Efficient Online Summarization of Microblogging Streams

Andrei Olariu
Faculty of Mathematics and Computer Science
University of Bucharest
andrei@olariu.org

Abstract

The large amounts of data generated on microblogging services are making summarization challenging. Previous research has mostly focused on working in batches or with filtered streams. Input data has to be saved and analyzed several times, in order to detect underlying events and then summarize them. We improve the efficiency of this process by designing an online abstractive algorithm. Processing is done in a single pass, removing the need to save any input data and improving the running time. An online approach is also able to generate the summaries in real time, using the latest information. The algorithm we propose uses a word graph, along with optimization techniques such as decaying windows and pruning. It outperforms the baseline in terms of summary quality, as well as time and memory efficiency.

1 Introduction

Coined in 2006-2007, the term microblogging is used to describe social networks that allow users to exchange small elements of content. The widespread use of services like Facebook or Twitter means users have access to information that is otherwise unavailable. Yet, as popular events commonly generate hundreds of thousands of tweets, following them can be difficult. Stream summarization – generating a short text based on a sequence of posts – has been seen as the best approach in solving this problem.

This paper introduces Twitter Online Word Graph Summarizer. TOWGS is the first online abstractive summarization algorithm and is capable of state-of-the-art processing speeds. Most previous algorithms process a stream in batches. They require several passes through the data or a feed specifically filtered for an event. Batch summarization is suitable for small experiments, but it is not capable of efficiently handling thousands of tweets per second.

We collect a 3.4 million tweets dataset for evaluation purposes. We choose as baseline an algorithm designed to summarize related tweets. We determine a set of important events relative to the input data. A group of judges rate the summaries generated by both algorithms for the given events. Our solution is not only capable of online summarization, but it also outperforms the batch-based event-filtered baseline in terms of result quality. The code for our algorithm is available online, along with the summaries, event keywords, ratings and tweet IDs: `https://github.com/andreiolariu/online-summarizer`.

2 Related Work

2.1 Summarization

We distinguish two approaches in performing multi-document summarization: extractive and abstractive. With the risk of oversimplification, we view extractive summarization as a process of selecting sentences from the documents, while abstractive summarization generates phrases that may not appear in the input data.

The extractive approach is usually modeled as an optimization problem (Erkan and Radev, 2004). It can be combined with other techniques, such as clustering (Silveira and Branco, 2012) or topic modeling (Li and Li, 2013).

Although actually performing word-level extraction, we consider word graph summarization algorithms abstractive because they are able to generate summaries not found among the input sentences. Word graphs are used in compressing similar sentences (Filippova, 2010) or summarizing product reviews (Ganesan et al., 2010).

A relevant reference for the problem of up-

Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, pages 236–240,
Gothenburg, Sweden, April 26-30 2014.

date summarization is TAC update summarization track (Dang and Owczarzak, 2008).

2.2 Summarization on Twitter

Regarding summarizing Twitter streams, we notice that all approaches are either restricted to specific filtered streams, or combined with event detection.

Extractive summarization is predominant when working with Twitter data. It was first used for streams following simple and structured events, such as sports matches (Takamura et al., 2011; Chakrabarti and Punera, 2011; Nichols et al., 2012; Zubiaga et al., 2012).

The Phrase Reinforcement algorithm, introduced by Sharifi et al. (2010a; 2010b), extracts frequently used sequences of words. It was first applied in summarizing topic streams. Subsequent research emphasized evolving topics (Gao et al., 2013) or event decomposition (Olariu, 2013).

Other approaches are based on integer linear programming (Liu et al., 2011) or LDA (Khan et al., 2013). Yang et al. (2012) develop a framework for summarization, highlighting its scalability. Shou et al. (2013) introduce Sumblr, capable of cluster-based online extractive summarization.

Abstractive summarization is difficult on Twitter streams. It is easily affected by noise or by the large variety of tweets. Olariu (2013) showed that abstractive summarization is feasible if posts are clustered based on similarity or underlying events.

3 Twitter Online Word Graph Summarizer

3.1 Building the Word Graph

By employing a word graph, TOWGS doesn't have to save any of the tweets, like extractive approaches do. It can also skip the clustering step applied by the other online algorithm (Shou et al., 2013), leading to faster summarization.

Previous word graph algorithms are based on bigrams. Words are mapped to nodes in the graph, while an edge is added for each bigram. When applied to Twitter messages, the results depend on the similarity of the summarized tweets (Olariu, 2013). A set of related tweets generates a quality summary. When applied to unrelated tweets, the generated summary lacks any meaning. This happens because event-related signals (in our case bigrams) stand out when analyzing similar tweets, but get dominated by noise (bigrams of common words) when analyzing unrelated tweets.

