TweetMotif: Exploratory Search and Topic Summarization for Twitter

Brendan O'Connor
Carnegie Mellon University
brenocon@gmail.com

Michel Krieger
Meebo, Inc.
mikekrieger@gmail.com

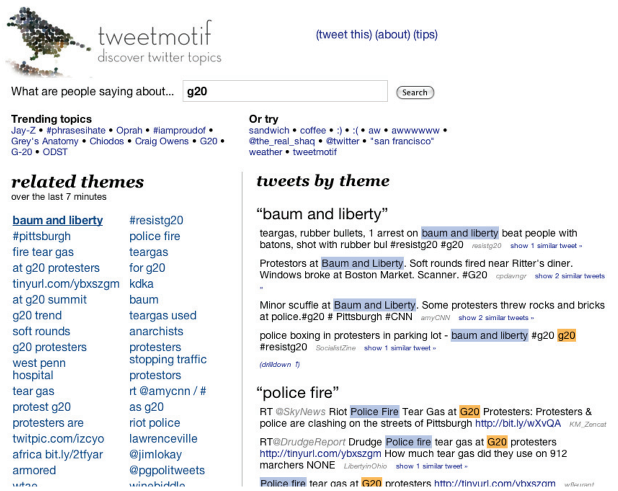
David Ahn
Microsoft, Inc.
daviddahn@gmail.com

Abstract

We present TweetMotif, an exploratory search application for Twitter. Unlike traditional approaches to information retrieval, which present a simple list of messages, TweetMotif groups messages by frequent significant terms — a result set's subtopics — which facilitate navigation and drilldown through a faceted search interface. The topic extraction system is based on syntactic filtering, language modeling, near-duplicate detection, and set cover heuristics. We have used TweetMotif to deflate rumors, uncover scams, summarize sentiment, and track political protests in real-time. A demo of TweetMotif, plus its source code, is available at http://tweetmotif.com.

Introduction and Description

On the microblogging service Twitter, users post millions of very short messages every day. Organizing and searching through this large corpus is an exciting research problem. Since messages are so small, we believe microblog search requires summarization across many messages at once.



tweetmotif
discover twitter topics

(tweet this) (about) (tips)

What are people saying about... g20 Search

Trending topics
Jay-Z • #phrasesihate • Oprah • #iamproudof • Grey's Anatomy • Chiodos • Craig Owens • G20 • G-20 • ODST

Or try
sandwich • coffee • :) • :(• aw • awwwwww • @the_real_shaq • @twitter • "san francisco" weather • tweetmotif

related themes
over the last 7 minutes

baum and liberty, #pittsburgh, fire tear gas, at g20 protesters, tinyurl.com/ybxszgm, at g20 summit, g20 trend, soft rounds, g20 protesters, west penn hospital, tear gas, protest g20, protesters are, twitpic.com/izcyo, africa bit.ly/2tfyar, armored, #resistg20, police fire, teargas, for g20, kdka, baum, teargas used, anarchists, protesters stopping traffic, protestors, rt @amycnn / #, as g20, riot police, lawrenceville, @jimlokay, @pgpolitweets

tweets by theme

"baum and liberty"

teargas, rubber bullets, 1 arrest on baum and liberty beat people with batons, shot with rubber bul #resistg20 #g20 resistg20 show 1 similar tweet »

Protestors at Baum and Liberty. Soft rounds fired near Ritter's diner. Windows broke at Boston Market. Scanner. #G20 cpdavngr show 2 similar tweets »

Minor scuffle at Baum and Liberty. Some protesters threw rocks and bricks at police.#g20 # Pittsburgh #CNN amyCNN show 2 similar tweets »

police boxing in protesters in parking lot - baum and liberty #g20 g20 #resistg20 SocialistZine show 1 similar tweet »

(drilldown ↑)

"police fire"

RT @SkyNews Riot Police Fire Tear Gas at G20 Protesters: Protesters & police are clashing on the streets of Pittsburgh http://bit.ly/wXvQA KM_Zencat

RT@DrudgeReport Drudge Police fire tear gas at G20 protesters http://tinyurl.com/ybxszgm How much tear gas did they use on 912 marchers NONE LibertyinOhio show 1 similar tweet »

Police fire tear gas at G20 protesters http://tinyurl.com/ybxszgm wfleurant

Figure 1: Screenshot of TweetMotif.

Our system, TweetMotif, responds to user queries, first retrieving several hundred recent matching messages from a simple index; we use the Twitter Search API.

Instead of simply showing this result set as a list, TweetMotif extracts a set of themes (topics) to group and summarize these messages. A topic is simultaneously characterized by (1) a 1- to 3-word textual label, and (2) a set of messages, whose texts must all contain the label.

TweetMotif's user interface is inspired by faceted search, which has been shown to aid Web search tasks (Hearst et al. 2002). The main screen is a two-column layout. The left column is a list of themes that are related to the current search term, while the right column presents actual tweets, grouped by theme. As themes are selected on the left column, a sample of tweets for that theme appears at the top of the right column, pushing down (but not removing) tweet results for any previously selected related themes. This allows users to explore and compare multiple related themes at once.

The set of topics is chosen to try to satisfy several criteria, which often conflict:

1. Frequency contrast: Topic label phrases should be frequent in the query subcorpus, but infrequent among general Twitter messages. This ensures relevance to the query while eliminating overly generic terms.
2. Topic diversity: Topics should be chosen such that their messages and label phrases minimally overlap. Overlapping topics repetitively fill the same information niche; only one should be used.
3. Topic size: A topic that includes too few messages is bad; it is overly specific.
4. Small number of topics: Screen real-estate and concomitant user cognitive load are limited resources.

