Experiments in Microblog Summarization

Beaux Sharifi, Mark-Anthony Hutton and Jugal K. Kalita
Department of Computer Science
University of Colorado
Colorado Springs, CO 80918 USA
{bsharifi,mhutton86}@gmail.com, jkalita@uccs.edu

Abstract—**This paper presents algorithms for summarizing microblog posts. In particular, our algorithms process collections of short posts on specific topics on the well-known site called Twitter and create short summaries from these collections of posts on a specific topic. The goal is to produce summaries that are similar to what a human would produce for the same collection of posts on a specific topic. We evaluate the summaries produced by the summarizing algorithms, compare them with human-produced summaries and obtain excellent results.**

I. Introduction

Twitter, the microblogging site started in 2006, has become a social phenomenon, with more than 20 million visitors each month. While the majority posts are conversational or not very meaningful, about 3.6% of the posts concern topics of mainstream news[1]. At the end of 2009, Twitter had 75 million account holders, of which about 20% are active[2]. There are approximately 2.5 million Twitter posts per day[3]. To help people who read Twitter posts or tweets, Twitter provides a short list of popular topics called *Trending Topics*. Without the availability of such topics on which one can search to find related posts, for the general user, there is no way to get an overall understanding of what is being written about in Twitter.

Twitter works with a website called WhatTheTrend[4] to provide definitions of trending topics. WhatTheTrend allows users to manually enter descriptions of why a topic is trending. However, WhatTheTrend suffers from spam and irrelevant posts that reduce its utility to some extent.

In this paper, we discuss an effort to automatically create "summary" posts of Twitter trending topics. In short, we perform searches for Twitter on trending topics, get a large number of posts on a topic and then automatically create a short post that is representative of all the posts on the topic. We create summaries in various ways and evaluate them using metrics for automatic summary evaluation.

II. Related Work

Automatically summarizing microblog topics is a new area of research and to the authors' best knowledge, approaches have not been published. However, summarizing microblogs can be viewed as an instance of automated text summarization, which is the problem of automatically generating a condensed version of the most important content from one or more documents for a particular set of users or tasks [1].

As early as the 1950s, Luhn was experimenting with methods for automatically generating extracts of technical articles [2]. Edmundson [3] developed techniques for summarizing a diverse corpora of documents to help users evaluate documents for further reading. Early research in text summarization focused primarily on simple statistical techniques that relied upon lexical features such as word frequencies (e.g. [2]) or formatting clues such as titles and headings (e.g. [3]).

Later work integrated more sophisticated approaches such as machine learning, and natural language processing. In most cases, text summarization is performed for the purposes of saving users time by reducing the amount of content to read. However, text summarization has also been performed for purposes such as reducing the number of features required for classifying (e.g. [4]) or clustering (e.g. [5]) documents. Following another line of approach, [6] and [7], generated textual summaries of results of database queries. With the growth of the Web, interest has grown to improve summarization while also to summarize new forms of documents such as web pages (e.g. [8]), emails (e.g. [9]–[11]), newsgroups (e.g. [12]) discussion forums (e.g. [13]–[15]), and blogs (e.g. [16], [17]). Recently, interest has emerged in multiple document summarization as well (e.g. [18]–[21]) thanks in part to annual conferences that aim to further the state of the art in summarization by providing large test collections and common evaluation of summarizing systems[5].

III. Problem Description

On a microblogging service such as Twitter, a user can perform a search for a topic and retrieve a list of posts that contain the phrase. The difficulty in interpreting the results is that the returned posts are only sorted by recency. The motivation of the summarizer is to automate this process and generate a more representative summary in less time and effort.

Most search engines built into microblogging services only return a limited number of results when querying for a particular topic or phrase. For example, Twitter only returns a maximum of 1500 posts for a single search phrase. Our problem description is as follows:

[1]http://www.pearanalytics.com/blog/tag/twitter/
[2]http://themetricsystem.rjmetrics.com/2010/01/26/new-data-on-twitters-users-and-engagement/
[3]http://blog.twitter.com/2010/02/measuring-tweets.html
[4]http://www.whatthetrend.com/
[5]http://www.nist.gov/tac/

Given a set of posts that are all related by containing a common search phrase (e.g. a topic), generate a summary that best describes the primary gist of what users are saying about that search phrase.

IV. Selected Approaches

The first decision in developing a microblog summarization algorithm is to choose whether to use an abstractive or extractive approach. We choose an extractive approach since its methodologies more closely relate to the structure and diversity of microblogs. Abstractive approaches are beneficial in situations where high rates of compression are required. However, microblogs are the antithesis to long documents. Abstractive systems usually perform best in limited domains since they require outside knowledge sources. These approaches might not work so well with microblogs since they are unstructured and diverse in subject matter. Extractive techniques are also known to better scale with more diverse domains [22].

We implement several extractive algorithms. We start by implementing two preliminary algorithms based on very simple techniques. We also develop and implement two primary algorithms. First, we create the novel Phrase Reinforcement algorithm that uses a graph to represent overlapping phrases in a set of related microblog sentences. This graph allows the generation of one or more summaries and is discussed in detail in Section VI-A. We develop another primary algorithm based on a well established statistical methodology known as TF-IDF. The Hybrid TF-IDF algorithm is discussed at length in Section VI-B.

