Summarization of Twitter Microblogs

Beaux Sharifi*, David Inouye+ and Jugal K. Kalita*

*Department of Computer Science, University of Colorado, Colorado Springs, CO 80918
+Department of Computer Science, University of Texas at Austin, Austin, TX 78712
Email: {beaux.sharifi,davidinouye}@gmail.com, jkalita@uccs.edu

Due to the sheer volume of text generated by a microblog site like Twitter, it is often difficult to fully understand what is being said about various topics. This paper presents algorithms for summarizing microblog documents. Initially, we present algorithms that produce single-document summaries but later extend them to produce summaries containing multiple documents. We evaluate the generated summaries by comparing them to both manually produced summaries and, for the multiple-post summaries, to the summarization results of some of the leading traditional summarization systems.

Keywords: Twitter; Informal Text; Summarization

Received 00 January 2009; revised 00 Month 2009

1. INTRODUCTION

Twitter[1], the microblog site started in 2006, has become a social phenomenon. More than 340 million Tweets are sent out every day[2]. While a majority of posts are conversational or not particularly meaningful, about 3.6% of the posts concern topics of mainstream news[3]. Twitter has been credited with providing the most current news about many important events before traditional media, such as the attacks in Mumbai in November 2008. Twitter also played a prominent role in the unfolding of the troubles in Iran in 2009 subsequent to a disputed election, and the so-called Twitter Revolutions[4] in Tunisia and Egypt in 2010-11.

To help people who read Twitter posts or tweets, Twitter provides two interesting features: an API that allows users to search for posts that contain a topic phrase and a short list of popular topics called *Trending Topics*. A user can perform a search for a topic and retrieve a list of most recent posts that contain the topic phrase. The difficulty in interpreting the results is that the returned posts are only sorted by recency, not relevancy. Therefore, the user is forced to manually read through the posts in order to understand what users are primarily saying about a particular topic.

A website called WhatTheTrend[5] attempts to provide definitions of trending topics by allowing users to manually enter descriptions of why a topic is trending. Here is an example of a definition from WhatTheTrend:

Why is Justin Bieber popular right now? The young Canadian R&B/pop singer has a lot of fans who like to tweet about him!

The Twitter API only allows users to see the most recent posts on a topic in chronological order; it does not provide any relevancy based ordering of posts. In addition, WhatTheTrend is limited because it requires manual input and both the Twitter API and WhatTheTrend suffer from spam and irrelevant posts that reduce their utility to some extent. Otherwise, for the general user, there is no way to get an overall understanding of what is being written about on Twitter. The motivation to create the summarizer is to automate this process and generate a more representative summary in less time and effort.

Twitter has grown by about 1500% every year since its founding[6]. According to the same story, the Twitter API fielded about 19 billion searches a month in 2010 compared to about 90 billion for Google[7]. The massive rise of microblogging as a new form of communication and data generation has opened up a new domain for natural language processing that could be aimed at discovering real time public opinion, news stories or conversational topics. In order to understand and use this flood of information, we feel that just being able to search for a topic and receive the most recent posts whose text matches the keywords is not enough. To get a snapshot of what is being written about a topic in terms of a large number of posts, it is necessary to obtain a summary or a gist of these posts. If we use the traditional term *document* to refer to each post (or "tweet"), for our purposes, obtaining the summary of a large set of documents on a certain topic specified

[1]`http://www.twitter.com`
[2]`http://blog.twitter.com/2012/03/twitter-turns-six.htm`
[3]`http://www.pearanalytics.com/blog/tag/twitter/`
[4]`http://en.wikipedia.org/wiki/Twitter_Revolution`
[5]`http://www.whatthetrend.com`
[6]`http://www.networkworld.com/news/2010/041410-biz-stone-says-twitter-has.html`
[7]`http://www.google.com`

by keywords means selecting the most significant one or more documents from the set. It is also possible to construct new documents by piecing together parts of original documents or posts.

Data reduction or summarization is essential for understanding and exploring any data. Even if data summarization is only used in the initial phases of data analysis and exploration, it is critical for providing intuition about what questions would be interesting to ask and what other data analysis methods could be useful. In the case of microblogs, summarization is necessary before any other data analysis can be done because it would simply be impossible to read through millions of tweets. In addition, summarization can be a useful exploratory tool by itself. For example, a short summary could be attached to each trending topic and a user would then only have to read a short one page Twitter page to understand all the trending topics rather than reading thousands of somewhat random posts. Each of the topics on the summary page would be linked to the raw Twitter posts in case someone wanted to explore the topic further. Another motivating example of the utility of Twitter summarization is using Twitter for market research. Many companies are embedding unique hashtags (e.g. #AmericanIdol) in their advertisements to track what viewers are saying about a particular product or service. Combining automatic summarization with Twitters search API, companies could generate—in real time—condensed summaries about what the majority of users are saying about the hashtag topics they have defined. These could then be used for understanding what the majority of viewers are saying about their products of services with less time and effort. One can provide copious additional examples of situations where summarization of Twitter documents would be useful, although we do not do so since our main focus is the presentation and comparison of a number of algorithms for summarization of twitter documents.

In this paper, we discuss an effort to create "summary" documents for Twitter trending topics. In short, we perform a search on the Twitter API based on trending topics to get a large number of documents on a topic and then automatically create a summary that is representative of all the documents on the topic. We start by discussing algorithms to create single-document summaries. We evaluate the single-document summaries using two metrics for automatic summary evaluation. We then extend the work to obtain summaries containing multiple documents since a single document may not be able to reflect the subtopics surrounding a certain trending topic. We compare our multi-document summaries with ones produced by leading multi-document summarizers.

2. RELATED WORK

Although automatic summarization of longer documents has been researched extensively, processing short and informal microblog posts has only recently been considered. Automatically summarizing microblog topics is a new area of research. Of course, summarizing microblogs can be viewed as an instance of the more general problem 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 [1–3].

As early as the 1950's, Luhn was experimenting with methods for automatically generating extracts of technical articles using surface features such as word frequency [4]. Edmundson developed techniques that added positional features such as formatting clues like titles and headings for summarizing a diverse corpora of documents to help users evaluate documents for further reading [5]. Other early summarizing programs include the FRUMP system [6], which analyzed UPI news stories in about 50 domains and summarized them using a script-based extraction of relevant information in the specific domains. The TOPIC system [7] created a hierarchical text graph from an input document and then condensed the text based on the topics to form summaries. The SCISOR system [8] created conceptual summaries using heuristics that chose parts of a concept graph created from Down Jones news wire stories. Kupiec used a Naïve Bayes classifier trained on a collection of 188 scientific documents and their corresponding human created summaries to learn the probabilities that a document sentence will be chosen in the human summaries [9]. Kupiec's work started a trend that increasingly led to the use of supervised learning and statistical methods in summarization. Yeh et al. presented a summarizer that took into account several features such as position, presence of positive and negative keywords, centrality, and resemblance to title. After considering these features, the summarizer then used a genetic algorithm to create a scoring function [10].

A number of algorithms have been developed for various aspects of document summarization during recent years. For the selection of summary content, notable algorithms were developed including SumBasic [11] and the centroid algorithm of Radev et al. [12]. SumBasic's underlying premise is that words that occur more frequently across the documents have a higher probability of being selected for human created multi-document summaries than words that occur less frequently. This idea is similar to Luhn's idea but there are some significant differences. Vanderwende et al. [11] introduce a new method called SumFocus which imposes constraints on topic changes during summary generation. The centroid algorithm [12] takes into consideration a centrality measure of a sentence in relation to the overall topic of the document cluster or in relation to a document in the case of single document summarization.

Ordering the sentences or other components in any generated text including generated summaries is an important area of research as well and several algorithms have been proposed regarding this topic. Althaus et al. assign costs to transitions in a discourse using a cost function and compute optimal ordering of sentences to create a maximally coherent discourse in a local sense [13]. They show the problem to be NP-complete but use modern and efficient heuristic algorithms for the traveling salesman problem to compute an optimal order. The LexRank algorithm for computing the relative importance of sentences or other textual units in a document (or a set of documents) creates an adjacency matrix among the textual units using an IDF-modified cosine distance measure and then computes the stationary distribution considering it to be a Markov chain [14]. The TextRank algorithm [15] is also a graph-based approach that creates an adjacency matrix and then finds the most highly ranked sentences (or keywords) in a document using the PageRank algorithm [16]. Barzilay and Lapata use an entity-grid representation of discourse to capture the pattern of entity distribution in a text and use this information to create a representation of the text as a set of entity transition sequences that can be used to create summaries [17].

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., [18]) or clustering (e.g., [19]) documents. Following another line of approach, early work by Kalita et al. generated textual summaries of database query results [20, 21]. Instead of presenting a table of data rows as the response to a database query, they generated textual summaries from predominant patterns found within the data table.

With the growth of the Web, interest has grown on how to improve summarization and how to summarize new forms of documents such as Web pages (e.g., [22], [23], [24], [25]), discussion forums [26] and blogs (e.g., [27], [28], [29]). In the context of the Web, multi-document summarization is useful in combining information from multiple sources. For example, if one wants to summarize news items on a topic, the number of source articles may be quite large. Information may have to be extracted from many articles and pieced together to form a comprehensive and coherent summary. Thus, multiple document summarization is an important and timely topic of research (e.g., [30], [31], [32], [33], [34], [35], [36]). The major difference between single document summarization and multi-document summarization is the greater amount of redundancy caused by using multiple source texts [37, pp. 797]. A method of avoiding redundant sentences in a summary is computing similarity between a sentence already chosen to be in the summary and one being proposed to be selected. Another solution may involve clustering the important sentences picked out from the various source texts and using only a representative sentence from each cluster [38, Chapter 14], [39, Chapter 5]. For multi-document summarization, the ordering of extracted sentences (e.g., [13, 17], mentioned earlier in this section) is important as well. In the area of publicly available programs, MEAD [40] is a flexible platform for multi-document multi-lingual summarization. MEAD implements multiple summarization algorithms and provides metrics for evaluating multi-document summaries.

There have been a few recent efforts in summarizing Twitter posts. We have been able to find work on Twitter summarization by three groups of researchers [41–43]. Harabagiu and Hickl [41] focus on summarization of microblog posts, related to complex world events. To summarize, they capture event structure information from tweets and user behavior information that captures how individual users describe information relevant to a topic. Following Huang and Mitchell [44], they infer event structure using a generative model with the help of an Expectation Maximization algorithm [45]. They represent user actions in terms of tweet chains considering retweets, responds and quoted tweets. They use the “sleeping experts” learning framework [46] to assess the relevance of content expressed by groups of users linked by chains. Takamura et al. [42] summarize Japanese Twitter posts on soccer games during which people comment and express opinions play by play as the game progresses on a timeline. They observe two main ideas. First, similar timestamped documents can mention different events if they are temporally distant. Second, documents on a single topic such as goals scored at a specific time can occur with some temporal delay. They solve an optimization problem called the p-median problem in an approximate manner after imposing conditions based on these two observations. Our prior work [43, 47–50] has focused on developing and comparing algorithms for producing summaries of Twitter posts. This paper puts together work in several papers into a coherent whole and also extends it.

3. PROBLEM DESCRIPTION

By design, Twitter documents are very short because they are limited to 140 characters. Our work is in the context of a collection of such documents.

Our description of the first problem addressed in this paper is as follows:

Given a set of Twitter documents that are all related by containing a common search phrase (i.e., a topic), generate a short summary in terms of a representative document that best describes the primary ”gist” of what users are saying about that search phrase.

However, a summary containing a single document can only represent one idea surrounding a topic. With this limited coverage of a specified microblog topic, important or interesting information about a topic may be easily overlooked. These short summaries may provide simple *indicative* summaries that give enough information to spark the interests of users, but summaries containing multiple documents that cover multiple subtopics of the original topic would push the summaries towards being *informative* [51]. Therefore, this research further explores several methods for producing these summaries with multiple representative documents.

The problem considered in the second part of this paper can be defined as follows:

> *Given a topic keyword or phrase and a number k, retrieve a set of k most representative documents from a set of Twitter documents.*

4. APPROACHES FOR SINGLE-DOCUMENT TWITTER SUMMARIES

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. Microblog documents are already highly condensed leading to the greater potential of finding an extract from a collection to serve as the summary. In addition, because microblogs are unstructured and diverse in subject matter, abstractive approaches would not be appropriate since they tend to do best in very limited domains and require outside knowledge sources. Extractive techniques are also known to better scale with more diverse domains [51].

We implement several extractive algorithms. We start by implementing two preliminary algorithms for single-document summarization based on very simple techniques. We also develop and implement two new algorithms for single-document summarization. First, we create an algorithm that we call the Phrase Reinforcement algorithm which 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 6.1. We develop another new algorithm for single-document summaries based on a well established statistical methodology known as TF-IDF. The Hybrid Document TF-IDF algorithm is discussed at length in Section 6.2. We evaluate these algorithms extensively.

