Date of acceptance Grade

Instructor

Event detection in interaction network

Han Xiao

Helsinki June 3, 2016

UNIVERSITY OF HELSINKI
Department of Computer Science

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Faculty of Science	Department of Computer Science

Tekijä — Författare — Author
Han Xiao

Työn nimi — Arbetets titel — Title

Event detection in interaction network

Oppiaine — Läroämne — Subject
Computer Science

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
	June 3, 2016	52 pages + 3 appendices

Tiivistelmä — Referat — Abstract

We study the problem of detecting top-k *events* from digital interaction records (e.g, emails, tweets). We first introduce *interaction meta-graph*, which connects associated interactions. Then, we define an *event* to be a subset of interactions that (i) are topically and temporally close and (ii) correspond to a tree capturing information flow. Finding the best event leads to one variant of prize-collecting Steiner-tree problem, for which three methods are proposed. Finding the top-k events maps to maximum k-coverage problem. Evaluation on real datasets shows our methods detect meaningful events.

ACM Computing Classification System (CCS):
H.4 [Information Systems Applications],
H.2.8 [Database Applications],
G.2.2 [Graph Theory]

Avainsanat — Nyckelord — Keywords
graph mining, social-network analysis, temporal networks, event detection, graph summarization

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — övriga uppgifter — Additional information

Contents

1 Introduction **1**

2 Background **4**

2.1 Discrete optimization 4

2.2 Approximation algorithms 5

2.3 Prize-collecting Steiner-tree problem and its variants 6

2.4 Problem relationships 7

3 Model **9**

3.1 Interaction network 9

3.2 Interaction meta-graph 10

4 Problem formulation **13**

4.1 Finding the best event 13

4.2 Finding the top-k events 14

5 Algorithms **15**

5.1 Finding the best event 15

5.2 Finding the top-k events 18

5.3 Root sampling strategy 18

6 Experimental evaluation **21**

6.1 Datasets and preprocessing 21

6.2 Evaluation on synthetic datasets 23

6.3 Parameter effects on real datasets 25

6.4 Evaluation on sports tweet dataset 27

6.5 Case study in English Letter 29

6.6 Case study in Enron dataset 31

6.7 Case study in Twitter datasets 32

7 Discussion **36**

7.1 Problem formulation 36

7.1.1 Mixed topics 36

7.1.2 Broken stories 37

7.1.3 Alternative problem formulations 38

7.2 Interaction meta-graph construction 39

7.2.1 Similarity = causality? 39

7.2.2 *Self-talking* problem 39

7.3 Other interesting directions 40

8 Conclusions **42**

8.1 Contribution 42

8.2 Related work 43

8.2.1 Event detection 43

8.2.2 Text summarization 45

8.2.3 Information diffusion 46

References **47**

Appendices

1 NP-hardness of Problem 4

2 Code and scripts

1 Introduction

Event detection is a fundamental problem in data mining and numerous methods have been applied to a variety of application areas, including detecting events in time series and data streams [GS99], point clouds and vector spaces [BKNS00], and networks [BGHS12]. In this paper we focus on the problem of event detection in networks, in particular networks that contain both *content* and *time* information about interactions between the nodes . We define an *interaction* (u, V, α, t) to be a piece of information represented by a vector α sent by a node u to other nodes V at time t. Examples of interaction networks include data communication networks, such as email, Twitter, or online messaging systems.

At a high level, our goal is to summarize the network activity by finding the top-k events that take place. We consider an interaction network $H = (N, \mathcal{I})$ representing a set of interactions $\mathcal{I}$ that take place among a set of nodes N. The interactions in $\mathcal{I}$ are directed, annotated with content information, and time-stamped. We define an *event* in the interaction graph H to be a subset of interactions, $\mathcal{I}' \subseteq \mathcal{I}$ that are (i) temporally close, (ii) topically similar, and (iii) correspond to a tree that captures the information flow in the network.

We convert the interaction network $H = (N, \mathcal{I})$ into a weighted *interaction meta-graph* $G = (\mathcal{I}, E)$, that is, a graph whose vertices are the interactions $\mathcal{I}$. Two interactions $i, j \in \mathcal{I}$ are connected in G if it is possible to explain the information flow between i and j. In particular, we consider three types of flow: *broadcast*, *relay* and *reply*. The edge weights of the interaction meta-graph G measure the topic dissimilarity between connected interactions. Our transformation from the interaction network to the interaction meta-graph has the interesting property that an *event* in the interaction graph H corresponds to a tree T in the interaction meta-graph G. The root of the tree T is interpreted as the source of the event. Downstream interactions that occur inside the tree are attributed to the information-propagation mechanism.