We solve this issue by building the word graph from trigrams. In our version, each node in the graph is a bigram. Having a sentence (w_1, w_2, w_3, w_4), we will first add two special words (to mark the beginning and end of the sentence) and generate the following edges: $(S, w_1) \rightarrow (w_1, w_2)$, $(w_1, w_2) \rightarrow (w_2, w_3)$, $(w_2, w_3) \rightarrow (w_3, w_4)$ and $(w_3, w_4) \rightarrow (w_4, E)$. Weights are added to nodes and edges in order to store the count for each bigram or trigram.

A negative effect of building the word graph from trigrams is that it significantly increases the number of nodes, leading to an increase in both memory and time. We approach this issue by pruning the graph. We implement pruning by periodically going through the whole graph and removing edges that were not encountered in the previous time window. The length of this hard window can be set based on how much memory we would like to allocate, as well as on the size of the soft window introduced in the next subsection.

3.2 Word Graph Online Updating

In previous work, word graphs are discarded after generating the summary. For our online summarization task, the graph is being constantly updated with tweets. It can also respond, at any time, to queries for generating summaries starting from given keywords.

In order to keep the results relevant to what is popular at query time, we would like the graph to *forget* old data. We implement this behavior by using decaying windows (Rajaraman and Ullman, 2011). They are applied not only to graph weights (counts of bigrams and trigrams), but also to counts of words and word pair cooccurrences.

At each time step (in our case, each second), all counts are multiplied by $1 - c$, where c is a small constant. For example, after one hour (3600 seconds), a value of 1 would become 0.48 with $c = 0.0002$ (given by $(1 - c)^{3600}$) and 0.05 with $c = 0.0008$.

In order to optimize the implementation, we explicitly multiply the counts only when they are read or incremented. For each record, we keep the timestamp for its latest update t_k. Knowing the current timestamp t_n, we update the count by multiplying with $(1 - c)^{t_n - t_k}$.

The size of the decaying window influences the

results and the memory requirements for TOWGS. A larger window requires less pruning and more memory, while also leading to more general summaries. For example, given a stream of tweets related to a sporting event, summaries generated over very narrow windows would probably highlight individual goals, touchdowns or penalties. The summary for a two hour window would instead capture just the final score.

3.3 Generating Summaries

Given a word graph, generating a summary involves finding the highest scoring path in the graph. That path connects the special words which mark the beginning and end of each sentence. Since finding the exact solution is unfeasible given our real time querying scenario, we will employ a greedy search strategy.

The search starts by selecting the node (bigram) with the highest weight. If we are interested in summarizing an event, we select the top ranking bigram containing one of the event's keywords.

At this point, we have a path with one node. We expand it by examining forward and backward edges and selecting the one that maximizes the scoring function:

$$
\begin{aligned}
&score(n, e, m, p, k) = \\
&\quad c_1\, frequency(n) && \text{(1a)}\\
&\quad + c_2\, edge\_score(e, m) && \text{(1b)}\\
&\quad + c_3\, word\_score(n, p) && \text{(1c)}\\
&\quad + c_4\, word\_score(n, k) && \text{(1d)}\\
&\quad - c_5\, frequent\_word\_pen(n) && \text{(1e)}\\
&\quad - c_6\, repeated\_word\_pen(n) && \text{(1f)}
\end{aligned}
$$

where p is a path representing a partial summary, n is a node adjacent to one of the path's endpoints m by edge e and k is a list of keywords related to an event. The constants c_1 through c_6 determine the influence each helper function has on the overall score. The node n represents a bigram composed of the words w_i (already in the path as part of m) and w_o (currently being considered for extending p). The helper functions are defined as:

$$frequency(n) = \log(W_b[n]) \quad \text{(2a)}$$

$$edge\_score(e, m) = \log\left(\frac{W_t[e]}{W_b[m]}\right) \quad \text{(2b)}$$

$$word\_score(n, p) = \sum_{w \in p} \frac{1}{|p|} \log\left(\frac{W_d[w, w_o]}{\sqrt{W_w[w] W_w[w_o]}}\right) \quad \text{(2c)}$$

$$frequent\_word\_pen(n) = \log(W_w[w_o]) \quad \text{(2d)}$$

$$repeated\_word\_pen(n) = \mathbf{1}_p(w_o) \quad \text{(2e)}$$

where $W_w[w]$ is the weight for word w, $W_b[m]$ is the weight for the bigram represented by node m, $W_t[e]$ is the weight for the trigram represented by edge e and $W_d[w, w_o]$ is the weight for the cooccurrences of words w and w_o in the same tweets. $\mathbf{1}_p(w_o)$ is the indicator function. In all these cases, weights are counts implemented using decaying windows (subsection 3.2).