Next, we implement algorithms that produce summaries containing multiple documents reflecting the multi-faceted nature of Twitter documents on a single topic. We develop several multi-document summarization algorithms. Our two primary multi-document summarization algorithms are a clustering-based algorithm and a direct extension of the Hybrid TF-IDF single-document summarization algorithm. We perform an extensive evaluation of the multi-document summarization algorithms as well.

5. NAIVE ALGORITHMS FOR SINGLE-DOCUMENT SUMMARIES

While the two preliminary approaches are simplistic, they serve a critical role towards allowing us to evaluate the results of our primary algorithms.

5.1. Random Approach

Given a filtered collection of Twitter documents (i.e., posts) and topic sentences that are each related to a single topic, we generate a summary by simply choosing at random either a document or sentence from the set of inputs. Though unlikely given the short nature of a Twitter document, we note that sometimes a document may contain more than one sentence.

5.2. 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 Twitter documents 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 documents (i.e., posts) from the collection. For the remaining two, we choose both the shortest and longest topic sentences from the collection.

6. PRIMARY ALGORITHMS FOR SINGLE-DOCUMENT SUMMARIES

We discuss two primary algorithms: the Phrase Reinforcement Algorithm and an adaptation of the well-known TF-IDF approach.

6.1. The Phrase Reinforcement Algorithm

The Phrase Reinforcement (PR) algorithm generates summaries by looking for the most commonly occurring phrases. By representing these common phrases as a weighted and directed acyclic graph, the algorithm generates summaries by searching for the most weighted paths through the graph.

6.1.1. Motivation

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

1. *Our first Soupy Sales RIP cartoon*
2. *Aw, Soupy Sales died.*
3. *Just read that my favorite comedian Soupy Sales died.*
4. *Soupy Sales – RIP – I always watched his show when I was a kid - classic!*

In these documents, we consider the words that immediately follow the topic phrase *Soupy Sales*. Notice that all documents contain words immediately after the phrase *Soupy Sales* that in some way refer to his death. Furthermore, documents 1 and 4 share the word *RIP* and posts 2 and 3 share the word *died*. Therefore, there exists some overlap in word usage adjacent to the phrase *Soupy Sales*. This overlap occurs because there exists only so many ways to express the main idea of Soupy Sales' death in a very short Twitter document.

The second observation is that microbloggers often repeat the most relevant documents for a trending topic by quoting others. Quoting has its own special convention in Twitter and uses the following form: *RT @[TwitterAccountName]: Quoted Message*. *RT* refers to *Re-Tweet* and indicates one is copying a document from the indicated Twitter account. For example, the following are some quoted documents.

1. *RT @dcagle: Our first Soupy Sales RIP cartoon*
2. *RT @RadioPages: Soupy Sales has died at 83.*

Retweeting significantly reinforces the overlap of word usage around a topic phrase. While microbloggers 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 PR algorithm capitalizes on these behaviors. The main idea of the algorithm is to determine the most heavily overlapping phrase centered about the topic phrase. This phrase is used as the topic's summary. The justification is that repeated information is often a good indicator of its relative importance [4].

6.1.2. The Algorithm

The algorithm begins with the topic phrase for which one desires to generate a summary. 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.

Building a Word Graph We focus only on English tweets in this paper. We use simple tools we have developed ourselves to recognize word and sentence boundaries.

The root node contains the topic phrase—in this case *soupy sales*. The root node is shown in Figure 1a. We build a graph showing how words occur before and after the phrase in the root node after processing all the posts on the topic. The graph can be thought of 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 posts to the set of posts 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 posts 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.

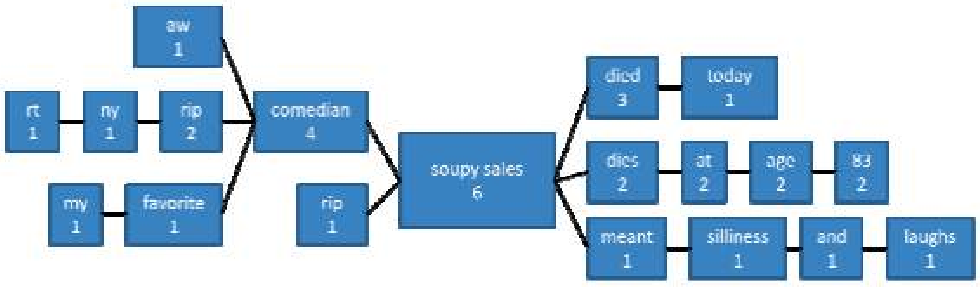
In the graph, all the nodes are all in lower-case and stripped of any non-alpha-numeric characters. Words are tokenized in this manner in order to increase the amount of overlap among words. The count of the associated node's phrase is exactly the number of matches within the set of input posts that have the same position and word sequence relative to the root node. Nodes with a count less than two are not actually 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 1b. Usually, nodes with a count less than two are not even added to the graph but in order to help illustrate our graph without a volume of example data, we temporarily add these nodes to the graph. Since we originally allow non-overlapping phrases, we now reinstate that restriction by pruning the graph of any sub-trees containing nodes with a count less than 2. This reduces the graph to the one in Figure 1c.

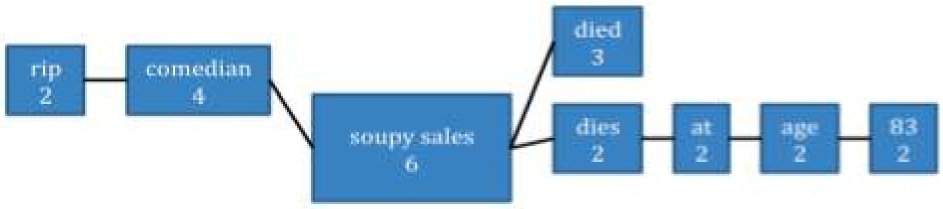
Weighting Individual Word Nodes The algorithm prepares for the generation of summaries by weighting individual nodes. Node weighting is performed to account for the fact that some words have more informational content than others. For example, the root node *soupy sales* contains no information since it is common to every input sentence so we give it a weight of zero. Common stop words are noisy features that do not help discriminate between phrases so 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

soupy sales

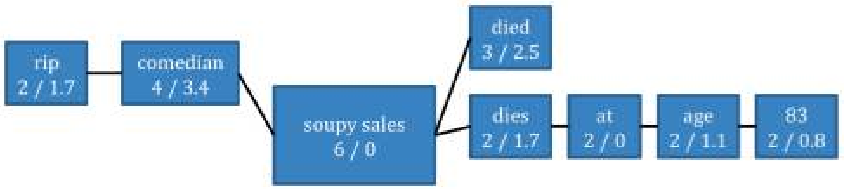
(a) Phrase Reinforcement root node for topic *Soupy Sales*.



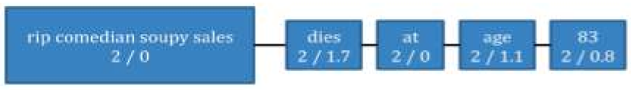
(b) Fully constructed PR graph (allowing non-overlapping words/phrases).



(c) Fully constructed PR graph (requiring overlapping words/phrases).



(d) Fully constructed weighted PR graph (requiring overlapping words/phrases).



(e) Fully constructed PR graph for second half of summary. This demonstrates best complete summary.

FIGURE 1: Progression during the Phrase Reinforcement Algorithm

by an amount that is proportional to their distance:

$$\text{weight}(node) = \text{count}(node) - \text{distance}(node) * \log_b \text{count}(node). \quad (1)$$

The logarithm base b is a parameter and 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 1d. We assume the logarithm base b is set to 10 for helping generate longer summaries.

Generating Summaries In order to generate a summary, the algorithm looks for the most overlapping phrase within the graph. Since the nodes' weights are proportional to their overlap, the algorithm has to search for the phrase with the most weight. 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 a node on the left. The path with the most weight is chosen. Assuming the algorithm produces the graph shown in Figure 1d, the path with most weight on the left-hand side of the root node has a weight of 5.1 including the root node and 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. This way, when we filter our input sentences to those containing this phrase, we can only generate phrases found within the input sentences. Assume we get the path shown in Figure 1e as the most heavily weighted path on the right hand side. The full summary generated by the algorithm for our example is: *rip comedian soupy sales dies at age 83.*

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. To find such a phrase, we divide our summary into an ordered list of tokens. For each input sentence, we search for the occurrence of each token in order that they occur in the produced summary token list. However, since we do not know exactly how many formatting characters are between each token, we permit up to a threshold number of intervening characters between each summary token. Once, we find the first matching phrase, this phrase is our final summary. For our example, the algorithm would produce the following final summary:

RIP Comedian Soupy Sales dies at age 83.

6.2. Hybrid Document TF-IDF Summarization

After analyzing the results obtained by the Phrase Reinforcement approach, we notice that it significantly improves upon our earlier results obtained using naïve single-document(-sentence) summarizers but still leaves room for improvement as its performance only halves the difference between the random and manual summarization methods (see Section 8.3). Therefore, we develop another approach based upon a classical technique dating back to early summarization work performed by [4].

6.2.1. TF-IDF

Term Frequency Inverse Document Frequency (TF-IDF) is a statistical weighting technique that has been applied to many types of information retrieval problems. For example, it has been used for automatic indexing [52], query matching of documents [53], and automated summarization [54]. Generally TF-IDF is not known as one of the leading algorithms in automated summarization.

For straightforward automated summarization, the application of TF-IDF is fairly simplistic. The idea is to assign each sentence within a document a weight that reflects the sentence's saliency within the document. Once each sentence has been weighted, the sentences are ordered by their weights from which the top k 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 [55]. To determine the weight of a term, we use the formula:

$$TF\_IDF = tf_{ij} * \log_2 \frac{N}{df_j} \qquad (2)$$

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 [52].

The TF-IDF value is composed of two primary parts. The term frequency component (TF) assigns more weight to words that occur frequently within a document because important words are often repeated [4]. 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 [55]. This does not have any ill effects on single document summarization. When generating a summary from multiple documents, however, this becomes an issue because the terms within the longer documents have more weight. To compensate for this problem, many normalization methods have been proposed. Some example methods include cosine normalization, maximum *tf* normalization, byte length normalization and pivoted document length normalization [55]. We experimented with several schemes and found that the simple normalization method given in (Equation 7) works well for us. This ensures that even documents that are really short have a minimum amount of contribution to the weight.

6.2.2. Algorithm

Equation (2) defines the weight of a term in the context of a document. However, we do not really have a document in the conventional sense although we have been using the term *document* to refer to a short Twitter post so far. We are dealing with a set of microblog posts that are each limited to at most 140 characters in length. Each document expresses a short thought on a topic and several such short documents may be needed to expresses an even a moderately complex thought or discussion. So, one question we may pose is whether it is necessary to tweak what we consider a document for the TF-IDF computation to work. One option is to define a single (macro-) document that encompasses all the documents (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 in this (macro-) document. However, doing so causes us to lose the IDF component since we only have a single (macro-) document. On the other extreme, we could characterize each post as a document—following the traditional approach we have used in our discussions in this paper so far—making the IDF component's definition clear. But, the TF component now has a problem because each document (post) contains only a handful of words, and therefore, most term frequencies will be a small constant for a given document.

To handle this situation, we redefine TF-IDF in terms of a hybrid document. We primarily define a document as a single post. However, when computing the term frequencies, we assume a (macro-) document containing the entire collection of these short documents (posts). Therefore, the TF component of the TF-IDF formula uses the entire collection of documents (posts) while the IDF component treats each post as a separate document. This way, we have differentiated term frequencies but also do not lose the IDF component. For our purposes, a term is a single word in a document.

We next choose a normalization method since otherwise the TF-IDF algorithm will always bias towards longer documents. We normalize the weight of a document by dividing it by a normalization factor. Since common stop words do not help discriminate the saliency of documents, we give each of these types of words a weight of zero by comparing them with a prebuilt list. Given this, our definition of the TF-IDF summarization algorithm is now complete for microblogs. We summarize this algorithm below in Equations (3)-(7).

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

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

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

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

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

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.