We formalize the task of interaction-network summarization as the problem of finding top-k trees in the transformed interaction meta-graph $G = (\mathcal{I}, E)$. We decompose our problem into two sub-problems. First, we find a set of independent candidate events, which are temporally and topically coherent. Since our goal is to summarize the interaction network we aim to find events that satisfy temporal and topical coherence constraints while *maximizing* the number of event nodes. We show

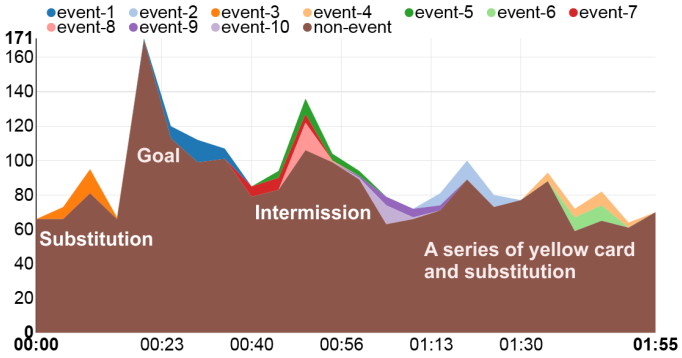
this problem is the budget version of *prize-collecting Steiner-tree* (PCST) problem in directed acyclic graphs. We provide three algorithms, the best performing of which is one based on a greedy approach.

The second sub-problem is to select k events that maximize the overall node coverage. This task maps to the *maximum k-coverage* problem, and it can be approximated using a standard greedy algorithm. To further speed up the running time of our algorithm, we also propose a search strategy that avoids evaluating candidate events at all possible tree roots, but heuristically selects the most promising ones.

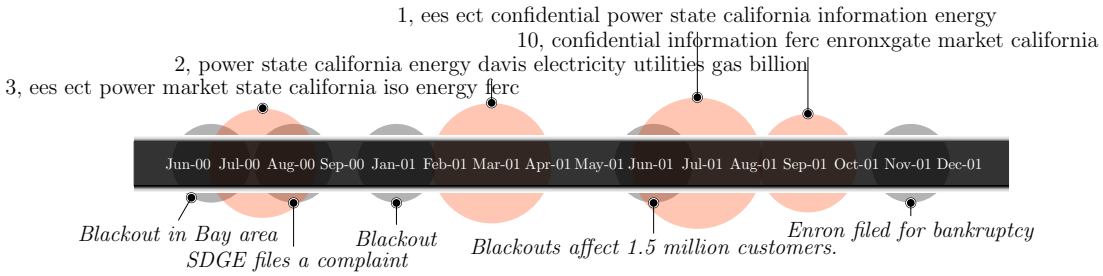
The problem defined in this paper can be applied to a variety of application scenarios, such as online social media, bioinformatics, or political science. In our experimental evaluation, we mainly focus on analyzing textual data in social media. We experiment with one letter dataset, one email dataset and several Twitter-based datasets. We provide a comparison of the different approaches, as well as several examples that our methods discover meaningful events. As an example, we demonstrate discovered events for two datasets in Figure 1.1. The first dataset contains tweets about one football game. The second contains email records from Enron Corporation[1]. The emails are mainly about energy crisis in California[2] and Enron scandal[3].

In the following sections, we will first give background on discrete optimization and related problems in Section 2. In Section 3 and 4, we describe our model and problem formulations respectively. Later in Section 5, corresponding algorithms are proposed and analyzed. Then, experiment results on both synthetic and real datasets are given in Section 6. Finally, we discuss possible improvements and conclusions in Section 7 and Section 8.

[1] https://en.wikipedia.org/wiki/Enron
[2] https://en.wikipedia.org/wiki/California_electricity_crisis
[3] https://en.wikipedia.org/wiki/Enron_scandal



(a)



(b)

Figure 1.1: Detected events by our methods. (a) the temporal volume of detected events on tweets about a football match and (b) timeline description on *Enron* email datasets . In (a), tweet frequency is plotted against time. Top-10 events as well as non-event messages are colored. Manual annotation of the actual events is displayed. In (b), external events (smaller black circles and labeled with italic text) and extracted events in larger and red circles are plotted. Extracted events are summarized by their top topic terms. (*ees*: Enron Energy Service, *ect*: Enron Capital and Trade Resources, *iso*: California Independent System Operator, *ferc*: Federal Energy Regulatory Commission, *SDGE*: San Diego Gas & Electri Company.)

