Better Twitter Summaries?

Joel Judd & Jugal Kalita
Department of Computer Science
University of Colorado
Colorado Springs, Colorado
Email: {jjudd2,jkalita}@uccs.edu

Abstract

This paper describes an approach to improve summaries for a collection of Twitter posts created using the Phrase Reinforcement (PR) Algorithm (Sharifi et al., 2010a). The PR algorithm often generates summaries with excess text and noisy speech. We parse these summaries using a dependency parser and use the dependencies to eliminate some of the excess text and build better-formed summaries. We compare the results to those obtained using the PR Algorithm.

1 Introduction

Millions of people use the Web to express themselves and share ideas. Twitter is a very popular micro blogging site. According to a recent study approximately 340 million Tweets are sent out every day[1]. People mostly upload daily routines, fun activities and other words of wisdom for readers. There is also plenty of serious information beyond the personal; according to a study approximately 4% of posts on Twitter have relevant news data[2]. Topics that may be covered by reputable new sources like CNN (Cable News Network) were considered relevant. A topic is simply a keyword or key phrase that one may use to search for Twitter posts containing it. It is possible to gather large amounts of posts from Twitter on many different topics in short amounts of time. Obviously, processing all this information by human hands is impossible. One way to extract information from Twitter posts on a certain topic is to automatically summarize them. (Sharifi et al., 2010a; Sharifi et al., 2010b; Sharifi et al., 2010c) present an algorithm called the Phrase Reinforcement Algorithm to produces summaries of a set of Twitter posts on a certain topic. The PR algorithm produces good summaries for many topics, but for sets of posts on certain topics, the summaries become syntactically malformed or too wordy. This is because the PR Algorithm does not pay much attention to syntactic well-formedness as it constructs a summary sentence from phrases that occur frequently in the posts it summarizes. In this paper, we attempt to improve Twitter summaries produced by the PR algorithm.

[1] http://blog.twitter.com/2012/03/twitter-turns-six.htm

[2] http://www.pearanalytics.com/blog/wp-content/uploads/2010/05/Twitter-Study-August-2009.pdf

2 The PR Algorithm Revisited

Given a number of Twitter posts on a certain topic, the PR algorithm starts construction of what is called a word graph with a root node containing the topic phrase. It builds a graph showing how words occur before and after the phrase in the root node, considering all the posts on the topic. It builds a subgraph to the left of the topic phrase and another subgraph to its right in a similar manner. To construct the left graph, the algorithm starts with the root node and obtains the set of words that occur immediately before the current node's phrase. 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. 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 to construct the right subgraph. Once the full graph is there, the algorithm weights individual nodes. The weights are initialized to the same values as their frequency counts. Then, to account for the fact that some phrases are naturally longer than others, they penalize nodes that occur farther from the root node by an amount that is proportional to their distance. To generate a summary, the algorithm looks for the most overlapping phrases within the graph. Since the nodes' weights are proportional to their overlap, the algorithm searches for the path within the graph

Proceedings of NAACL-HLT 2013, pages 445–449,
Atlanta, Georgia, 9–14 June 2013. ©2013 Association for Computational Linguistics

with the highest cumulative weight. The sequence of words in this path becomes the summary.

3 Problem Description

We start by making some observations on the phrase-reinforcement algorithm. Certain topics do not produce well-formed summaries, while others yield very good summaries. For the posts that have a well-centered topic without a huge amount of variation among the posts, the algorithm works well and creates good summaries. Here is an example summary produced by the PR algorithm.

> *Phillies defeat Dodgers to take the National League Championship series.*

(Sharifi et al., 2010a; Sharifi et al., 2010b; Sharifi et al., 2010c) provide additional examples. The PR algorithm limits the length of the summary to approximately 140 characters, the maximum length of a Twitter post. However, often the summary sentence produced has extraneous parts that appear due to the fact that they appear frequently in the posts being summarized, but these parts make the summary malformed or too wordy. An example with some wordiness is given below.