6.2.3. *Discussion*

The essence of the Hybrid TF-IDF algorithm is its ability to assign meaningful term frequency and inverse-document frequency values to a very short document type while also being able to carefully control the overall weight and length of the target document. By defining the term frequencies in terms of the entire set of words within all of the candidate Twitter documents, we have a much more representative set in order to judge a word's natural frequency compared to using a single document. Using only a single Twitter document would have resulted in all of the words having about the same small term frequency since it is unlikely a word would ever occur more than once within a single document or two. Conversely, by choosing to use individual documents for defining the inverse document frequency, we are able to measure and weigh terms based on how often those terms occur across documents. Words that occur too frequently across the majority of documents are most likely either the candidate phrase (which by definition is in every document) or unknown stop words. In either of these cases, these types of words do not provide much discriminatory power. Alternatively, words that occur with some higher frequency but not in every document provide much greater discriminatory power. Therefore, by defining a document in a hybrid way, we have maximized the amount of information available for both the term frequency and inverse-document frequency components within the classical TF-IDF equation.

One other important contribution of the Hybrid TF-IDF equation is its normalization factor which allows it to carefully control the overall target summary length. As noted by [55] and noted earlier in this paper, classical TF-IDF is very sensitive to document length and often over-weighs terms from longer documents. In our application of TF-IDF, we observed the same effect. Without a normalization factor, classical TF-IDF awarded the most weight to the longer documents since the weight of a document is the simple sum of the weights of the composing words. Therefore, initially our implementation of TF-IDF resulted in simply the longest documents always awarded the most weight. In order to counteract this effect, we divide the weight of a document by our normalization factor which is just the maximum of either a predefined constant (i.e. the Minimum Threshold) or the number of words in the document. The Minimum Threshold is simply the average desired target document length. In our case, we set the constant to the average document length of our manual summaries of around 10 words or so.

The document weight normalization factor has the following effect. For documents that are longer than the target summary length, these documents become increasingly penalized for every word that is longer than the target length. Where the longest document would before have had the greatest overall weight, these sentence weights get reduced to their average term weight since we are dividing the sum of term weights by the number of terms. The longer the document, the less average term weight it will likely contain. Alternatively, for documents shorter than the target summary length, these documents will also get penalized since they will be divided by a number larger than the number of terms in the document. This is important since if we had just divided by the number of terms like the previous case, the formula would bias towards the shortest documents since they would have the highest average term weight. Therefore, we penalize shorter documents as well by dividing the weight of shorter documents by the minimum threshold which has the effect of adding $(MinimumThreshold - \#WordsInSentence)$ zero-weight terms to the document. Finally, the combination of meaningful term frequencies, inverse document frequencies, and deliberate control of the overall target summary length give the Hybrid TF-IDF algorithm the discriminatory power to differentiate the most salient documents of a desired length within the collection of available documents for a given topic phrase.

7. EXPERIMENTAL SETUP AND EVALUATION METRICS FOR SUMMARIES

7.1. 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 of documents (approximately 1500). Therefore, we had 50 trending topics with a set of 1500 documents for each.

7.2. Preprocessing the Posts

Because microblog documents are an unstructured and informal way of communicating, the documents were preprocessed to remove spam and other noise features. These pre-processing steps included the following.

1. Convert any HTML-encoded characters into ASCII.
2. Convert any Unicode characters (e.g. "\u24ff") into their ASCII equivalents and remove. Since we are dealing with English tweets only, we remove Unicode characters outside the ASCII range.
3. Filter out any embedded URL's (e.g. "http://"), HTML (e.g. "<a.../a>"), headings (e.g. "NEWS:"), references (e.g. "[...]"), tags (e.g. "<...>"), and retweet phrases (e.g. "RT" and "@AccountName").
4. Discard the document if it is spam. We developed a Naïve Bayes classifier for spam detection among Twitter posts [47].
5. Discard the document if it is not in English. We use some simple heuristics to determine the language of a post.
6. Discard the document if another document by the same user has already been acquired. This is to obtain a wider variety of posts.
7. Reduce the remaining number of documents by choosing the first 100 posts. This is done so that our volunteers can write manual summaries needed for evaluation using the same 100 posts as used by the algorithms.
8. Break the document into sentences. Most tweets have only one sentence.
9. Detect the longest sentence that contains the topic phrase and use it to represent the document. This step helps filter out extraneous noise sentences such as "Wow!" or "That's funny."

These pre-processing steps and their rationale are described more fully in [47].

7.3. 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 [3]. Extrinsic evaluations measure how well a summary enables a user to perform some form of task [51].

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 measure the similarity between an automated summary and a set of manual summaries [56]. One of the simplest ROUGE metrics is the ROUGE-N metric:

$$\text{ROUGE-N} = \frac{\sum_{s \in MS} \sum_{n\text{-}grams \in s} \text{match}(n\text{-}gram)}{\sum_{s \in MS} \sum_{n\text{-}grams \in s} \text{count}(n\text{-}gram)}. \quad (8)$$

Here, *MS* is the set of manual summaries, 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.

Both precision and recall of the automated summaries can be computed using related formulations of the ROUGE metric. Given that MS is the set of manual summaries and u is the set of unigrams in a particular manual summary, recall can be defined as

$$r = \text{ROUGE-1} = \frac{\sum_{m\in MS}\sum_{u\in m}\text{match}(u)}{\sum_{m\in MS}\sum_{u\in m}\text{count}(u)} \left(= \frac{matched}{relevant}\right), \quad (9)$$

where $count(u)$ is the number of unigrams in the manual summary and $match(u)$ is the number of co-occurring unigrams between the manual and automated summaries. The ROUGE metric can be slightly altered so that it measures the precision of the auto summaries such that

$$p = \text{ROUGE-1}' = \frac{\sum_{m\in MS}\sum_{u\in m}\text{match}(u)}{\mid MS \mid * \sum_{u\in a}\text{count}(u)} \left(= \frac{matched}{retrieved}\right), \quad (10)$$

where $\mid MS \mid$ is the number of manual summaries and a is the auto summary. Because both recall and precision are important in summaries, the F-measure, which is a type of average of precision and recall, is computed such that

$$\text{F-measure} = \frac{2pr}{p+r}. \quad (11)$$

We weigh precision and recall equally since that is the practice in all summarization literature as far as we know.

Lin performed evaluations to understand how well different forms of ROUGE's results correlate with human judgments [56]. 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. In DUC 2003, the documents summarized were proper newspaper articles from sources such as AP newswire (1998-2000), New York Times (1998-2000), Financial Times of London (1991-1994) and The Los Angeles Times (1989-1990). The source articles were of normal length although various types of short summaries were produced. In our case, the source documents themselves are at most 140 characters or about 10 words or so long. Lin found ROUGE-1 and other ROUGEs to correlate highly with human judgments for very short summaries. Since this task is similar to creating microblog summaries in that the resulting summaries are short in both cases, we implement ROUGE-1 as a metric.

Evaluation of text summarization is a complex and controversial issue in computational linguistics [57]. There have been a few studies trying to determine a good measure of summary quality for extractive summaries besides ROUGE. Liu and Liu [58, 59] demonstrate that ROUGE metrics correlate well with human summaries when applied to transcripts of meetings transcripts, after the metrics were slightly modified to take into account characteristics of the domain such as disfluencies and speaker information. Many complex methods have been proposed (e.g., [57, 60]), but no one has conclusively shown that any of the newly proposed methods work well in all situations. For example, Saggion et al. [57] show that the correlation between metrics such as Pyramids, Responsiveness and ROUGE is strong for some datasets, but not for others. By all accounts, ROUGE is still the most widely used summarization evaluation framework [57]. We want to leave this complex task of determining if metrics other than ROUGE-1 work better for tweets to a future researcher since it will be a paper by itself. In fact, our current understanding is that it is difficult to find a perfect metric. To overcome limitations of ROUGE, we use manual evaluations as well.

Since we want some 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 complete an automated summary expresses the meaning of a human summary.

8. EVALUATION OF SINGLE-DOCUMENT SUMMARIES

8.1. Manual Summaries

We asked two volunteers to generate a complete set of 50 manual summaries for all topics. The volunteers were instructed to generate the best summary possible in 140 characters or less while using only the information contained within the documents (see Table 1).

8.1.1. 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 compare the manual single-document summaries against one another bi-directionally by assuming either set was the set of automated summaries. We do the same using the Content metric in order to understand the semantic similarity between the two summaries. By evaluating our two manual summaries against one another, we help establish practical upper limits of performance for automated single-document summaries. These results in addition to the results of the preliminary algorithms collectively establish a range of expected performance for our primary algorithms for single-document summarization.

To generate the Content performance, we asked a volunteer to evaluate how well one single-document 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 2.

8.2. Performance of Naïve Algorithms for Single-Document Summarization

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

TABLE 1: 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
#MM	#mm: Users post about their favorite song, band, or line. Same as #MusicMonday	It's Music Monday on Twitter!
Gossip Girl	Gossip Girl's fans hope that Chuck and Blair make up their relationship by the next episode.	Chuck and Blair's relationship in question after Gossip girl episode

TABLE 2: Average ROUGE-1, Content scores, 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
Average	0.34	0.34	0.34	4.2	10

TABLE 3: Content Performance for Naive Summaries

	Content Performance
Manual Average	4.1
Random Sentence	3.0
Shortest Sentence	2.2

originally anticipated given that our average manual F-measure is only 0.34 (See Table 2). Careful examination explains these results by considering two factors. Some overlap is explained because the ROUGE-1 metrics include 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 single-document summaries, we present Content performance in Table 3. This table indicates that 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 documents (posts)—which include the non-topic sentences. As expected, 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 document (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).

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 (i.e., F-measure), the number of tokens in the automated and manual summaries must be similar to each other since ROUGE-1 is simply comparing unigrams. Therefore, when using the ROUGE-1 metric, one must assure that the manual and automated summaries have similar lengths.

8.3. Performance of Phrase Reinforcement Algorithm for Single-Document Summarization

The PR (Phrase Reinforcement) algorithm generates summaries using complete documents as input since it builds a connected graph of words around the central topic. The algorithm ignores punctuation and treats documents as a sequence of words.

Table 4 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 table also displays ROUGE-1 results we computed earlier for the manual and randomly generated summaries which represent our expected range of results. The algorithm produces an average recall of 0.30, an average precision of 0.31, and a combined F-measure of 0.30. This is a significant improvement over the random sentence approach. However, it still leaves some room for improvement since the manual summaries had an F-measure of 0.34.

The PR algorithm generates an average Content metric score of 3.66 (see Table 4). This score also shows an improvement over the random summaries. Table 4 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 4: 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

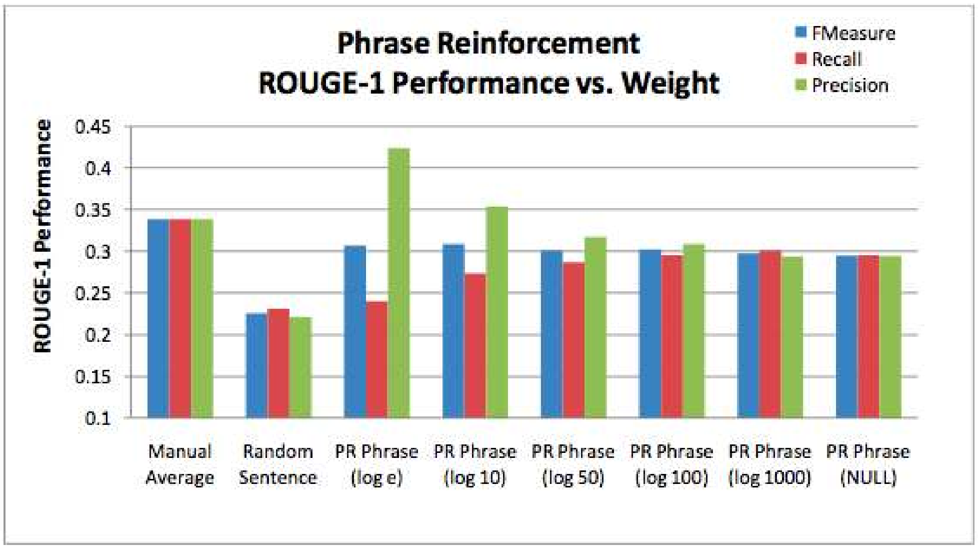


FIGURE 2: ROUGE-1 performance for different weightings of the Phrase Reinforcement algorithm.

Table 5.

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. We vary the logarithm base for a variety of values while measuring its effect on performance and average summary length.

In Figures 2 and 3, we can see the effect of varying the weighting parameter b within Equation 1. There appears to be a threshold (in our case of 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. The Content performance also decreases as b decreases. Above this threshold, the average summary length does not increase while the ROUGE-1 and Content performance begins to degrade slightly. While the particular threshold may vary for different sizes of testing data, we can generalize that the weighting parameter b affects the resulting length and performance of the algorithm. However, beyond a certain threshold, the performance of the PR algorithm is relatively stable. In Figure 2, the label “PR Phrase (NULL)” indicates the absence of the weighting parameter altogether. In these cases, we simply use a node’s count as its weight

8.4. Performance of the Hybrid TF-IDF Algorithm

In Figure 5, we present the results of the Hybrid TF-IDF algorithm for the ROUGE-1 metric in relation to all of our previous results. The TF-IDF results are denoted as ”TF-IDF Sentence (11)” to distinguish the fact that the TF-IDF algorithm produces one sentence instead of a phrase for summaries and that a threshold of 11 words for the normalization factor is used. 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 better than our Phrase Reinforcement results.

