Navigating Large Comment Threads With CoFi

Christine Doran, Guido Zarrella and John C. Henderson
The MITRE Corporation
Bedford, MA
{cdoran,jzarrella,jhndrsn}@mitre.org

Abstract

Comment threads contain fascinating and useful insights into public reactions, but are challenging to read and understand without computational assistance. We present a tool for exploring large, community-created comments threads in an efficient manner.

1 Introduction

The comments made on blog posts and news articles provide both immediate and ongoing public reaction to the content of the post or article. When a given site allows users to respond to each other ("threaded" responses), the comment sets become a genuine public conversation. However, this information can be difficult to access. Comments are typically not indexed by search engines, the volume is often enormous, and threads may continue to be added to over months or even years. This makes it hard to find particular information of interest (say, a mention of a particular company in a set of thousands of YouTube comments), or to understand the gist of the discussion at a high-level.

Our goal in this work was to create a simple tool which would allow people to rapidly ingest useful information contained in large community-created comment threads, where the volume of data precludes manual inspection. To this end, we created CoFi (**Co**mment **Fi**lter), a language-independent, web-based interactive browser for single comment threads.

2 How CoFi works

For a given set of comments, we create a distinct CoFi instance. Each instance is over a natural data set, e.g. all comments from a particular discussion group, comments attached to an individual news article, or tweets resulting from a topical search. Creating a CoFi instance has three steps: harvesting the comments, clustering the comments, and responding to user interactions while they visualize and navigate (sorting and filtering) the dataset.

2.1 Harvesting the data

Our comments are harvested from individual web sites. These need not be in English, or even in a single language. Typically, sites use proprietary javascript to present comments. Each web site has a unique interface and formatting to serve the comments to web browsers, and there is no general purpose tool to gather comments everywhere. The CoFi approach has been to factor this part of the problem into one harvesting engine per web site. Some sites provide an API that simplifies the problem of harvesting comments that contain particular keywords. On other sites, there seems to be no reliable alternative to developer ingenuity when it comes to altering the harvesting engines to accommodate data formats. Thus, we note that the harvesting activity is only *semi*-automated.

2.2 Clustering the data

Once harvesting is complete, the rest of the process is automatic. Clusters are generated and labeled using a pipeline of machine learning tools. The open source package MALLET provides many of our document ingestion and clustering components (McCallum, 2002). Our processing components are language-independent and can be used with non-English or mixed language data sets.

Specifically, we use a combination of Latent Dirichlet Allocation (LDA), K-Means clustering, and calculation of mutual information. LDA models each document (aka comment) as a mixture of latent topics, which are in turn comprised of a probability distribution over words (Chen, 2011, gives a good overview). It's an unsupervised algorithm that performs approximate inference. The topics it infers are the ones that best explain the statistical distributions of words observed in the

Proceedings of the NAACL-HLT 2012: Demonstration Session, pages 9–12,
Montréal, Canada, June 3-8, 2012.

data. It is highly parallelizable and so it scales well to very large data sets. In practice we ask LDA to search for *5k* topics, where *k* is the number of clusters we will eventually display to the user.

The second step is to perform K-Means clustering on the documents, where the documents are represented as a mixture of LDA topics as described above, and the clustering chooses *k* clusters that minimize the differences between documents in the cluster while maximizing the difference between documents that are not in the same clusters. This step is fast, in part because of the fact that we have already reduced the number of input features down to *5k* (rather than having one feature for each word observed in the entire dataset.)

Finally, we give the clusters titles by performing a calculation of mutual information (MI) for each word or bigram in each cluster. Specifically, clustering terms (both words and bigrams) that occur frequently in one cluster but rarely in other clusters will receive high scores. The terms with the highest MI scores are used as cluster labels.

One significant advantage of this completely unsupervised approach is that CoFi is more robust to the language of comment data, e.g. grammatical and spelling inconsistency, informal language, which are a challenge for rule-based and supervised NLP tools.