> *today is day for vote obama this election day*

Some "raw" PR summaries are a lot more wordy than the one above. The goal we address in this paper is to create grammatically better formed summaries by processing the "raw" summaries formed by the PR Algorithm. We drop this excess text and the phrases or extract pieces of text which make sense grammatically to form the final summary. This usually produces a summary with more grammatical accuracy and less noise in between the words. This gets the main point of the summary across better.

4 Approach

The idea behind creating the desired summary is to parse the "raw" summary and build dependencies between the dependent and governor words in each summary. We perform parts of speech tagging and obtain lists of governing and dependent words. This data forms the basis for creating a valid summary. For example given the Twitter post, *today is day for vote obama this election day*, a dependency parser produces the governor-dependent relationships as given in Table 1. Figure 1 also shows the same grammatical dependencies between words in the phrases.

We believe that a word which governs many words is key to the phrase as a whole, and dependent words which are closely related, or in other words, lay close to each other in the phrase should be left in the order they appear. Conceptually, our approach works as follows: look at every word and see if it makes sense with the word before and after it. This builds dependencies between the word in question with the words around it. If a word before or after the word being analyzed does not make sense grammatically, it can be removed from that grammatically correct phrase. Dependent words that are not close to each other may not be as important as words that lay close to each other and have more dependencies, and thus may be thrown out of the summaries. Through this process grammatically correct phrases can be formed.

The dependencies are built by tagging each word as a part of speech and seeing if it relates to other words. For example, it checks whether or not the conjunction "and" is serving its purpose of combining a set of words or ideas, in other words, if those dependencies exist. If dependencies exist with the nearby words, that given collection of words can be set aside as a grammatically correct phrase until it reaches words with no dependencies, and the process

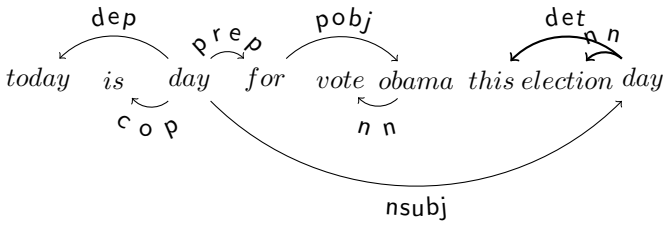
Table 1: Governor and Dependent Words for *today is day*[1] *for vote obama this election day*[2]

Governor	Dependent
day[1]	today
	is
	for
	day[2]
obama	vote
for	obama
day[2]	this
	election

Algorithm 1 Algorithm to Fix "Raw" PRA Summaries

I. For each word, check grammatical compatibility with words before and after the word being checked.
II. If a word has no dependencies immediately before or after it, drop the word.
III. After each word has been checked, check for the words that form a grammatical phrase.
IV. Write out the summary without the dropped words and without phrases with only two words.
V. If needed, go back to step III, because there shouldn't be any more single words with no dependencies to check, and repeat as many times as necessary.

Figure 1: Dependency Parse for *today is day*[1] *for vote obama this election day*[2]



can continue. The phrases with few words can be dropped, as well as single words. These new phrases can be checked for grammatical accuracy in the same way as the previous phrases, and if they pass, can remain combined forming a longer summary that should be grammatically correct. The main steps are given in Algorithm 1.

Now, take the example summary produced by the PR Algorithm for the election Twitter posts. Looking at this summary, we, as humans, may make changes and make the summary grammatically correct. Two potential ideal summaries would be the following.

today is the day to vote for obama

vote for obama this election day

The actual process used in the making of the grammatical summaries is as follows. Two main lists are created from lists of governor and dependent words, one with the governor words and another with the dependent words. The governor words are checked to see how many dependent words are linked to them. The governing words with the highest number of dependent words are kept for later. For example using the above phrase about the elections, the word "day" was the governing word with the highest amount of dependent words and was thus kept for the final summary. The superscripts on the word "day" differentiate its two occurrences. The dependent words are kept in groups of closely linked dependent words. Using the same example about the election, an intermediate list of closely related dependent words is "today," "is," "for," "vote," "obama," "this," "election," and "day." And the final list of closely related dependent words is "for," "vote," "obama," "this," "election" and "day." After these two lists are in the final stages the lists are merged placing the words in proper order.