V. Naïve Algorithms

While the two preliminary approaches are simplistic, they serve a critical role towards allowing us to evaluate the results of our primary algorithms since no prior results yet exist.

A. Random Approach

Given a filtered collection of relevant posts that are each related to a single topic, we generate a summary by simply choosing at random either a post or a sentence.

B. Length Approach

Our second preliminary approach serves as an indicator of how easy or difficult it is to improve upon the random approach to summarization. Given a collection of posts and sentences that are each related to a single topic, we generate four independent summaries. For two of the summaries, we choose both the shortest and longest post from the collection. For the remaining two, we choose both the shortest and longest topic sentence from the collection.

VI. Primary Algorithms

We discuss two primary algorithms: one is called the Phrase Reinforcement Algorithm that we develop. The other is an adaptation of the well-known TF-IDF approach.

A. The Phrase Reinforcement Algorithm

The Phrase Reinforcement (PR) algorithm generates summaries by looking for the most commonly occurring phrases. The algorithm has been discussed briefly in [23], [24]. More details can be found in [25].

1) Motivation: The algorithm is inspired by two observations. The first is that users tend to use similar words when describing a particular topic, especially immediately adjacent to the topic phrase. For example, consider the following posts collected on the day of the comedian Soupy Sales' death:

1) *Aw, Comedian Soupy Sales died.*
2) *RIP Comedian Soupy Sales dies at age 83.*
3) *My favorite comedian Soupy Sales died.*
4) *RT @NY: RIP Comedian Soupy Sales dies at age 83.*
5) *RIP: Soupy Sales Died Today.*
6) *Soupy Sales meant silliness and laughs.*

Notice that all posts contain words immediately after the phrase *Soupy Sales* that in some way refer to his death. Furthermore, posts 2 and 4 share the word *RIP* and posts 1, 3 and 5 share the word *died*. Therefore, there exists some overlap in word usage adjacent to the phrase *Soupy Sales*.

The second observation is that microbloggers often repeat the most relevant posts for a trending topic by quoting others. Quoting uses the following form: *RT @[TwitterAccountName]: Quoted Message*. *RT* refers to *Re-Tweet* and indicates one is copying a post from the indicated Twitter account. For example, the following is a quoted post: *RT @dcagle: Our first Soupy Sales RIP cartoon.* While users writing their own posts occasionally use the same or similar words, retweeting causes entire sentences to perfectly overlap with one another. This, in turn, greatly increases the average length of an overlapping phrase for a given topic. The main idea of the algorithm is to determine the most heavily overlapping phrase centered about the topic phrase. The justification is that repeated information is often a good indicator of its relative importance [2].

2) The Algorithm: The algorithm begins with the topic phrase. These phrases are typically trending topics, but can be other non-trending topics as well. Assume our starting phrase is the trending topic *Soupy Sales*. The input to the algorithm is a set of posts that each contains the starting phrase. The algorithm can take as input any number of posts returned by Twitter. In the running example, we pick the first 100 posts returned by the Twitter search engine.

a) Building a Word Graph: The root node contains the topic phrase, in this case *soupy sales*. We build a graph showing how words occur before and after the phrase in the root node, considering all the posts on the topic. We think of the graph as containing two halves: a sub-graph to the left of the root node containing words occurring in specific positions to the left of the root node's phrase, and a similar sub-graph to the right of the root node.

To construct the left-hand side, the algorithm starts with the root node. It reduces the set of input sentences to the set of sentences that contain the current node's phrase. The current node and the root node are initially the same. Since every input sentence is guaranteed to contain the root phrase, our list of sentences does not change initially. Subsequently, the algorithm isolates the set of words that occur immediately before the current node's phrase. From this set, duplicate words are combined and assigned a count that represents how many instances of those words are detected. For each of these unique words, the algorithm adds them to the graph as nodes with their associated counts to the left of the current node.

In the graph, all the nodes are in lower-case and stripped of any non-alpha-numeric characters. This increases the amount of overlap among words. Each node has an associated count. The associated node's phrase has exactly count number of occurrences within the set of input sentences at the same position and word sequence relative to the root node. Nodes with a count less than two are not added to the graph since the algorithm is looking for overlapping phrases. The algorithm continues this process recursively for each node added to the graph until all the potential words have been added to the left-hand side of the graph.

