Small Text Data Analysis through Application of User and Group Modeling

Faisal Alquaddoomi

October 2015

Abstract

Small data are digital traces that we generate from interacting with the world, and is an active area of exploration in modeling individual and group behavior. Small text data is a form of computer-mediated communication that we exchange with others, but with a particular focus on the individual and how that text relates to their other traces. We consider conversational text in order to model aspects of the user's state, their groups' states, and the dynamic relationship between the two. In addition to modeling tasks, the work addresses scaling and logistical issues in processing large amounts of natural-language text; the application of research in distributed/optimized learning and model representation is discussed as a means to improve throughput. Finally, we discuss the issue of sparsity in certain modes of conversation, such as email and text messages, and discuss the potential for transfer learning from large, similar corpora to improve inference on smaller, yet important datasets.

1 Introduction

Small data – the data which we generate as a byproduct of our daily digital interactions with people, services, and the world in general – have been identified as an important source of insights about ourselves (Estrin [2014]). Typical sources of small data include location traces, financial transaction records, biometrics from fitness and sleep trackers, and browser/app logs. These data, when considered in full, paint a comprehensive picture of the individual which could be used for personal reflection, goal-setting, or prediction of trends and future behavior.

A specific subset of small data is *small text data*, the set of natural-language text that we exchange through various modalities, e.g. social media, email, text messaging, or online forums. This text belongs to a larger family of *computer-mediated communication* ("CMC"), which includes all informal text that people post or exchange online.

Where other small data allows inference about a user's physical health, interests, and habits, small text data allows us to model how individuals relate to each other and how these relationships create and support groups. We can learn about the character of these groups, for instance what effect on a user's latent state membership in the group induces, and how the aggregate state of the individuals characterizes the group, as well as more superficial characteristics such as the topic of the group. Being able to characterize both users and

groups can aid in such tasks as identifying sub-communities of users present in larger forums, which can facilitate administrating these large communities. Identifying language correlation structures between groups and users can help in peer discovery, and considering these changes in language correlation over time can predict a user's trajectory through membership in various communities as well as larger migrations of groups of users through communities, and the lasting impacts on communities that these migrations may cause. Once discovered in large corpora, the models that identify these correlation structures can be generalized to other, sparser modes of individual-specific communication such as text messaging and email.

To achieve this vision, this work consists of three emphases:

1. Modeling users and the groups to which they belong, both in terms of their language model and latent state, and examining the effects on these models from interactions between the two. This is accomplished primarily through language modeling, but potentially also chronemics (temporal effects in human communication) and other metadata present in these exchanges.
2. What infrastructure is necessary to perform this analysis at scale. This need manifests in batch processing large corpora, where distributing the load significantly improves turn-around time for experiments, and in continuously processing an individual's small text data in an interactive fashion that enables both human-in-the-loop learning and user-facing applications of learned hypotheses.
3. What methods can be applied to circumvent the sparsity of an individual's data through transfer learning from large public corpora and multitask learning across other individuals' data.

These emphases and their applications are described in more detail in the following.

NLP for user modeling and vice versa As mentioned, small text data can provide insight into the factors that influence the text we generate. Whether it be our interests, our relationships with others, our personal/projected image, or latent cognitive state, there are myriad modeling targets; the systematic analysis of natural language can provide data for training these models. Once trained, these models can be used to, for instance, predict the evolution of a personal relationship, reveal differences in how we present ourselves in different communities, or identify factors that affect our sociability with others. The context in which individuals communicates also provides information about that context: for instance, certain fora or peer groups might engender a style of conversation from their participants that characterizes the group.

Continuous Acquisition, Application, and Learning One of the more intriguing properties of this text data is that it accumulates over time – individuals are constantly interacting with services to produce text that becomes part of their ongoing record. This record can provide a broad longitudinal view of the user's state over time across all the records to which they have access. Most services retain these histories and provide mechanisms to access them, enabling both retrospective and ongoing learning tasks. Since the user is required

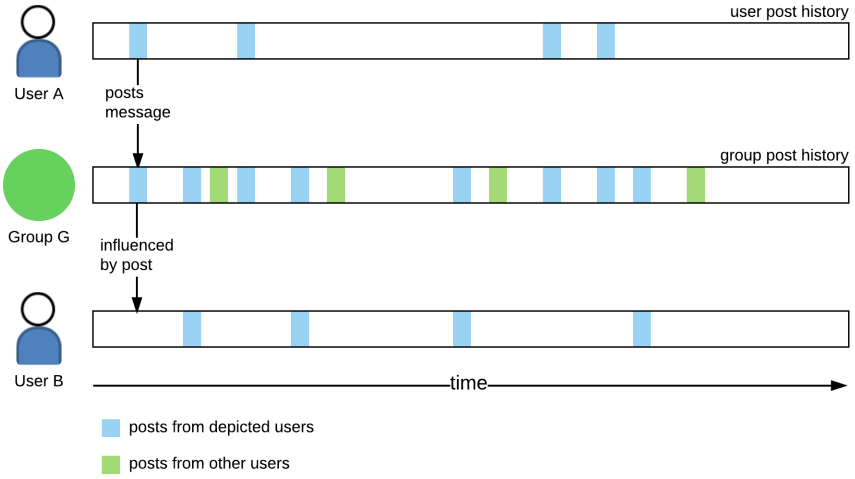


Figure 1: Timeline of interaction effects between sample users and a group. Users are effectively text generation processes with memory that are correlated with each other in time via their groups (as well as external, un-modeled effects.)

to provide this access, they can be harnessed as a resource in human-in-the-loop learning to label and interpret their own data. Continuous processing also allows these models to be applied by the user over their data for their own purposes; in fact, providing continuing value to the user is essential to keeping them in the loop. The problem of *concept drift* (Tsymbal [2004]), the gradual shift in the meaning of topics as further discourse clarifies and expands their definitions, further stresses the need to continuously keep models up-to-date with new information and selectively decay the influence of older data.

