Realtime Event Summarization from Tweets with Inconsistency Detection

Lingting Lin, Chen Lin

Department of Computer Science, Xiamen University, China chenlin@xmu.edu.cn

Abstract. The overwhelming amount of event relevant tweets highlights the importance of realtime event summarization systems. When new information emerges, former summaries should be updated accordingly to deliver most recent and authoritative information. Existing studies couldn't preserve the integrity of a realtime summary. For example, for an ongoing earthquake event, existing studies might generate a summary including new and old estimates of number of injuries, which are inconsistent. In this contribution we present a realtime event summarization system with explicit inconsistency detection. We model the realtime summarization problem as multiple integer programming problems and solve the relaxed linear programming form by an improved simplex update method. To reduce the storage and computational cost of expensive inconsistency detection, we embed a novel fast inconsistency detection strategy in the simplex update algorithm. We conduct comprehensive experiments on real twitter sets. Compared with state-of-the-art methods, our framework produces summaries with higher ROUGE scores and lower inconsistency rates. Furthermore our framework is more efficient and more scalable.

1 Introduction

Emergence of microblogging platforms, such as Twitter, has resulted in a large community of microblogging users and a massive amount of instant reports about live events all over the world. Event summarization system is needed to facilitate knowledge management and improve user experiences. We have witnessed rapidly increasing popularity of research efforts in event summarization from tweets [15, 6, 12, 14, 7, 3, 17, 13]. Most previous studies are based on extractive methods, i.e. they extract a smallest set of representative tweets to form a brief summary. Extractive methods are easy to implement and have shown to perform well [15, 6, 12, 14, 7, 3, 17]. Our arguments in this paper are founded on extractive summarization methods.

Beyond the usual requirements for text summarization systems, such as coverage and representativeness of the summary, event summary must also be **realtime**. On one hand, the response must be fast. The summary must be efficiently updated as new tweets arrive. On the other hand, the summary must report the current status of the event. As an ongoing event often involves changing information, the old summary must be updated to include new information when it emerges.

During the update process, the integrity of the summary must be preserved. An outdated tweet report must be replaced if it leads to **inconsistency** in the summary. An inconsistent summary is harmful for most live events because it is confusing and misleading. Examples include (1) in an earthquake the number of injuries is increasing over time. Thus the numbers in previous summaries become obsolete and they should be replaced by the most up-to-date numbers. (2) The number of injuries is estimated by several parties, such as bystanders, hospitals and so on. When a more authoritative source, such as the local government announces the new estimate, the old estimates in previous summaries are no longer credible and must be replaced.

Realtime event summarization from tweets is still an open problem. In the literature, most of previous works treat the problem as producing different forms of summaries from a static set of tweets [15, 6, 12, 7, 3]. A few recent research works focused on efficient algorithms to summarize the tweet streams [14, 17]. Their summarization systems are based on coarse grained semantic analysis, and thus are not able to detect inconsistency. Though we have shown that integrity of the event summary is crucial, to the best of our knowledge, none of the previous works is able to produce a realtime event summarization which is guaranteed to exclude inconsistent information.

Two challenges arise in producing realtime event summarization without inconsistent information.

The first challenge lies in the macro-level algorithm. Realtime summarization requires an efficient algorithm to analyze the streaming tweets. As the amount of available tweets constantly increases to infinity, re-computation based on a complete set of all tweets up to the current timestamp is infeasible. The ideal algorithm is to incorporate new tweets as they become available, and discard old tweets when possible to limit the storage and speed up processing.

The second challenge is related to the micro-level analysis to detect inconsistency. Inconsistency detection is based on pair-wise similarity. Coarse grained semantic analysis, such as the cosine similarity measure based on the bag of words representation in previous works [15, 6, 12, 14, 7, 3, 17, 13] is suitable to capture topic similarity in a summarization. However, it is not able to detect inconsistent information. We need to design new similarity metrics for this purpose. Furthermore, inconsistency detection is computationally expensive. It is important to avoid unnecessary pair-wise comparisons.

Our goal in this paper is to design a system that delivers realtime summary with integrity from tweets. To address the first challenge, we assume that, the realtime summarization problem given a small batch of new tweets can be modeled as two integer programming problems, one of which on the old tweets, and another on the new batch. Both integer programming problems can be relaxed to linear programming problems and be solved by the simplex method. In each update we first optimize the problem on the new batch. We use the solution on the new batch to modify the problem on the old tweets and incrementally update the summary. In this manner, we do not need to store or operate on the complete tweet set and the full similarity matrix.

To address the second challenge, we propose skeleton similarity: a new similarity metric to assess information similarity between any pair of tweets. An inconsistency detection strategy, which is a combination of the skeleton similarity and authority estimation heuristics, is then adopted in the pivoting operation in the simplex method. It has two advantages in embedding the skeleton similarity computation in the simplex algorithm. (1) It significantly reduces the number of information similarity comparisons. (2) It ensures that the former summary will be replaced by most up-to-date and authoritative information.

Our contributions are three folds. (1) The integrity of event summary is a relatively unexplored area in Tweet summarization. We propose to improve the integrity of event summarization by explicit inconsistency detection. (2) Our system is targeted towards text streams. We model the realtime summarization problem as integer programming problems in small batches. We differ from existing work in that we enable incremental updates in the simplex framework. (3) We propose a novel skeleton similarity to efficiently and effectively capture inconsistency.