2 Background

Before diving deeper, we first give the definition and examples of discrete optimization, which lies at the heart of our methods. As many such problems are **NP**-hard, we briefly describe *approximation algorithms*, which approaches **NP**-hard problems faster but in sacrifice of optimality. We also give definitions to the *prize-collecting Steiner-tree* problem and its variants. Last, we describe the relationships among a list of discrete optimization problems related to this work.

2.1 Discrete optimization

According to the definition in [BV04], an *optimization problem* has the form:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f_0(x) \\ \text{subject to} \quad & f_i(x) \leq b_i, i = 1, \ldots, m \end{aligned}$$

The vector $x = (x_1, \cdots, x_n)$ is the *optimization variable*. $f_0 : \mathbb{R}^n \to \mathbb{R}$ is the *objective function* to be optimized. $f_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, \cdots, m$ are the *constraint functions* and $b_1, \cdots, b_m$ are the limits for the constraints. A vector x^* is *optimal* if it has the smallest objective function value meanwhile satisfying all the constraints. *Discrete optimization problems* refer to a subset of optimization problems where values for $x_1, \cdots x_n$ are discrete.

Example 1 ***Set Cover****: given a set of ground elements,* $U = \{u_1, \ldots, u_n\}$ *and* $\mathbf{S} = \{S_i \subseteq U \mid i = 1, \ldots, n\}$*, find a set* $\mathbf{S}' \subseteq \mathbf{S}$ *that minimizes* $\left|S'\right|$ *such that* $\underset{S_i \in \mathbf{S}'}{\cup} S_i = U$

We denote $x_i = 1$ if $S_i \in \mathbf{S}'$ and 0 otherwise and have the following form for this problem:

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1,\ldots,m} x_i \\ \text{subject to} \quad & \sum_{S_i : u \in S_i} x_i > 0 \text{ for } u \in U \\ & x_i \in \{0, 1\} \end{aligned}$$

2.2 Approximation algorithms

Many discrete optimization problems are **NP**-hard, therefore, unless P = **NP**, there are no efficient algorithms solving those problems optimally with running time bounded polynomially by the input size [WS11]. In this case, people usually relax either of the following requirements, *optimality* or *efficiency*. For example, one approach relaxes the efficiency requirement and is often successful if one puts more values on optimality and is willing to pay time cost.

However, people are often daunted by the time cost which can be several days even months. For tasks that need to be done in real-time, such as control system and digital game, waiting several seconds is intolerable. The other approach relaxes the optimality requirement meanwhile trying to find a "good enough" or even near-optimal solution.

α-approximation algorithms belong to the latter approach and is defined in [WS11] as:

Definition 1 *An α-approximation algorithm for an optimization problem is a polynomial-time algorithm that for all instances of the problem produces a solution whose value is within a factor of α of the value of an optimal solution.*

Intuitively, α measures how much the optimality requirement is relaxed. For example, a $\frac{1}{2}$-approximation algorithm for a maximization problem always returns a solution with value at least half of the optimal value. Note that for minimization problems, $\alpha > 1$ and for maximization problems $\alpha < 1$ is used. By convention, the notion of an algorithm with approximation guarantee α is the same as an α-approximation algorithm. α-approximation algorithms differ from heuristic algorithms, such as simulated annealing and genetic algorithms, in that the former provide approximation guarantee while the latter do not.

The study of approximation algorithms gives us the following benefits. First of all, we need to solve many **NP**-hard problems in practice and in some cases, approximation algorithms give good even near-optimal results. Second, instead of evaluating heuristics empirically, approximation factor gives a quantified approximation guarantee in a rigorous way. Last, it gives a way to measure how hard various **NP**-hard problems can be approximated.

2.3 Prize-collecting Steiner-tree problem and its variants

Prize collecting Steiner tree problem (PCST) is one famous **NP**-hard discrete optimization problem. It was first studied by Goemans et al. [GW95]. In the original work, undirected graphs (thus unrooted) are studied. Without loss of generality, we consider the rooted case in directed graphs.

