Summarizing Microblogs Automatically

Beaux Sharifi, Mark-Anthony Hutton, and Jugal Kalita
University of Colorado at Colorado Springs
1420 Austin Bluffs Parkway
Colorado Springs, CO 80918, USA
{bsharifi, mhutton, jkalita}@uccs.edu

Abstract

In this paper, we focus on a recent Web trend called microblogging, and in particular a site called Twitter. The content of such a site is an extraordinarily large number of small textual messages, posted by millions of users, at random or in response to perceived events or situations. We have developed an algorithm that takes a trending phrase or any phrase specified by a user, collects a large number of posts containing the phrase, and provides an automatically created summary of the posts related to the term. We present examples of summaries we produce along with initial evaluation.

1 Introduction

Since Twitter's inception in 2006, it has grown at an unprecedented rate. In just four years, the service has grown to approximately 20 million unique visitors each month with users sending short 140-character messages (known as "tweets") approximately 40 million times a day. While the majority of these tweets are pointless babble or conversational, approximately 3.6% of these posts are topics of mainstream news (Pear Analytics, 2009). For example, Twitter has been cited as breaking many important events before traditional media, such as the attacks in Mumbai and the crash of the US Airways flight into the Hudson River.

In order to help users sort through the vast number of tweets that occur each day, Twitter.com has added a number of tools. For instance, Twitter's homepage displays important topics for three different ranges of time in order to see what topics are popular. For most topics, users are forced to read through related posts in order to try and understand why a topic is trending. In order to help users further, Twitter has partnered with the third-party website WhatTheTrend[1] in order to provide definitions of trending topics. WhatTheTrend allows users to manually enter descriptions of why a topic is trending. Unfortunately, WhatTheTrend suffers with spam and rants as well as lag time before a new trending topic is defined by a user.

While WhatTheTrend is a step in the right direction, a better approach is to automatically summarize important events as they occur in real time. We have developed such a method. Our method can automatically summarize a collection of microblogging posts that are all related to a topic into a short, one-line summary. Our results show that our automated summarizer produces summaries that are close to human-generated summaries for the same set of posts. For example, Table 1 below contains a sample of automatically produced summaries for some recently trending topics on Twitter.

2 Related Work

Some early work focused on summarizing results of database queries for presentation during natural language interactions (e.g., Kalita et al., 1986). Most summaries are generated for the purposes of providing a "gist" of a document or a set of documents to human readers (e.g., Luhn, 1958; Brandow et al., 1995). Summaries are sometimes also used as inputs to machine learning approaches, say for categorization. Kolcz et al. (2001) summarize textual documents in order to classify them, using the summaries as a feature to be input to a classifier. Most early studies used a news corpus like the Reuters dataset. As the Web started growing in size, the focus moved to Web pages.

[1] http://www.whatthetrend.com

Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the ACL, pages 685–688,
Los Angeles, California, June 2010.

Topic	Automated Summary	Date
Ice Dancing	Canadians Tessa Virtue and Scott Moir clinch the gold in Olympic ice dancing; U.S. pair Davis and White win silver	2/22/2010
Dodgers	Phillies defeat Dodgers to take the National League Championship series.	10/21/2009
Limbaugh	Limbaugh dropped from group bidding for St. Louis Rams	10/14/2009
Dow Jones	The Dow Jones Industrial Average passes 10,000 for the first time since October 7th, 2008.	10/14/2009
Captain Lou	Wrestler, personality Captain Lou Albano dies at 76	10/14/2009
Bloomberg	Bloomberg Acquires Businessweek for Less Than $5 million	10/13/2009
G20	Trouble breaks out at G20 summit: Protesters and riot police have clashed ahead of the G20 summit in Pittsburgh	09/24/2009
AT&T	AT&T plans for iPhone MMS to arrive Friday	09/23/2009

Table 1. Example Summaries Produced by the Phrase Reinforcement Algorithm.

For example, Mahesh (1997) examines the effectiveness of Web document summarization by sentence extraction. Recently, there has been work on summarizing blogs (e.g. Zhou and Hovy, 2006; Hu et al., 2007). Most techniques focus on extraction: the selecting of salient pieces of documents in order to generate a summary. Applying extraction on microblogs at first appears irrelevant since a microblog post is already shorter than most summaries. However, extraction is possible when one considers extracting from multiple microblogs posts that are all related to a central theme.

3 Approach

3.1 Twitter API

Through an entirely HTTP-based API provided by Twitter, users can programmatically perform almost any task that can be performed via Twitter's web interface. For non-whitelisted users, Twitter restricts a user to 150 requests/hour. Furthermore, searches are limited to returning 1500 posts for a given request. Our summarizer has been shown to produce comparable automated summaries to human summaries with as few as 100 posts.