The goal is to provide the user a concise summary of themes and variation in the query subcorpus, then allow the user to navigate to individual topics to see their associated messages, and allow recursive drilldown. The approach is related to document clustering (though a message can belong to multiple topics) and text summarization (topic labels are a high-relevance subset of text across messages). We heuristically proceed through several stages of analysis.

Step 1: Tokenization and syntactic filtering

Tokenization is difficult in the social media domain, and good tokenization is absolutely crucial for overall system performance. Standard tokenizers, usually designed for newspapers or scientific publications, perform poorly. Our regex-based tokenizer treats hashtags, @-replies, abbreviations, strings of punctuation, emoticons and unicode glyphs (e.g. musical notes) as tokens.

Unigrams are too narrow a unit of analysis; ideally, we want to extract all phrases and subphrases. We begin with all unigrams, bigrams, and trigrams, discarding unigrams that are function words, and discarding bigrams and trigrams that cross syntactic boundaries. These rules flag n-grams including certain types of punctuation tokens in certain positions, and ones that end with certain right-binding function words like "the" and "of." This simple syntactic filtering greatly improves the coherency of extracted n-grams.

Step 2: Score and filter topic phrase candidates

TweetMotif takes a simple language modeling approach to identifying topic phrases that are most distinctive for a tweet result set, scoring them by the likelihood ratio:

$$\frac{Pr(\text{ phrase } \mid \text{ tweet result set })}{Pr(\text{ phrase } \mid \text{ general tweet corpus })}$$

As is usually the case in language modeling, a given phrase does not necessarily occur in a corpus, so probabilities must be estimated with smoothing. We tried several simple estimation methods, settling on Lidstone smoothing:

$$Pr(\text{ phrase } \mid \text{corpus}) = \frac{\text{phrase count in corpus} + \delta}{N + \delta n}$$

where for a phrase of length m, N is the count of all phrase instances of length m in the corpus, $\delta = 0.5$ is the smoothing parameter, and n is the count of all phrase types of length m in the corpus. Essentially, there are independent models for unigram, bigram, and trigram phrases.

The background corpus consists of 150,000 Twitter messages collected in April 2009 from search queries for *the*, *of* and several other common English function words.

It is interesting to compare our approach to TF/IDF for document retrieval, which estimates document relevance by balancing the frequency of query terms against their frequencies in a background corpus. Note that the average Twitter message is 11 words long, and words rarely occur more than once in a message; thus, the count of a word is virtually the same as the count of messages it occurs in (DF and TF are the same). If messages are considered documents, the notion of document TF is not very useful. Our approach is more like TF for one giant document consisting of the concatenation of all query subcorpus messages. This too is an odd analogy. In general, we believe the microblog search problem will require creative formulations of cross-message phonemena beyond current paradigms in IR.

Step 3: Merge similar topics

Every candidate phrase defines a topic, a set of messages that contain that phrase. Many phrases, however, occur in roughly the same set of messages, thus their topics are repetitive. We seek to merge similar topics.

First, there are easy merges between subsumed n-gram phrases of differing sizes. Each of an n-gram's label-subsumed (n-1)-grams must conversely subsume its message set. For example, the message set for the bigram topic "swine flu" must be a subset or equal to the two unigram topics "swine" and "flu." If the "swine flu" topic is in fact equal to the "flu" topic, then we discard the "flu" topic, since "swine flu" is strictly more informative.

Furthermore, ignoring topic labels, we merge topics if their message sets have more than 90% Jaccard similarity. All pairs of topics are compared, so final topics are connected components of the pairwise $Jacc \geq 0.9$ graph — i.e., single-link clustering. When several topics are merged, only the intersection of messages is included in the new topic. There is a label choice problem/opportunity for merged topics: any of the old topics' labels are now legitimate. Our heuristic solution usually picks longer and higher scoring labels, and sometimes combines short labels into a skip n-gram.

Step 4: Group near-duplicate messages

Grouping messages by topic reveals a massive amount of message duplication on Twitter. People forward ("retweet") interesting messages such as jokes and news headlines; and furthermore, a seemingly huge number of bots repeat advertisements, spam, weather reports, news feeds, other people's tweets, songs being played on personalized Internet radio stations, templated messages, etc. It is a waste of space to always show near-duplicates to the search user; therefore we detect clusters of near-duplicates, display them with a single representative and numeric size, and allow them optionally to be viewed. The algorithm groups messages whose sets of trigrams have a pairwise Jaccard similarity exceeding 65%. (For this and the topic merging step, using phrase-message indexes cuts down on the potentially quadratic runtime.)

This technique seems to reliably find retweets and other forms of repetition; it also naturally groups together spam. This was very important in mid-2009 when we developed TweetMotif, when spam was very prevalent and retweets were not explicitly marked in the data.

Step 5: Finalize topics

We are now left with a ranked list of topics containing messages in near-duplicate clusters. After eliminating topics that contain only one near-duplicate cluster, the list is cut off to the top 40 topics, and all messages that did not end up in a topic are put in a catch-all "more..." topic.

All the above analysis is conducted on the result set while the user waits. A demo, examples, and source code are available at http://tweetmotif.com.

References

Hearst, M.; Elliott, A.; English, J.; Sinha, R.; Swearingen, K.; and Yee, K. 2002. Finding the flow in web site search. *Commun. ACM* 45(9):42–49.