5 Experiments and Results

To begin, the Twitter posts were collected manually and stored in text files. The topics we chose to focus on important current events and some pop culture. Approximately 100 posts were collected on ten different topics. These topics are "The Avengers," "Avril Lavigne," "Christmas," "the election," "Election Day," "Iron Man 3," "president 2012," "Hurricane Sandy," "Thanksgiving," and "vote."

Table 2: ROUGE-L without Stopwords, Before

Task	Recall	Precision	F-score
Task 1	0.667	0.343	0.453
Task 2	1.000	0.227	0.370
Task 3	0.353	0.240	0.286
Task 4	0.800	0.154	0.258
Task 5	1.000	0.185	0.313
Task 6	0.667	0.150	0.245
Task 7	0.889	0.125	0.219
Task 8	0.636	0.125	0.209
Task 9	0.500	0.300	0.375
Task 10	0.455	0.100	0.164
Average	0.696	0.195	0.289

The collections of posts were passed on to three volunteers to produce short accurate summaries that capture the main idea from the posts. The collections of posts were also first run through the PR Algorithm and then through the process described in this paper to try and refine the summaries output by the PR Algorithm. The Stanford CoreNLP parser[3] was used to build the lists of governor and dependent words.

We use ROUGE evaluation metrics (Lin 2004) just like (Sharifi et al., 2010a; Sharifi et al., 2010b; Sharifi et al., 2010c), who evaluated summaries obtained with the PR Algorithm. Specifically, we use ROUGE-L, which uses the longest common subsequence (LCS) to compare summaries. As the LCS of the two summaries in comparison increases in length, so does the similarity of the two summaries.

We now discuss results using ROUGE-L on the summaries we produce. Tables 2 through 5 show the results of four different ROUGE-L evaluations, comparing them to the results found using the PR

[3] `http://nlp.stanford.edu/software/corenlp.shtml`

Table 3: ROUGE-L without Stopwords, After

Task	Recall	Precision	F-score
Task 1	0.667	0.480	0.558
Task 2	0.400	0.500	0.444
Task 3	0.000	0.000	0.000
Task 4	0.400	0.333	0.363
Task 5	0.900	0.600	0.720
Task 6	0.389	0.350	0.368
Task 7	0.556	0.250	0.345
Task 8	0.545	0.500	0.522
Task 9	0.417	0.417	0.417
Task 10	0.363	0.200	0.258
Average	0.464	0.363	0.400

Algorithm, and Table 6 shows the comparisons of the averaged scores to the scores (Sharifi et al., 2010a) obtained using the PR Algorithm. Table 2 shows the regular ROUGE-L scores, meaning the recall, precision and F-scores for each task and the average overall scores, for the collection of posts before using the dependency parser to refine the summaries. Table 3 displays the results after using the dependency parser on the summaries formed by the PR Algorithm. One of the options in ROUGE is to show the "best" result, for each task. Table 4 has this result for the PR Algorithm results. Table 5 shows the results of the "best" scores, after running it through the dependency parser. Table 6 shows the averages from Tables 3 and 5, using the dependency parser, compared to Sharifi et al.'s results using the PR Algorithm. Stopwords were not removed in our experiments.

