The Email Analysis Framework: Aiding the Analysis of Personal Natural Language Texts

Faisal Alquaddoomi
UCLA Comp. Science Dept.
4732 Boelter Hall
Los Angeles, CA, USA
faisal@cs.ucla.edu

Cameron Ketcham
Cornell NYC Tech
111 8th Avenue #302
New York, NY 10011
cketcham@cornell.edu

Deborah Estrin
Cornell NYC Tech
111 8th Avenue #302
New York, NY 10011
destrin@cs.cornell.edu

ABSTRACT

Free-form email text streams are a rich, yet seldom-tapped, source of information about an individual's internal state. The difficulty in using this source of information is due partially to issues with obtaining and parsing these streams, and the sensitivity of the personal data they may contain.

This work presents a framework for allowing a user to authorize the acquisition and processing of emails from their Gmail account in order to model the user's use of language. The framework exposes a RESTful HTTPS API for third-party apps to produce personal analytics for the user from their language model, over which the user maintains fine-grained control by selectively granting access via OAuth2. Candidate applications that consume the language models are discussed, including how they may derive personal analytics from the provided data.

Categories and Subject Descriptors

I.2.7 [**Natural Language Processing**]: Text analysis

General Terms

Design

1. INTRODUCTION

As we interact with the world, we produce a profusion of data across different modalities. Of particular interest is the data we produce in communicating with other human beings, which could if collected and analyzed provide insight into our relationships with others as well our own internal state. This data often takes the form of free text which by its nature is qualitative, and thus challenging to analyze with quantitative methods. It is also frequently strewn across various services. Some of these services expose the data for public consumption, as in the case of social networking sites like Twitter, Facebook, or posts on personal blogs. Other services are more private, such as email and text messaging, and special care must be taken to gain access to the data as well as to preserve its privacy.

To summarize, the primary concerns are to securely collect, integrate, and analyze this often sensitive qualitative data. This paper proposes the implementation of a framework, the "Email Analysis Framework" (EAF), that consumes a user's sent email and produces a set of quantitative models and statistics informed by the field of natural language processing. While the scope of the project is currently to collect and process email, the intent is to expand the framework to collect and integrate other sources of free text, for instance from social networking sites. It is hoped that the EAF will be used as a proxy for these qualitative data sources, providing a foundation upon which user-facing tools can be built to derive insights about this data for the individual in a privacy-preserving way.

The EAF is currently under active development, but an alpha version of the tool is available[1], as is the source code [2]. This paper principally describes the structure and design choices in acquiring, analyzing, and disbursing sensitive data. Applications are discussed in 4, which currently consist of a completed sample EAF consumer that produces trivial visualizations as well as two more significant applications that are currently in development.

2. APPROACH AND RELATED WORK

As described in [13], the overarching intent of quantifying the self is to collect, integrate, and analyze data streams that may be indicative of an individual's physical, emotional, and psychological state. The purpose of this analysis is to promote awareness of how these measurable quantities both affect and can be affected by the individual's state, and to provide support for decisions that change that state. As mentioned previously, free text is both relatively easy to collect and clearly carries much information about how we feel about others and ourselves; indeed, it has been demonstrated that even our choices of words reflect our psychological state [11]. While this data may be present, it is in an opaque form that must be parsed into usable quantitative data.

The analysis of free text has been extensively addressed in the field of natural language processing (NLP). NLP con-

[1] `https://eaf.smalldata.io`
[2] `https://github.com/falquaddoomi/social_text_processor/`

cerns itself with the broad task of comprehending (that is, unambiguously parsing) and extracting structured information from human language, which is accomplished through two main approaches: rule-based (aka grammatical) and statistical methods. The EAF primarily makes use of these statistical methods, specifically n-gram language modeling, to build a sequence of generative models of an individual's use of language over time.

n-gram models are sufficiently descriptive of an individual's use of language that they can be used to discriminate one author from another purely by comparing descriptive statistics computed over them, such as the entropy or the perplexity of the distributions [10, 14]. Descriptive statistics, such as the entropy of a language model mentioned previously, are of special appeal to privacy because they provide an essential determination about the author without compromising the original content from which the statistic was derived.