Sparsity and Transfer Learning However, we face several challenges in creating useful NLP models for small text data. The most significant problem in analyzing an individual's small text data is sparsity: individuals on their own simply do not produce a sufficiently large volume of text to robustly train predictive models. Secondly, many of the tasks that we would like to perform – sentiment analysis, entity-relationship detection, etc. – are still topics of active research. Many standard NLP tasks such as parts-of-speech tagging and named-entity recognition require large amounts of data from a corpus similar to the target domain; most off-the-shelf classifiers are trained on articles or news-wire, not conversational text. It is therefore of interest to be able to perform learning on large, public corpora such as forum and social media posts that are similar enough to the target domain of personal conversational text.

Once learning has been accomplished on these large corpora, there is potential to perform *transfer learning* to the target domain of an individual's text. When transfer learning is appropriate, what should be transferred, and how it is to be transferred, are still open questions that vary depending on the domain and task at hand (Pan and Yang [2010]). There is also the possibility of performing *multi-task learning* on a single user's text, or across multiple users for the same

task.

Tasks and datasets The relationship between users and groups will be explored through a number of experiments on user and group models and how they evolve with respect to each other. The recently-released Reddit corpus (the Cow [2015]) is particularly well-suited to serve as the large corpus for this exploration; it consists of approximately 1TB of conversational text data, and is a collection of all comments posted to the link-sharing service from August 2007 to October 2015. Each comment is tagged with the user who posted it and the forum ("subreddit") to which it was posted. Many users have long-term presence on the site, with comment histories spanning the entire archive.

The experiments are described in detail in section 4, and generally consist of building and comparing these models over time.

Supporting platform In order to support this research and to answer future questions, this work proposes a modular processing pipeline which centralizes the secure acquisition and processing of text data. The subsequent stages of the pipeline are modular in the sense that they consist of discrete processing units with well-defined input and output types, which can be easily composed into custom pipelines. The system can be extended by implementing new processing units, and managing non-modeling details such as how and when they are executed is left to the framework. Generally, pipelines can be configured to run in batch mode on a specific set of input data, or be attached to live data sources (e.g. one's email account or SMS messages) and produce output continuously. This pipeline has been partially realized in previous work on the Email Analysis Framework (the EAF) – specifically, the secure acquisition, processing, and continuous operation portions – but the modular subunit composition, batched mode, and generalization to non-email data sources is still future work. The prior work, as well as the new modular pipeline, are discussed in section 5.

2 Background

Central to statistical NLP is the concept of a **language model**, an assignment of probability over sequences of words, e.g. sentences. This can alternatively be phrased (as in Shannon's original definition) as a sequence-prediction problem in which the next word is conditioned on all of the prior words in the sequence. Formally, for a sequence of length m, the probability of the sequence is $P(w_1, \ldots, w_m) = \prod_{i=1}^{m} P(w_i \mid w_1, \ldots, w_{i-1})$.

Granted that the space of possible word combinations is vast, that the space of words itself is not closed, and thus that most possible sentences have not been observed, it is extremely difficult to address the problem of modeling sentence probabilities directly. Instead, a common approximation is to compute the probabilities of *n-grams*, sets of sequential words of length n, and then reconstruct the probabilities of longer sequences by combining their pairwise probabilities. This comes at a cost to accuracy – it is possible to assign high probability to sequences that consist of highly-probably sub-sequences, but are together not probable. This method also does not take into account any notion of semantic meaning. Finally, some amount of probability mass must be allocated to sequences that were unobserved during training to avoid erroneously

assigning them (and thus the entire sequence) zero probability. This process is known as *smoothing*; a variety of approaches for it exist, each with its own set of assumptions, although Kneser-Ney smoothing is generally accepted to be the state-of-the-art – albeit expensive – approach.

Comparison of language models In this work, language modeling is used extensively to compare users and groups to each other as captured by their language models. Le and Ho [2005] propose the use of the *Kullback-Leibler* (KL) divergence from information theory to determine the similarity of two categorical distributions to each other. The KL divergence, denoted $D_{KL}(P||Q)$, measures how many more bits are required in expectation to encode a sample from Q, the comparison distribution, than when using the "true" distribution P. While the strict formulation is non-symmetric (and thus is not a well-formed distance metric), there exists a commonly-used symmetric form, $D_{KL}(P||Q) + D_{KL}(Q||P)$, which can be used as a metric. The KL divergence is used in this work to compute the pairwise similarity of language models, both with respect to an out-of-band factor predicted to induce similarity (e.g. shared membership of users between two groups) and to instances of the same user or group at various points in time to detect divergence based on an impulse event (for instance, a user's change in membership within a group or vice versa.)

Choices in defining a token The purpose of the language model also influences what one considers a token in the system. For instance, information retrieval systems typically strip punctuation, normalize case, and often perform 'stemming' of words to their root forms in order to increase the overlap between queries and document contents. In contrast, named-entity recognition tasks often retain capitalization and punctuation for detecting proper nouns or currency denominations. Since our objective is to characterize language similarity, and because CMC is notoriously prone to neologisms, intentional misspellings, and other creative uses of typed language, we do not collapse the token space. Punctuation specifically is of interest; the "CMC cues" described in Kalman and Gergle [2010] may be of use in capturing *how* people speak versus *what* they are saying. On a more philosophical note, Eisenstein [2013] makes an interesting point that normalization presupposes the existence of an identifiable norm, which may not be the case in the varied and ever-changing world of online discourse. Unsupervised methods that make a mininum of assumptions about the domain may be more appropriate and accurate.