We evaluate the automated summaries against the manual summaries using the Content metric in order to understand whether or not we are truly achieving human-comparable summaries. The results of the Content evaluation are in Figure 6. Remarkably, the TF-IDF algorithm’s Content performance of 4.1 is also very similar to the manual summaries’ performance of 4.2. This score is also higher than the average Content score of the Phrase Reinforcement algorithm which was 3.66.

Figure 7 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,

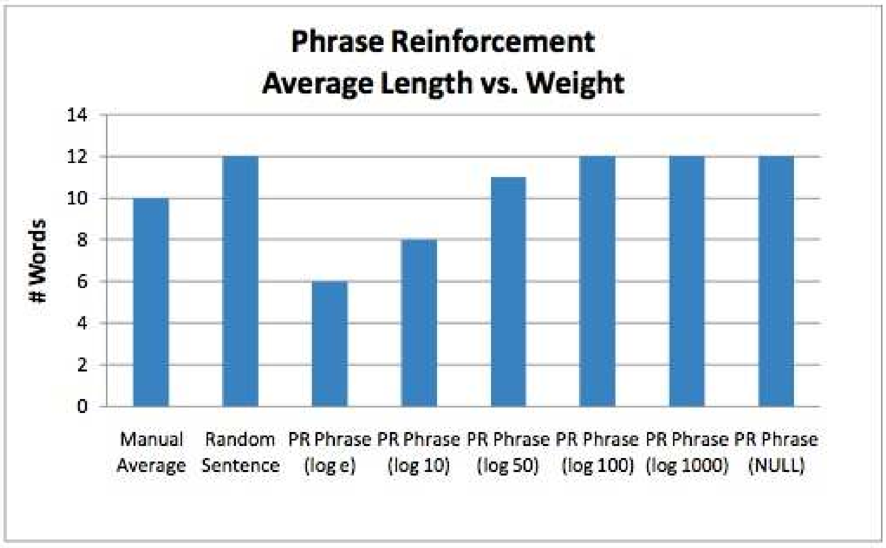


FIGURE 3: Average summary lengths for different weightings of the Phrase Reinforcement algorithm.

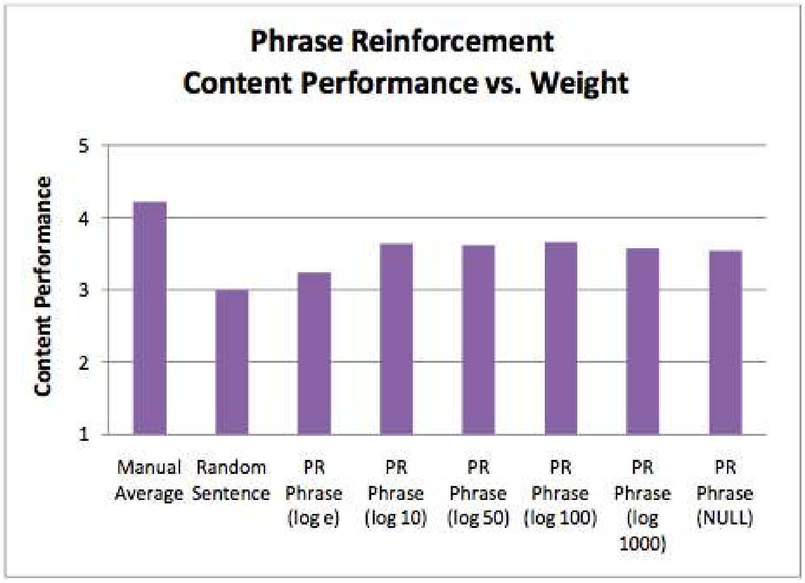


FIGURE 4: Content performance for different weightings of the Phrase Reinforcement algorithm.

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 6.

8.4.1. Normalization

In Section 6.2, we noted that the TF-IDF equation is sensitive to a document's length, and so we decided it was necessary to normalize the length of the sentence. 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. 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-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.

After many experiments, we settled on our normalization factor: the maximum of a minimum threshold and the number of words in the sentence. The minimum threshold is simply an integral number of words and approximately represents the desired average summary length. Given this definition, the normalization factor is then used as a divisor into the sentence weight. By choosing the maximum of either a threshold or the sentence length, we are able to control the average summary length. In fact, we can produce a spectrum of average summary lengths simply by using a range of

TABLE 5: Summaries produced by the Phrase Reinforcement algorithm

Topic	PR Phrase (log 100)
#BeatCancer	For every tweet tagged with #BeatCancer, eBay & Miller/Coors will donate $.01 to cancer.
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.
Russian Roulette	listening to rihanna's new song russian roulette
Dodgers	Phillies dump Dodgers for World Series return
Glee	Thong song on Glee!
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...
Balloon Boy	Balloon Boy Mom Admits to Hoax : According to court records made public today, Mayumi Heene, Ball..

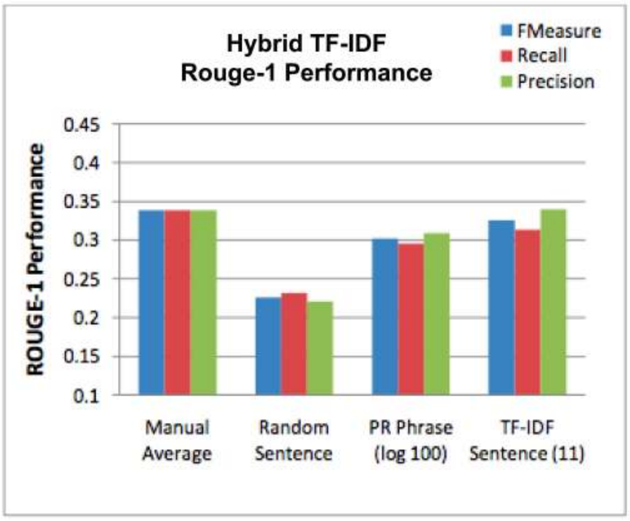


FIGURE 5: ROUGE-1 performance for Hybrid TF-IDF algorithm.

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 Figures 8 and 9, by varying the normalization threshold, we are able to control the average summary length and resulting ROUGE-1 metrics. Furthermore, there appears to be an inflection point at a threshold of 12 words where the precision and recall performance levels cross. This inflection point is where the average TF-IDF summary length is equal to the average manual summary length. Therefore, we can use the threshold to control whether we desire better precision, recall, or a balance of these two values for the produced TF-IDF summaries. More interestingly,

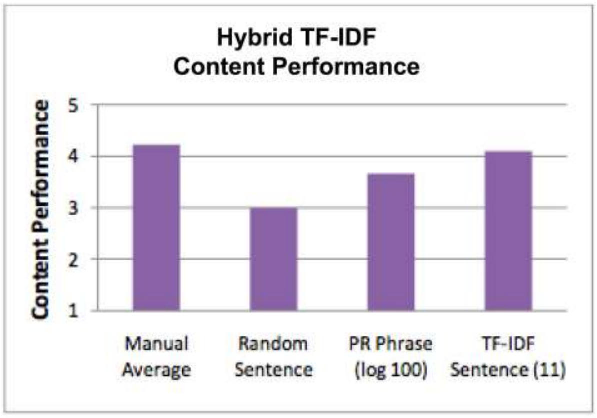


FIGURE 6: Content performance for the Hybrid TF-IDF algorithm.

TABLE 6: Summaries produced by the Hybrid TF-IDF algorithm

Topic	TF-IDF Sentence (11)
#musicmonday	#musicmonday damn have you guys heard the new gaga song?
#BeatCancer	Every tweet that includes #beatcancer raises money for cancer research.
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!

we do not necessarily need to compromise on Content performance while varying the summary length as seen in Figure 10.

9. ALGORITHMS FOR MULTI-DOCUMENT SUMMARIZATION

Having produced single-document summaries using several algorithms and having analyzed them exhaustively, we now want to consider producing summaries containing multiple documents, which we call multi-document

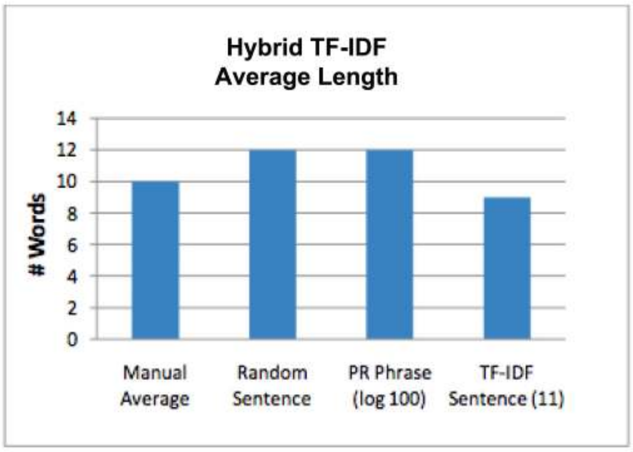


FIGURE 7: Average summary length for Hybrid TF-IDF algorithm.

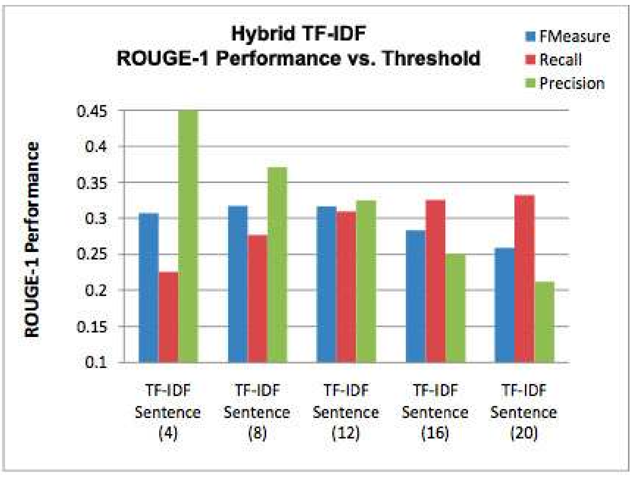


FIGURE 8: Rouge-1 performance vs. threshold for Hybrid TF-IDF algorithm

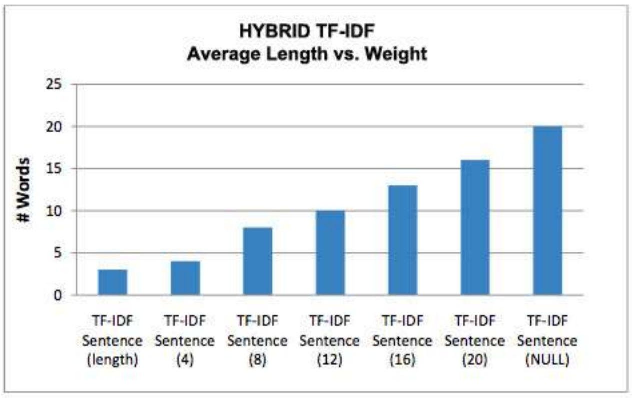


FIGURE 9: Average Length Vs. Weight for Hybrid TFIDF algorithm for various normalization thresholds

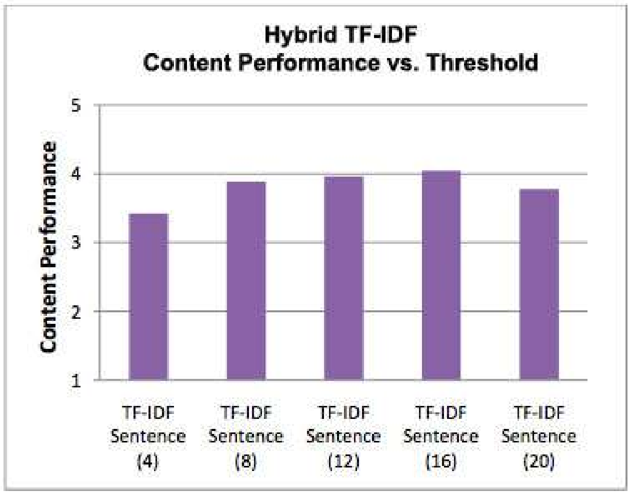


FIGURE 10: Content performance vs. threshold for Hybrid TF-IDF algorithm

summaries. As discussed in the beginning of this paper, the set of documents returned by a Twitter API search for a certain topic may actually represent several sub-topics or themes. Thus, instead of producing single-document summaries, it may be more appropriate to produce several summaries corresponding to the multiple themes present within a set of posts. Therefore, we extend our algorithms for single-document summaries to produce multi-document summaries. We compare the performance of our multi-document summarization algorithms with several baselines as well as other well-known state-of-the-art multi-document summarization algorithms.