A user's email is a unique corpus in that each document (i.e. email) is tagged with a host of metadata, including the time it was sent. Thus, computing language models over brackets of emails close in time can provide "snapshots" of the evolution of a user's use of language over time. These snapshots can be compared against each other to determine if there are shifts in the style of the user's written communications which could perhaps correlate to life events. There may be regularities in the changes of these models, or similarities to other people's models with whom the individual interacts. The snapshots can be filtered by recipient or by communication mode to determine if the audience or medium determines the way an individual writes, or if there are detectable groupings. Many more examples could be proposed for these temporal language models, especially when other sources of time-based data (location, activity, calendar events, etc.) are introduced. One of the EAF's main goals is to provide infrastructure to build and maintain these models, as well as allow them, and the descriptive statistics derived from them, to be released at the user's discretion for comparison to other data sources.

There are other frameworks which provide similar analytical capabilities, notably the General Architecture for Text Engineering (GATE) [2]. There are also numerous libraries and toolkits [3, 6] that include the same features that the EAF provides – in fact, the EAF makes use of the popular nltk library [1] to perform many of its functions. The EAF differs from these projects in its context: it is a deployable system focused on centralizing the secure acquisition and processing of emails for many users. It provides user-facing administrative interfaces to control it, and app-facing APIs to make use of its results. The EAF's intent is to allow users to make sense of their own data, and uses a fine-grained opt-in permission system fully controlled by the user to help protect against malicious or unintended use of the user's email data.

In the context of email analysis, the MIT Media Lab's Immersion project[7] shares the EAF's goal of using one's email for the purpose of personal insight and self-reflection. Unlike the EAF, the Immersion project restricts itself to analysis of the user's social group through reading the "From" and "To" fields of email header – no examination of the body text is performed. Further, the output of the Immersion project is an infographic and not raw data that can be reused by other components, whereas the EAF's purpose is to facilitate analysis by other tools.

3. ARCHITECTURE

The EAF's first task is to transform a user's sent email messages into a series of tokens, where each token is tagged with the time at which it was sent. This series of time-tagged tokens constitutes a "stream", from which the n-gram models mentioned previously are built. The stream is quantized into intervals; the ordering of tokens within these intervals is not preserved from their originating messages (outside of their order in the n-grams), with the express intention of making it difficult to reconstruct the original text. After quantization, the stream is then made available at the user's discretion to third-party applications ("consumers"), with the ability for the user to configure per-consumer filters that control what information that consumer can access. A few candidate consumers are discussed in the "Applications" section 4. In order to mitigate the danger of storing sensitive user credentials, the EAF makes extensive use of the OAuth2[4] standard, both as an OAuth2 consumer (of Gmail, currently) and as an OAuth2 provider. The use of OAuth2 also allows the user the freedom of revoking access to the EAF should they wish to discontinue its use, or to revoke access to third-party apps that had been authorized to consume the EAF's API. After the initial synchronization, future emails that the user sends are automatically acquired by the system by periodically polling the provider.

3.1 Structure

The EAF consists of three main components, as depicted in figure 1: a web interface through which the user authorizes access to their Gmail account and performs administrative tasks, a task processor which acquires the user's email and produces a token stream from it, and a second web interface which faces consumers of the token stream. Both web interfaces are implemented in Django 1.7, a framework for rapid development of web applications in Python. Authorization to third-party services is facilitated by Django-Allauth, a project that allows Django's built-in authentication system to interoperate with a host of OAuth2 providers, including Gmail. The task processor makes use of Celery, a distributed task queue processor that is often used in concert with Django. Both components communicate via a shared database, specifically PostgreSQL, which was chosen for its performance under heavy, highly concurrent loads.

