that for each person, one canonical Person node contains all of their communications, and the auxiliary Person nodes connect to the canonical node with Alias edges. This result can be seen in *(Fig. 2)*.

Finally, we have a module that allows the user to label their contacts as friends or acquaintances, updating Relations edges in the database. All contacts are classified, and subsets of these are extracted and marked as 'Unknown' during training and cross-validation.

### B. Features

Our features for every sample (person) are the frequency of each word that this person said to our user. The vocabulary begins as the entire stemmed collection of words from the data set, then is curated to lower the number of features. We explored three methods to extract the most important features:

1. A minimum and maximum total frequency cutoff, to eliminate overly rare and overly common words.

2. Principal Component Analysis (PCA) [1, 2] to extract the most relevant features.

3. The chi-squared statistic to extract the k-best features.

To set our minimum and maximum frequency limits we first computed the total frequency (times each word was heard or said) of all distinct words. We then calculated the mean, $\mu$, and standard deviation, $\sigma$, of the words on their total frequencies. The maximum frequency cutoff was set as a chosen number of standard deviations above the mean frequency. The low frequency cutoff was set as a fixed constant value. This proved to be a good way to remove 30-60% of features early on in the process, especially because many words are said only once or twice, and contribute little to classification.

The PCA method performed linear dimensionality reduction using Singular Value Decomposition of the data and keeping only the most significant singular vectors to project the data to a lower dimensional space. The SVD used factorizes the matrix $A$ into two unitary matrices $U$ and $V$, and a 1-D array $s$ of singular values (real, non-negative) such that $A == U * S * V$, where $S$ is a suitably shaped matrix of zeros with main diagonal $s$. This method did not prove to be useful to us for feature selection.

Finally, we selected the k-best features using the chi-squared statistic, which measures dependence between stochastic variables and eliminates features most likely to be independent of the known labels. This method was very effective for reducing the number of features without losing much in classification performance.

### C. Models

We explored the following classifiers: SVM with Linear Kernel, SVM with Exponential Kernel, Logistic Regression (using batch and stochastic gradient descent), Ridge Classifier, and Multinomial Naive Bayes. Below we show the basic method of a few of these:

1. **SVM with Linear Kernel** [3] Given a set of instance-label pairs $(x_i, y_i)$, $i = 1, \ldots, l$, $x_i \in R^n$ $y_i \in \{-1, +1\}$ it solves the primal unconstrainted optimization problem

$$\min_w \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi(w; x_i, y_i)$$

where $C > 0$ is the penalty parameter and loss function

$$\xi(w; x_i, y_i) = \max(1 - y_i w^T x_i, 0)^2$$

2. **Logistic Regression** [3] Performs same minimization as SVM with Linear Kernel but the loss function is instead defined as

$$\xi(w; x_i, y_i) = log(1 + e^{-y_i w^T x_i})$$

3. **SVM with Exponential Kernel** Given a training vector $x_i \in R^p$, $i = 1, \ldots, n$ and a vector

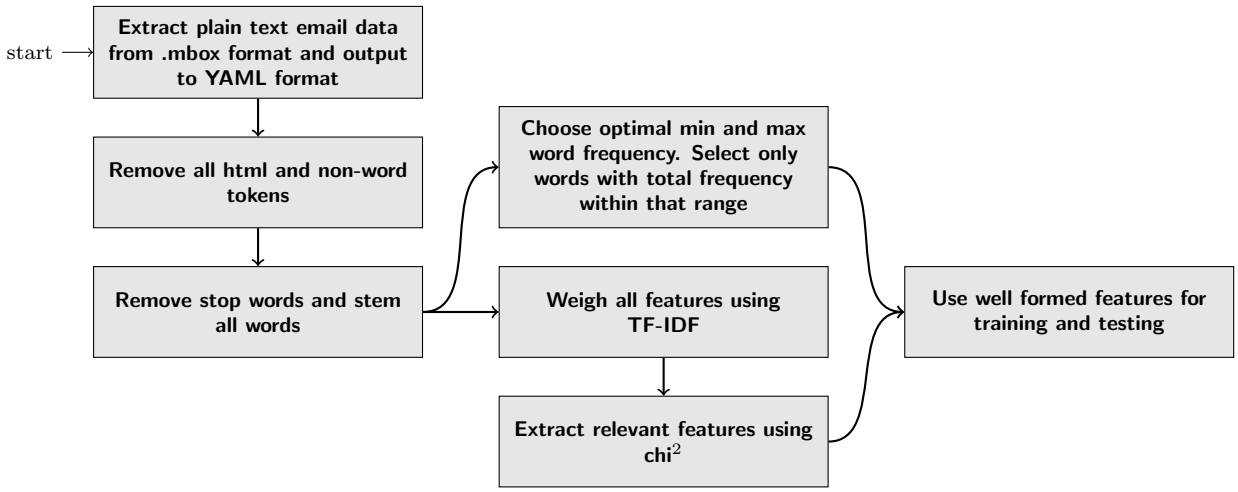


FIG. 3. Process for extracting features from email dataset

$y_i \in \{-1, +1\}$ it solves the dual optimization problem

$$\min_{\alpha} \frac{1}{2}\alpha^T Q\alpha - e^T\alpha$$

$$\text{subject to } y^T\alpha = 0$$

$$0 \leq \alpha_i \leq C,\ i = 1, \ldots, l$$

where $e$ is a vector of all ones, $C > 0$ is the upper bound, $Q$ is a $nxn$ positive semidefinite matrix, $Q_{ij} = K(x_i, x_j)$ and $\phi(x_i)^T\phi(x) = e^{-\gamma|x-x'|^2}$ is the kernel.