This paper is organized as follows. We briefly survey the related work in Sec. 2. In Sec. 3, we first introduce the idea of modeling a summarization problem as an integer programming problem and the standard simplex procedure to solve the relaxed linear programming problem. In Sec. 4, we give the problem definition for realtime event summarization given a small batch of new tweets and the improved simplex solution. The inconsistency detection strategy is a component of the simplex update algorithm. We present and analyze the experimental results on a real data set in Sec. 5. We conclude our work and suggest future directions in Sec. 6.

2 Related Work

Multi-document summarization conveys the main and most important meaning of several documents. There are generally two types of summarization techniques. One type is extraction-based summarization, which extracts objects from the entire collection and combines the objects into a summary without modifying the objects themselves. The other type is abstraction-based summarization which rephrases the source document. The majority of summarization systems are extractive. The extracted objects are often sentences. The selection is usually based on the representativeness of sentences, i.e. with significant frequency [16], or is a structural centroid in a sentence graph [6], or is considered important by a submodularity function [22].

The emergence of Twitter motivates recent research works on summarizing microblogging contents. Tweet summarization systems are successfully applied in entity-centric opinion summarization [9], personal summarization of interesting content [10, 1], search results grouping [8], and summarizing tweets for natural or social events [15, 6, 12, 14, 7, 3, 17].

At the algorithm level, tweet summarization also uses extractive and abstractive methods. Except a few works which are based on abstractive methods [13],

most tweet summarization methods are extractive, i.e. they rank and select the most representative tweets. A few recent works start to improve general multi-document summarization methods for better efficiency. In [14], an incremental clustering method is presented. In [17] the selection range is shrinked by detecting sub-events and selecting one sentence with maximal similarity to any new sub-event.

To achieve a better performance, the noisy and social nature of microblogs must be taken into consideration. Most tweet summarization systems identify influential tweets [4], promote most recent tweet [2], and circumnavigate spam and conversational posts [3]. However, the integrity of summary has not yet been fully studied. The work that is most related to ours is the classification and summarization of situational information in [12, 11]. However they do not explicitly identify inconsistency as they simply provide all versions of inconsistent information. And they do not accelerate algorithms for realtime responses.

3 Static Summarization

The standard summarization task is conducted on a static set of tweets. We refer this task as a static summarization. In this section, we first model the static summarization problem as an integer programming problem. We then present a high level explanation about the simplex method. We finally give an outline for the algorithm to solve static summarization.

3.1 Problem Definition

Suppose that we have a universe of N *candidate* tweets, within which M tweets are credible and relevant. The credible and relevant tweets are important so they are considered to be the *seeds* of the summary. The extractive method for any static summarization is to select a few representative tweet *reports*[1] from the tweet universe to form the summary. To model this problem, we use a vector $\mathbf{x} \in \mathcal{R}^N$, where each element $\mathbf{x}_j \in \{0, 1\}$ is a binary variable. If a candidate i is chosen to be a report in the summary, the corresponding $x_i = 1$. Otherwise, we set $x_i = 0$. We use another N-dimensional vector $\mathbf{c} \in \mathcal{R}^N$ to describe the loss of choosing each candidate as a report. $\mathbf{A} \in \mathcal{R}^{M \times N}$ is a similarity matrix, where $A_{i,j}$ is the similarity between a seed i and a candidate j in the tweet universe. $\mathbf{b} \in \mathcal{R}^M$ is a weight vector, where $b_i > 0$ indicates the importance of seed i being covered in the summary. Our objective is to

$$\min \mathbf{c}^T \mathbf{x} \text{ subject to } \mathbf{A}\mathbf{x} \geq \mathbf{b}, \forall i \; x_i \in \{0, 1\}. \tag{1}$$

The optimization problem in Equ. 1 aims to find a minimal set of reports that delivers all credible and relevant information. Thus the result summary achieves brevity, coverage and representativeness. The decision of b, c and the M indices

[1] To distinguish the three types of tweets, we will refer the tweets to be summarized as candidates, the tweets in the summary as reports, and the credible and relevant tweets as seeds.

affects the performance of summarization. We allow the flexibility of choosing any sounding technique to determine the credible and relevant set of tweets, their importance values and the loss of choosing any candidate. For example, in the experiment, we choose the top M results from a search engine, determine $\mathbf{b}$ by estimating the authority of each tweet account, and assign $\mathbf{c}$ by assessing the textual quality of each candidate. The technical details are discussed in Sec. 5.