The framework exposes a RESTful HTTPS interface to allow third-party applications to consume the token stream. The implementation of this interface was aided by the Django-REST-framework, and the specifications of the interface follow the openmHealth DSU specification v1.0, [9]. The user-facing web interface makes use of the RESTful interface itself for querying the token stream. In order to allow registered third-party sites to gain access to the user's email data for analysis and visualization, the EAF acts as an OAuth2 provider; third-party sites must involve the user in their request for a temporary access token, which they can subsequently use to make requests on the user's behalf.

3.2 User Interaction

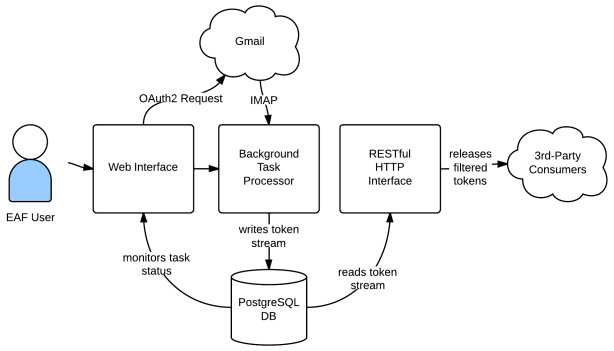


Figure 1: Structure of the Email Analysis Framework

Prior to using the system the user first creates an EAF site account which acts as an aggregation point for the multiple email accounts they might want to use. At the moment this account creation is performed automatically when the user authorizes their first email account; the account they authorize (or any other linked account) then implicitly logs them in to their site account, although this behavior is subject to change in the future.

In their interaction with the system, the user proceeds through three stages:

1. **Authorization**, in which the user is prompted to release temporary credentials used to access their email account via OAuth2.
2. **Acquisition**, during which the user monitors the progress of the system as it downloads their emails and performs filtering/transformations before inserting them into the database as a stream of tokens.
3. **Release**, in which the user selects which consumers can access their token stream and what filtering/transformations will be applied for that consumer.

3.2.1 Authorization

The authorization stage is initiated when the user visits the web interface. Using a standard OAuth2 handshake, the user is redirected to Google's Gmail authorization page, where they log in (or use a previously logged-in session) and then accept the permissions which the framework requests, specifically access to the user's email. If the user provides their consent, they are returned to the EAF where they can proceed to acquisition. If the user does not provide consent or some other error occurs, they are returned to the framework with an error message and are prompted to try again. Multiple email accounts can be associated with a single EAF site account, in which case selecting an account from the list of registered accounts begins the next stage, acquisition.

3.2.2 Acquisition

Initial Acquisition. Acquisition starts immediately after authorization and is handled by the background task processor. The user is shown a view of the task's progress which

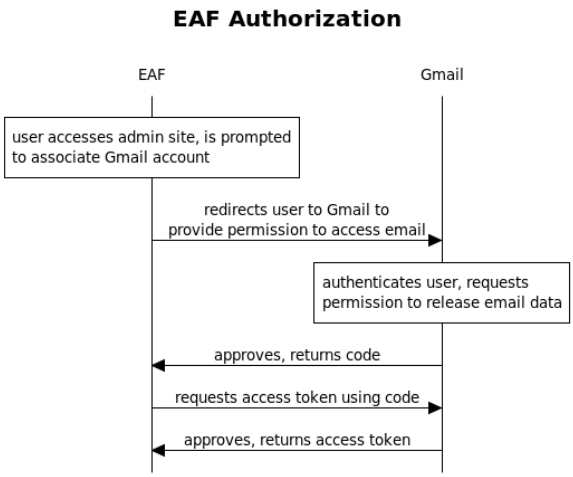


Figure 2: Gmail Authorization

is periodically updated. The process can be quite lengthy, especially in the case where there is a large backlog of messages to process, so the user is permitted to close the view and return to the site at their convenience to check in on the task's progress. Upon completion, the framework sends a notification email which includes the total duration of the acquisition task. At this point, the user can view the results of the acquisition process in a small built-in visualization dashboard that shows a few summarizing statistics about their token stream plotted over time. Incremental acquisition tasks that occur after the initial acquisition do not trigger a notification.