The algorithm repeats these steps symmetrically for the right-hand side of the graph. At the completion of the graph-building process, the graph looks like the one in Figure 1.

b) Weighting Individual Word Nodes: The algorithm prepares for the generation of summaries by weighting individual nodes. Node weighting is performed to account for the fact that some words have more informational content than others. For example, the root node *soupy sales* contains no information since it is common to every input sentence. We give it a weight of zero. Common stop words are noisy features that do not help discriminate between phrases. We give them a weight of zero as well. Finally, for the remaining words, we first initialize their weights to the same values as their counts. Then, to account for the fact that some phrases are naturally longer than others, we penalize nodes that occur farther from the root node by an amount that is proportional to their distance:

$$Weight(Node) = \\ Count(Node) - Distance(Node) * log_b\, Count(Node). \tag{1}$$

The logarithm base b can be used to tune the algorithm towards longer or shorter summaries. For aggressive summarization (higher precision), the base can be set to small values (e.g. 2 or the natural logarithm ln). While for longer summaries (higher recall), the base can be set to larger values (e.g. 100). Weighting our example graph gives the graph in Figure 2. We assume the logarithm base b is set to 10 for helping generate longer summaries.

c) Generating Summaries: The algorithm looks for the most overlapping phrase within the graph. First, we create the left partial summary by searching for all paths (using a depth-first search algorithm) that begin at the root node and end at any other node, going backwards. The path with the most weight is chosen. Assume in Figure 2, the path with most weight on the left-hand side of the root node (here 5.1 including the root node) on the left side of the graph contains the nodes *rip*, *comedian* and *soupy sales*. Thus, the best left partial summary is *rip comedian soupy sales*.

We repeat the partial summary creation process for the right-hand side of the current root node *soupy sales*. Since we want to generate phrases that are actually found within the input sentences, we reorganize the tree by placing the entire left partial summary, *rip comedian soupy sales* in the root node. Assume we get the path shown in Figure 3 as the most heavily weighted on the right hand side. The full summary generated by the algorithm for our example is: *rip comedian soupy sales dies at age 83*.

Strictly speaking, we can build either the left or right partial summary first, and the other one next. The full summary obtained by the algorithm is the same in both cases.

d) Post-processing: This summary has lost its case-sensitivity and formatting since these features were removed to increase the amount of overlap among phrases. To recover these features, we perform a simple best-fit algorithm between the summary and the set of input sentences to find a matching phrase that contains the summary. We know such a matching phrase exists within at least two of the input sentences since the algorithm only generates summaries from common phrases. Once, we find the first matching phrase, this phrase is our final summary. It produces the following final summary:

RIP Comedian Soupy Sales dies at age 83.

B. Hybrid TF-IDF Summarization

After analyzing the results obtained by the Phrase Reinforcement approach, we notice that it significantly improves upon our earlier results, but it still leaves room for improvement as its performance only halves the difference between the random and manual summarization methods (see Section VII-E). Therefore, we use another approach based upon a technique dating back to early summarization work [2].

1) TF-IDF: Term Frequency-Inverse Document Frequency, is a statistical weighting technique that has been applied to many information retrieval problems. It has been used for automatic indexing [26], query matching of documents [27], and automated summarization [28]. Generally TF-IDF is not known as a leading algorithms in automated summarization.

For straightforward automated summarization, the application of TF-IDF is simplistic. The idea is to assign each sentence within a document a weight that reflects the sentence's saliency within the document. The sentences are ordered by their weights from which the top m sentences with the most weight are chosen as the summary. The weight of a sentence is the summation of the individual term weights within the sentence. Terms can be words, phrases, or any other type of lexical feature [29]. To determine the weight of a term, we use the formula:

$$TF\_IDF = tf_{ij} * log_2 \frac{N}{df_j} \tag{2}$$

where tf_{ij} is the frequency of the term T_j within the document D_i, N is the total number of documents, and df_j is the number of documents within the set that contain the term T_j [26].

The TF-IDF value is composed of two primary parts. The term frequency component (TF) assigns more weight to words

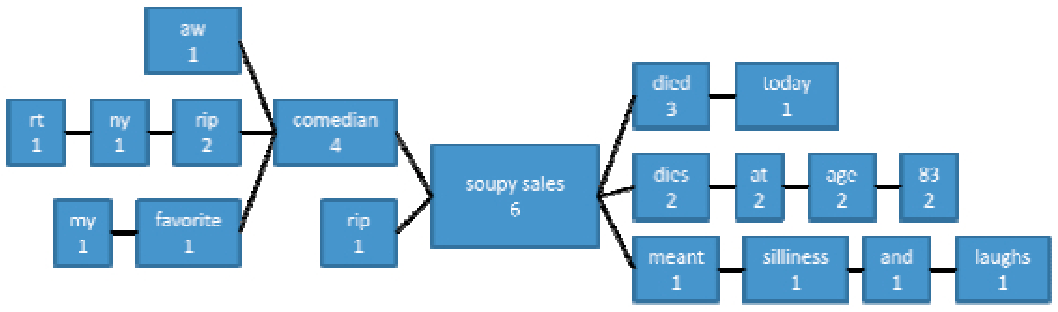

Fig. 1. Fully constructed PR graph (allowing non-overlapping words/phrases).

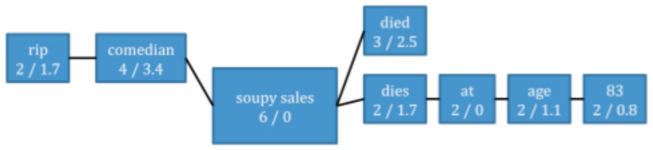

Fig. 2. Fully weighted PR graph (requiring overlapping words/phrases).