3.2 Phrase Reinforcement Algorithm

Given a trending topic, one can query Twitter.com for posts that contain the topic phrase. Presently, users would have to read these posts in order to comprehend and manually summarize their content. Instead, we automate this process using our Phrase Reinforcement Algorithm.

The central idea of the Phrase Reinforcement (PR) algorithm is to find the most commonly used phrase that encompasses the topic phrase. This phrase is then used as a summary. The algorithm was inspired from two simple observations: (1) users will often use the same word or sets of words adjacent to the topic phrase when describing a key idea and (2) users will often "re-tweet" (a Twitter form of quoting) the most relevant content for a trending topic. These two patterns create highly overlapping sequences of words when considering a large number of posts for a single topic. The PR algorithm capitalizes on these behaviors in order to generate a summary.

The Phrase Reinforcement algorithm begins with a starting phrase. This is typically a trending topic, but can be non-trending as well. Given the starting phrase, the PR algorithm submits a query to Twitter.com for a list of posts that each contains the phrase. Once the posts are retrieved, the algorithm filters the posts to remove any spam or other sources of irrelevant data (e.g. hyperlinks). Filtering is an important step in order to focus the algorithm on the most relevant content. We filter any spam by using a Naïve Bayes classifier which we trained using previously gathered spam content from Twitter.com. Next, non-English posts as well as duplicate posts are removed since we are concerned with English summaries only and want to prevent a single user from dominating a topic. Finally, given a set of relevant posts, we isolate the longest sentence from each post that contains the topic phrase. These sentences form the input into the PR algorithm.

Once we have the set of input sentences, the PR algorithm formally begins. The algorithm starts by building a graph representing the common sequences of words (i.e. phrases) that occur both be-

fore and after the topic phrase. The graph is generated such that it centers about a common root node representing the topic phrase. Adjacent to the root node are chains of common sequences of words found within the input sentences. In particular, each word is represented by a node and an associated count that indicates how many times the node's phrase occurs within the set of input sentences. The phrase of a node is simply the sequence of words generated by following the path from the node to the root node. To illustrate, consider the following set of input sentences for the topic "Ted Kennedy".

1. A tragedy: Ted Kennedy died today of cancer
2. Ted Kennedy died today
3. Ted Kennedy was a leader
4. Ted Kennedy died at age 77

Using these sentences, the PR algorithm would generate a graph similar to the one shown below in Figure 1.

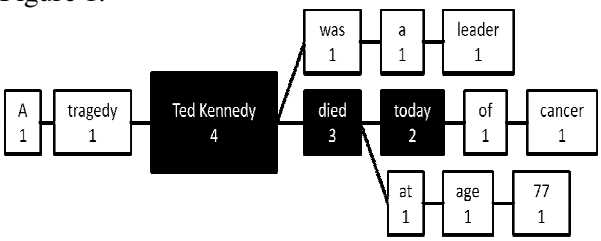

Figure 1. Example Phrase Reinforcement Graph.

In Figure 1, we see the node "today" has a count of two. This indicates that the phrase "Ted Kennedy died today" occurs exactly two times within the set of input sentences (in sentences 1 and 2). Likewise, the node "tradegy" has a count of one indicating the phrase "tragedy Ted Kennedy" only occurs one time (in sentence 1). In actuality, the PR algorithm would only add nodes to the graph with a count of at least two since it is looking for the most *common* phrase. These are shown as the black nodes in Figure 1. However, Figure 1 also includes unique nodes (shown in white) for helping illustrate the graph's structure.

After the graph is constructed, the PR algorithm assigns a weight to every node in order to prevent longer phrases from dominating the output. In particular, stop words are given a weight of zero while remaining words are given weights that are both proportional to their count and penalized the farther they are from the root node:

$$Weight(node) = Count(node) - [RootDistance(node) * \log_b Count(node)]$$

In the above equation, the *RootDistance* of a node is simply the number of hops to get from the node to the root node and the logarithm base, b, is a parameter to the algorithm. Smaller values of b (e.g. 2) can be used for preferring shorter summaries over longer summaries.

Finally, once the graph is constructed and weighted, the PR algorithm is ready to generate a *partial summary*. To do so, the PR algorithm searches for the path with the most total weight by searching all paths that begin with the root node and end with a non-root node. This path is denoted as the best partial path since it only represents one half of the summary (i.e. the most common phrase occurring either before or after the topic phrase). In order to generate the remaining half of the summary, the PR algorithm is essentially repeated by initializing the root node with the partial summary and rebuilding the graph. The most heavily weighted path from this new graph is the final summary produced by the PR algorithm.