Since the framework is intended to model the user's use of language and not the language of other individuals with whom the user is conversing, it is necessary to strip quotations and reply text from the emails prior to processing. Isolating only the user's text in sent mail is accomplished through an adapted version of the email_reply_parser [3] library, developed by GitHub.

Ongoing Acquisition. In the background task processor, the acquisition task consists of using the previously-obtained OAuth2 credentials to authenticate to Google's IMAP server. The task then proceeds to download the user's sent email (that is, the contents of "GMail\[Sent Mail]") in chronological order, skipping messages which have been recorded as processed in a previous iteration of the task. Each email is passed through a series of filters, called the "pre-filter chain", which ultimately results in a sequence of tokens that are associated with the email account, the user's EAF site account, and the time at which the email was sent. By default, the first filter in the chain performs tokenization: each email is split on newlines and punctuation into tokens, which are converted to lowercase to reduce the number of possible tokens due to capitalization differences, and stripped of numbers and quotation marks. The second filter is the "ignored words" filter, which allows the user to selectively prohibit certain words from ever entering the database. At the mo-

[3] `https://github.com/github/email_reply_parser`

ment, the ignored words must be manually entered, which makes filtering passwords and other sensitive information problematic, given that the ignored list itself is then sensitive. This will be addressed in the subsection on filter types, 3.3.

After the filter chain runs, the tokens are then written to the database. Rather than store repeated tokens individually, each token is stored with a count of the number of times it occurred within its message. If same token occurs in different messages, it is stored separately for each message. This choice was made as a compromise between allowing for flexible choice of the interval into which tokens are combined when the stream is consumed and consuming less space in the database; if the system were designed with a fixed interval rather than a flexible one, the tokens would simply be combined into a histogram for each interval.

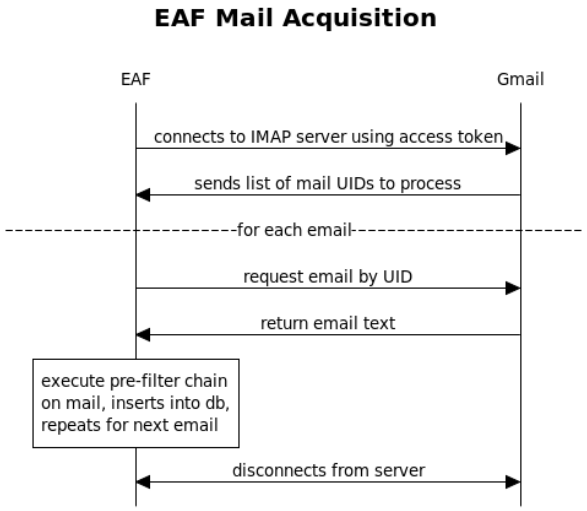


Figure 3: Mail Acquisition

3.2.3 Release

Once the user has found an EAF-compatible application, they can authorize that application to access their token stream via OAuth2. In this stage the EAF acts as an OAuth2 provider, providing a page to which the third-party application can redirect the user to be prompted for authorization of their identity via Gmail (used also as credentials to access their EAF data) and permission to access their token stream. In the case where the user has multiple token streams, they will be prompted to choose the streams to which they are granting access. On this page, the user selects a filter chain for each stream that will be used specifically with this consumer, or simply opt not to filter the stream at all. The process is detailed in figure 4.

After this point, the consumer can request updates from the token stream at any time. The EAF audits all accesses, displays the last time of access, and allows the user to revoke the consumer's access at any time or change the filter chain associated with that consumer.