10. PRIMARY ALGORITHMS FOR MULTI-DOCUMENT SUMMARIZATION

We develop and explore two primary algorithms for producing summaries containing multiple documents in this paper. The first method clusters documents into k clusters and then summarizes each cluster individually. The second method computes the Hybrid TF-IDF weights of documents and selects the top k documents filtered by a similarity threshold so that the documents in the summary are not too similar. We perform preliminary testing of microblog document clustering to select the best clustering method for the cluster summarizer.

10.1. Cluster Summarizer

10.1.1. Clustering Algorithms

The standard centroid based k-means algorithm is the foundation of all the clustering algorithms tested in this paper. Because the standard k-means algorithm generally performs well and is easy to implement [39], it was tested first. The bisecting k-means algorithm was tested next because it may perform better than the direct k-means algorithm as suggested in [39, 61]. The k-means++ algorithm, which is a new variation of k-means algorithm proposed and initially tested by [62], was then tested. Finally, an algorithm that combines the k-means++ algorithm with the bisecting algorithm was tested hoping to combine the benefits of the k-means++ algorithm and the bisecting k-means algorithm.

The primary measure of similarity used is the cosine similarity measure:

$$\text{sim}(s_i, s_j) = \cos(v_i, v_j) = \frac{v_i^t v_j}{\|v_i\| \|v_j\|}. \tag{12}$$

The centroid $c_i \in C$ is defined as

$$c_i = \frac{\sum_{v \in V_i} v}{n_i} \tag{13}$$

in which n_i is the number of posts and V_i is the set of feature vectors in ith cluster. The steps of each clustering algorithm are outlined as follows.

1. Standard k-means algorithm
   (a) Randomly choose k initial centroids $c_i \in V$ from the computed feature vectors $v_i \in V$.
   (b) Assign each document p_i to the centroid that is most similar to its corresponding feature vector v_i.
   (c) Compute the centroid of each cluster.
   (d) Repeat steps 1b and 1c until no documents are reassigned.
2. Bisecting k-means algorithm
   (a) Bisect the set of documents P into 2 clusters using the standard k-means algorithm ($k' = 2$) defined by algorithm 1 above.
   (b) Choose the largest already formed cluster to bisect.
   (c) Repeat steps 2a and 2b until the kth cluster has been formed.
3. k-means++ algorithm
   (a′) Choose initial centroids based on probability.
      i. Choose an initial centroid c_1 uniformly at random from V.
      ii. Choose the next center c_i, selecting $c_i = v' \in V$ with the probability $\frac{D(v')^2}{\sum_{v \in V} D(v)^2}$ where $D(v)$ is the shortest Euclidean distance from v to the closest center which is already known.
      iii. Repeat step 3a′ii until k initial centroids have been chosen.
   (b-d) Continue with steps 1b-1d of algorithm 1.
4. Bisecting k-means++ algorithm
   (a) Follow step 2a of the bisecting algorithm above except use the k-means++ algorithm instead of using the standard k-means algorithm.
   (b-c) Continue with the bisecting k-means clustering algorithm defined in steps 2b-2c.

10.1.2. Cluster Summarizer Used

We decided to use the bisecting k-means++ algorithm since it seems to provide the best results for our purposes. The results of comparing various clustering algorithms with Twitter documents are discussed in Section 12.2.2. Thus, our first multi-document summarizer is a cluster summarizer that clusters the input documents based on the bisecting k-means++ algorithm. Then, it summarizes each cluster with the original Hybrid TF-IDF algorithm.

10.2. Hybrid TF-IDF with Similarity Threshold

The single-document Hybrid TF-IDF algorithm developed has been discussed in Section 6.2. It weights all the sentences based on the Hybrid TF-IDF weighting of sentences explained. Originally, the algorithm only selected the best summarizing topic sentence, but we modified it to select the top k most weighted documents that do not have a similarity above a given threshold t because the top most weighted sentences of the modified Hybrid TF-IDF summarizer may be very similar or discuss the same subtopic. This similarity threshold filters out a possible summary post s'_i if it satisfies the following condition:

$$\text{sim}(s'_i, s_j) > t$$

$\forall s_j \in R$ where R is the set of posts already chosen for the final summary and t is the similarity threshold. Using the cosine similarity measure defined in section 10.1.1, the threshold was varied from 0 to 0.99 in increments of 0.01 for a total of 100 tests to find the best threshold to use.

11. ADDITIONAL SUMMARIZATION ALGORITHMS TO COMPARE RESULTS

We compare the results of summarization of the two newly introduced algorithms with baseline algorithms and well-known multi-document summarization algorithms. The baseline algorithms include a Random summarizer and a Most Recent summarizer. The other algorithms we compare our results with are SumBasic, MEAD, LexRank and TextRank.

11.1. Random Summarizer

This summarizer randomly chooses k posts for each topic as a summary. This method was chosen in order to provide worst case performance and set the lower bound of performance.

11.2. Most Recent Summarizer

This summarizer chooses the most recent k documents from the selection pool as a summary. It is analogous to choosing the first part of a news article as summary. It was implemented because often intelligent summarizers cannot perform better than simple summarizers that just use the first part of the document as summary.

11.3. SumBasic

SumBasic [11] uses simple word probabilities with an update function to compute the best k posts. It was chosen because it depends solely on the frequency of words in the original text and is conceptually very simple.

11.4. MEAD

This summarizer[8] [40] is a well-known flexible and extensible multi-document summarization system and was chosen to provide a comparison between the more structured document domain—in which MEAD works fairly well—and the domain of Twitter posts being studied. In addition, the default MEAD program is a cluster based summarizer so it will provide some comparison to our cluster summarizer.

11.5. LexRank

This summarizer [14] uses a graph-based method that computes pairwise similarity between two sentences—in our case two documents—and makes the similarity score the weight of the edge between the two sentences. The final score of a sentence is computed based on the weights of the edges that are connected to it. This summarizer was chosen to provide a baseline for graph based summarization instead of direct frequency summarization. Though it does depend on frequency, this system uses the relationships among sentences to add more information and is therefore a more complex algorithm than the frequency-based ones.

11.6. TextRank

This summarizer [15] is another graph-based method that uses the PageRank [16] algorithm. This provided another graph-based summarizer that incorporates potentially more information than LexRank since it recursively changes the weights of documents. Therefore, the final score of each document is not only dependent on how it is related to immediately connected documents but also how those documents are related to other posts. TextRank incorporates the whole complexity of the graph rather than just pairwise similarities.

12. EXPERIMENTAL SETUP FOR MULTI-DOCUMENT SUMMARIZATION

12.1. Manual Summarization

12.1.1. Choice of k

An initial question that we must answer before using any multi-document extractive summarizer on a set of Twitter documents is the question of how many documents are appropriate in a summary. Though it is possible to choose k automatically for clustering [63], we decided to focus our experiments on summaries with a predefined value of k for several reasons. First, we wanted to explore other summarization algorithms for which automatically choosing k is not as straightforward as in the cluster summarization algorithm. For example, the SumBasic summarization does not have any mechanism for choosing the right number of documents in the summary. Second, we thought it would be difficult to perform evaluation where the manual summaries were two or three documents in length and the automatic summaries were five or six documents in length—or vice versa—because the ROUGE evaluation metric is sensitive to length even with some normalization.

To get a subjective idea of what people thought about the value of $k = 4$ after being immersed in manual clustering for a while, we took a survey of the volunteers after they performed clustering of 50 topics—2 people for each of the 25 topics—with 100 documents in each topic. We asked them "How many clusters do you think this should have had?" with the choices "3 (Less)", "4 (About Right)" or "5 (More)". The results of our volunteer survey for this question are given in Figure 11. This survey is probably biased towards "4 (About Right)" because the question does not allow for numbers other than 3, 4 or 5. Therefore, these results must be taken tentatively but they at least suggest that there is some significant variability about the best value for k. Our bias is also based on the fact that our initial 1500 Twitter posts on each topic were obtained within a small interval of 15 minutes so we thought a small number would be good.

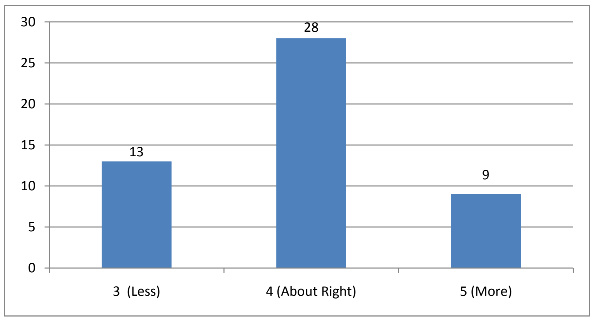


FIGURE 11: Volunteers' answers to question about whether the specified number of clusters ($k = 4$) was right after they had clustered the documents for the given topic.

[8]http://www.summarization.com/mead/

Since the volunteers had already clustered the documents into four clusters, the manual summaries were four documents long as well. This kept the already onerous manual summary creation process somewhat simple. However, this also means that being dependent on a single length for the summaries may impact our evaluation process described next in an unknown way.

12.1.2. Manual Summarization Method

Our manual multi-document summaries were created by volunteers who were undergraduates from around the US gathered together in an NSF-supported REU program. Each of the first 25 topics was manually summarized by two different volunteers[9] by performing steps parallel to the steps of the cluster summarizer. First, the volunteers clustered the documents into 4 clusters ($k = 4$). Second, they chose the most representative document from each cluster. And finally, they ordered the representative documents in a way that they thought was most logical or coherent. These steps were chosen because it was initially thought that a clustering based solution would be the best way to summarize the Twitter documents and it seemed simpler for the volunteers to cluster first rather than simply looking at all the documents at once. These procedures probably biased the manual summaries—and consequently the results—towards clustering based solutions but since the cluster summarizer itself did not perform particularly well in the evaluations, it seems that this bias was not particularly strong.

12.2. Summarization Tests Setup

Because the manual summaries were 4 documents in length, the automated summaries were restricted to producing 4-document summaries as well because the ROUGE metric is sensitive to summary length. For the summarizers that involve random seeding (e.g., random summarizer and cluster summarizer), 100 summaries were produced for each topic to avoid the effects of random seeding. For MEAD, each document was formatted to be one document with a single sentence inside of it. For LexRank—which is implemented in the standard MEAD distribution—the documents for each topic were added to one document as separate sentences. Because the exact implementation of TextRank was unavailable, the TextRank summarizer was implemented internally using the formulas described in [15].

For the Hybrid TF-IDF summarizer, in order to keep the documents from being too similar in content, a preliminary test to determine the best cosine similarity threshold was conducted. The F-measure scores when varying the similarity threshold t of the Hybrid TF-IDF summarizer from 0 to 0.99 are shown in Figure

[9]A total of 16 volunteers produced manual summaries in such a combination that no volunteer would be compared against another specified volunteer more than once.

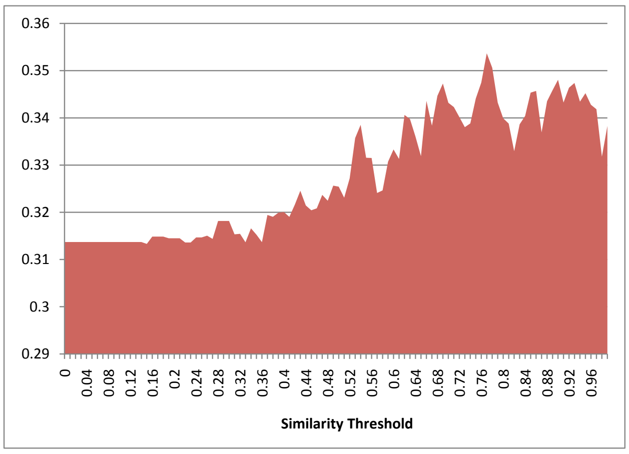


FIGURE 12: F-measures of Hybrid TF-IDF algorithm over different thresholds.

12. The best performing threshold of $t = 0.77$ seems to be reasonable because it allows for some similarity between final summary posts but does not allow them to be nearly identical. We believe that the algorithm would have worked with other choices of thresholds near 0.77.

Because some of the volunteers suggested that some topics had significant noise or miscellaneous posts, the cluster summarizer test varied the value of k for k-way clustering from 4 to 10 ($4 \leq k' \leq 10$) in order to see if changing the number of clusters affected the results of the cluster summarizer. The largest k clusters out of the k' clusters were then chosen, and each of these k clusters was summarized with the Hybrid TF-IDF algorithm like in the original cluster summarizer.