Using our example above and assuming that node weights are equal to their counts, the path with the most total weight is the path "Ted Kennedy died today of cancer" with a total weight of 11. This phrase would then be used as the root node of a new graph and the PR algorithm would be repeated. For this new graph, the only input sentence that contains this root phrase would be sentence 1. Therefore, the final summary for our example (assuming we allow unique phrases) would be sentence 1: "A tragedy: Ted Kennedy died today of cancer". However, if we only allow non-unique phrases in our graph (the black nodes), then our final summary would be "Ted Kennedy died today".

4 Results

In order to evaluate the PR algorithm, we gathered a set of testing data by collecting the top ten currently trending topics from Twitter's home page every day for five consecutive days. For each of the 50 trending topics, we retrieved the maximum

number of posts from Twitter using its own API and then filtered the number of posts to 100 posts per topic. These posts were then given to two volunteers. The volunteers were instructed to simply generate the best summary possible using only the information contained within the posts and in 140 characters or less. Furthermore, automated summaries for each topic were also produced using the same 100 posts per topic. These summaries were then compared.

For comparing the manual and automated summaries, we adopted two of the metrics used by the Document Understanding Conference (DUC) of 2002 and 2004 (Lin and Hovy, 2003). First, we used their *Content* metric which asks a human judge to measure how completely an automated summary expresses the meaning of the manual summaries on a five point scale where 1 represents no meaning overlap and 5 represents complete meaning overlap. Next, we also used the automated evaluation metric ROUGE-1 developed by Lin (2004) which measures co-occurring unigram overlap between a set of manual and automated summaries. We restricted our automated evaluation to ROUGE-1 as opposed to the other ROUGE metrics since Lin indicates that this metric correlates highly with human judgments for very short summary tasks similar to the one we are performing (Lin, 2004).

For the 50 trending topics we used as our evaluation corpus, the PR algorithm produced an average Content score of 3.72 using $b = 100$ for our weighting measure. This result indicates our automated summaries express slightly less than most of the meaning of the manual summary content. To compare, we also used this same metric on our two sets of manual summaries which produced an average content score of 4.25. For the ROUGE-1 metric, the PR algorithm produced an average precision score of 0.31 and an average recall score of 0.30. Combining these scores using F_1-Measure, the PR algorithm produced a combined F_1 score of 0.30. Comparing the manual summaries against one another using ROUGE-1, they produced the same average precision, recall, and F_1 score of 0.34.

5 Future Work

Presently, we are working on extending our PR algorithm to providing real-time summaries within specific topics. We are experimenting with using a front-end classifier for producing trending topics within distinct categories and then summarizing around these topics in order to generate an automated real-time newspaper.

Acknowledgments

The work reported in this paper is partially supported by the NSF Grant ARRA: : CNS 0851783.

References

Brandow, R., Mitze K., and Rau, L.F. Automatic Condensation of Electronic Publications by Sentence Selection, Information Processing and Management, Vol 31, No 5, pp. 675-685, 1995.

Hu, M. and Sun, A. and Lim, E.P. Comments-oriented blog summarization by sentence extraction, ACM CIKM, pp. 901-904, 2007.

Kalita, J.K., Jones, M.L., and McCalla, G.I., Summarizing Natural Language Database Responses, Computational Linguistics, Volume 12, No. 2, pp. 107-124, 1986.

Kolcz, A., Prabhakarmurthi, V, and Kalita, J. Summarizing as Feature Selection for Text Categorization, CIKM '01, pp. 365-370, 2001.

Lin, C.Y. ROUGE: a Package for Automatic Evaluation of Summaries, Proceedings of Workshop on Text Summarization, 2004.

Lin, C.Y. and Hovy, E. Automatic evaluation of summaries using n-gram co-occurrence statistics, NAACL, pp. 71-78, 2003.

Luhn, P. The Automatic Creation of Literature Abstracts, in IRE National Convention, pp. 60-68, 1958.

Mahesh, K. Hypertext Summary Extraction for Fast Document Browsing, Working Notes of the AAAI Spring Symposium for the WWW, pp. 95-103, 1997.

Pear Analytics. Twitter Study, Retrieved 03 31, 2010, from http://www.scribd.com/doc/18548460/Pear-Analytics-Twitter-Study-August-2009, 2009.

Zhou, L. and Hovy, E. On the summarization of dynamically introduced information: Online discussions and blogs, AAAI-2006 Spring Symposium on Computational Approaches to Analyzing Weblogs, 2006.