3.3 Filtering

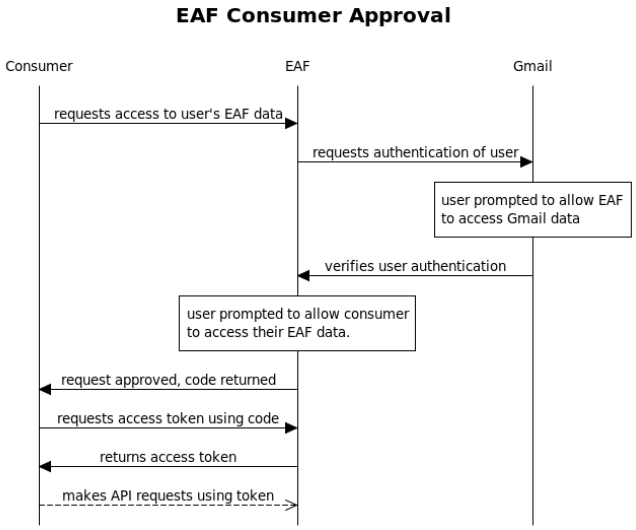


Figure 4: Release to Consumer

As previously mentioned, both the acquisition and release stages employ a sequence of filters that allow the input data to be selectively screened for sensitive terms and otherwise transformed to better suit the requirements of the consumers. The acquisition stage's filter chain is referred to as the "pre-filter chain" and the release stage's is the "post-filter chain". There is only a single pre-filter chain, but there can be as many as one post-filter chain for each registered consumer.

The pre-filter chain always has a special "tokenize" filter as its first filter, which produces the initial sequence of tokens for filtering and transformation, and may only be used in the pre-filter chain. A second special filter that may only be used in the pre-filtering step is the "ignore word sequence" filter, which ignores the sequence of tokens configured in the filter, and was initially created to ignore signature blocks. This filter can only function in the pre-filtering step as the exact sequence of the tokens is lost upon insertion into the database.

Aside from the special "tokenize" filter, there are a few other filters which can only be used in the pre-filtering step, namely:

- **Parts-of-Speech Tagger**, which replaces each token with its detected part of speech (noun, verb, etc.)
- **Fork**, which produces an identical stream to the current one, but with its own sub-filter chain. The tokens that are produced from a fork are tagged with a unique ID corresponding to that fork.

The "fork" filter is especially useful in conjunction with the part-of-speech tagger, as both the original text and the parts-of-speech stream can be either individually released or released together, which allows for analysis of the user's grammar. Note that the parts-of-speech stream does preserve the order of tokens in the original stream, but not the text of the tokens themselves.

The filter framework is modular, with the potential to add new filters easily in the future. At the moment, a few parameterizable filters are implemented to change the case of tokens, strip specific characters, and to remove words that are either on a user-specified list or not contained within aspell's "en_US" dictionary. Detecting and ignoring named entities is a work in progress.

- **Change Case**, which transforms the case of the tokens;
- **Strip Characters**, which can be used to remove numbers and other special characters;
- **Ignore Listed Words**, which removes tokens on an "ignore" list from the token stream; and
- **Ignore Non-Dictionary Words**, which removes tokens not found in a common English dictionary

By utilizing the "ignore words" filters, the user is allowed fine-grained control of both the contents of the EAF's database and the views of the token streams presented to different consumers.