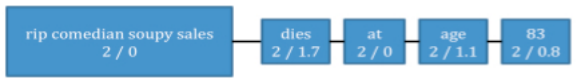

Fig. 3. Fully constructed PR graph for right half of summary). This also demonstrates best complete summary

that occur frequently within a document because important words are often repeated [2]. The inverse document frequency component (IDF) compensates for the fact that some words such as common stop words are frequent. Since these words do not help discriminate between one sentence or document over another, these words are penalized proportionally to their inverse document frequency (the logarithm is taken to balance the effect of the IDF component in the formula). Therefore, TF-IDF gives the most weight to words that occur most frequently within a small number of documents and the least weight to terms that occur infrequently or occur within the majority of the documents.

One noted problem with TF-IDF is that the formula is sensitive to document length. Singhal et al. note that longer documents have higher term frequencies since they often repeat terms while also having a larger number of terms [29]. This doesn't have any ill effects on single document summarization. However, when generating a summary from multiple documents this becomes an issue because the terms within the longer documents have more weight.

C. Algorithm

Equation 2 defines the weight of a term in the context of a document. However, we don't have a traditional document. Instead, we have a set of microblogging posts that are each related to a topic. So, one question we must first answer is how we define a document. An option is to define a single document that encompasses all the posts together. In this case, the TF component's definition is straightforward since we can compute the frequencies of the terms across all the posts. However, doing so causes us to lose the IDF component since we only have a single document. On the other extreme, we could define each post as a document making the IDF component's definition clear. But, the TF component now has a problem: Because each post contains only a handful of words, most term frequencies will be a small constant for a given post.

To handle this situation, we redefine TF-IDF in terms of a hybrid document. We primarily define a document as a single sentence. However, when computing the term frequencies, we assume the document is the entire collection of posts. This way, we have differentiated term frequencies but also don't lose the IDF component. A term is a single word in a sentence.

We next choose a normalization method since otherwise the TF-IDF algorithm always biases towards longer sentences. We normalize the weight of a sentence by dividing it by a normalization factor. Since stop words do not help discriminate the saliency of sentences, we give each of these types of words a weight of zero by comparing them with a prebuilt list. We summarize this algorithm below in Equations (3)-(7).

$$W(S) = \frac{\sum_{i=0}^{\#WordsInSentence} W(w_i)}{nf(S)} \tag{3}$$

$$W(w_i) = tf(w) * log_2(idf(w_i)) \tag{4}$$

$$tf(w_i) = \frac{\#OccurrencesOfWordInAllPosts}{\#WordsInAllPosts} \tag{5}$$

$$idf(w_i) = \frac{\#SentencesInAllPosts}{\#SentencesInWhichWordOccurs} \tag{6}$$

$$nf(S) = max[MinimumThreshold, \#WordsInSentence] \tag{7}$$

where W is the weight assigned to a sentence or a word, nf is a normalization factor, w_i is the ith word, and S is a sentence.

VII. Experimental Setup and Evaluation

A. Data Collection and Pre-processing

For five consecutive days, we collected the top ten currently trending topics from Twitter's home page at roughly the same time every evening. For each topic, we downloaded the maximum number (approximately 1500) of posts. Therefore, we had 50 trending topics with a set of 1500 posts for each.

We performed several forms of preprocessing in order to filter the posts into a usable form. Details can be found in [25].

B. Evaluation Methods

There is no definitive standard against which one can compare the results from an automated summarization system. Summary evaluation is generally performed using one of two methods: intrinsic, or extrinsic. In intrinsic evaluation, the quality of the summary is judged based on direct analysis using a number of predefined metrics such as grammaticality, fluency, or content [1]. Extrinsic evaluations measure how well a summary enables a user to perform some form of task [22].

To perform intrinsic evaluation, a common approach is to create one or more manual summaries and to compare the automated summaries against the manual summaries. One popular automatic evaluation metric that has been adopted by the Document Understanding Conference since 2004 is ROUGE. ROUGE is a suite of metrics that automatically measures the similarity between an automated summary and a set of manual summaries [30]. One of the simplest ROUGE metrics is the ROUGE-N metric:

$$ROUGE\text{-}N = \frac{\sum_{s \in MS} \sum_{n\text{-}grams \in S} Match(n\text{-}gram)}{\sum_{s \in MS} \sum_{n\text{-}grams \in S} Count(n\text{-}gram)}. \tag{8}$$

Here, n is the length of the n-grams, $Count(n\text{-}gram)$ is the number of n-grams in the manual summary, and $Match(n\text{-}gram$ is the number of co-occurring n-grams between the manual and automated summaries.

Lin [30] performed evaluations to understand how well different forms of ROUGE's results correlate with human judgments. One result of particular consequence for our work is his comparison of ROUGE with the very short (around 10 words) summary task of DUC 2003. Lin found ROUGE-1 and other ROUGEs to correlate highly with human judgments. Since this task is very similar to creating microblog summaries, we implement ROUGE-1 as a metric.

Since we want certainty that ROUGE-1 correlates with a human evaluation of automated summaries, we also implement a manual metric used during DUC 2002: the Content metric which asks a human judge to measure how completely an automated summary expresses the meaning of a human summary on a $1 \cdots 5$ scale, 1 being worst and 5 being best.