3.2 The Simplex Method

We transform the integer programming problem in Equ. 1 to a bounded linear programming problem by making the following adjustments: $\tilde{\mathbf{c}} = [\mathbf{c}, \mathbf{0}], \tilde{\mathbf{x}} = [\mathbf{x}^T, \mathbf{z}]^T, \tilde{\mathbf{A}} = [\mathbf{A}, -\mathbf{I}]$, where $\mathbf{z} \in \mathcal{R}^M$, $\mathbf{I}$ is the $M \times M$ identity matrix. Therefore we have the following objective

$$\min \tilde{\mathbf{c}}^T \tilde{\mathbf{x}} \text{ subject to } \tilde{\mathbf{A}}\tilde{\mathbf{x}} = \mathbf{b}, \forall 1 \leq j \leq N,\ \tilde{x}_j \leq 1, , \tilde{\mathbf{x}} \geq \mathbf{0}. \tag{2}$$

The linear programming problem in Equ. 2 can be solved by the simplex method. Each iterate in the simplex method is a *basic feasible point* that (1) it satisfies $\tilde{\mathbf{A}}\tilde{\mathbf{x}} = \mathbf{b}, \forall 1 \leq i \leq M+N, \tilde{x}_i \geq 0, \forall 1 \leq j \leq N, \tilde{x}_j \leq 1$ and (2) there exists three subsets $\mathcal{B}, \mathcal{U}, \mathcal{L}$ of the index set $\mathcal{A} = \{1, 2, \cdots M+N\}$ such that $\mathcal{B}$ contains exactly M indices, $\mathcal{A} = \mathcal{B} \cup \mathcal{U} \cup \mathcal{L}$ and $i \in \mathcal{U} \Rightarrow \tilde{x}_i = 1, i \in \mathcal{L} \Rightarrow \tilde{x}_i = 0$.

The major issue at each simplex iteration is to decide which index to be removed from the basis $\mathcal{B}$ and replaced by another index outside the basis $\mathcal{B}$. As the optimal is achieved when the KKT conditions are satisfied, in each simplex iteration first a **pricing** step is conducted to check on the KKT conditions. If the KKT conditions are satisfied then the problem is solved. Otherwise a **pivoting** operation is implemented to select entering and leaving indices.

To obtain an easy start of basic feasible points, we solve the following linear programming problem.

$$\min \mathbf{e^T s} \text{ subject to } \tilde{\mathbf{A}}\tilde{\mathbf{x}} + \mathbf{I}\mathbf{s} = \mathbf{b}, \forall 1 \leq j \leq N,\ \tilde{x}_j \leq 1, , \tilde{\mathbf{x}} \geq \mathbf{0}, \mathbf{s} \geq \mathbf{0}, \tag{3}$$

where $\mathbf{e}$ is the vector of all ones, $\mathbf{I}$ is a diagonal matrix whose diagonal elements are $I_{ij} = 1$. $\mathbf{s}$ are called artificial variables. It is easy to see that the solution to Equ. 3 is a basic feasible point for Equ. 2, if the objective $\mathbf{e^T s} = 0$.

The remaining problem is that the solution we obtained for Equ. 2 are not integers. As our interest is on the variables $\mathbf{x}$ in Equ. 1, we can interpretate the values obtained for Equ. 2 $\tilde{x}_j, \forall 1 \leq j \leq N$ as the probability of choosing j as a response. For indices that are in the upper bound set $j \in \mathcal{U}$, the corresponding candidates are deemed in the summary. For indices that are in the lower bound set $j \in \mathcal{L}$, the corresponding candidates are filtered. For the remaining candidates, we perform a **rounding** algorithm. We first sort the solutions we obtained for Equ. 2 in descending order of their values $0 < \tilde{x}_j < 1, \forall 1 \leq j \leq N$ and then sample x_j based on the probability of assigned value $\tilde{x}_j$.

$$p(x_j = 1) = \tilde{x}_j, \forall 1 \leq j \leq N \tag{4}$$

To conclude this section, we present Algorithm. 1. The framework of static summarization includes three stages. The first stage (step 1 to step 2) is to obtain a basic feasible point for the linear programming problem Equ. 2, the second stage (step 3 to step 4) is to optimize the linear programming problem in Equ. 2, and the third stage (step 5) is to obtain the integer solutions for Equ. 1.

Input: $\mathbf{c}, \mathbf{b}, A$
Output: $\mathbf{x}$
1 Initialize $\tilde{\mathbf{x}} = \mathbf{0}, \tilde{\mathbf{A}} = [\mathbf{A}, -\mathbf{I}], \mathbf{s} = \mathbf{b}$;
2 Solve $\min \mathbf{e^T s}$ subject to $\tilde{\mathbf{A}}\tilde{\mathbf{x}} + \mathbf{Is} = \mathbf{b}, \forall 1 \leq j \leq N,\ \tilde{x}_j \leq 1, , \tilde{\mathbf{x}} \geq \mathbf{0}, \mathbf{s} \geq \mathbf{0}$ by simplex method;
3 Initialize $\tilde{\mathbf{x}}$ by the solution above, set $\tilde{\mathbf{c}} = [\mathbf{c}, \mathbf{0}]$;
4 Solve $\min \tilde{\mathbf{c}}^T \tilde{\mathbf{x}}$ subject to $\tilde{\mathbf{A}}\tilde{\mathbf{x}} = \mathbf{b}, \forall 1 \leq j \leq N,\ \tilde{x}_j \leq 1, , \tilde{\mathbf{x}} \geq \mathbf{0}$ by simplex method;
5 $\tilde{\mathbf{x}} = [\mathbf{x}, \mathbf{z}]^T$, Rounding $\mathbf{x}$;

Algorithm 1: The framework for static summarization

4 Dynamic Summarization

For streaming texts, the realtime summarization system must be able to update the summary as new contents arrive. The summary is dynamic because it changes over time. We here consider a batch setting: the summary is updated when a predefined number T of tweets are received. Note that if we set $T = 1$ we enable fully online update. In this section, we first model the dynamic summarization problem and then we present the improved simplex method.