4. APPLICATIONS

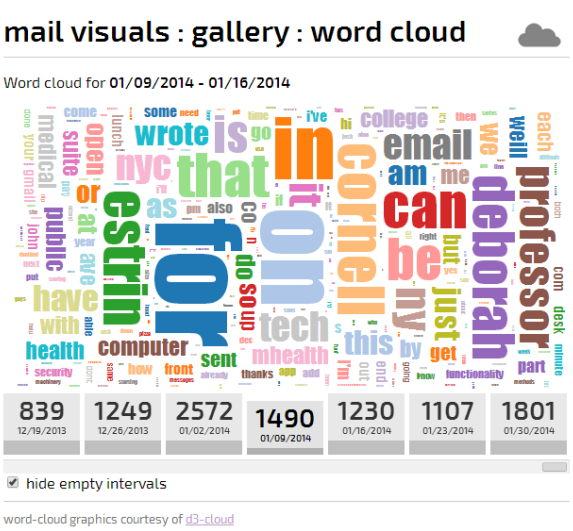
As mentioned, third-party applications can gain temporary access to a user's data for the purpose of visualizing or otherwise processing it. Granting this access is currently at the user's discretion; the user should make use of the per-consumer post-filter controls to limit the release of sensitive information to potentially untrustworthy parties. Consumers sign requests to the EAF's RESTful JSON API with an access token, obtained through the process described in the "Release" section above 3.2.3.

4.1 Example: Mail Visualization Front-End

In order to demonstrate the functionality of the framework, a visualization front-end was developed that consumes the framework's API and produces a few example visualizations. The front-end also serves as a reference implementation of EAF client authentication; it first requests permission from the user before gaining access to their token stream. The visualization front-end currently offers the following modules:

- **Word Cloud** - a "word cloud" infographic that can be viewed on a per-week basis (figure 5).
- **Rhythm** - a table of the days of the week as the columns and hours of the day as the rows is colored according to the number of emails sent within each interval, with darker colors corresponding to more emails sent (a heatmap, essentially; figure 6).
- **Alters** - a bar chart of the number of people contacted per week; when a week is selected, displays a bar chart of emails sent to each person.

These visualization modules are intended to be a jumping-off point for more useful visualizations to be constructed, which would ideally incorporate data from other sources to strengthen inferences about the user's overall state.



t]

Figure 5: "Word Cloud" Visualization

4.2 Pulse and Partner

In addition to the sample application discussed above, our group is currently developing two applications that make use of statistics computed against the user's email. The first is "Pulse", which makes use of location traces from the user's smartphone as well as the frequency and variety of individuals with whom one communicates to compute a score that indicates **how** rather than **what** the individual is doing. This score is visualized as a waveform over a short window of time (i.e. a week), which can be shared with family members and friends. The second is "Partner", which is intended to measure the degree to which linguistic style matching occurs among individuals who interact with each other face to face, a fairly well-documented phenomenon [8], [5]. Partner makes use of the location traces of two or more individuals as well as computed statistics over their emails to produce two scores, a "proximity" and a "language-style matching" score, which will be visualized as individual timeseries. A third timeseries will display their computed correlation over time.

5. CONCLUSION, FUTURE WORK

The Email Analysis Framework, a system for extracting structured, easily-consumed data from time-tagged natural-language text was proposed in this work. At the moment it is limited to acquiring text from Gmail, computing, and exposing language models to other tools via a RESTful HTTPS API, but it is hoped to be extended to other sources of personal natural-language text, such as Facebook and Twitter streams. A few candidate visualizations were described to both demonstrate how the data could be used and to stimulate investigation into more novel applications.

In terms of future work, there are extensions planned to all the stages of the framework. As mentioned, the scope of providers is intended to be expanded to other text providers, which will allow analysis to be performed on how different media affect the language model. Additional streams can be extracted in the processing phase, such as identifying