NLP tasks for enriching the language model In addition to the n-gram language modeling approach discussed above, there are a number of sequence- and feature-based tasks that form a basis for many higher-level NLP tasks. **Part-of-speech Tagging** is a sequence-based method that annotates each token in a sequence with its grammatical part of speech (noun, verb, adjective, etc.) Building upon this, statistical **Named-Entity Recognition** (NER) is a feature-based method that attempts to determine which tokens correspond to proper nouns or fixed-format quantities such as dates or currency amounts. PoS tagging and NER, in addition to many other tagging methods, can be used in conjunction with n-gram modeling by including the tags as annotations to the entries in the language model.

Feature design for modeling other latent state Finally, *feature representation* is important when modeling the latent state of users and groups. A "good" feature set is one that allows sufficient separation of classes in classification tasks, or discovering a function with a low-error fit in regression tasks. In both cases, the feature representation should not be overly specific to the training data, so that unseen future examples are still likely to conform to the trend that was determined during training. Features can either be hand-crafted from knowledge about the domain (e.g. capitalization is known to be important to distinguish proper nouns in English, ergo it is a good feature for a named-entity recognizer), they can be selected from a larger pool of features using a feature selection/dimensionality reduction framework such as ElasticNet (Zou and Hastie [2005]) or Principle Component Analysis, or they can be learned from the input for a given task and dataset. The latter approach has become popular in deep learning architectures, multilayer neural network architectures, in which intermediate layers encode a feature representation that is successively adjusted via back-propagation while minimizing the error of the input data. Deep neural networks have been applied with success to NLP tasks such as machine translation with great success, and may be of some use in this work when attempting to correlate language models with user and group state.

3 Prior and Related Work

Related work in the area covers three foci: 1) instances of NLP applied to social media, which corresponds to the theme of studying user/group dynamics, 2) what approaches exist to continuously acquire and learn from text data, corresponding to our emphasis on human-in-the-loop learning over personal text, and 3) what techniques and features promote transfer learning across different corpora.

3.1 NLP and Group Dynamics Modeling

NLP and Author/Topic Clustering Addressing the problem of community/topic detection specifically, McCallum et al. [2005] propose the Author-Recipient-Topic model, which allows the discovery of topic models on a per-author/recipient pair basis. Similar work by Zhou et al. [2006] performs author/topic modeling on a coarser scale, determining communities of authors by collaboration in semantically-similar topic groups in the Enron email dataset. Liu et al. [2009] provides an interesting alternate formulation of joint author-topic relationship learning, using shared citations as the underlying peering relationship between authors that connects topics, and documents from unconnected authors on the same topic as suggesting peers.

Our context in this work is slightly dissimilar, since we are already aware of the groups and our users' membership in them, but the problems are similar in that they both seek to determine the nature of the relationship between users and their respective groups.

Inferring Latent State via NLP There is a large body of work in applying NLP to social media to uncover psychological and social dynamics. Danescu-Niculescu-Mizil et al. [2012] investigated how linguistic coordination in conversa-

tions carried out on Facebook can indicate social power structures. Coppersmith et al. [2015] correlated language used on twitter and a variety of conditions, such as ADHD and seasonal affective disorder, by using self-reporting individuals as a labeled training set. In a similar study, Eichstaedt et al. [2015] employed tweet content to predict heart disease mortality on a per-county basis. The gist of these studies is that by aggregating a large amount of personal conversation, albeit produced by unique individuals, one can determine global effects.

While the present work involves discovering effects specifically on the language models of users and groups, and not correlating language to latent state as the studies above describe, these models of latent state could also be explored as affecting text generation both in the individual case and in aggregate within a social group.

3.2 Pipeline Processing of User Data

Our proposed system architecture is broadly inspired by the *flow-based programming* paradigm (Morrison [2010]), in which computation is characterized by the flow of data rather than the flow of control. Other projects, notably Apache Storm (Storm [2014]) address the issue of reliably processing streaming data at scale. Our requirements differ from theirs in that the volume of our data is relatively low and only needs to be processed continuously, not in real-time. Instead, we endorse the flexible creation, integration, and execution of components, which is difficult to accomplish in a statically defined high-performance processing topology.

Distributed machine learning Distributing machine learning algorithms is also an open research question. Some methods, such as "bagging" methods that linearly aggregate multiple weak classifiers, are embarrassingly parallelizable, and cross-validation methodology that performs training and validation on the entire dataset with different parameters are similarly straightforward to distribute. In some cases it is unclear how to partition the input and aggregate the resulting sub-classifiers, but progress has been made on specific popular algorithms. For instance, Forero et al. [2010] have developed a distributed version of the popular support-vector machine (SVM) algorithm which assumes limited exchange between nodes and produces a high-quality final classifier through consensus across the sub-classifiers. An approach known as "feature hashing", also known as the "hashing trick" in reference to the "kernel trick"), is used to reduce the size of the often immense feature sets used in many NLP tasks. Feature hashing has been demonstrated to improve the speed and memory efficiency of learning to perform spam detection (Attenberg et al. [2009]). This approach will likely improve the efficiency of distributed learning in general and should be considered as part of the pipeline.

Distributed and high-performance language modeling Given that building language models over a large corpus can be time- and resource-intensive, it is advantageous to be able to optimize and/or distribute the task of compiling and querying these models. Brants et al. [2007] describe a high-throughput distributed language modeling architecture that also introduces a new smoothing formulation, so-called *Stupid Backoff* that is empirically tested to be of slightly

lower accuracy than the state-of-the-art Kneser-Ney smoothing, but with much better runtime performance. A complementary line of research by Talbot and Osborne [2007] on efficient n-gram modeling employs a novel formulation of (usually binary) Bloom filters to store frequency information along with n-gram membership information in the model at significant space savings over traditional methods. They extend their work in Levenberg and Osborne [2009] to include online updating from streaming sources of data, and in Levenberg et al. [2011] to include multiple streams in a single architecture for further space savings. Both extensions are of special interest for continuously acquiring and processing streams of small text data.