C. Manual Summaries

Two volunteers generated a complete set of 50 manual "best" summaries possible for all topics in 140 characters or less while using only the information contained within the posts (see Table 1).

1) Manual Summary Evaluation: The manual summaries generated by our two volunteers are semantically very similar to one another but have different lengths and word choices. We use ROUGE-1 to compare the manual summaries against one another. We do the same using the Content metric in order to understand how semantically similar the two summaries are. By evaluating our two manual summaries against one another, we help establish practical upper limits of performance for

TABLE I
EXAMPLES OF MANUAL SUMMARIES

Topic	Manual Summary 1	Manual Summary 2
#BeatCancer	Every retweet of #BeatCancer will result in 1 cent being donated towards Cancer Research.	Tweet #beatcancer to help fund cancer research
Kelly Clarkson	Between Taylor Swift and Kelly Clarkson, which one do you prefer.....?	Taylor Swift v. Kelly Clarkson

TABLE II
ROUGE-1, CONTENT, AND LENGTH FOR MANUAL SUMMARIES

	Rouge-1			Content	Length
	F	Precision	Recall		
Manual 1	0.34	0.31	0.37	4.4	11
Manual 2	0.34	0.37	0.31	4.1	9
Manual Avg	0.34	0.34	0.34	4.2	10

TABLE III
CONTENT PERFORMANCE FOR NAIVE SUMMARIES

	Content Performance
Manual Average	4.2
Random Sentence	3.0
Shortest Sentence	2.3

automated summaries. These results in addition to the results of the preliminary algorithms collectively establish a range of expected performance for our primary algorithms. We compare the manual summaries against one another bi-directionally by assuming either set was the set of automated summaries.

To generate the Content performance, we asked a volunteer to evaluate how well one summary expressed the meaning of the corresponding manual summary. The average results for computing the ROUGE-1 and Content metrics on the manual summaries are shown in Table II.

D. Performance of Naïve Algorithms

The generation of random sentences produces an average recall of 0.23, an average precision of 0.22, and an F-measure of 0.23. These results are higher than we originally anticipated given our average manual F-measure is only 0.34. Some overlap is explained by that ROUGE-1 measures common words. Second, while we call our first preliminary approach "random", we introduce some bias into this approach by our preprocessing steps discussed earlier.

To understand how random sentences are compared to manual summaries, we present Content performance in Table III. The random sentence approach generated a Content score of 3.0 on a scale of 5. In addition to choosing random sentences, we also chose random posts. This slightly improves the recall scores over the random sentence approach (0.24 vs. 0.23), but worsens the precision (0.17 vs. 0.22).

The random post approach produces an average length of 15 words while the random sentence averaged 12 words. Since the random sentence approach is closer to the average manual summary length, it scores higher precision. Overall, the random sentence approach produces a summary that is more balanced in terms of recall and precision and a higher F-measure as well (0.23 vs. 0.20).

TABLE IV
ROUGE-1, CONTENT AND LENGTH FOR THE PR SUMMARIES

	Rouge-1			Content	Length
	F	Precision	Recall		
Manual Average	0.34	0.34	0.34	4.2	10
Random Sentence	0.23	0.22	0.23	3.0	12
PR Phrase (log 100)	0.30	0.31	0.30	3.7	12

TABLE V
SUMMARIES PRODUCED BY THE PHRASE REINFORCEMENT ALGORITHM

Topic	PR Phrase (log 100)
#iusuallylieabout	#iusuallylieabout my age
A-Rod	A-Rod homers in third straight game
National League	Phillies defeat Dodgers to take the National League Championship Series.
Angels	#Dodgers lost 11-0, #Angels lost 10-1.
Dodgers	Phillies dump Dodgers for World Series return
Apple Fires Back	Apple Fires Back at Windows 7 in New Ads - Apple's "I'm a PC" and "I'm a Mac" dynamic ad duo are at it again i...

The length-based second preliminary approach is unsurprisingly disappointing. The shortest sentence and shortest post approaches generate far too short summaries, averaging only two or three words in length. Because of their short length, these two approaches generate very high precision but fail horribly at recall, scoring less than either of the random approaches. Choosing the longest sentence and longest post has the exact opposite problem. These two approaches generate fairly good recall (approximately the average manual recall), but very low precision.

These results demonstrate a limitation of ROUGE-1: to achieve a high combined ROUGE-1 score, the number of tokens in the automated and manual summaries must be similar to each other since ROUGE-1 is simply comparing unigrams.

E. Performance of Phrase Reinforcement Algorithm

Table IV displays ROUGE-1 performance for the PR algorithm using a logarithm base of 100 for the weight of a node as described in Equation 1. The algorithm produces an average recall of 0.30, an average precision of 0.31, and a combined F1-measure of 0.30. This is a significant improvement over the random sentence approach. However, it still leaves some room for improvement since manual summaries had F1-measure of 0.34.

The PR algorithm generates an average Content metric score of 3.66 (see Table IV). This score is an improvement over the random summaries. Table IV also indicates that the PR summaries are on average only two words longer in length than the average manual summary and equal to the length of the average random sentence.

A sample of summaries is presented below in Table V.