Table 4: ROUGE-L Best without Stopwords, Before

	Recall	Precision	F-score
Task 1	1.000	0.429	0.600
Task 2	1.000	0.227	0.370
Task 3	0.500	0.200	0.286
Task 4	1.000	0.154	0.267
Task 5	1.000	0.167	0.286
Task 6	1.000	0.200	0.333
Task 7	1.000	0.125	0.222
Task 8	1.000	0.071	0.133
Task 9	1.000	0.400	0.571
Task 10	1.000	0.100	0.182
Average	0.950	0.207	0.325

As one can see, the use of our algorithm on the summaries produced by the PR Algorithm improves the F-score values, at least in the example cases we tried. In almost every case, there is substantial rise in the F-score. As previously mentioned, some collections of Tweets do not produce good summaries. Task 3 had some poor scores in all cases, so one can deduce that the posts on that topic (Christmas) were widely spread, or they did not have a central theme.

Table 5: ROUGE-L Best without Stopwords, After

	Recall	Precision	F-score
Task 1	1.000	0.600	0.750
Task 2	0.400	0.500	0.444
Task 3	0.000	0.000	0.000
Task 4	0.500	0.333	0.400
Task 5	1.000	0.600	0.750
Task 6	0.600	0.600	0.600
Task 7	0.667	0.400	0.500
Task 8	1.000	0.333	0.500
Task 9	1.000	0.667	0.800
Task 10	1.000	0.250	0.400
Average	0.718	0.428	0.515

Table 6: ROUGE-L Averages after applying our algorithm vs. Sharifi et al.

	Recall	Precision	F-score
Sharifi (PRA)	0.31	0.34	0.33
Rouge-L after reconstruction	0.46	0.36	0.40
Rouge-L best after reconstruction	0.72	0.43	0.52

6 Conclusion

The PR Algorithm is not a pure extractive algorithm. It creates summaries of Twitter posts by piecing together the most commonly occurring words and phrases in the entire set of tweets, but keeping the order of constituents as close to the order in which they occur in the posts, collectively speaking. As we noted in this paper, the heuristic method using which the PR Algorithm composes a summary sentence out of the phrases sometimes leads to ungrammatical sentences or wordy sentences. This paper shows that the "raw" summaries produced by the PR Algorithm can be improved by taking into account governor-dependency relationships among the constituents. There is nothing in this clean-up algorithm that says that it works only with summaries of tweets. The same approach can potentially be used to improve grammaticality of sentences written by humans in a sloppy manner. In addition, given several sentences with overlapping content (from multiple sources), the same process can potentially be used to construct a grammatical sentence out of all the input sentences. This problem often arises in general multi-document summarization. We believe

that a corrective approach like ours can be used together with a sentence compression approach, such as (Knight and Marcu 2002), to produce even better summaries in conjunction with the PR or other summarization algorithms that work with socially-generated texts which are often malformed and short.

We have shown in this paper that simply focusing on grammatical dependency tends to make the final summaries more grammatical and readable compared to the raw summaries. However, we believe that more complex restructuring of the words and constituents would be necessary to improve the quality of the raw summaries, in general.

References

Knight, K. and Marcu, D. 2004. Summarization beyond sentence extraction: A probabilistic approach to sentence compression, Artificial Intelligence, Vol. 139, No. 1, pp. 91–107.

Lin, C.Y. 2004. Rouge: A package for automatic evaluation of summaries, Text Summarization Branches Out: Proceedings of the ACL-04 Workshop, pp. 74–81.

Sharifi, Beaux, Mark-Anthony Hutton, and Jugal Kalita. 2010. Summarizing Microblogs Automatically, Annual Conference of the National Association for Advancement of Computational Linguistics-Human Language Technology (NAACL-HLT), pp. 685-688, Los Angeles.

Sharifi, Beaux, Mark-Anthony Hutton, and Jugal Kalita. 2010. Experiments in Microblog Summarization, Second IEEE International Conference on Social Computing (SocialCom 2010), pp. 49-56, Minneapolis.

Sharifi, Beaux, Mark-Anthony Hutton and Jugal Kalita. 2010. Automatic Summarization of Twitter Topics, National Workshop on Design and Analysis of Algorithms, NWDAA 10, Tezpur University, Assam, India.