12.2.1. Evaluating Clustering Algorithms

Entropy and purity as explained in [61] were the metrics used for testing the competing clustering algorithms, and they are defined as follows: given a particular cluster X_i of size n_r, the entropy of the cluster is defined as

$$E(X_r) = -\frac{1}{\log q}\sum_{i=1}^{q} \log \frac{n_r^i}{n_r}, \tag{14}$$

where q is the number of classes in the pre-classified posts and n_r^i is the number of posts of the ith class that were assigned to the rth cluster. The total entropy of the clustering solution is

$$E(X) = \sum_{r=1}^{k} \frac{n_r}{n} E(X_r). \tag{15}$$

In general, the smaller the entropy values, the better the clustering solution. Similarly, the purity of a particular cluster is defined as

$$P(X_r) = \frac{1}{n_r}\max(n_r^i), \tag{16}$$

TABLE 7: Average Entropies and Purities

	Avg. Entropy	Avg. Purity
k-means	0.732	0.499
k-means++	0.724	0.508
Bisecting k-means	0.720	0.514
Bisecting k-means++	0.709	0.525

which represents the fraction of the cluster that is made up of the largest class of documents. The total purity of the clustering solution is

$$P(X) = \sum_{r=1}^{k} \frac{n_r}{n} P(X_r). \tag{17}$$

In general, the larger the purity values, the better the clustering solution.

For the clustering tests, the 50 sets of posts based on specified topics were split into 10 test data sets of 5 topics each.

12.2.2. Preliminary Clustering Results and Analysis
In order to avoid the sensitivity of random seeding, 100 k-way clustering solutions were computed for each of the 10 different data sets for a total of 1000 iterations per clustering algorithm. The average purities and entropies of the clustering algorithms are given in Table 7.

The bisecting k-means++ algorithm seemed to perform the best by producing approximately 4% better entropy and 7% better purity than the base k-means algorithm. The bisecting k-means++ algorithm performed the best most likely because it combines the strengths of both the k-means++ algorithm and the bisecting k-means algorithm. As suggested by [39,61], the bisecting k-means algorithm did work better than the standard k-means algorithm. Similarly, as suggested by [62], the k-means++ algorithm performed better than the standard k-means algorithm.

Thus, although there was not a single winner by far among the clustering algorithms, the bisecting k-means++ algorithm performs the best, and hence we decided to use the bisecting k-means++ algorithm for our cluster summarizer.

13. RESULTS AND ANALYSIS OF MULTI-DOCUMENT SUMMARIZATION

As discussed in section 7.3, we use ROUGE-1 as the main evaluation measure for our summaries. In addition, since we want certainty that ROUGE-1 correlates with a human evaluation, we implemented a human evaluation using Amazon Mechanical Turk[10], a paid system that pays human workers small amounts of money for completing a short Human Intelligence Task, or HIT. The HITs used for summary evaluation displayed the summaries to be compared side by side with the topic specified. Then, we asked the user, "The auto-generated summary expresses ______ of the meaning of the human produced summary." The possible answers were "All," "Most," "Some," "Hardly Any" and "None" which correspond to a score of 5 through 1, respectively.

[10]http://www.mturk.com

TABLE 8: Evaluation Numbers for ROUGE and MTurk Evaluations.

Number of summaries	Randomly seeded*	Others
Number of topics	25	25
Summaries per topic	100	1
Total summaries computed	**2500**	**25**
ROUGE evaluation		
ROUGE scores computed	**2500**	**25**
MTurk evaluation		
Number of summaries evaluated	25^+	25
Number of manual summaries per topic	2	2
Evaluators per manual summary	2	2
Total MTurk evaluations	**100**	**100**

* The randomly seeded summaries were the Random Summarizer and the Cluster Summarizer.
$^+$An average scoring post based on the F-measure for each topic was chosen for the MTurk evaluations because evaluating 2500 summaries would have been impractical.

13.1. Summarization Results and Analysis

For the summarizers that involve random seeding (e.g., random summarizer and cluster summarizer), 100 summaries were produced for each topic to avoid the effects of random seeding. These numbers can be seen more clearly in Table 8. Also, because we realized that the overlap of the topic keywords in the summary is trivial since every post contains the keywords, we ignored keyword overlap in our ROUGE calculations.

For the human evaluations using Amazon Mechanical Turk, each automatic summary was compared to both manual summaries by two different evaluators. This leads to 100 evaluations per summarizer as can be seen in Table 8. The manual summaries were evaluated against each other by pretending that one of them was the automatic summary.

The reported average F-measure of all the iterations was computed as the simple average

$$\mathrm{F_{avg}} = \frac{1}{F_n} \sum_{f \in F} f,$$

where F is the set of all F-measures and F_n is the number of F-measures being averaged.

TABLE 9: Average values of F-measure, recall and precision ordered by F-measure.

	F-measure	Recall	Precision
LexRank	0.2027	0.1894	0.2333
Random	0.2071	0.2283	0.1967
Mead	0.2204	0.3050	0.1771
Manual	0.2252	0.2320	0.2320
Cluster	0.2310	0.2554	0.2180
TextRank	0.2328	0.3053	0.1954
MostRecent	0.2329	0.2463	0.2253
Hybrid TF-IDF	0.2524	0.2666	0.2499
SumBasic	0.2544	0.3274	0.2127

Our experiments evaluated eight different summarizers: random, most recent, MEAD, TextRank, LexRank, cluster, Hybrid TF-IDF and SumBasic. Both the automatic ROUGE-based evaluation and the MTurk human evaluation are reported for all eight summarizers in Figures 13 and 14, respectively. The values of average F-measure, recall and precision can be seen in Table 9. The values of average MTurk scores can be seen at the top of Table 11.

13.2. Analysis of Results

13.2.1. General Observations

We performed a paired two-sided T-test for each summarizer compared to each other summarizer for both the ROUGE scores and the human evaluation scores. For the ROUGE scores, the twenty five average F-measure scores corresponding to each topic were used for the paired T-test. For the human evaluation, all hundred evaluation scores were used for the paired T-test. The pairwise matrix of p-values for these tests as well as the average score for each summarizer can be seen in Tables 10 and 11. The bolded p-values indicate that the summarizers are statistically different at the 95% confidence level.

Since the number of unigrams in the automated summary could significantly affect the ROUGE scores, the average number of characters for each summarizer is shown in Figure 15. The high average number of characters for the MEAD, TextRank and SumBasic summarizers—approximately 50% higher than the manual summaries—explains why the recall values of the MEAD, TextRank and SumBasic summarizers are particularly high. In addition, the results help explain why the recall of every summarizer except the LexRank summarizer is not higher than their corresponding precision measures since the average number of characters for all the other summarizers is greater than the average number of characters for the manual summaries.

Overall, it seems from these results that the simple frequency based summarizers, namely SumBasic and Hybrid TF-IDF, perform better than summarizers that incorporated more information or more complexity such as LexRank, TextRank or MEAD. This probably has much to do with the special nature of Twitter documents in which documents often have very little structure and have so few words that forming relationships between pairs of documents is not particularly helpful. In addition, since each document is mostly uncorrelated with any other document—except for replies and retweets—, thematic or progressive development of a topic is rare, and therefore, more complex relational models will probably not capture more topical information than frequency models. More specific analysis on each of the summarizers is described in the following sections.

13.2.2. Manual Summaries

Though the manual to manual F-measure scores seem low at 0.3291, this may be explained by several factors. First, the instructions given to the volunteers for summarizing did not give any guidelines on how to cluster the documents except whatever themes or subtopics the volunteers thought could be good clusters. Therefore, the clusters for a topic may have been significantly different from one person to another depending on how they wanted to differentiate the posts. Second, some topics only had thematic overlap rather than unigram overlap. For example, the topic "#MM" was a topic that stood for "Music Mondays" and the tweets would simply have names of songs or names of artists. Obviously, the names of songs or artists do not tend to overlap naturally. In addition, these results seem to agree with the fairly low F-measure scores computed for one sentence summaries in [43, 47–49].

It may seem odd that the manual to manual scores are actually lower than some of the other summarizers' scores. However, this is possible with the F-measure scores because if the summarizer produced a summary that was very similar to one of the manual summaries, it would score a very high F-measure compared to the first manual summary. Then, when compared to the second summary, it could also be decently similar since it is not exactly like the other manual summary. In this way, F-measure scores could be higher than the manual to manual F-measure scores. In a similar manner, the human evaluation scores that are higher on average than manual could mean that the generated summaries captured some of the good ideas from both manual summaries better than each manual summary captured the ideas of the other manual summary.

13.2.3. Random and Most Recent Summarizer

The seemingly high F-measure and human score of the random summarizer may be explained by a few characteristics of microblog documents. First, microblog documents on a given topic tend to use similar words so a fair amount of incidental word and theme overlap seems reasonable. Second, a particularly

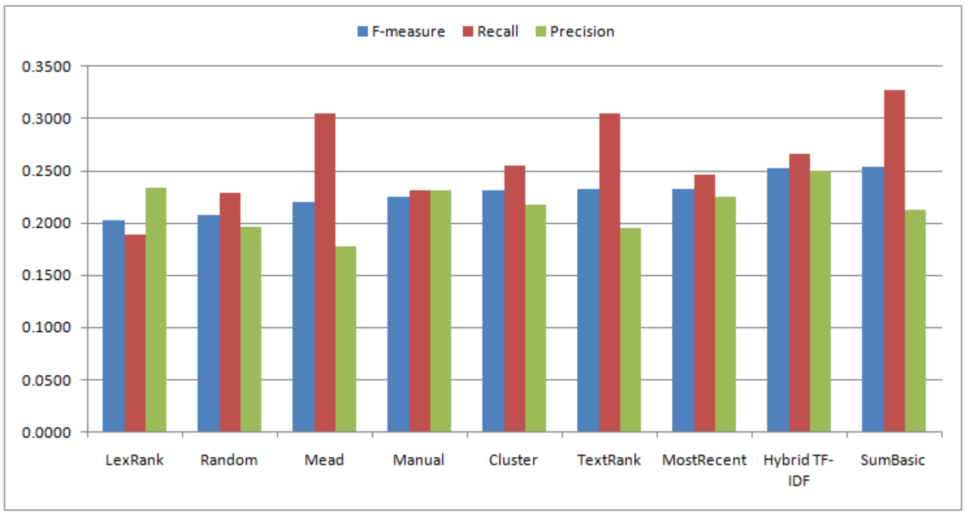


FIGURE 13: Average F-measure, precision and recall ordered by F-measure.

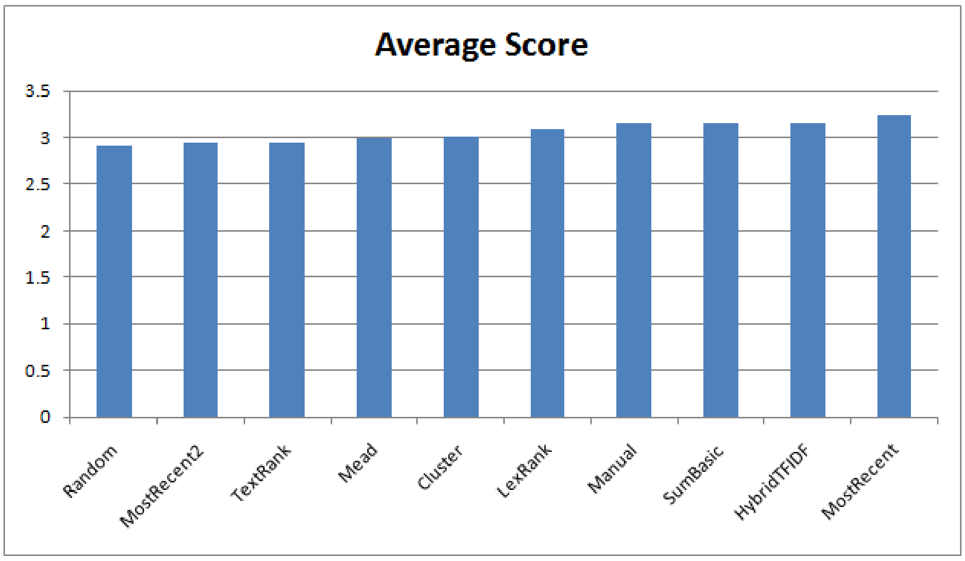


FIGURE 14: Average scores for human evaluation using Amazon Mechanical Turk ordered by average score.

interesting document on a given topic can often be quoted verbatim by many other microblog users so the random summarizer has a better chance of selecting one of these informative retweeted documents. Third, the preprocessing of the documents helped reduce the original set of documents to a set of less noisy documents.