The PR algorithm is able to tailor the length of its summaries by adjusting the weights of nodes as defined by Equation 1. For example, by assigning less weight to nodes farther from the root phrase, the algorithm prefers more common shorter phrases over less common longer phrases.

In figures 4 and 5, we can see the effect of varying the weighting parameter b within Equation 1. There appears to be a threshold (when $b \approx 100$) for which smaller values of b begin reducing the average summary length. As b decreases, the algorithm begins trading ROUGE-1 recall performance for precision performance and reducing the Content performance as well. Above this threshold, the average summary length does not increase while the ROUGE-1 and Content performance begins to degrade slightly. In Figure 4, the label "PR Phrase (NULL)" indicate the absence of the weighting parameter all together. In these cases, we simply use a nodes count as its weight

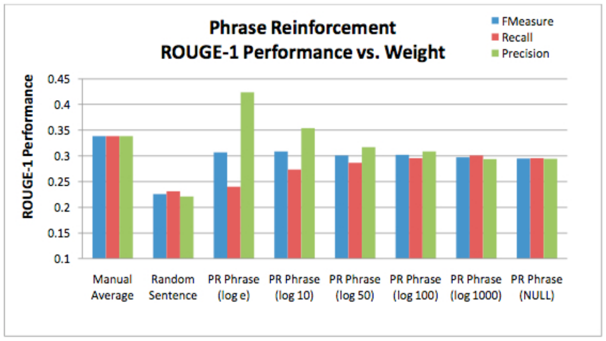

Fig. 4. ROUGE-1 performance for different weightings of the Phrase Reinforcement algorithm

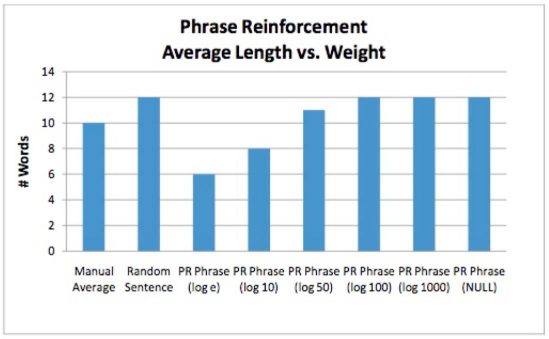

Fig. 5. Average summary lengths for different weightings of the Phrase Reinforcement algorithm

F. Performance of the Hybrid TF-IDF Algorithm

In Figure 7, we present the results of the TF-IDF algorithm for the ROUGE-1 metric. The TF-IDF results are denoted as TF-IDF Sentence (11) to distinguish the fact that the

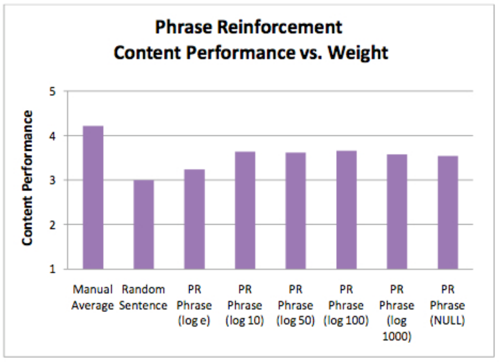

Fig. 6. Content performance for different weightings of the Phrase Reinforcement algorithm

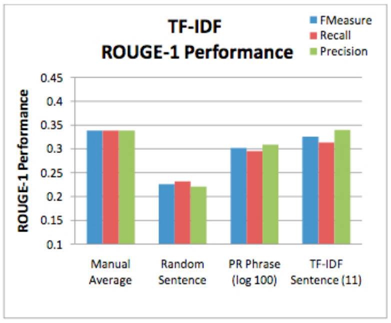

Fig. 7. Rouge performance for Hybrid TFIDF algorithm

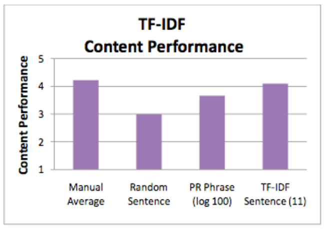

Fig. 8. Content performance for Hybrid TFIDF algorithm

TABLE VI
SUMMARIES PRODUCED BY THE HYBRID TFIDF ALGORITHM

Topic	TF-IDF Sentence (11)
#BeatCancer	Every tweet that includes #beatcancer raises money for cancer research.
DWTS	the michael jackson tribute on #dwts tonight was AWESOME!
Paranormal	Activity Paranormal Activity iz a scary movie!
#clubrules	#clubrules ladies don't wear smudgy make-up.
Balloon Boy	Balloon Boy's Mom Admitted Hoax to the Cops!

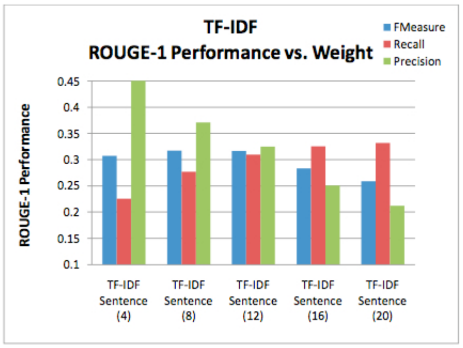

Fig. 10. ROUGE-1 performance vs. weight for Hybrid TFIDF algorithm

TF-IDF algorithm produces sentences instead of phrases for summaries and that we are using a threshold of 11 words as our normalization factor. The TF-IDF algorithm produces an average recall of 0.31, an average precision of 0.34, and a combined F1-Measure of 0.33. These values are very close to the performance levels of our manual summaries of 0.34. Furthermore, they are also better than our Phrase Reinforcement results.