In this problem, one is given directed graph $G = (V, E)$, tree root $r \in V$, edge cost function: $c : E \to \mathbb{R}$ and node prize function: $p : V \to \mathbb{R}$. He needs to find a subtree $T = (V^{'}, E^{'})$ rooted at r that minimizes $\sum_{e \in E^{'}} c(e) + \sum_{v \notin V^{'}} p(v)$. In other words, the optimal subtree should minimize the cost of edges in the tree plus the prizes of nodes *not* covered by the tree.

There are several related variants to the original form:

- *Net Worth Maximization*: one is asked to find subtree $T^{'}$ rooted at r that maximizes $\sum_{v \in V^{'}} p(v) - \sum_{e \in E^{'}} c(e)$.
- *PCST-Quota*: one is given *quota* Q and asked to find subtree $T^{'}$ rooted at r that minimizes $\sum_{e \in E^{'}} c(e)$ under the constraint $\sum_{v \in V^{'}} p(v) \geq Q$.
- *PCST-Budget*: one is given *budget* B and asked to find subtree $T^{'}$ rooted at r that maximizes $\sum_{v \in V^{'}} p(v)$ under the constraint $\sum_{e \in E^{'}} c(e) \leq B$.

Note that the original problem is the Lagrangian relaxation form for both *PCST-Quota*.

A motivating application for PCST is the fiber-optic network design problem. Suppose we want to lay a fiber-optic network along the roads in a neighborhood. The street map corresponds to the graph, where graph node corresponds to road intersection or customer building. The prize corresponds to the estimated revenues we can earn by connecting the node. Meanwhile, edge cost corresponds to the cost of laying network cable along that edge.

The original version does not make much economical sense, while its variants suits better for this application. For example, the *Net Worth Maximization* aims for the most profitable network. For *PCST-Quota*, the network should cost as least as possible while satisfying a minimum revenue requirement. For *PCST-Budget*, the goal is to maximize the revenue without surpassing a given budget for the construction cost.

Though the original formulation is equivalent to the *Net Worth Maximization* problem in terms of optimization as the sum of their objective functions equal to the prize sum of all nodes. However, they differ in terms of approximation. For example, Goemans [GW95] proposes $2 - \frac{1}{|V|-1}$ algorithm for the original formulation. However, [FPS00] proved there is no constant α-approximation algorithm for the *Net Worth Maximization* problem unless **P**=**NP**.

2.4 Problem relationships

Several problems directly relate to *PCST-Budget* in a way that one can be reduced to another or approximated using another. In this part, we describe the relationships between these problems. Specifically, we consider *k-minimum spanning tree* (*k-MST*), *Steiner-Tree* (*Steiner-Tree*), *PCST-Budget* and *PCST-Quota* and show how one can be reduced to or approximated using another. The results are summarized in Figure 2.1.

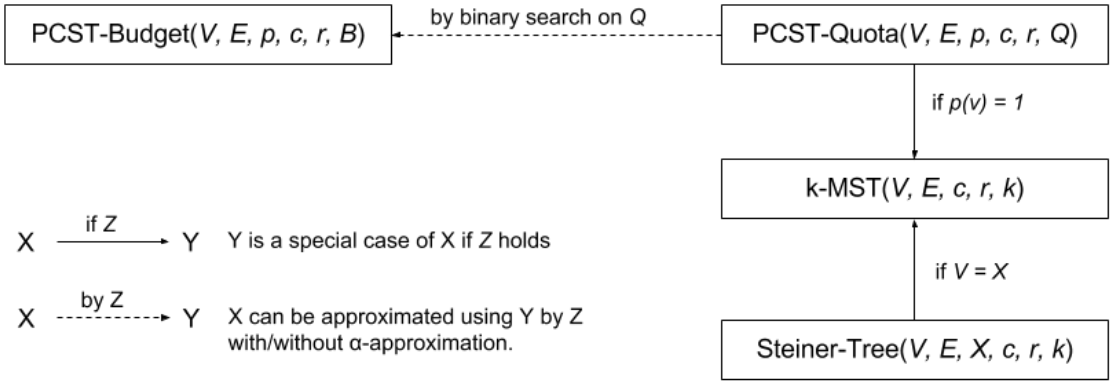


Figure 2.1: Existing problems related to *PCST-Budget* and their relationships.