The proposed architecture will make use of these advancements in distributed learning and language modeling to cope with the large amounts of data involved with batch-processing large corpora. Additionally, these architectural improvements can help scale up to acquiring large numbers of users' small text data, as well as enabling the creation of "live" corpora from Reddit, Twitter, and other sources.

3.3 Transfer/Multitask Learning and NLP

Both transfer and multitask learning are contexts in which discrete learning instances are performed that differ in their target task, domain, hyperparameters, or feature representation, but share some commonality that can potentially improve performance over learning in isolation (Pan and Yang [2010]). Multitask learning is the process of jointly optimizing across multiple learning instances, in which either the task, the feature representation, or the dataset differs. Transfer learning, on the other hand, is the application of learning from a source context to a target context; no back-propagation of information from target to source occurs. Multitask learning has been applied by Collobert and Weston [2008] as a means to improve performance on the usual suite of NLP tasks – part-of-speech tagging, named-entity recognition, semantic annotation, to name a few – by jointly optimizing these tasks on a shared, learned feature representation. Multitask learning has also been applied to learning a shared spam classifier across multiple users' email (as mentioned in the distributed learning section) by Attenberg et al. [2009]; in this case, the task is the same, but each instance expresses idiosyncratic differences in the distribution of its data, but can still be combined to reduce sparsity.

4 Data and Experiments

4.1 Dataset

In this work, the Reddit corpus will be employed as a large dataset that resembles the domain of interest – it is conversational text between individuals, much like email or text messaging is. It differs in that there is a predefined topic of interest for each set of comments, and exchanges occur predominantly between strangers rather than acquaintances. The corpus consists of approximately 1.7 billion comments posted by users to the website reddit.com. In addition to the text itself, each comment includes the following metadata: a username, the time at which it was posted, the subreddit to which it was posted, the comment or

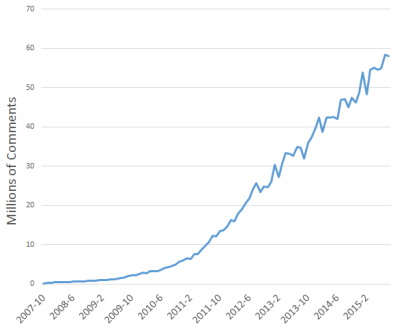


Figure 2: Comments per month from August 2007 to October 2015

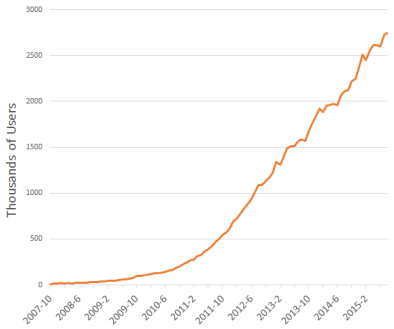


Figure 3: Distinct users per month from August 2007 to October 2015

topic to which it is a reply, the number of up/down votes the comment has received, and its "controversiality" score, a proprietary heuristic. Comments are moderated by other users of the site by awarding upvotes or downvotes, with the display of comments on the site being roughly ordered by score (the sum of upvotes and downvotes). The thread of conversation can be reconstructed from each comment's parent pointer, providing more contextual information.

The trend in number of comments motivates the previous argument for a continuous pipeline that refreshes the corpus with new data pulled from the site. Figure 2 shows the number of comments per month and figure 3 shows the number of distinct authors. Both display a similar pattern of growth, with the number of comments being several order of magnitudes larger than the authors. While the growth is sub-exponential, it still appears to be occurring at an increasing rate. Without a mechanism in place to refresh it, the proportion of data in the corpus will quickly fall behind the current state of the site. The audience of Reddit users for applications that apply the analysis and perform human-in-the-loop learning is likewise increasing.

The experiments described below rely on two key properties of this dataset: the posting history of each user on the site, and the history of comments on each subreddit.

4.2 Experiments

The experiments proposed below will explore how user membership influences a subreddit's language model, and how subreddits influence user language models. Most rely upon language models trained on the text of each user and subreddit; some rely on a series of language models trained on segmented intervals of time in order to determine temporal effects on the model. The computational cost and size of these models motivates distributing the problem, as mentioned in theory in section 3 and applied in section 5.

Correlation of Subreddit Language Model to Userbase The purpose of this experiment will be to determine the degree of correlation between subreddits with similar user-bases and the similarity of their language models. It is hypothesized that subreddits' language models are characterized by the aggregate language models of their users, and thus that subreddits which share users

should also share similar language models. By computing the Kullback-Leibler divergence of each pairwise subreddit, we can populate a matrix of correlations of language models between each subreddit. Similarly, we can use the number of users in common between each pairwise subreddits to simply compute its degree of shared membership (although note that this does not take into account change in membership over time.) We can use a chi-squared test to determine the degree of similarity between these matrices, which will determine whether a shared userbase implies a shared language model.

Note that language model similarity of the subreddits may very well be conflated by the fact that they share a topic, and thus share the same pool of users interested in that topic. It remains to be determined how this "topic effect" on the language model can be removed while still preserving the background "user effect"; focusing on CMC cues might be one means of circumventing this issue.

Userbase-induced Language Model Changes To test the hypothesis that users characterize the style of language in a particular subreddit, we consider subreddits which have undergone a dramatic change in their userbase over a short period of time. The null hypothesis in this case is that the language model should remain unchanged, granted that the subreddit is still focused on the same topic. In this setup, a language model A would be trained on a time period prior to the userbase change, and a model B would be trained on a period after. The difference between these two models (computed using KL divergence) would then be compared to two periods of the same duration during an interval in which the userbase remained relatively constant. If a statistically significant difference was found, it would indicate that a change in the userbase provokes a change in the language model, even when the topic is held constant.