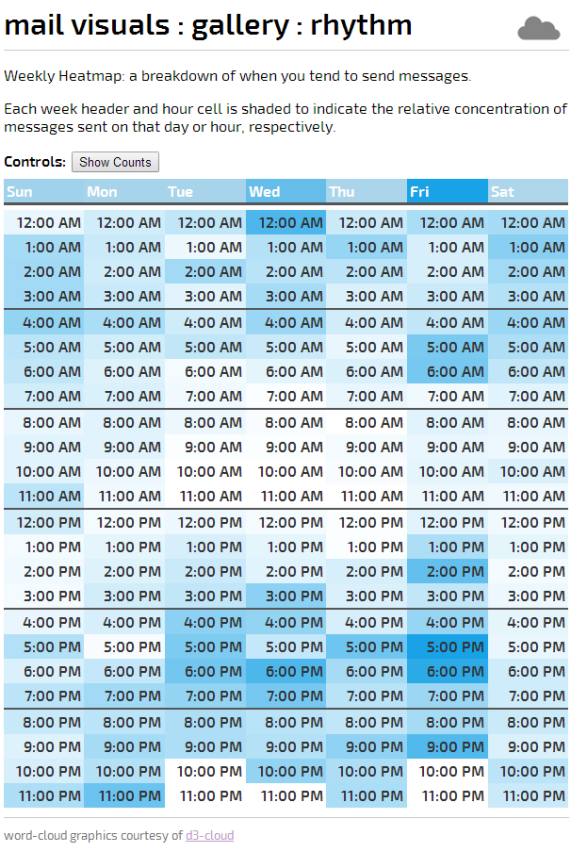


Figure 6: "Rhythm" Visualization

named entities and topics, all of which can be analyzed over time, audience, etc. Industry-standard information extraction techniques such as autoslog [12] could be applied to discover meeting arrangements, events that occur to named entities or topics mentioned in the emails, and so on. Sentiment analysis could be computed and exposed as another temporal stream, to attempt to model the user's disposition as a function of time. Additional third-party applications are planned, such as a tool for determining points of inflection in the descriptive statistics computed on the language model, and a tool to easily correlate other time-based data against the statistics streams.

6. REFERENCES

[1] S. Bird. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pages 69–72. Association for Computational Linguistics, 2006.

[2] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. Gate: an architecture for development of robust hlt applications. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 168–175. Association for Computational Linguistics, 2002.

[3] D. Ferrucci and A. Lally. Uima: an architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering*, 10(3-4):327–348, 2004.

[4] D. Hammer-Lahav and D. Hardt. The oauth2. 0 authorization protocol. 2011. Technical report, IETF Internet Draft, 2011.

[5] M. E. Ireland, R. B. Slatcher, P. W. Eastwick, L. E. Scissors, E. J. Finkel, and J. W. Pennebaker. Language style matching predicts relationship initiation and stability. *Psychological Science*, 22(1):39–44, 2011.

[6] A. Mallet. Java-based packed for statistical nlp toolkit. *Available at (accessed 26.01. 10)*, 2010.

[7] Mit media lab's immersion project. `https://immersion.media.mit.edu/`. Accessed: 2014-02-04.

[8] K. G. Niederhoffer and J. W. Pennebaker. Linguistic style matching in social interaction. *Journal of Language and Social Psychology*, 21(4):337–360, 2002.

[9] Ohmage dsu 1.0 specification. `https://github.com/openmhealth/developer/wiki/DSU-1.0-Specification`. Accessed: 2014-02-04.

[10] F. Peng, D. Schuurmans, V. Keselj, and S. Wang. Automated authorship attribution with character level language models. In *10th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2003)*, pages 267–274, 2003.

[11] J. W. Pennebaker, M. R. Mehl, and K. G. Niederhoffer. Psychological aspects of natural language use: Our words, our selves. *Annual review of psychology*, 54(1):547–577, 2003.

[12] E. Riloff and W. Phillips. An introduction to the sundance and autoslog systems. Technical report, Technical Report UUCS-04-015, School of Computing, University of Utah, 2004.

[13] M. Swan. The quantified self: Fundamental disruption in big data science and biological discovery. *Big Data*, 1(2):85–99, 2013.

[14] Y. Zhao, J. Zobel, and P. Vines. Using relative entropy for authorship attribution. In *Information Retrieval Technology*, pages 92–105. Springer, 2006.