## III. RESULTS

All results were obtained using grid searching in scikit-learn. In addition to the named classifiers, all trials used a TfidfTransformer to normalize the data and a SelectKBest feature selection algorithm using the chi-squared test. All results are from 10-fold cross validation, using the following data set:

- Emails: 3000
- Total words spoken: 225000
- Samples (contacts): 309
- Min. frequency cutoff: 5
- Max. frequency cutoff stddev: 20

Our results show excellent classification of the sample data. Logistic regression was the most accurate classifier, at 5.9% testing error. In general, linear classifiers did a little better, likely because the number of features was higher than the number of samples. Nonlinear classifiers tended to over-fit, and would often have 100% training error while having 10% testing error. Reducing the vocabulary by usage frequency was more effective than PCA, but the chi-squared test was the most effective method of feature selection. Reducing the number of features with the chi-squared test reduces the resulting accuracy, but we can get a significant reduction in the feature count without much of a hit. For example, using the top 1000 features we can still achieve over 93% accuracy, and using the top 100 features we achieve 88.9% accuracy.

| Classifier | Accuracy |
|---|---|
| **Logistic Regression** | 94.1% |
| **Logistic Regression w/ SGD** | 93.4% |
| **SVM w/ Linear Kernel** | 93.1% |
| **SVM w/ Exp. Kernel** | 92.5% |
| **Ridge Classifier** | 92.5% |
| **Multinomial Naive Bayes** | 91.5% |

We think our accuracy would certainly improve with a larger data set. However, much of the difficulty comes because it is hard to classify people in black and white as friend vs acquaintance, and there is always a gray area. We can directly see this by looking at the misclassified contacts. It may be more effective to use a continuous 'friendliness' scale, perform a regression, and then compare our results using some closeness test to the labeled value.

Finally, it is interesting to look at the most useful features for classification. Here are the top twenty word stems for this data set:

['putnam' 'unbeliev' 'red' 'there' 'boop' 'iphon' 're' 'subject' 'probabl' 'fwd' 'forward' 'candid' 'guy' 'drink' 'im' 'hugh' 'sox' 'nun' 'i' 'yeah']

These words provide some nice insights. 'fwd', 'forward', 're' are the tokens we replaced forwarded and replied sections in emails with, and signify that how much

we forward/reply vs send new emails is an important factor. 'yeah', 'i', 'im', and 'guy' suggest that informal language is very important. 'candid' likely comes from candidate, which points towards job interviews, and is likely a huge marker towards acquaintance.

## IV. FUTURE

In the future there are a variety of things we would like to accomplish:

1. Use meta-features from the email header and message statistics.
2. Classify into more categories or use continuous 'friendliness' value.
3. Improve insert performance to handle 10,000s of emails.
4. Create a useful open-source visualization tool.

One possible improvement would would be to add message statistics and meta-features as part of our feature vector. Adding features such as email length, email frequency, number of people to, cc'd, and bcc'd on an email, and who was typically sent an email together might lead to interesting results. It would also be interesting to consider heard vs sent words separately.

Another interesting area would be to move away from only a binary classification such as friend or acquaintance and into multinomial classification set such as [friend, family, acquaintance, newsletter]. Alternatively, we could stick with friend vs acquaintance but explore a continuous 'friendliness' scale to avoid the tough gray area with contacts who are almost-but-not-quite friends. One way to provide insight into this is to further explore clustering algorithms such as K-means and see if we can produce some compelling patterns in the data.

Another useful goal is to improve the speed of our database queries and inserts, especially when dealing with email datasets on the order of 60k-100k emails. While our current speeds are fine for our own testing if we ever want to allow other users to begin to take advantage of our software this shortcoming will have to be taken care of. This would also allow us to load in a much larger sample set which could help improve both supervised and unsupervised approaches.

Finally, integrating a tool which would allow us to better visualize our results would be enormously helpful. While neo4j's features are sufficient for our current needs it is overall quite slow and only able to display a small number of nodes. Again if we plan to have external users using our system such a feature is a necessity.

## V. CONCLUSION

We are able to very effectively classify email contacts as friends vs acquaintances based exclusively on a bag-of-words model of the communicated words. This method can potentially be used to improve email or mobile clients by predicting relationships with contacts. There is great promise for future work that incorporates different communication mediums, message meta-data, and larger data sets. The greatest challenges are dealing with the noisy data (preprocessing, alias creation), time required for labeling, and the difficult gray area between friends and acquaintances. We plan to extend this work into an open-source tool that will allow users to build a graph of their personal network and visualize various aspects of it to gain useful insights.

## VI. REFERENCES

[1] Jones E, Oliphant E, Peterson P, et al. SciPy: Open Source Scientific Tools for Python, 2001

[2] Scikitlearn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 28252830, 2011.

[3] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification Journal of Machine Learning Research 9(2008), 1871-1874.