In addition to the machine-generated topic clusters, CoFi allows user-defined topics. These are search terms and topic labels hand-created by a domain expert. CoFi partitions the comments into machine-generated topics and *also* assigns each comment to any of the matching predefined topics. This approach is useful for domain experts, enabling them to quickly find things they already know they want while allowing them to also take advantage of unexpected topics which emerge from the system clustering.

2.3 Creating the visualizations

CoFi uses the **`JQuery`**, **`Flot`**, and **`g.Raphael`** javascript libraries to provide a dynamic, responsive interface. When the user visits a CoFi URL, the data is downloaded into their browser which then computes the visualization elements locally, allowing fast response times and offline access to the data. The **`JQuery`** library is central to all of the javascript processing that CoFi performs, and ensures that all features of the interface are cross-compatible with major browser versions.

The interface provides the ability to drill down further into any data, allowing the user to click on any aspect of the analysis to obtain more detail. Since the visualization is calculated locally, the software can create dynamically updated timelines that show the user how any subset of their data has changed over time.

It is also important to prioritize all data presented to the user, allowing them to focus on the most useful documents first. CoFi applies an automatic summarization technique to perform relevance sorting. We evaluated several state-of-the-art automatic document summarization techniques and settled on a Kullback-Leibler divergence inspired by techniques described in Kumar et al. (2009). The "relevance" sort relies on a measure of how representative each comment is relative to the entire collection of comments that the user is viewing at the time. This allows us to rapidly rank tens of thousands of comments in the order of their relevance to a summary. Several of the approaches we tested were chosen from among the leaders of NIST's 2004 Document Understanding Conference (DUC) summarization evaluation. Many of them used slight variants of KL divergence for sentence scoring. We also implemented Lin & Bilmes' (2010) Budgeted Maximization of Submodular Functions system, which performed best according to the DUC evaluation. However, even after applying a scaling optimization inspired by the "buckshot" technique of Cutting et al. (1992) the processing speed was still too slow for dealing with datasets containing more than 10000 small documents. The KL divergence approach scales linearly in the number of comments while still offering cutting edge qualitative performance. This means that the calculation can be done on the fly in javascript in the browser when the user requests a relevance sort. This allows CoFi to tailor the results to whatever sub-selection of data is currently being displayed. For CoFi's typical use cases this computation can be completed in under 2 seconds.