13.2.4. Most Recent Summarizer

Because the test documents were collected by getting documents for trending topics—very active topics—, most of the documents for each topic were generated within fifteen minutes of each other or even less. Therefore, it seemed unlikely that the most recent documents would be particularly more relevant than random documents. The p-values for the F-measure scores seem to agree with only a p-value of 0.162 when comparing the Random summarizer to the most recent summarizer. However, the human scores showed that maybe the most recent summarizer was better than random. One possible reason for the most recent summarizer doing better than expected is that the manual summaries may be biased towards the more recent documents because these were displayed to the volunteers first and may have biased their judgments of the best documents to use for the summary.

Initially, the first set of human scores for evaluating the most recent summarizer—the results marked "MostRecent"—was surprisingly higher than expected outperforming SumBasic and Hybrid TF-IDF even though its F-measure scores were significantly lower than SumBasic or Hybrid TF-IDF. Because of this, we considered that maybe the results were skewed by one or two Mechanical Turk workers who rated a few of the documents significantly higher than most people would.

Therefore, we decided to retrieve a second set of

TABLE 10: P-values for two-sided paired T-test for F-measures in the experiments.

Avg. F-meas.	0.187	0.207	0.220	0.225	0.232	0.233	0.233	0.252	0.254
	LR	Rand.	Mead	Man.	Clust.	TR	MR	Hyb.	SB
LexRank		0.102	**0.040**	0.140	**0.004**	**0.010**	**0.043**	**0.003**	**0.001**
Random	0.102		0.226	0.318	**0.014**	**0.021**	0.162	**0.011**	**0.002**
Mead	**0.040**	0.226		0.789	0.364	0.408	0.554	0.109	**0.027**
Manual	0.140	0.318	0.789		0.714	0.644	0.783	0.229	0.141
Cluster	**0.004**	**0.014**	0.364	0.714		0.946	0.963	0.095	0.069
TextRank	**0.010**	**0.021**	0.408	0.644	0.946		0.995	0.329	0.098
MostRecent	**0.043**	0.162	0.554	0.783	0.963	0.995		0.429	0.333
HybridTFIDF	**0.003**	**0.011**	0.109	0.229	0.095	0.329	0.429		0.920
SumBasic	**0.001**	**0.002**	**0.027**	0.141	0.069	0.098	0.333	0.920	

TABLE 11: P-values for two-sided paired T-test for human evaluation for the experiments.

Avg. Score	2.91	2.94	2.94	3.00	3.01	3.09	3.15	3.15	3.16	3.24
	Rand.	MR2	TR	Mead	Clus.	LR	Man.	SB	Hyb.	MR
Random		0.801	0.765	0.331	0.305	**0.049**	**0.019**	**0.011**	**0.007**	**0.000**
MostRecent2*	0.801		1.000	0.615	0.531	0.156	0.058	0.077	0.072	**0.011**
TextRank	0.765	1.000		0.534	0.461	0.116	**0.042**	**0.048**	**0.030**	**0.005**
Mead	0.331	0.615	0.534		0.919	0.314	0.108	0.096	0.103	**0.012**
Cluster	0.305	0.531	0.461	0.919		0.407	0.167	0.123	0.092	**0.021**
LexRank	**0.049**	0.156	0.116	0.314	0.407		0.463	0.488	0.461	0.104
Manual	**0.019**	0.058	**0.042**	0.108	0.167	0.463		1.000	0.921	0.358
SumBasic	**0.011**	0.077	**0.048**	0.096	0.123	0.488	1.000		0.910	0.307
HybridTFIDF	**0.007**	0.072	**0.030**	0.103	0.092	0.461	0.921	0.910		0.304
MostRecent*	**0.000**	**0.011**	**0.005**	**0.012**	**0.021**	0.104	0.358	0.307	0.304	

* Please see Section 13.2.4 for an explanation of the two different results for the Most Recent summarizer.

one hundred evaluations with essentially the same procedure—the results marked "MostRecent2"—and got significantly lower scores than the previous time. In fact, the p-value for comparing the two sets of results is 0.011. Therefore, it seems that we may have been correct that the first set of results was skewed. Even if both results are valid, the average of both result sets would be 3.09, which is a little lower than the manual scores. Therefore, in general, it seems that the most recent summarizer does not perform better than the random summarizer. This would agree with our original idea that the most recent documents should not be inherently more relevant than random documents because of the nature of Twitter documents being generated generally haphazardly without order or specific thematic development.

13.2.5. Frequency Based Summarizers (SumBasic and Hybrid TF-IDF)

The simple frequency based summarizers seemed to outperform all other algorithms both in F-measure scores and human evaluation scores. For the F-measure scores, both are significantly different from the LexRank and Random summarizers, and SumBasic is significantly different than the MEAD summarizer. For the human evaluation scores, both are significantly different from Random and TextRank.

The Hybrid TF-IDF summarizer can be seen as adding a little complexity to the simple SumBasic algorithm by including information regarding the IDF component of the term frequency calculation. From the results, it seems that this added complexity is not particularly helpful in computing summaries. However, it should be noted that the Hybrid TF-IDF has a closer balance between precision and recall whereas the SumBasic algorithm has a higher recall than precision. This suggests that the SumBasic algorithm may be biased towards longer summaries but does not necessarily affect its performance or overall usefulness.

Both algorithms employ a redundancy reduction method in order to avoid summaries that have very similar documents in them. In addition, both use a greedy approach to this reduction by selecting the next best document—either the best weighted non-redundant document in the case of Hybrid TF-IDF or the best weighted document after weight recalculation in SumBasic. Therefore, it seems that simple word frequency calculations and redundancy reduction are particularly important for summarizing Twitter topics.

13.2.6. Cluster Based Summarizers (MEAD and Cluster)

The two cluster based summarizers—the MEAD summarizer and our implementation of a cluster

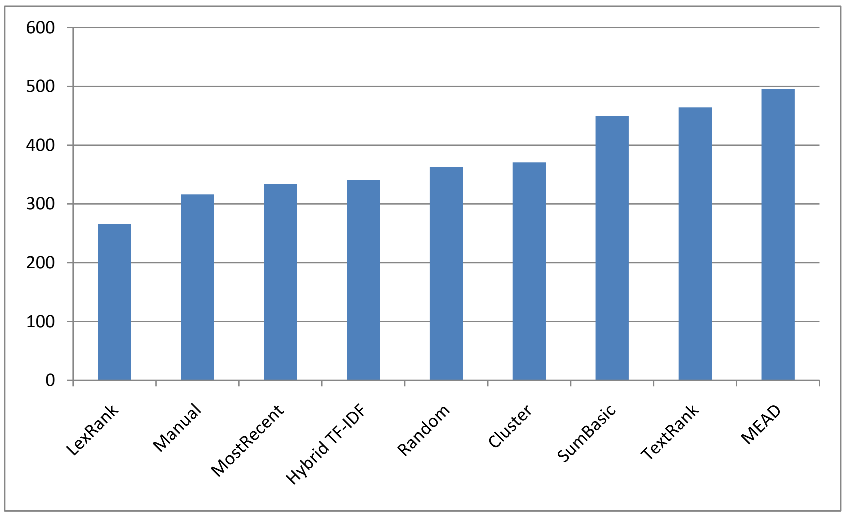


FIGURE 15: Average number of characters for each summarizer.

summarizer—did not do as well as expected. They performed significantly better than the Random summarizer in the F-measure scores but did not perform significantly better in the human evaluation. Like with the frequency summarizers, these summarizers attempted to reduce redundancy but did so by clustering the documents first and then summarizing based on these clusters. However, clustering did not seem to increase performance. This could be true because of the short, unstructured and informal nature of Twitter documents that does not correlate with the expectations of more traditional techniques for summarizing that use clustering. Also, the documents may be particularly difficult to cluster unlike more structured or longer document collections because they have so few non-zero features. Since the default MEAD summarizer has much better recall than precision, it may be improved upon by normalizing the weights of sentences more strongly than the default.

13.2.7. Graph Based Summarizers (LexRank and TextRank)

The results of the two graph based summarizers—LexRank and TextRank—were intriguing. In the F-measure scores, LexRank did worse than random but TextRank did decently well by at least being significantly different than random. However, in the human scores, TextRank did not even perform significantly better than random whereas LexRank performed significantly better than random. One interesting aspect that may explain some of this is that LexRank was the only algorithm that had a better precision than recall. This may suggest that LexRank in general chose shorter more concise documents for its summaries. TextRank was the opposite by having a much higher recall than precision. Because the human evaluation scores suggest that LexRank performed better than TextRank, these algorithms may suggest that the F-measure score is biased towards longer summaries.

In general, however, since the frequency based summarizers did better—and in a several cases significantly better—than the graph based summarizers, it seems that the added complexity of interrelationships did not help in summarizing Twitter posts.

14. CONCLUSION

In the first part of this paper, we presented two algorithms for microblog summarization. The Phrase Reinforcement algorithm we develop provides very good summaries by creating two partial summary graphs on both sides of the topic phrase by picking out the most frequently used words, indexed by position away from the center phrase, on the two sides. However, we find, after exhaustive experimentation, that an adaptation of the TF-IDF algorithm produces as good summaries as the PR algorithm or even better.

Encouraged by the results in the first part of the paper, we extended our goal to produce a multiple post summary of a given topic in order to understand better the information that microblogs provide. For multiple post summaries, the Hybrid TF-IDF summarizer with a similarity threshold of 0.77 produced significantly better results than the random summarizer and seems to be competitive with manually generated summaries. In addition, it outperforms several of the leading summarization systems MEAD, LexRank and TextRank. This suggests that microblog posts cannot be treated as traditional documents. A simple extension of the work reported in this paper may be to attempt to penalize longer posts especially for the MEAD and TextRank summarizers to see if it improves their F-measures.

This project could be further extended in many ways. First, alternative IR-ranking algorithms such as BM25 and its variations [53, 64, 65] should be considered. BM25 term weighting has been used widely and successfully across a range of collections for search tasks. Second, methods for dynamically discovering a good value for k of given topic could be researched. Third, from the clustering results, it seems that clustering microblog posts is not as simple or clean as clustering normal structured documents, and therefore, some new ways of clustering or computing feature vectors could be explored. For better clustering tests, the manually produced clusters from the manual summarization task could be used as predefined classes instead of using classes defined by a topic phrase. Fourth, if a list of the most significant current topics could be computed (e.g., see [66]), a summary of all the most significant topics could be generated in real time. It may also be possible to produce a topic browsing and summarization tool that will help people have a more comprehensive idea about real time microblog information. Fifth, the coherence of the multiple post summary could be researched in depth. Sophisticated methods for ordering standard documents have been explored by [67,68], and these advanced methods could be applied to microblog summary cohesion. Also, other coherence issues such as pronoun resolution and fragmented arguments are issues that all summarization techniques need to consider [51]. Finally, it would be interesting to evaluate the usefulness of the Twitter summaries in the context of a question-answering system, in the spirit of [69–71], where a user asks a question and the answering system analyzes the corpus of tweets and summarizes them to produce a relevant and useful answer.

ACKNOWLEDGMENT

The work reported in this paper has been partially supported by NSF grant ARRA: 0851783. David Inouye was an undergraduate researcher supported by this NSF Research Experience for Undergraduates (REU) Grant during the summer of 2010.

REFERENCES

[1] Radev, D., Hovy, E., and McKeown, K. (2002) Introduction to the Special Issue on Summarization. *Computational Linguistics*, **28**, 408.

[2] Jones, K. S. (2007) Automatic Summarising: The State of the Art. *Information Processing & Management*, **43**, 1449–1481.

[3] Lin, J., Ozsu, M., and Liu, L. (2009) Summarization. *Encyclopedia of Database Systems*. Springer.

[4] Luhn, H. P. (1958) The Automatic Creation of Literature Abstracts. *IBM Journal of Research and Development*, **2**, 159–165.

[5] Edmundson, H. P. (1969) New Methods in Automatic Extracting. *Journal of the Association for Computing Machinery*, **16**, 264–285.

[6] DeJong, G. (1982) An Overview of the FRUMP System. *Strategies for Natural Language Processing.* Lawrence Erlbaum Associates, Mahwah, NJ, USA.

[7] Reimer, U. and Hahn, U. (1988) Text Condensation as Knowledge Base Abstraction. *Fourth IEEE Conference on AI Applications*, pp. 338–344, . San Diego, CA, USA.

[8] Rau, L., Jacobs, P., and Zernik, U. (1989) Information Eextraction and Text Summarization using Linguistic Knowledge Acquisition. *Information Processing & Management*, **25**, 419–428.

[9] Kupiec, J., Pedersen, J., and Chen, F. (1995) A trainable document summarizer. *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Seattle, WA, USA, pp. 68–73.