The scoring function gives a higher score to frequent bigrams (equations 1a and 2a). In the same time, individual words are penalized on their frequency (equations 1e and 2d). Such scores favor words used in specific contexts as opposed to general ones. Trigrams are scored relative to bigrams (equations 1b and 2b). Again, this favors context specific bigrams. The word score function (equation 2c) computes the average correlation between a word (w_o from the bigram represented by node n) and a set of words. The set of words is either the current partial summary (equation 1c) or the event-related keywords (equation 1d).

We use logarithms in order to avoid floating point precision errors.

4 Evaluation

4.1 Corpus and Baseline

Our corpus is built using the Twitter Search API. We gathered an average of 485000 tweets per day for a total of seven days, between the 4th and the 10th of November 2013. This volume of tweets represents around 0.1% of the entire Twitter stream. Because of Twitter's terms of service, sharing tweets directly is not allowed. Instead, the source code we've released comes with the tweet IDs needed for rebuilding the corpus.

The algorithm chosen as baseline is Multi-Sentence Compression (or MSC), as presented in (Olariu, 2013). MSC is a batch algorithm for abstractive summarization. It performs best on groups of similar tweets, such as the ones related to an event. After receiving a summarization query for a set of keywords, the tweets are filtered based on those keywords. MSC processes the remaining tweets and generates a word graph. After building the summary, the graph is discarded.

Because it has to store all tweets, MSC is not as memory-efficient as TOWGS. It is also not time-efficient. Each summarization query requires fil-

tering the whole stream and building a new word graph. The advantage MSC has is that it is working with filtered data. Olariu (2013) has shown how susceptible word graphs are to noise.

4.2 Evaluation Procedure

The list of 64 events to be summarized was determined using a frequency based approach. A simple procedure identified words that were used significantly more in a given day compared to a baseline. The baselines were computed on a set of tweets posted between the 1st and the 3rd of November 2013. Words that often appeared together were grouped, with each group representing a different event.

The MSC algorithm received a cluster of posts for each event and generated summaries of one sentence each. TOWGS processed the posts as a stream and answered to summarization requests. The requests were sent after the peak of each event (at the end of the hour during which that event registered the largest volume of posts).

The metrics used for assessing summary quality were completeness (how much information is expressed in the summary, relative to the event tweets) and grammaticality. They were rated on a scale of 1 (lowest) to 5 (highest).

We asked five judges to rate the summaries using a custom built web interface. The judges were not native English speakers, but they were all proficient. Three of them were Twitter users. While the judges were subjective in assessing summary quality, each one did rate all of the summaries and the differences between the two algorithms' ratings were consistent across all judges.

The constants c_1 through c_6 (introduced in subsection 3.3) were set to 2, 3, 3, 10, 1 and 100, respectively. These values were manually determined after experimenting with a one day sample not included in the evaluation corpus.

5 Results

The average ratings for completeness are very similar, with a small advantage for TOWGS (4.29 versus MSC's 4.16). We believe this is a good result, considering TOWGS doesn't perform clustering and summarizes events that account for less than 1% of the total volume. Meanwhile, MSC processes only the event-related tweets. The average rating for grammaticality is significantly higher for TOWGS (4.30), as compared to MSC (3.78).

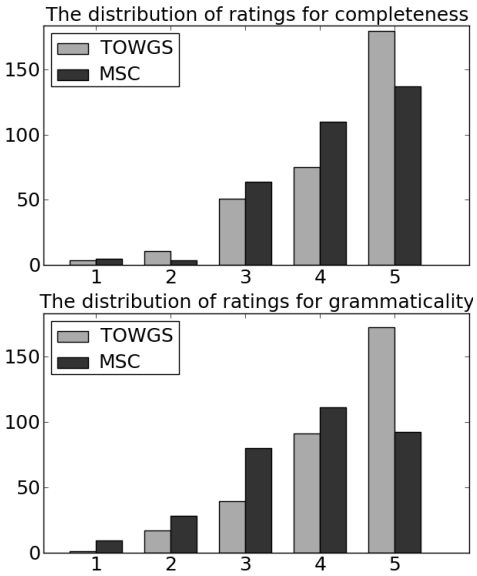

Figure 1: The ratings distribution by algorithm and metric.

While not engineered for speed, our implementation can process a day of data from our corpus (around 485000 tweets) in just under three minutes (using one 3.2 GHz core). In comparison, Sumblr (Shou et al., 2013) can process around 30000 tweets during the same interval. TOWGS requires an average of 0.5 seconds for answering each summarization query. Regarding memory use, pruning kept its value constant. In our experiments, the amount of RAM used by the algorithm was between 1.5 - 2 GB.

The code for TOWGS is available online, along with the summaries, keywords, ratings and tweet IDs: `https://github.com/andreiolariu/online-summarizer`.