The results of the Content evaluation are in Figure 8. Remarkably, the TF-IDF algorithm's Content performance is also very similar to the manual summaries. This score is also higher than the average Content score of the Phrase Reinforcement algorithm which was 3.66.

Figure 9 displays the average length of the TF-IDF summaries. Interestingly, the TF-IDF summaries are one word shorter, on average, than the manual summaries with an average length of 9 words. In fact, this is the exact average length of the second set of manual summaries.

Finally, we present a selection of summaries for the TF-IDF algorithm from our testing topics in Table VI.

1) Normalization: The TF-IDF equation is sensitive to a document's length. We normalize the length of a sentence rather than normalizing the term weights for two reasons: (1) the sentence length had a much greater influence than the over-

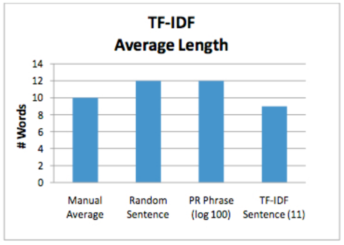

Fig. 9. Average summary length for Hybrid TFIDF algorithm

estimation of term weights and (2) we use a hybrid definition of a document which may or may not have been relevant to the standard normalization techniques. Prior to normalizing the length of a sentence, our average sentence length of a summary using TF-IDF was 20 words, much greater than our average manual sentence length of 10 words.

After many experiments, we settled on our normalization factor: the maximum of a minimum threshold and the number of words in a sentence. The minimum threshold is simply an integral number of words and approximately represents the desired average summary length. The normalization factor is then used as a divisor into the sentence weight. It enables us to control the average summary length. In fact, we can produce a spectrum of average summary lengths simply by using a range of minimum thresholds.

We next compute the average ROUGE-1 and Content performance levels for different minimum thresholds in order to determine an optimal threshold for our testing data. Since ROUGE-1 measures unigram overlap between the manual and automated summaries, our initial guess of an optimal threshold is one that produces an average summary length equal to the average manual summary length of 10 words. Exhaustive experiments show 11 to be the ideal length for our purpose.

As seen in Figure 10, by varying the normalization threshold, we are able to control the average summary length and resulting ROUGE-1 precision and recall. Furthermore, there appears to be an inflection point where the precision and recall performance levels cross at the threshold value of 12 words. Therefore, we can use the threshold to control whether we desire better precision or recall or a balance of these two values for the produced TF-IDF summaries. More interestingly, we don't necessarily need to compromise on Content performance while varying the summary length. Figures 11 and 12 show how content and ROUGE-1 perform as weight is changed.

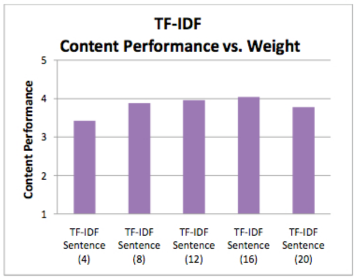

Fig. 11. Content performance vs. weight for Hybrid TFIDF algorithm

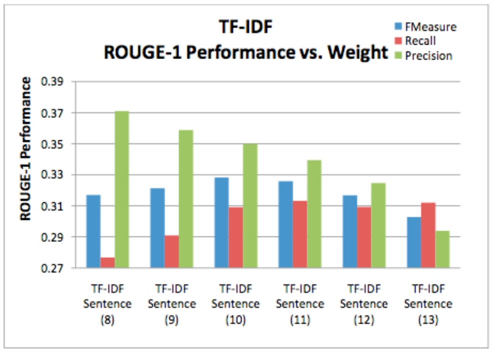

Fig. 12. ROUGE-1 performance vs. weight for Hybrid TFIDF algorithm for various normalization thresholds

VIII. Conclusion

In this paper, we have presented two primary approaches to microblog summarization. The graph-based Phrase Reinforcement algorithm finds the most common phrase on one side of the search phrase, selects those posts that contain this phrase, and then finds posts with the most common phrase on the other side as well. The Hybrid TF-IDF algorithm defines a document differently for different formulas used in the computation of a post's total weight. We find, after exhaustive experimentation, that the Hybrid TF-IDF algorithm produces as good summaries as the PR algorithm or even better.

We want to extend our work in various ways. One idea is to produce multi-sentence or multi-post summaries. We can do so using the PR algorithm or the Hybrid TF-IDF algorithm, by picking posts with the top n weighs. One challenge will be to produce a coherent multi-sentence summary because of issues such as presence of anaphora and other coherence issues. We also want to cluster the posts on a specific topic into k clusters to find various themes and sub-themes that are present in the posts and then find a summary that may be one-post or multi-post. Of course, one of the biggest problems we have observed with microblogs is redundancy; in other words, quite frequently, a large number of posts are similar to one another.