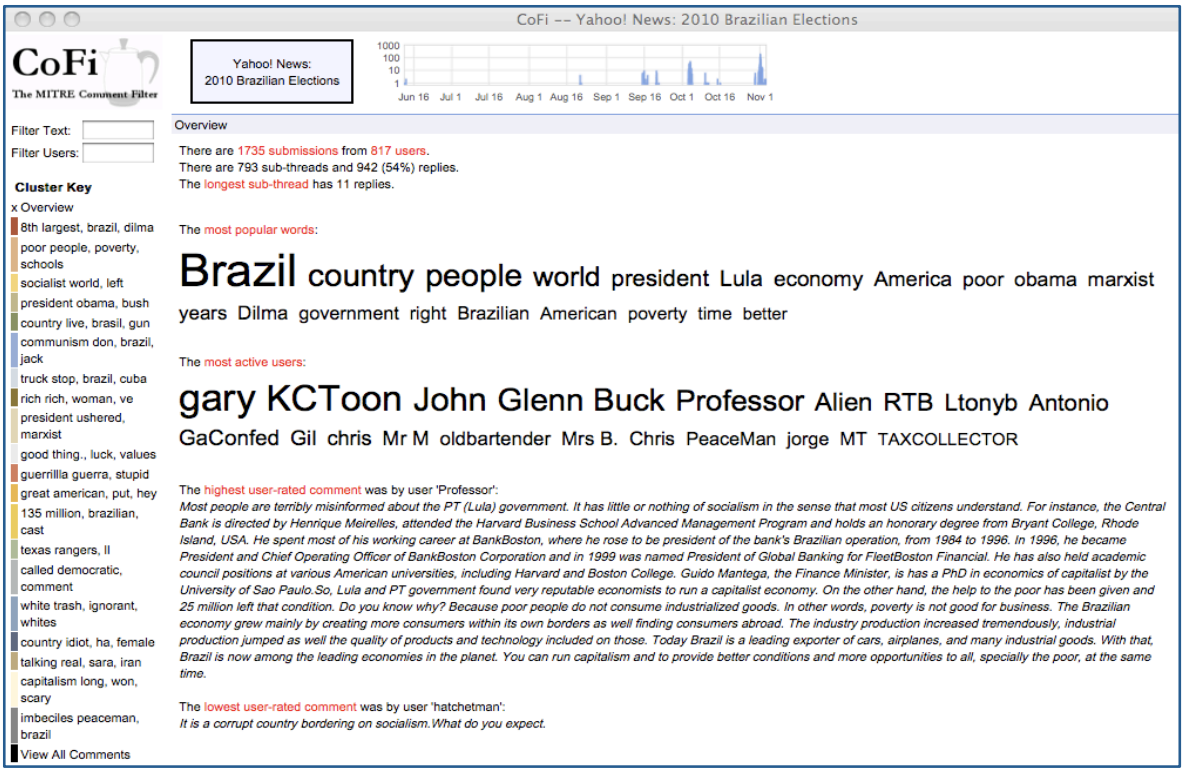

Figure 1: CoFi top level summary view

3 The CoFi Interface

CoFi takes a set of comments and produces the interactive summary you see in Figure 1. CoFi works best when a user is operating with between 200 and 10,000 comments. With small numbers of comments, there may not be enough data for CoFi to find interesting topic clusters. With very large numbers of comments, a user's web browser may struggle to display all comments while maintaining sufficient responsiveness.

The raw data is available for inspection in many ways. The summary screen in Figure 1 presents a list of automatically-discovered clusters on the left-hand side (typically 10-30, this is a parameter of the clustering algorithm), the posting volume timeline on the top, and some overall statistics and characteristic words and posters in the middle. The user can return to this view at any point using the *Overview* button. At the top of the page, CoFi presents the total number of comments and participants, and a summary of the level of threading, which is a good indicator of how interactive the data set is. Where community ratings appear on a site, we also present the highest and lowest rated comments (this is solely based on the community rating, and not on our relevance calculation). In the middle of the display are two hyperlinked word clouds containing the highest frequency words and users. Selecting one of the top words or users has the same effect as searching for that term in one of the *Search* boxes—both of these approaches will present the user with matching comments with the term highlighted, and color coding to indicate cluster membership. The links from most popular words and most active users bring up a multi-graph view as in Figure 3.

Each time a set of comments is selected, either via a cluster, full text search, or filtering on a particular commenter, the set is presented to the user in a sorted order with the comments most representative of the set ordered above those that are less representative. In this way, the user can quickly get a handle on what the set is about without reading all of the items in detail. The comments can also be sorted into the original temporal order, which can be useful to see how a comment thread evolves over time, or to view an original comment and threaded replies in a nested ordering. Figure 2 shows a single cluster in CoFi. The full thread timeline now has a red overlay for the selected subset of comments.