6 Conclusion

Summarizing tweets has been a popular research topic in the past three years. Yet developing efficient algorithms has proven a challenge, with most work focused on small filtered streams.

This paper introduces TOWGS, a highly efficient algorithm capable of online abstractive microblog summarization. TOWGS was tested on a seven day 0.1% sample of the entire Twitter stream. We asked five judges to rate the summaries it generated, along with those from a baseline algorithm (MSC). After aggregating the results, the summaries generated by TOWGS proved to have a higher quality, despite the fact that MSC processed just the batches of event-filtered tweets. We also highlighted the state-of-the-art time efficiency of our approach.

References

Deepayan Chakrabarti and Kunal Punera. 2011. Event summarization using tweets. In *Proceedings of the 5th Int'l AAAI Conference on Weblogs and Social Media (ICWSM)*.

Hoa Trang Dang and Karolina Owczarzak. 2008. Overview of the tac 2008 update summarization task. In *Proceedings of text analysis conference*, pages 1–16.

Günes Erkan and Dragomir R. Radev. 2004. Lexrank: graph-based lexical centrality as salience in text summarization. *J. Artif. Int. Res.*, 22(1):457–479, December.

Katja Filippova. 2010. Multi-sentence compression: finding shortest paths in word graphs. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 322–330, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. 2010. Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 340–348, Stroudsburg, PA, USA. Association for Computational Linguistics.

Dehong Gao, Wenjie Li, and Renxian Zhang. 2013. Sequential summarization: A new application for timely updated twitter trending topics. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, ACL '13, pages 567–571. Association for Computational Linguistics.

Muhammad Asif Hossain Khan, Danushka Bollegala, Guangwen Liu, and Kaoru Sezaki. 2013. Multi-tweet summarization of real-time events. In *SocialCom*, pages 128–133. IEEE.

Jiwei Li and Sujian Li. 2013. Evolutionary hierarchical dirichlet process for timeline summarization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, ACL '13, pages 556–560. Association for Computational Linguistics.

Fei Liu, Yang Liu, and Fuliang Weng. 2011. Why is "sxsw" trending?: exploring multiple text sources for twitter topic summarization. In *Proceedings of the Workshop on Languages in Social Media*, LSM '11, pages 66–75, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jeffrey Nichols, Jalal Mahmud, and Clemens Drews. 2012. Summarizing sporting events using twitter. In *Proceedings of the 2012 ACM international conference on Intelligent User Interfaces*, IUI '12, pages 189–198, New York, NY, USA. ACM.

Andrei Olariu. 2013. Hierarchical clustering in improving microblog stream summarization. In *Proceedings of the 14th international conference on Computational Linguistics and Intelligent Text Processing - Volume 2*, CICLing'13, pages 424–435, Berlin, Heidelberg. Springer-Verlag.

Anand Rajaraman and Jeffrey David Ullman. 2011. *Mining of Massive Datasets*. Cambridge University Press, New York, NY, USA.

Beaux Sharifi, Mark-Anthony Hutton, and Jugal Kalita. 2010a. Summarizing microblogs automatically. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 685–688, Stroudsburg, PA, USA. Association for Computational Linguistics.

Beaux Sharifi, Mark-Anthony Hutton, and Jugal K. Kalita. 2010b. Experiments in microblog summarization. In *Proceedings of the 2010 IEEE Second International Conference on Social Computing*, SOCIALCOM '10, pages 49–56, Washington, DC, USA. IEEE Computer Society.

Lidan Shou, Zhenhua Wang, Ke Chen, and Gang Chen. 2013. Sumblr: Continuous summarization of evolving tweet streams. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '13, pages 533–542, New York, NY, USA. ACM.

S.B. Silveira and A. Branco. 2012. Combining a double clustering approach with sentence simplification to produce highly informative multi-document summaries. In *Information Reuse and Integration (IRI), 2012 IEEE 13th International Conference on*, pages 482–489.

Hiroya Takamura, Hikaru Yokono, and Manabu Okumura. 2011. Summarizing a document stream. In *Proceedings of the 33rd European conference on Advances in information retrieval*, ECIR'11, pages 177–188, Berlin, Heidelberg. Springer-Verlag.

Xintian Yang, Amol Ghoting, Yiye Ruan, and Srinivasan Parthasarathy. 2012. A framework for summarizing and analyzing twitter feeds. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '12, pages 370–378, New York, NY, USA. ACM.

Arkaitz Zubiaga, Damiano Spina, Enrique Amigó, and Julio Gonzalo. 2012. Towards real-time summarization of scheduled events from twitter streams. In *Proceedings of the 23rd ACM Conference on Hypertext and Social Media*, HT '12, pages 319–320, New York, NY, USA. ACM.