Problem 1 ***k-minimum spanning tree(k-MST)****: Given an directed graph* $G = (V, E)$ *with edges cost:* $c : E \to \mathbb{R}$, $r \in V$ *and an integer* k, *find a spanning tree* $T = (V', E')$ *rooted at* r *with* $\left|V'\right| = k$ *that minimizes* $\sum_{e \in E'} c(e)$

k-MST reduces to *PCST-Quota* when $p : V \to 1$. In this case, k in *k-MST* maps to Q in *PCST-Quota*.

For *Steiner-Tree* problem, we use the definition in [CCC+99].

Problem 2 ***Steiner Tree****: Given a directed graph $G = (V, E)$, $r \in V$, a set of terminal nodes: $X \subseteq V$, edges cost: $c : E \to \mathbb{R}$ and an integer k, find a subtree $T = (V', E')$ rooted at r such that $\left|V' \cap X\right| = k$ and $\sum_{e \in E'} c(e)$ is minimized.*

k-MST is a special case of *Steiner-Tree* when $X = V$. Therefore, α-approximation algorithm for *Steiner-Tree* (such as [CCC+99]) also applies for *k-MST*.

Algorithms for *PCST-Quota* can be used to approximate *PCST-Budget*. Johnson et al. [JMP00] propose $(5+\epsilon)$-approximation algorithm for *PCST-Budget* in undirected graph. The $(5+\epsilon)$ factor is derived by reduction from *k-MST* to *PCST-Budget*. The algorithm repeatedly calls algorithm for *k-MST* using binary search on Q. If the resulting subtree violates the budget constraint in *PCST-Budget*, then Q is divided by $1 + \epsilon$ to a smaller value. Otherwise, Q is multiplied by $1 + \epsilon$ to a larger one. Finally, the output tree using Q has total edge cost smaller than B while the tree using $(1 + \epsilon)Q$ has cost larger than B.

For directed graph, a similar binary search procedure can be used to approximate *PCST-Budget*. Even though there is an α-approximation algorithm [CCC+99] for *PCST-Quota* in directed graph, to our knowledge, there does not exist any approximation guarantee for the corresponding *PCST-Budget*.

3 Model

An interaction network is represented by a list of interactions, each consist of a sender, one or more recipients, a content vector and timestamp information. We propose a transformation of an interaction network into an *interaction meta-graph*, which captures temporal and topical association as well as the information flow in the network. This transformation helps to provide a cleaner abstraction to the event-detection problem.

3.1 Interaction network

An *interaction network* $H = (N, \mathcal{I})$ consists of a set of n nodes, N and a set of m time-stamped interactions, $\mathcal{I}$ between one *sender* node and a set of *recipient* nodes. We assume that interactions are annotated with textual content. In all our experiments we apply text modeling techniques, such as Latent Dirichlet Allocation (LDA) [BNJ03], on all the available text in the network to learn global topic-to-token probability for L topics. Therefore, each interaction is associated with a topic vector. We note that the topic-modeling choice is independent to our main methodology; in practice any text-representation scheme can be used, such as, simple bag-of-words representation.

The set of interactions $\mathcal{I}$ in the network is represented as

$$\mathcal{I} = \{(u_i, V_i, \alpha_i, t_i)\}\,, \text{ with } i = 1, \ldots, m, \text{ such that } u_i \in N, V_i \subseteq N, t_i \in \mathbb{R},\ \alpha_i \in \mathbb{R}^L,$$

indicating that nodes u_i interacts with V_i at time t_i, and the content of their interaction is described by the vector α_i. We consider that interactions are directed. In other words, more than one interaction may take place between a pair of nodes, with different time stamps. Conversely, more than one interaction may take place at the same time, between different nodes. Online communication networks, such as email networks, Facebook networks are examples of interaction networks. In these networks, interactions correspond to messages being sent and where one message can have multiple recipients, e.g, $|V_i| > 1$. Letter networks is another example, where there is usually one recipient for each interaction, e.g, $|V_i| = 1$.