4.1 Problem Definition

When a small batch of tweets arrive, the candidate universe will be a new set of N_1 candidates with importance weights denoted by $\mathbf{c_1}$ and the former candidates. We keep previous reports $\mathbf{x_0}$ with $x_i = 1$ in the former summary and their loss value $\mathbf{c_0}$ and discard everything else from former corpus to reduce storage space.

To produce a summary that covers the demanded information, the model in Sec. 3 requires a predefined set of credible and relevant seed tweets and their importance weights $\mathbf{b}$. It is crucial here to mention that in the dynamic setting the seed set is priorly unknown due to inconsistency. For example, if a new credible tweet is found to conflict with a former credible tweet, then the former one must be excluded. Suppose we have M seeds, but the integrity is not guaranteed. It is possible to check on inconsistency to prune the M seeds to identify a consistent new set. However, the inconsistency detection algorithm on $M \times (M-1)$ pairs is expensive. Henceforth, our idea is to call the inconsistency detection algorithm only when it is necessary.

Suppose we segment the timeline of tweets into two divisions. M_0 represents the set of seeds that are covered by the former summary, M_1 is the set of seeds published after the former summary was generated. M_1 is generated as follows. We first retrieve R relevant tweets published after the summary. Then we can adopt any heuristic to fast prune inconsistent seeds in M_1. We can also adopt any external strategy to exclude seeds that are less credible than any seed in M_0. Therefore the seeds in M_1 must be covered with priority if they are inconsistent with the seeds in M_0, because any seed in M_1 is newer and at least as credible

as any seed in M_0. To deliver an up-to-date summary, we set $\mathbf{c}$ so that the loss for new candidates is smaller than the old candidates.

Below we give the architecture of the dynamic summarization system. We model the dynamic summarization as sequential integer programming problems. Given the whole set of candidates $\mathbf{x_0}, \mathbf{x_1}$ and the associated loss $\mathbf{c_0}, \mathbf{c_1}$, the importance of possible seeds $\mathbf{b_1}$ in M_1, at stage I, we first solve the optimization problem for the higher division of seeds.

$$\min {\mathbf{c_0}}^T \mathbf{x_0} + {\mathbf{c_1}}^T \mathbf{x_1} \text{ subject to } S\mathbf{x_0} + B\mathbf{x_1} \geq \mathbf{b_1}, \forall i, \forall j \in \{0,1\}\ x_{j,i} \in \{0,1\}. \quad (5)$$

In solving Equ. 5 we will obtain the refined seed set $\mathbf{b_0}$. In stage II we solve the optimization problem:

$$\begin{aligned} &\min {\mathbf{c_0}}^T \mathbf{x_0} + {\mathbf{c_1}}^T \mathbf{x_1} \\ &s.t. \begin{bmatrix} F & D \\ S & B \end{bmatrix} \begin{bmatrix} \mathbf{x_0} \\ \mathbf{x_1} \end{bmatrix} \geq \begin{bmatrix} \mathbf{b_0} \\ \mathbf{b_1} \end{bmatrix}, \\ &\forall i, \forall j \in \{0,1\}\ x_{j,i} \in \{0,1\}, \end{aligned} \quad (6)$$

where $F \in \mathcal{R}^{M_0 \times N_0}$ is the similarity matrix between former reports and old seeds, $D \in \mathcal{R}^{M_0 \times N_1}$ is the similarity matrix between new candidates and old seeds, $S \in \mathcal{R}^{M_1 \times N_0}$ is the similarity matrix between old reports and new seeds, $B \in \mathcal{R}^{M_1 \times N_1}$ is the similarity matrix between new candidates and new seeds.

4.2 Simplex Update Method

As in Sec. 3, we introduce new variables. $\tilde{\mathbf{c}} = [\mathbf{c_0}, \mathbf{c_1}, \mathbf{0}], \tilde{\mathbf{x}} = [\mathbf{x_0}, \mathbf{x_1}, \mathbf{z}]^T, \tilde{\mathbf{A}} = [S, B, -\mathbf{I}], \mathbf{s}$, where $\mathbf{z} \in \mathcal{R}^{M_1}$, $\mathbf{I}$ is the $M_1 \times M_1$ identity matrix. To initialize, we set $\mathbf{x_0} = \mathbf{1}$ and keep $\mathbf{x_0}$ fixed while solving Equ. 5. We use the solution as the initialized guess for the second stage optimization.

In each iterate of simplex, we first operate the pricing step. Let's compute $\bar{c}_j = \tilde{c}_j - y^T \tilde{\mathbf{A}}_{,j}$ for every $\tilde{x}_j$ not in the basis, where $\mathbf{y} = {\mathcal{B}^T}^{-1} \mathbf{c}_{\mathcal{B}}$. If $\tilde{x}_j = 0, \bar{c}_j > 0$ and $\tilde{x}_j = 1, \bar{c}_j < 0$, then the objective function is optimized. Otherwise we implement the pivoting operation.