[10] Yeh, J., Ke, H., Yang, W., Meng, I., et al. (2005) Text Summarization Using a Trainable Summarizer and Latent Semantic Analysis. *Information Processing & Management*, **41**, 75–95.

[11] Vanderwende, L., Suzuki, H., Brockett, C., and Nenkova, A. (2007) Beyond SumBasic: Task-focused Summarization with Sentence Simplification and Lexical Expansion. *Information Processing & Management*, **43**, 1606–1618.

[12] Radev, D., Blair-Goldensohn, S., and Zhang, Z. (2001) Experiments in Single and Multi-document Summarization using MEAD. *Document Understanding Conference, DUC-01*, New Orleans, LA, USA.

[13] Althaus, E., Karamanis, N., and Koller, A. (2004) Computing Locally Coherent Discourses. *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, Barcelona, Spain.

[14] Erkan, G. and Radev, D. (2004) LexRank: Graph-based Centrality as Salience in Text Summarization. *Journal of Artificial Intelligence Research*, **22**, 457–480.

[15] Mihalcea, R. and Tarau, P. (2004) TextRank: Bringing Order into Texts. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 404–411. Barcelona, Spain.

[16] Brin, S. and Page, L. (1998) The Anatomy of a Large-scale Hypertextual Web Search Engine. *Computer Networks and ISDN Systems*, **30**, 107–117.

[17] Barzilay, R. and Lapata, M. (2008) Modeling Local Coherence: An Entity-based Approach. *Computational Linguistics*, **34**, 1–34.

[18] Kolcz, A., Prabakarmurthi, V., and Kalita, J. (2001) Summarization as Feature Selection for Text Categorization. *Proceedings of the Tenth International Conference on Information and Knowledge Management*, Atlanta, GA, USA, pp. 365–370.

[19] Ganti, V., Gehrke, J., and Ramakrishnan, R. (1999) CACTUS—Clustering Categorical Data Using Summaries. *KDD '99: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 73–83.

[20] Kalita, J., Colbourn, M., and McCalla, G. (1984) A Response to the Need for Summary Responses. *Proceedings of the 10th International Conference on Computational Linguistics and 22nd Annual Meeting of the Association for Computational Linguistics*, Stanford, CA, USA, pp. 432–436.

[21] Kalita, J., Jones, M., and McCalla, G. (1986) Summarizing Natural Language Database Responses. *Computational Linguistics*, **12**, 107–124.

[22] Mahesh, K. (1997) Hypertext Summary Extraction for Fast Document Browsing. *Proceedings of the AAAI Spring Symposium on Natural Language Processing for the World Wide Web*, Stanford, CA, USA, pp. 95–103.

[23] Berger, A. and Mittal, V. (2000) OCELOT: A System for Summarizing Web Pages. *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Athens, Greece, pp. 144–151.

[24] Buyukkokten, O., Garcia-Molina, H., and Paepcke, A. (2001) Seeing the Whole in Parts: Text Summarization for Web Browsing on Handheld Devices. *Proceedings of the 10th International Conference on World Wide Web*, Hong Kong, China, pp. 652–662.

[25] Sun, J., Shen, D., Zeng, H., Yang, Q., Lu, Y., and Chen, Z. (2005) Web-page Summarization Using Clickthrough Data. *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Salvador, Brazil, pp. 194–201.

[26] Tigelaar, A., op den Akker, R., and Hiemstra, D. (2010) Automatic Summarisation of Diiscussion Fora. *Natural Language Engineering*, **16**, 161–192.

[27] Zhou, L. and Hovy, E. (2006) On the Summarization of Dynamically Introduced Information: Online Discussions and Blogs. *Proceedings of AAAI-2006 Spring Symposium on Computational Approaches to Analyzing Weblogs*, Stanford, CA, USA, pp. 237–242.

[28] Ku, L., Liang, Y., and Chen, H. (2006) Opinion Extraction, Summarization and Tracking in News and Blog Corpora. *Proceedings of AAAI-2006 Spring Symposium on Computational Approaches to Analyzing Weblogs*, Stanford, CA, USA, pp. 100–107.

[29] Hu, M., Sun, A., and Lim, E. (2007) Comments-oriented Blog Summarization by Sentence Extraction. *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management*, Lisbon, Portugal, pp. 901–904.

[30] Mani, I. and Bloedorn, E. (1997) Multi-document Summarization by Graph Search and Matching. *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Conference on Innovative Applications of Artificial Intelligence*, Providence, RI, USA, pp. 622–628.

[31] Carbonell, J. and Goldstein, J. (1998) The Use of MMR, Diversity-based Reranking for Reordering Documents and Producing Summaries. *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Melbourne, Australia, pp. 335–336.

[32] Barzilay, R., McKeown, K., and Elhadad, M. (1999) Information Fusion in the Context of Multi-document Summarization. *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, College Park, MD, USA, pp. 550–557.

[33] Goldstein, J., Mittal, V., Carbonell, J., and Kantrowitz, M. (2000) Multi-document Summarization by Sentence Extraction. *NAACL-ANLP 2000 Workshop on Automatic Summarization*, Seattle, WA, USA, pp. 40–48.

[34] Lin, C. and Hovy, E. (2002) From Single to Multi-document Summarization: A Prototype System and its evaluation. *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, Philadelphia, PA, USA, pp. 457–464.

[35] Radev, D., Jing, H., Sty, M., and Tam, D. (2004) Centroid-based Summarization of Multiple Documents. *Information Processing & Management*, **40**, 919–938.

[36] Madnani, N., Zajic, D., Dorr, B., Ayan, N., and Lin, J. (2007) Multiple Alternative Sentence Compressions for Automatic Text Summarization. *Proceedings of the 2007 Document Understanding Conference (DUC-2007) at NLT/NAACL*, Rochester, NY, USA.

[37] Jurafsky, D. and Martin, J. H. (2009) *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Prentice Hall, Upper Saddle River, NJ, USA.

[38] Manning, C. D. and Schuetze, H. (2002) *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, MA, USA.

[39] Dunham, M. H. (2003) *Data Mining: Introductory and Advanced Topics*. Prentice Hall, Upper Saddle River, NJ, USA.

[40] Radev, D., Allison, T., Blair-Goldensohn, S., Blitzer, J., Çelebi, A., Dimitrov, S., Drabek, E., Hakim, A., Lam, W., Liu, D., Otterbacher, J., Qi, H., Saggion, H., Teufel, S., Topper, M., Winkel, A., and Zhang, Z. (2004) MEAD—A Platform for Multidocument Multilingual Text Summarization. *The International Conference on Language Resources and Evaluation (LREC) 2004*, Lisbon, Portugal.

[41] Harabagiu, S. and Hickl, A. (2011) Relevance Modeling for Microblog Summarization. *Proceedings of the Fifth International Conference on Weblogs and Social Media (ICWSM)*, Barcelona, Spain, pp. 514–517.

[42] Takamura, H., Yokono, H., and Okumura, M. (2011) Summarizing a Document Stream. *Advances in Information Retrieval*, pp. 177–188. Springer, New York, NY, USA.

[43] Sharifi, B., Hutton, M.-A., and Kalita, J. K. (2010) Summarizing Microblogs Automatically. *Annual Conference of the National Association for Advancement of Computational Intelligence-Human Language Technology (NAACL-HLT)*, Los Angeles, CA, USA, pp. 685–688.

[44] Huang, Y. and Mitchell, T. M. (2006) Text Clustering with Extended User Feedback. *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Seattle, WA, USA, pp. 413–420.

[45] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977) Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, **39**, 1–38.

[46] Cohen, W. W. and Singer, Y. (1996) Context-sensitive Learning Methods for Text Categorization. *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Zurich, Switzerland, pp. 307–315.

[47] Sharifi, B. (2010) Automatic Microblog Classification and Summarization. Master's thesis. M.S. Thesis, Department of Computer Science, University of Colorado, Colorado Springs, CO, USA.

[48] Sharifi, B., Hutton, M.-A., and Kalita, J. K. (2010) Experiments in Microblog Summarization. *Second IEEE International Conference on Social Computing (SocialCom 2010)*, Minneapolis, MN, USA, pp. 49–56.

[49] Sharifi, B., Hutton, M., and Kalita, J. (2010) Automatic Summarization of Twitter Topics. *National Workshop on Design and Analysis of Algorithm*, Tezpur, Assam, India, pp. 121–128.

[50] Inouye, D. and Kalita, J. K. (2011) Comparing Twitter Summarization Algorithms for Multiple Post Summaries. *Privacy, Security, Risk and Trust (PASSAT), 2011 IEEE Third International Conference on Social Computing (SocialCom)*, Boston, MA, pp. 298–306.

[51] Hahn, U. and Mani, I. (2000) The Challenges of Automatic Summarization. *IEEE Computer*, **33**, 29–36.

[52] Salton, G. (1989) *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[53] Manning, C., Raghavan, P., and Schütze, H. (2008) *An Introduction to Information Retrieval*. Cambridge University Press, Cambridge, MA, USA.

[54] Seki, Y. (2002) Sentence Extraction by TF/IDF and Position Weighting from Newspaper Articles. *Proceedings of the 3rd National Institute of Informatics Test Collection Information Retrieval (NTCIR) Workshop*, Tokyo, Japan.

[55] Singhal, A., Buckley, C., and Mitra, M. (1996) Pivoted Document Length Normalization. *SIGIR '96: Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Zurich, Switzerland, pp. 21–29.

[56] Lin, C.-Y. and Hovy, E. (2003) Automatic Evaluation of Summaries Using N-gram Co-occurrence Statistics. *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, Edmonton, Alberta, Canada, pp. 71–78.

[57] Saggion, H., Torres-Moreno, J.-M., Cunha, I. d., and SanJuan, E. (2010) Multilingual Summarization Evaluation Without Human Models. *Proceedings of the 23rd International Conference on Computational Linguistics*, Beijing, China, pp. 1059–1067.

[58] Liu, F. and Liu, Y. (2008) Correlation between ROUGE and Human Evaluation of Extractive Meeting Summaries. *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies*, Columbus, OH, USA, pp. 201–204.

[59] Liu, F. and Liu, Y. (2010) Exploring Correlation between ROUGE and Human Evaluation on Meeting Summaries. *IEEE Transactions on Audio, Speech, and Language Processing*, **18**, 187–196.

[60] Louis, A. and Nenkova, A. (2009) Automatically Evaluating Content Selection in Summarization without Human Models. *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, Singapore, pp. 306–314.

[61] Zhao, Y. and Karypis, G. (2001) Criterion Functions for Document Clustering: Experiments and Analysis. Technical report. Department of Computer Science, University of Minnesota, Minneapolis, MN, USA.

[62] Arthur, D. and Vassilvitskii, S. (2007). *SODA '07: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, New Orleans, LA, USA, pp. 1027–1035.

[63] Tibshirani, R., Walther, G., and Hastie, T. (2001) Estimating the Number of Clusters in a Data Set via the Gap Statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **63**, 411–423.

[64] Robertson, S., Zaragoza, H., and Taylor, M. (2004) Simple BM25 Extension to Multiple Weighted Fields. *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*, Washington DC, USA, pp. 42–49.

[65] Pérez-Iglesias, J., Pérez-Agüera, J. R., Fresno, V., and Feinstein, Y. Z. (2009) *Integrating the Probabilistic Models BM25/BM25F into Lucene*. arXiv preprint arXiv:0911.5046.

[66] Benhardus, J. and Kalita, J. (2013) Streaming Trend Detection in Twitter. *International Journal of Web Based Communities*, **9**, 122–139.

[67] Barzilay, R., Elhadad, N., and McKeown, K. (2001) Sentence Ordering in Multidocument Summarization. *Proceedings of the First International Conference on Human Language Technology Research*, San Diego, CA, USA, pp. 1–7.

[68] Lapata, M. (2003) Probabilistic Text Structuring: Experiments with Sentence Ordering. *Proceedings of the annual meeting of the Association for Computational Linguistics*, Edmonton, Alberta, Canada, pp. 545–552.

[69] Melli, G., Wang, Y., Liu, Y., Kashani, M. M., Shi, Z., Gu, B., Sarkar, A., and Popowich, F. (2005) Description of SQUASH, the SFU question answering summary handler for the DUC-2005 summarization task. *Document Understanding Conference 2005*, Vancouver, BC, Canada.

[70] Moreno, J. M. T., St-Onge, P.-L., Gagnon, M., El-Bèze, M., and Bellot, P. (2009) Automatic summarization system coupled with a question-answering system (qaas). *Computing Research Repository*, **http://arxiv.org/abs/0905.29900**.

[71] Biryukov, M., Angheluta, R., and Moens, M.-F. (2005) Multidocument question answering text summarization using topic signatures. *Journal of Digital Information Management*, **3**, 27–33.