Acknowledgment

The work reported in this paper has been partially supported by NSF grant ARRA: 0851783

References

[1] J. Lin, M. Ozsu, and L. Liu, "Summarization," *Encyclopedia of Database Systems, Springer*, 2009.
[2] H. P. Luhn, "The automatic creation of literature abstracts," *IBM J. Res. Dev.*, vol. 2, no. 2, pp. 159–165, 1958.
[3] H. P. Edmundson, "New methods in automatic extracting," *J. ACM*, vol. 16, no. 2, pp. 264–285, 1969.
[4] A. Kolcz, V. Prabakarmurthi, and J. Kalita, "Summarization as feature selection for text categorization," in *10th CIKM*, 2001, pp. 365–370.
[5] V. Ganti, J. Gehrke, and R. Ramakrishnan, "Cactus—clustering categorical data using summaries," in *KDD '99*, pp. 73–83.
[6] J. Kalita, M. Colbourn, and G. McCalla, "A response to the need for summary responses," in *10th COLING and 22nd ACL Meeting*, 1984, pp. 432–436.
[7] J. Kalita, M. Jones, and G. McCalla, "Summarizing natural language database responses," *Computational Linguistics*, vol. 12, no. 2, pp. 107–124, 1986.
[8] K. Mahesh, "Hypertext summary extraction for fast document browsing," in *AAAI Spring Symposium on NLP for the World Wide Web*, 1997, pp. 95–103.
[9] D. Lam, S. Rohall, C. Schmandt, and M. Stern, "Exploiting e-mail structure to improve summarization," *Technical Paper at IBM Watson Research Center*, vol. 20, no. 02, 2002.
[10] O. Rambow, L. Shrestha, J. Chen, and C. Lauridsen, "Summarizing email threads," in *HLT/NAACL*, 2004, pp. 2–7.
[11] S. Wan and K. McKeown, "Generating overview summaries of ongoing email thread discussions," in *COLING*, 2004, p. 12.
[12] K. Lang, "Newsweeder: Learning to filter netnews," in *In Proceedings of the Twelfth International Conference on Machine Learning*, 1995.
[13] R. Farrell, "Summarizing electronic discourse," *Intelligent Systems in Accounting, Finance & Management*, vol. 11, no. 1, pp. 23–38, 2002.
[14] M. Klaas, "Toward indicative discussion fora summarization," Citeseer, Tech. Rep., 2005.
[15] A. Tigelaar, R. Op den Akker, and D. Hiemstra, "Automatic summarisation of discussion fora," *Natural Language Engineering*, vol. 16, no. 02, pp. 161–192, 2010.
[16] L. Zhou and E. Hovy, "On the summarization of dynamically introduced information: Online discussions and blogs," in *AAAI Spring Symposium on Computational Approaches to Analyzing Weblogs*, 2006.
[17] M. Hu, A. Sun, and E. Lim, "Comments-oriented blog summarization by sentence extraction," in *16th CIKM*, 2007, pp. 901–904.
[18] D. Radev, H. Jing, M. Sty, and D. Tam, "Centroid-based summarization of multiple documents," *Information Processing & Management*, vol. 40, no. 6, pp. 919–938, 2004.
[19] R. Barzilay, K. McKeown, and M. Elhadad, "Information fusion in the context of multi-document summarization," in *37th ACL Meeting*, 1999, pp. 550–557.
[20] J. Goldstein, V. Mittal, J. Carbonell, and M. Kantrowitz, "Multi-document summarization by sentence extraction," in *NAACL-ANLP Workshop on Automatic summarization*, 2000, pp. 40–48.
[21] R. Mihalcea and P. Tarau, "TextRank: Bringing order into texts," in *Proceedings of EMNLP*. Barcelona: ACL, 2004, pp. 404–411.
[22] U. Hahn and I. Mani, "The challenges of automatic summarization," *Computer*, pp. 29–36, 2000.
[23] B. Sharifi, M.-A. Hutton, and J. Kalita, "Automatic Summarization of Twitter Topics," in *National Workshop on Design and Analysis of Algorithm, Tezpur, India*, 2010.
[24] ——, "Summarizing Microblogs Automatically," in *NAACL-HLT*, June, 2010.
[25] B. Sharifi, "Automatic microblog classification and summaraization," Master's thesis, University of Colorado at Colorado Springs, 2010.
[26] G. Salton, *Automatic text processing: the transformation, analysis, and retrieval of information by computer*. Addison-Wesley Longman, 1989.
[27] C. Manning, P. Raghavan, and H. Schütze, "An introduction to information retrieval," 2008.
[28] Y. Seki, "Sentence Extraction by tf/idf and position weighting from Newspaper Articles," in *3rd NTCIR Workshop*, 2002.
[29] A. Singhal, C. Buckley, and M. Mitra, "Pivoted document length normalization," in *SIGIR '96*, 1996, pp. 21–29.
[30] C.-Y. Lin and E. Hovy, "Automatic evaluation of summaries using n-gram co-occurrence statistics," in *NAACL '03*, 2003, pp. 71–78.