We modify the conventional pivoting operation in the simplex method. In the initialization we keep $\mathbf{x_0} = \mathbf{1}$, which indicates that the constraint is satisfied with all the former reports kept. If the pricing step suggests the objective function is not optimized, there are usually multiple indices which violate the KKT conditions. Hence in selecting the entering index, we prefer a z index, because selecting an auxiliary variable leads to minimal modification of the summary. However, if an index from $\mathbf{x_0}$ is to leave and an index from $\mathbf{x_1}$ is to enter, a new tweet is able to cover the same set of seeds with smaller cost (given that we assign higher loss c_0 to old tweets). We can not rule out the possibility of inconsistency. In fact, inconsistency only happens when two tweets are semantically closely related. Therefore we conduct inconsistency test between the entering index and the leaving index. If the leaving index is found to be inconsistent of the new report, we delete the corresponding component $x_{0,i}$ from the problem. This strategy ensures that inconsistent candidates do not repeatedly enter and leave the summary. We remove all seeds that are related to $x_{0,i}$ in $\mathbf{b_0}$. The pivoting operation is described in Algorithm. 2.

$q = \arg\max_j \{10 \times \bar{c}_j \ \forall j \in \mathbf{z}_{\mathcal{N}}, \bar{c}_j \ \forall j \in \mathbf{x}_{\mathcal{L}}, -\bar{c}_j \ \forall j \in \mathbf{x}_{\mathcal{U}}\}$;
$d = \mathcal{B}^{-1}\tilde{\mathbf{A}}_{,q}$;
if $q \in \mathbf{z}_{\mathcal{N}}$ **then**
  $x_q^{new} = \min_i \left\{ \{\frac{x_{\mathcal{B}_i}}{d_i}, \frac{s_{\mathcal{B}_i}}{d_i}, \frac{z_{\mathcal{B}_i}}{d_i}\} \forall d_i > 0, \{\frac{x_{\mathcal{B}_i}-1}{d_i}\} \forall d_i < 0 \right\}$;
  $\mathbf{x}_{\mathcal{B}^{old}}^{new} = \mathbf{x}_{\mathcal{B}^{old}}^{old} - dx_q^{new}$;
  $p = \arg\min_i$;
end
if $q \in \mathbf{x}_{\mathcal{L}}$ **then**
  $x_q^{new} = \min_i \left\{ \{\frac{x_{\mathcal{B}_i}}{d_i}, \frac{s_{\mathcal{B}_i}}{d_i}, \frac{z_{\mathcal{B}_i}}{d_i}\} \forall d_i > 0, \{\frac{x_{\mathcal{B}_i}-1}{d_i}\} \forall d_i < 0, 1 \right\}$;
  $\mathbf{x}_{\mathcal{B}^{old}}^{new} = \mathbf{x}_{\mathcal{B}^{old}}^{old} - dx_q^{new}$;
  if $x_q^{new} \neq 1$ **then**
    $p = \arg\min_i$;
  end
end
if $q \in \mathbf{x}_{\mathcal{U}}$ **then**
  $x_q^{new} = 1 - \min_i \left\{ \{\frac{x_{\mathcal{B}_i}}{-d_i}, \frac{s_{\mathcal{B}_i}}{-d_i}, \frac{z_{\mathcal{B}_i}}{-d_i}\} \forall d_i < 0, \{\frac{1-x_{\mathcal{B}_i}}{d_i}\} \forall d_i > 0, 1 \right\}$;
  $\mathbf{x}_{\mathcal{B}^{old}}^{new} = \mathbf{x}_{\mathcal{B}^{old}}^{old} + d(1 - x_q^{new})$;
  if $x_q^{new} \neq 0$ **then**
    $p = \arg\min_i$;
  end
end
if $1 \leq p \leq N_0, N_0 + 1 \leq q \leq N_0 + N_1$ **then**
  Inconsistency detection on (p, q);
  if p, q *are inconsistent* **then**
    Refine $\mathbf{b_0}$;
    Remove p from index set $\mathcal{A}$;
  end
end
$\mathcal{B}^{new} \leftarrow \mathcal{B}^{old} - \{p\} \bigcup \{q\}$;

Algorithm 2: Pivoting in the simplex update method

4.3 Inconsistency Detection

To effectively deal with tweets which contain abbreviations, numerals, foreign words and symbols, we perform case-folding, lemmatization, stop-word removal, marker deletion, POS tagging [12] and Named Entity Recognition [24]. We ignore the Twitter-specific words (assigned by tag "G"). We replace all numerals by a special token "numeral". The named entity phrases, such as person, organization location are replaced by special tokens. A tweet is transformed to a sequence of tokens after pre-processing.

The skeleton of a pair of tweets is the longest common sequence (LCS) of tokens. Note that we do not consider the position of special tokens in finding the LCS. Special tokens, despite of their values, are considered to be identical and a component of the LCS. After the LCS is found, we append the specific tokens

to the LCS. As shown in Table 1, the LCS for the given pair of tweets is "least numeral location".

Table 1. Illustrative example of a pair of inconsistent tweets

Original tweet 1	At least 38 killed in Pakistan by blast outside Lahore park
Processed tweet 1	least (numeral) killed (location) blast (location)
Original tweet 2	TTP claims Lahore park explosion that killed at least 65
Processed tweet 2	(organization) claim (location) explosion killed least (numeral)

Two tweets are inconsistent, if they are similar at the sentence level, yet they are different in some of the key components. In a real event, numerals and named entities are often key components because they provide situational information about an event. Therefore we identify inconsistency if (1) the length ratio of LCS v.s. the shortest tweet in the pair is equal to or larger than 0.5 and (2) the special tokens, such as numerals, persons, organizations and locations have different values in the two tweets. For example, the two tweets in Tab. 1 are inconsistent, because the LCS has length 3 and the shortest tweet has length 6 and the numerals and locations are different (i.e. 38 v.s. 65). According to Algorithm 2 if tweet2 is newer (thus in x_1), we will remove tweet1 from the candidate index and delete indices in $\mathbf{b}_0$ which are covered by tweet1.