User/subreddit language Correlation Users' and groups' language models ostensibly vary over time. While the previous experiment assessed the effect of a changing userbase on the group's model, it is also of interest to examine the effect of changing group membership on an individual's language model, both overall and with respect to particular subreddits. In this experiment, a few candidate individuals with large amounts of longitudinal data are selected, and their posting history is used to produce a series of temporal language models, each providing a snapshot of a particular point in time. Each user model in the sequence is compared against each subreddit in which they have posts in that duration to determine if their language model becomes more convergent with the subreddit's language model over time. Also, their language models are compared to their prior language models to determine global drift as their membership changes.

It is quite possible that there are effects on a user's language model that are independent of the subreddits to which they belong, although convergence would suggest a causal relationship. Observed differences in the user language model for specific subreddits would provide an interesting launching point for extrapolating this technique to examining differences in language between recipients in, say, email, or more general differences in language between modes such as texting and Facebook posts.

Language model specificity A rough, albeit standard approach to evaluating the performance of one statistical language model versus another is to compute its *perplexity* on a test dataset. The perplexity of a language model is commonly as the average number of bits required to encode a word, and the ratio of two models' perplexities indicates which is more "confident" at choosing the next word in a sequence. We can apply perplexity to a language model trained a user's personal language text against a model first trained against the reddit corpus, then retrained on the user's personal text, to determine if knowledge transfer reduces perplexity.

5 System Architecture

The Email Analysis Framework Prior work explored the problem of securely and privately gaining access to a user's personal data streams, specifically their Gmail account. Appendix B describes work on this system, the *Email Analysis Framework* (EAF), in detail. Briefly, the system exposes a web-based administration interface through which the user provides an OAuth2 grant to access their Gmail (and conceivably other third-party services in the future) and monitors the acquisition and processing of their data. The EAF produces a set of standard NLP models and artifacts, most notably part-of-speech annotated n-gram models computed over the user's total dataset as well as discrete subsets of that dataset bucketed into week-long intervals. These artifacts are exposed through a RESTful HTTP API and protected via an OAuth2 grant, similar Gmail's protection scheme. Third-party applications that wish to use the EAF's output do so by requesting an OAuth2 grant from the user to access the EAF, which the user can customize on a per-application basis. Customizations of the grant include filtering out specific tokens, word classes as determined by PoS tagging, and other metadata associated with emails such as recipients or time.

EAF v2: A Modular, Distributed Processing Pipeline Appendix A proposes a full refactoring of the EAF into a distributed processing system in which processing units communicate with each other over a publish-subscribe bus. This alleviates the EAF's inability to process anything but Gmail data, its inability to easily scale horizontally under increasing load, and reduces the impedance of modifying the EAF by allowing new pipelines to be composed simply by registering new, standalone processing units with the system. Each unit is notified whenever data records of its input type become available, proactively computing and storing the resulting outputs onto a database attached to the bus. Computation continues when either a processing unit becomes available that registers these cached results as an input type, or an application makes use of them for visualization or user interaction.

6 Work Plan

6.1 Expected Chapters

1. Introduction and motivation of small text data

2. Personal Language Model generation techniques
3. End-to-end analysis with a data set/experiments
4. Modular pipeline implementation
5. Conclusions and future work
6. (Appendix) Discussion of software and data repositories

6.2 Timeline

- Fall 2015:
  - initial analysis of the reddit data set under the listed experiments using an ad-hoc analysis pipeline
- Winter 2015:
  - continue work on the publish-subscribe architecture and type library, begin to refactor parts of the EAF as small processing units
  - re-implement the reddit analysis pipeline within the new architecture
  - * explore modeling other aspects of latent user state as a function of group membership (and vice versa) via features derived from other small data sources and manual annotation
    * write an adapter to integrate the reddit corpus into the EAF as an additional set of users and source of data
- Spring 2016:
  - complete implementation of the architecture
  - focus on domain transfer of the reddit models to email and other sparse small text data
  - apply retrained models on this new corpora to attempt to predict known memberships/states
- Summer 2016: wrap up any remaining incomplete tasks

7 Conclusions and Future Work

In this work, we examine the problem of modeling language effects as a result of the dynamic interaction between users and groups over time. We propose a few experiments to identify the presence of these language effects in both users and groups, and discuss the potential to translate these findings from the studied corpus to small text data, which contain the same interesting user-group dynamics but with less supporting data on which to base inference. The architectural sections of the work propose a practical system for applying this research to continuously process both small text data and large "live" corpora, which can serve as a platform for enabling interactive applications and human-in-the-loop learning.

While identifying the existence of language transfer between users and groups is a necessary first step, there is still much work to be done in qualifying how

the language transfer effect occurs and what is being transferred in particular instances of its occurrence. For instance, one could examine a particular user migration (say, from one content platform to another) and attempt to determine which sub-community was implicated in the migration, what their interests were, and how they integrated into their new destination. Concept drift is still a very interesting question that remains unaddressed by this work: it is difficult to estimate how much and which historical data to retain to maintain a statistically significant yet relevant model, but is essential to ensuring that inferences on ever-changing users and corpora do not become historical artifacts.

References

Josh Attenberg, Kilian Weinberger, Anirban Dasgupta, Alex Smola, and Martin Zinkevich. Collaborative email-spam filtering with the hashing trick. In *Proceedings of the Sixth Conference on Email and Anti-Spam*, 2009.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz Josef Och, and Jeffrey Dean. Large language models in machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 858–867, 2007. URL `http://www.aclweb.org/anthology/D/D07/D07-1090`.

Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

Glen Coppersmith, Mark Dredze, Craig Harman, and Kristy Hollingshead. From adhd to sad: Analyzing the language of mental health on twitter through self-reported diagnoses. In *Proceedings of the Workshop on Computational Linguistics and Clinical Psychology: From Linguistic Signal to Clinical Reality, Denver, Colorado, USA, June. North American Chapter of the Association for Computational Linguistics*, 2015.

Cristian Danescu-Niculescu-Mizil, Lillian Lee, Bo Pang, and Jon Kleinberg. Echoes of power: Language effects and power differences in social interaction. In *Proceedings of the 21st international conference on World Wide Web*, pages 699–708. ACM, 2012.

Johannes C Eichstaedt, Hansen Andrew Schwartz, Margaret L Kern, Gregory Park, Darwin R Labarthe, Raina M Merchant, Sneha Jha, Megha Agrawal, Lukasz A Dziurzynski, Maarten Sap, et al. Psychological language on twitter predicts county-level heart disease mortality. *Psychological science*, 26(2): 159–169, 2015.

Jacob Eisenstein. What to do about bad language on the internet. In *HLT-NAACL*, pages 359–369, 2013.

Deborah Estrin. Small data, where n= me. *Communications of the ACM*, 57 (4):32–34, 2014.

Pedro A Forero, Alfonso Cano, and Georgios B Giannakis. Consensus-based distributed support vector machines. *The Journal of Machine Learning Research*, 11:1663–1707, 2010.

Cheng-Kang Hsieh, Hongsuda Tangmunarunkit, Faisal Alquaddoomi, John Jenkins, Jinha Kang, Cameron Ketcham, Brent Longstaff, Joshua Selsky, Betta Dawson, Dallas Swendeman, et al. Lifestreams: A modular sense-making toolset for identifying important patterns from everyday life. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, page 5. ACM, 2013.

Yoram M Kalman and Darren Gergle. Cmc cues enrich lean online communication: The case of letter and punctuation mark repetitions. 2010.

Si Quang Le and Tu Bao Ho. An association-based dissimilarity measure for categorical data. *Pattern Recognition Letters*, 26(16):2549–2557, 2005.

Abby Levenberg and Miles Osborne. Stream-based randomised language models for SMT. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 756–764, Singapore, August 2009. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/D/D09/D09-1079`.

Abby Levenberg, Miles Osborne, and David Matthews. Multiple-stream language models for statistical machine translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 177–186, Edinburgh, Scotland, July 2011. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W11-2122`.

Yan Liu, Alexandru Niculescu-Mizil, and Wojciech Gryc. Topic-link lda: joint models of topic and author community. In *proceedings of the 26th annual international conference on machine learning*, pages 665–672. ACM, 2009.

Andrew McCallum, Andrés Corrada-Emmanuel, and Xuerui Wang. The author-recipient-topic model for topic and role discovery in social networks: Experiments with enron and academic email. 2005.

J Paul Morrison. *Flow-Based Programming: A new approach to application development*. CreateSpace, 2010.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345–1359, 2010.

Apache Storm. Storm, distributed and fault-tolerant real-time computation. `https://storm.apache.org/`, 2014.

David Talbot and Miles Osborne. Randomised language modelling for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 512–519, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P/P07/P07-1065`.

Sketch the Cow. Complete public reddit comments corpus. `https://archive.org/details/2015_reddit_comments_corpus`, July 2015.

Alexey Tsymbal. The problem of concept drift: definitions and related work. *Computer Science Department, Trinity College Dublin*, 106, 2004.

Ding Zhou, Eren Manavoglu, Jia Li, C Lee Giles, and Hongyuan Zha. Probabilistic models for discovering e-communities. In *Proceedings of the 15th international conference on World Wide Web*, pages 173–182. ACM, 2006.

Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.

Appendix A Modular Pipeline Proposal

In the following, we describe a distributed system architecture which will coordinate our efforts to create a modular processing pipeline that permits both continuous and batch data processing. The architecture is based on the *publish-subscribe* design pattern, in which *publishers* (i.e. data producers, such as a smartphone app or API-polling tool) push data onto a shared storage medium (known as a *bus*). When data becomes available, *subscribers* that are interested in the data pull it from the bus and perform processing. The set of subscribers and publishers are collectively referred to as *components* of the system. The data that is exchanged on the bus are referred to as *messages*. Finally, the system includes local *services* that operate independently of the bus and are shared by all the components. All of the entities are implemented as OS processes, and may either run locally with each other or communicate over the network.

Figure 4 presents an example of how the core system components (specifically, the bus and the user identity service) can be augmented with data collection, processing, and visualization units. An example application of providing news article recommendations is implemented by acquiring email and twitter data, processing it to build a topic model, then matching the model against news articles to display a ranked list to the user. The query to Twitter for the follow graph is intended to demonstrate that DPUs can make use of external data sources as well as small data from the bus.