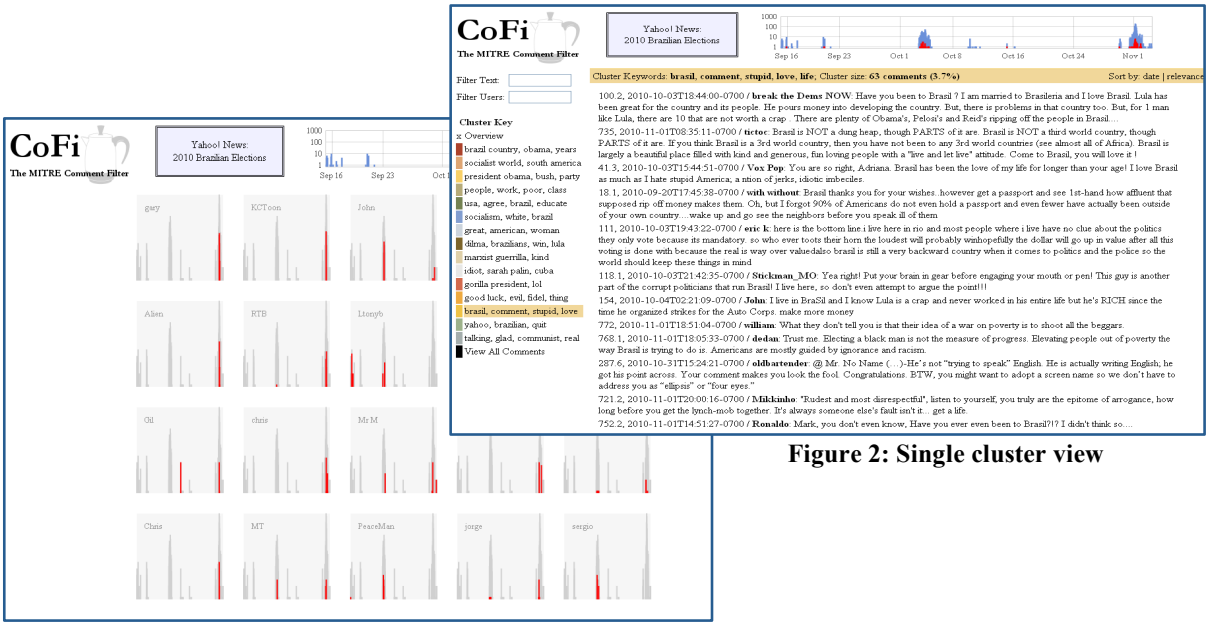

Figure 2: Single cluster view

Figure 3: The "small multiples" view of frequent contributors

At the bottom of the cluster lists, there is a *View All Comments* option. Sorting the entire set by relevance gives a good snapshot of most and least useful comments in the thread. From any of the views, clicking on a user name will display all comments from that user, and clicking on the comment ID will present that sub-thread; top-level comments are numbered X, while replies are labeled X.X. The CoFi interface also allows the user to export individual comments, marking those comments as having been "handled" and routed to a particular person. This makes it easier to incrementally process comments as they arrive.

We have applied CoFi to 72 distinct data sets, including forum discussions, news article, blog and YouTube comments, Twitter and comments on regulatory changes submitted to government offices via Regulations.gov. These last documents are much longer than those CoFi was intended to handle, but CoFi was nonetheless able to support interesting analysis. In one instance, we identified a clear case of "astroturfing" (fake grassroots movement) based on the CoFi clusters.

Acknowledgements

Over the course of this project, many people have supported our work. We'd particularly like to thank Mark Maybury, Robert Peller at USSOUTHCOM, and Marty Ryan, Robert Battle and Nathan Vuong at the MITRE Miami site. This technical data was produced for the U. S. Government under Contract No. W15P7T-11-C-F600, and is subject to the Rights in Technical Data-Noncommercial Items clause at DFARS 252.227-7013 (NOV 1995). Approved for Public Release: 12-1507. Distribution Unlimited. MITRE Document number MP120212.

References

Chen, Edwin (2011). Introduction to Latent Direchlet Allocation, http://blog.echen.me/2011/08/22/introduction-to-latent-dirichlet-allocation/#comments

Cutting, D., Karger, D., Pedersen, J., and Tukey, J. (1992). Scatter/Gather: a cluster-based approach to browsing large document collections. *Proceedings of 15th Annual International ACM SIGIR conference,* New York, NY, USA, 318-329

Kumar, C., P. Pingali, and V. Verma (2009). Estimating Risk Of Picking a Sentence for Document Summarization. *Proceedings of CICLing 2009*, LNCS 5449, 571-581.

Lin, H. and Bilmes, J. (2010). Multi-document summarization via budgeted maximization of submodular functions. *Proceedings of Human Language Technologies 2010*, Los Angeles, CA, USA, 912-920.

McCallum, Andrew Kachites (2002). MALLET: A Machine Learning for Language Toolkit.

Mishne, Gilard and Natalie Glance (2006). Leave a reply: An Analysis of Weblog Comments. *In Workshop on the Weblogging Ecosystem*, 15th International World Wide Web Conference, May.