5 Experiment

5.1 Experimental Setup

We use the twitter event data set [18]. The corpus includes tweets for 10 real world events, collected between 2014 and 2016, through the Twitter API using keyword matching. Details of the data set, including the description of each event, the number of tweets related to each event are shown in Tab. 2.

Different types of tweets are contained in the data set, including replies and retweets. The corpus is multilingual, including English, Japanese and so on. In pre-processing, we first filter non-English tweets. We use the Bloom Filter algorithm [19]to filter duplicate tweets. Emoji expressions, http links and mentions (@somebody) are removed from the vocabulary.

5.2 Summarization Performance

We first evaluate the summarization performance of the proposed algorithm. For each event, we chronologically order the tweets and segment the timeline. Each time segment contains 3000 tweets. We run the search engine Terrier [2], and retrieve top 100 related posts returned by Terrier. We assign $b = 0.2$ for tweets published by any user with more than 5 tweets in the data set, and $b = 0.1$ for other tweets. We assign $c_i = 0.25 \times I_i(u_i) + 0.25 \times \frac{nr_i}{5000} + 0.5 \times \frac{nt_i}{500}$, where $I_i(u_i)$

[2] www.terrier.org

Table 2. Statistics of the events and related tweets

Abbrevation	♯ Tweet	Time period (start end)	Event description
EOutbreak	12983	2014/07/01 2014/07/31	large-scale virus outbreaks in West Africa
GUAttack	21000	2014/06/02 2014/07/17	military operation launched by Israel in the Hamas-ruled Gaza Strip
HProtest	27000	2014/09/26 2014/10/17	a series of civil disobedience campaigns in Hong Kong
THagupit	10315	2014/12/05 2014/12/11	a strong cyclone code-named Typhoon Hagupit hit Philippines.
CHShoot	15000	2015/01/07 2015/01/07	two brothers forced their way into the offices of the French satirical weekly newspaper Charlie Hebdo in Paris
HPatricia	9288	2015/10/24 2015/12/08	the second-most intense tropical cyclone on record worldwide
RWelcome	19725	2015/09/02 2015/11/24	rising numbers of people arrived in the European Union because of European refugee crisis.
BAExplossion	15000	2016/03/22 2016/03/22	three coordinated suicide bombings occurred in Belgium.
HPCyprus	14917	2016/03/29 2016/03/30	a domestic passenger flight was hijacked by an Egyptian man in Cyprus.
LBlast	13423	2016/03/27 2016/03/30	a suicide bombing that hit the main entrance of Gulshan-e-Iqbal Park.

is an identity function which returns whether the tweet user u_i is a certified account, nr_i is the number of retweets of i, nt_i is the number of thumbs up i receives.

We compare our summarization scheme with four state-of-the-art summarization methods. (1)LPR [21]:summarization based on a sentence connectivity matrix. (2)MSSF [22]: multi-document summarization based on submodularity hidden in textual-unit similarity property. (3)SNMF [23]: summarization based on symmetric non-negative matrix decomposition. (4)Sumblr [14]: online tweet summarization based on incremental clustering.

The ground truth of summaries for each time segment is manually generated. For each time segment of each event, three human volunteers review the summaries generated by the above mentioned methods and individually select the tweets that can summarize the events of this time period. The final gold standard summary is selected by taking a simple majority vote on each tweet.

The measurement is ROUGE(Recall-Oriented Understudy for Gisting Evaluation). ROUGE is a standard evaluation toolkit for document summarization [20] which automatically determines the quality of a summary by comparing it with the human generated summaries through counting the number of their overlapping textual units (e.g., n-gram, word sequences, and etc.). There are four different ROUGE measures: ROUGE-N, ROUGE-L, ROUGE-W, and ROUGE-S, depending on the textual units to be compared.

We report the ROUGE F-score for all versions of ROUGE measures, averaged over the 10 events in our data set, by our simplex algorithm and the comparative state-of-the-art methods. As shown in Fig. 1, our system (marked as Simplex) outperforms state-of-the-art methods in terms of all ROUGE metrics. Our method outperforms Sumblr, which is also a realtime tweet summarization system, by more than 20% in terms of all ROUGE-F scores. It is clear that, despite of the nature of events, the proposed system which is based on integer programming problems is able to model the summarization task well.

Furthermore, by explicitly detecting for inconsistency, the quality of summary is significantly enhanced.