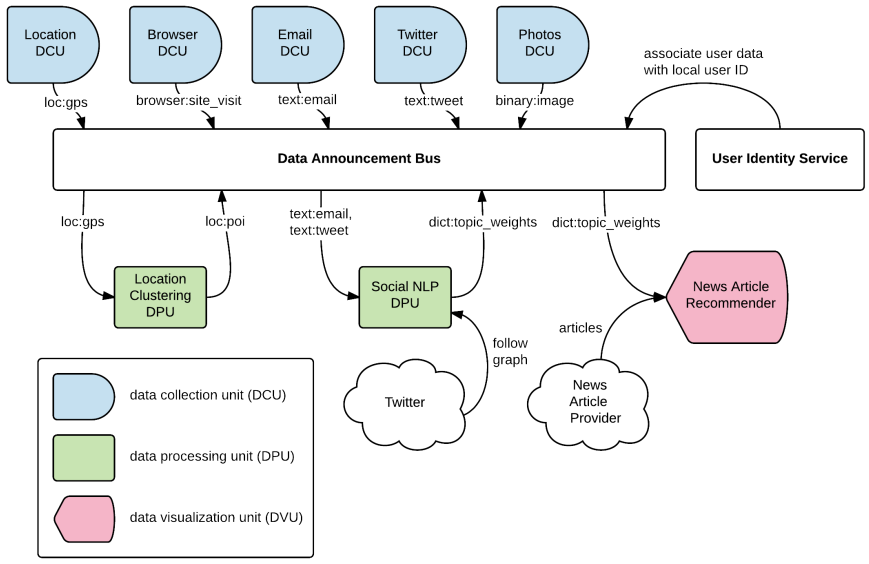


Figure 4: Modular system components and integration mechanisms of proposed system architecture

A.0.1 Components

There are three core types of components in our system, characterized by their roles on the bus: (1) *collectors,* which are solely publishers, (2) *transformers*, which are both subscribers and publishers, and (3) *consumers*, which are solely subscribers. We use the abbreviations "DCU", "DPU", and "DVU", respectively, to refer to these components.

Collectors are components that create data or transmit external data into the system; some examples are smartphone apps that record location or self-report, and daemons that poll, say, Gmail's API or the Twitter API, and push a user's emails and posts into the system as they become available. In some cases, it is possible to transform sensitive data on the client into a less sensitive aggregate form — for instance, one's browsing history could be transformed into a users' general interests. In both cases (where data is sent verbatim or pre-processed before sending), there is a server-side portion of the DCU that collects and publishes incoming data to the bus for the rest of the architecture to use.

Consumers are similarly straightforward: they take processed or raw data and export it somehow from the system, often in the form of a visualization or tool presented to the user, or a public API that third-party software can use. Transformers deserve a more in-depth discussion, as follows.

Transformers are the "workhorses" of the modular pipeline: they take "low-level" data and process them into more semantically meaningful forms. For example, a transformer might read in raw GPS coordinates (or Location data from new Location APIs) from a smartphone app and spatio-temporally cluster them to produce a set of "places of interest" for the user, e.g. their home or work locations. Transformers can consume multiple sources of input to produce a single output type, e.g. a location trace and accelerometer trace in order to produce an activity-annotated trace of locations where the user walked, sat, took public transportation, etc. Transformers can also consume processed data, allowing the composition of chains of processing units that produce a high-level output. The ability to factor complicated processing tasks into chains of discrete, independent units improves both the scalability and reusability of the system. Small independent processing units are also easier to test, improving robustness.

A.0.2 Message structure

We impose a *type system* over messages exchanged on the bus in order to allow each unit to discover data that is relevant to its task. The contents of each message is defined by a schema corresponding to its type; for instance, GPS locations are typed accordingly and are described as vectors of latitude, longitude, elevation, etc. The type identifies *what* the message is, whereas the schema describes *how* the payload of the message is formatted, i.e. the fields in the payload and their respective types. As such, all but the primitive types (numbers, strings, booleans, etc.) are compositions of existing types. We intend to use schema.org's schema ontology as a starting point, but will maintain our own library of custom schemas corresponding to our system's types.

For the sake of identifying related types, we suggest a hierarchical naming scheme where specific types include a more general type as their prefix, if applicable. For instance, both emails and Twitter posts contain mostly text, so their

respective types might be "text:email"and "text:twitter". Note that this is not a hard inheritance; there is no expectation of conformance on the contents of types with the same prefix and types do not inherit any fields from their parents.

In addition to the type system, each message is wrapped in a *frame* which includes metadata common to all small data. The frame specifies the ID of the user to whom the data belongs (explained in the "Services" subsection) and the time of collection. Additionally, the frame contains the data's *pedigree*, represented as an ordered list of component UUIDs and data type tuples, spanning from the original DCU to the most recent DPU that processed it.

A.1 Related architectures

In prior work, we developed a modular toolset for mobile-only sensing in a framework called Lifestreams Hsieh et al. [2013]. As part of the proposed work we will adapt and enhance Lifestreams to handle non-mobile small-data streams. The Lifestreams stack consisted of four layers: (i) feature extraction (ii) feature selection, (iii) inference, and (iv) visualization. Each layer consists of modular plug-ins or building blocks that can process the data provided by the lower layer, and then send the result to the next layer up in the stack. At the bottom of the stack are different personal data streams obtained via personal data capture mechanisms discussed earlier (e.g., consumer transactions, communication, entertainment, mobile data).

Appendix B Email Analysis Framework

(listed on the following pages...)

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

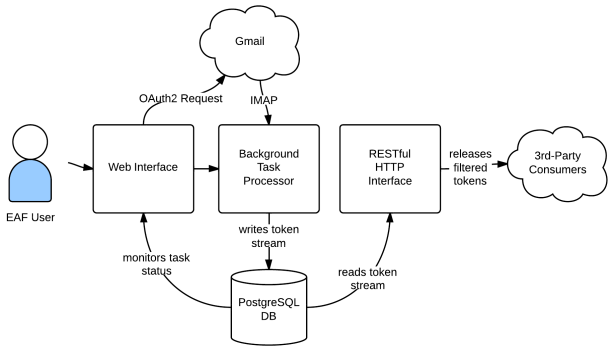


Figure 1: Structure of the Email Analysis Framework

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

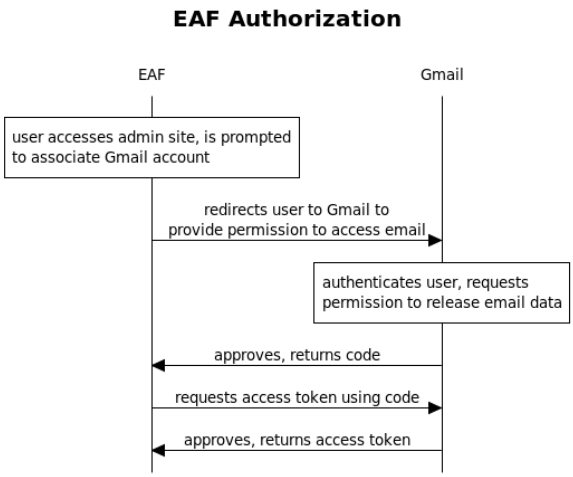


Figure 2: Gmail Authorization

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.