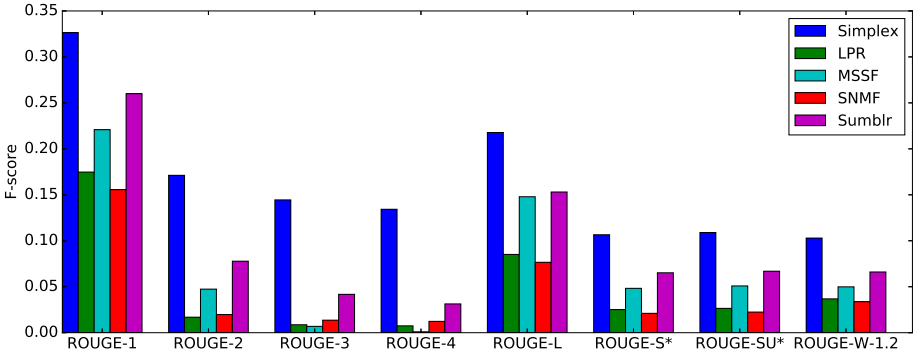


Fig. 1. Average ROUGE F-score on 10 events by different methods.

5.3 Inconsistency Detection Performance

Next we study the integrity of summaries generated by different methods. As events in our dataset vary in the length of their lifespan, we select the first 4 time segments for every event, and compute the inconsistency rate of the summary produced by each method. We first adopt the inconsistency detection strategy in Sec. 4 to identify inconsistent reports in each summary, and then we manually check the inconsistent reports. We finally obtain the inconsistency rate, which is the proportion of the number of inconsistent reports to the number of reports in a summary.

As shown in Fig. 2, the proposed method preserves the integrity of the summary. At each time segment, the median, 25th and 75th percentile of the inconsistency rate over all event summaries produced by the proposed methods are all near zero. On the contrary, the state-of-the-art comparative methods can not generate an consistent summary. LPR and SNMF are better than MSSF and Sumblr, possibly due to the fact that LPR and SNMF tend to select a smaller number of reports. Sumblr and MSSF tend to select more redundant reports, thus the possibility of inconsistency is increased.

5.4 Efficiency Study

We now analyze the efficiency of the various techniques. We implement the proposed algorithm in java and use the open source code for LPR,MSSF, SNMF and Sumblr. All comparative methods are given the set of candidates in each time segment. All experiments have been executed on a server with an Intel(R) Xeon(R) CPU E5-1660 3.20G Hz (4 cores)and main memory 64G bytes.

As some methods adopt incremental updating (i.e. Sumblr and ours) while some do not (i.e. LPR,MSSF,SNMF), efficiency analysis must be separated for the first summary and the summaries afterwards. We report execution times of each method in generating summary in the first time segment, averaged over all events in Fig. 3(a). The graph y-axes has logarithmic scale to illustrate the

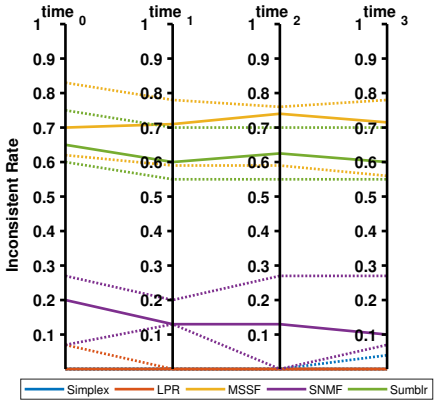


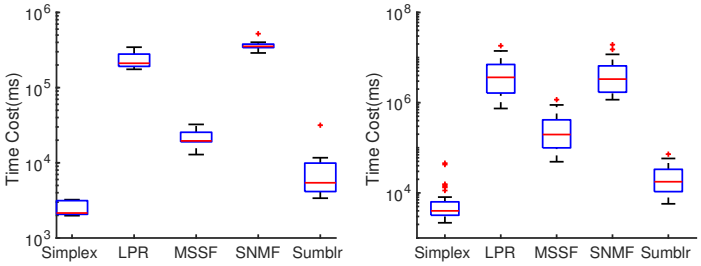
Fig. 2. Inconsistentcy rate, including median (solid line), the 25th percentile and the 75th percentile (dashed line) over ten events on first four time segments for each method .

difference more clear. We can see that the proposed algorithm is the fastest. We emphasize here that the proposed algorithm is 10 times faster than the second fast method Sumblr, which is also an incremental updating summarization system. This highlights the potential of our proposed method, since generating realtime summaries is critical during disaster events.

We then report the average execution times by different methods, over all events and all follow-up time segments in Fig. 3(b). We see that again our proposed method is the fastest. Compared with Fig. 3(a), the execution times for LPR,MSSF and SNMF to generate follow-up summaries differ by two orders of magnitude. This is expected as LPR,MSSF and SNMF are not incremental methods. Thus they are not suitable for delivering real-time summaries. For any ongoing event their computation time will increase exponentially with the size of accumulated tweets. However, Sumblr and our proposed method are more efficient in updating the summaries, as re-computation is limited to a small range of reports. Our method is more scalable than Sumblr, not only because averagely our method requires significantly less execution time (smaller median of the boxplot) but also because execution times of our method have a narrower distribution, which implies that our method is more stable with events of different sizes.

5.5 Effects of Parameters

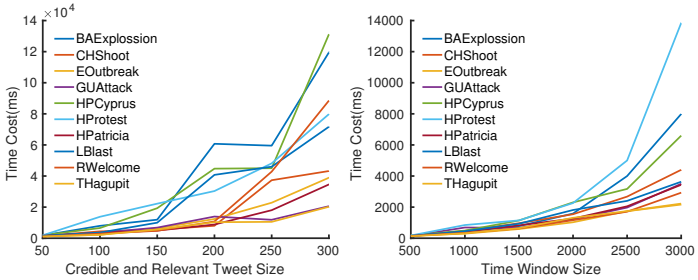
Finally we examine the effects of parameters in the efficiency of the proposed method. We study the effect of the size of credible and relevant tweets (the number of seeds M in Sec. 3) in Fig. 4(a). The execution time is the time cost to generate the first summary. We set $M = \{50, 100, 150, 200, 250, 300\}$. We can see that the execution time grows linearly with the size of credible and relevant tweets. However when the number of seeds is too big, the time cost is significantly higher. We observe similar patterns in Fig. 4(b), where the reported execution



(a) First time segment (b) Follow-up time segments

Fig. 3. Average execution times to generate a summary by various methods.

time is averaged over all time segments for each event. The time cost grows linearly with the size of time segments when the time window is small. However when the time window is too wide, the time cost is significantly higher. The above two observations suggest that in order to generate realtime summary, it is better to segment the timeline into more divisions and limit the size of seeds.



(a) M (b) T

Fig. 4. Execution times to generate summary for each event by the proposed method, given different values of parameters.

6 Conclusion

In this paper we study the problem of realtime event summarization. We implement inconsistency detection in the update procedure to preserve integrity of the summary. We model the realtime summarization problem as integer programming problems and solve the relaxed linear programming form by simplex method. We present a novel inconsistency detection strategy and embed it in the simplex algorithm. Our work sheds insights in intellectual information management in social media platforms, and is especially important for disaster monitoring systems. Our future direction includes exploring more advanced inconsistency detection strategies and exploiting state-of-the-art numerical optimization techniques to solve the integer programming problems.

References

1. J. Y. Chin, S. S. Bhowmick, and A. Jatowt. Totem: Personal tweets summarization on mobile devices. SIGIR 2017, pp. 1305-1308.
2. M. Efron and G. Golovchinsky. Estimation methods for ranking recent information. SIGIR 2011, pp. 495-504.
3. M. Gillani, M. U. Ilyas, S. Saleh, et al. Post summarization of microblogs of sporting events. WWW 2017, pp. 59-68.
4. J. Hannon, M. Bennett, and B. Smyth. Recommending twitter users to follow using content and collaborative filtering approaches. RecSys 2010, pp. 199-206.
5. H. Kwak, C. Lee, H. Park, et al. What is twitter, a social network or a news media? WWW 2010, pp. 591-600.
6. C. Lin, C. Lin, J. Li, et al. Generating event storylines from microblogs. CIKM 2012, pp. 175-184.
7. Z. Liu, Y. Huang, and J. R. Trampier. Leds: Local event discovery and summarization from tweets. GIS 2016, pp. 53:1–53:4.
8. M. Mathioudakis and N. Koudas. Twittermonitor: trend detection over the twitter stream. SIGMOD 2010, pp. 1155-1158.
9. X. Meng, F. Wei, X. Liu, et al. Entity-centric topic-oriented opinion summarization in twitter. KDD 2012, pp. 379-387.
10. Z. Ren, S. Liang, E. Meij, et al. Personalized time-aware tweets summarization. SIGIR 2013, pp. 513-522.
11. K. Rudra, S. Banerjee, N. Ganguly, et al. Summarizing situational tweets in crisis scenario. HT 2016, pp. 137-147.
12. K. Rudra, S. Ghosh, N. Ganguly, et al. Extracting situational information from microblogs during disaster events: A classification-summarization approach. CIKM 2015, pp. 583-592.
13. B. Sharifi, M.-A. Hutton, and J. Kalita. Summarizing microblogs automatically. HLT 2010, pp. 685-688.
14. L. Shou, Z. Wang, K. Chen, et al. Sumblr: continuous summarization of evolving tweet streams. SIGIR 2013, pp. 533-542.
15. H. Takamura, H. Yokono, and M. Okumura. Summarizing a document stream. ECIR 2011, pp. 177-188.
16. W. Yih, J. Goodman, L. Vanderwende, et al. Multi-document summarization by maximizing informative content-words. IJCAI, 2007, Vol. 7, pp. 1776-1782.
17. A. Zubiaga, D. Spina, E. Amigó, et al. Towards real-time summarization of scheduled events from twitter streams. HT 2012, pp. 319-320.
18. A. Zubiaga. A Longitudinal Assessment of the Persistence of Twitter Datasets. arXiv preprint arXiv:1709.09186, 2017.
19. B. Bloom. Space/time tradeoffs in hash coding with allowable errors. Communications of the ACM, 1970, 13(7): 422-426.
20. C. Y. Lin. Rouge: A package for automatic evaluation of summaries. Text Summarization Branches Out, 2004.
21. G. Erkan, D. R. Radev. LexPageRank: Prestige in Multi-Document Text Summarization. EMNLP 2004, pp. 365-371.
22. J. X. Li, L. Li, T. Li. MSSF: A Multi-Document Summarization Framework based on Submodularity. SIGIR 2011, pp. 1247-1248.
23. D. Wang, T. Li, S. Zhu, et al. Multi-document summarization via sentence-level semantic analysis and symmetric matrix factorization. SIGIR 2008, pp. 307-314.
24. J. Rose. Finkel, T. Grenager, and C. Manning. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. ACL 2005, pp. 363-370.