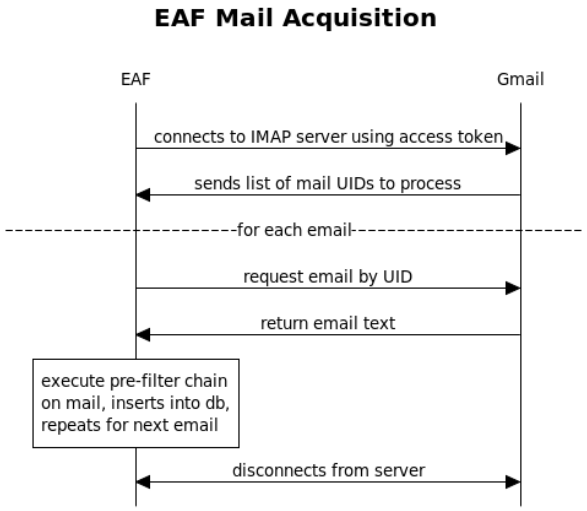


Figure 3: Mail Acquisition

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

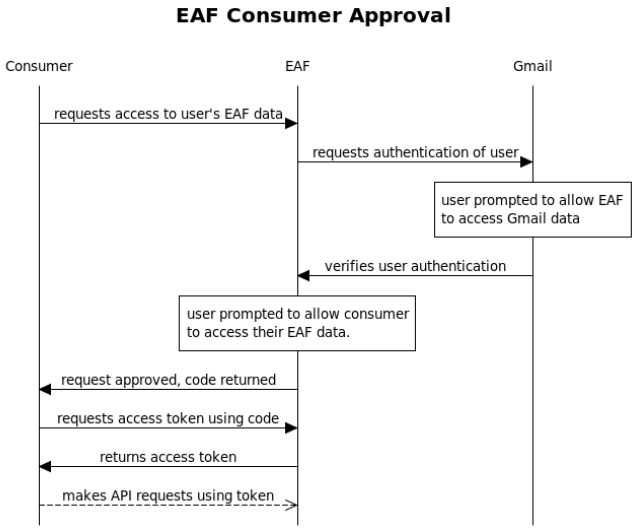


Figure 4: Release to Consumer

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

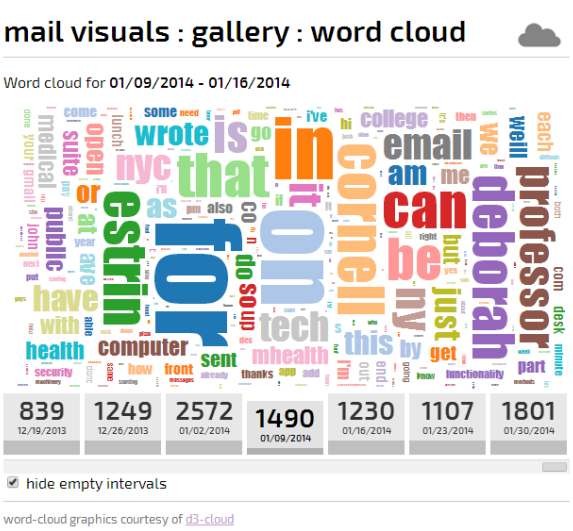
As mentioned, third-party applications can gain temporary access to a user's data for the purpose of visualizing or otherwise processing it. Granting this access is currently at the user's discretion; the user should make use of the per-consumer post-filter controls to limit the release of sensitive 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

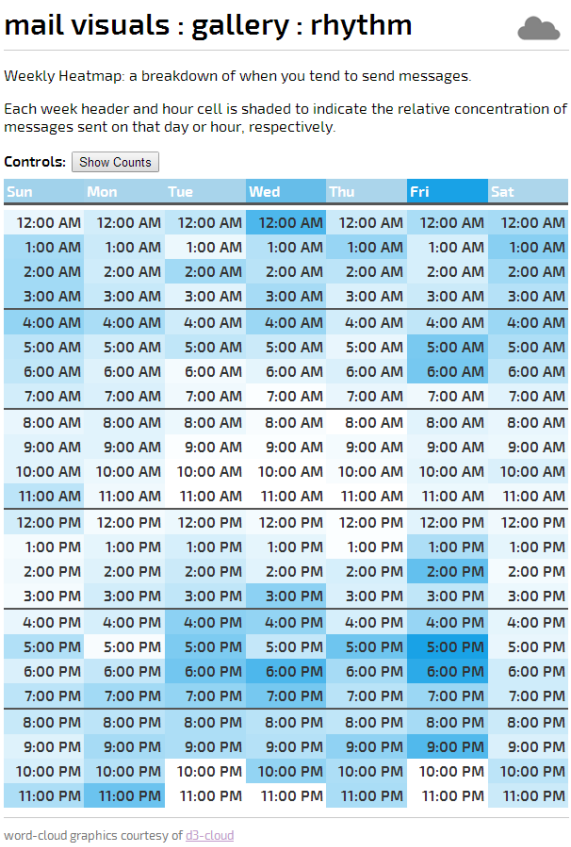


Figure 6: "Rhythm" Visualization

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.