3.2 Interaction meta-graph

Given an interaction network H we construct a directed weighted *interaction meta-graph* $G = (\mathcal{I}, E, c)$. The vertices of $\mathcal{I}$ in G correspond to the interactions $\mathcal{I}$ in H. There is an edge from vertex $i = (u_i, V_i, \alpha_i, t_i) \in \mathcal{I}$ to a vertex $j = (u_j, V_j, \alpha_j, t_j) \in \mathcal{I}$ if the following holds:

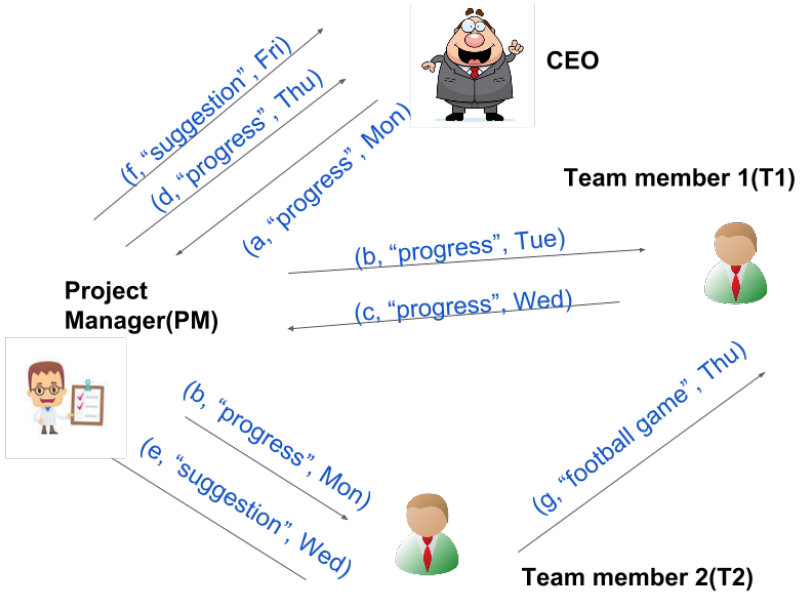
1. Interaction i takes place before interaction j (time comprehension): $t_i < t_j$.
2. Information comprehension takes place in one of the following ways:
   (a) interactions i and j share the same sender node in N: $u_i = u_j$ (*broadcast*);
   (b) the recipient nodes of the interaction i contain the sender node of the interaction j and the recipient nodes of j is not the sender node of i: $u_j \in V_i$ and $u_i \notin V_j$ (*relay*);
   (c) the recipient nodes of an interaction i contain the sender node of an interaction j and the recipient nodes of j contain the sender node if i: $u_j \in V_i$ and $u_i \in V_j$ (*reply*).

Example 2 ***A synthetic interaction meta-graph*** *(Figure 3.1): The edges between (d, f), (a, b), (b, c) (Figure 3.1 (c)) are the examples of three edge types respectively (broadcast, relay, reply). Two main events are happening: (1) progress: CEO asked PM about project progress in interaction a, which is forwarded to T1 and T2 in interaction b. Later, T1 reported back to PM in c. Last, PM reported to CEO in d. The information flow in this event is $a \to b \to c \to d$. (2) T2 came up with some suggestion and reported it to PM in e. The PM found the suggestion useful and later forwarded to CEO in f. The information flow in this event is $e \to f$. g is not in the top-2 events as the topic is different from the top-2 and it is the only football-related interaction. Note that due to time comprehension, G is a DAG.*

For the edges of the interaction meta-graph G, we assign weights to measure the topical dissimilarity between interactions. Given two interactions $(u_i, V_i, \alpha_i, t_i)$ and $(u_j, V_j, \alpha_j, t_j)$ connected by an edge in G, our edge-weighting function $c : E \to \mathbb{R}$ is a distance function between the topic vectors α_i and α_j.

Finally, given a meta-graph $G = (\mathcal{I}, E, c)$ and a time interval $[s, f]$ we define the *time-induced meta-graph* $G([s, f]) = (\mathcal{I}([s, f]), E, c)$, where $\mathcal{I}([s, f])$ are the interactions that occur in $[s, f]$

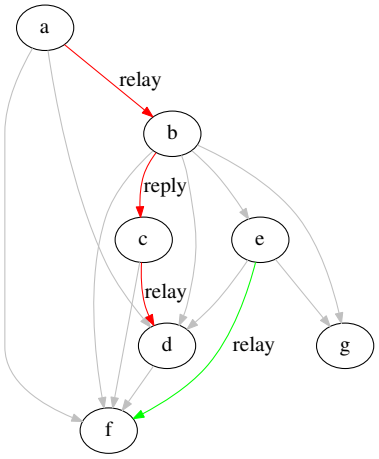
$$\mathcal{I}([s, f]) = \{(u, V, \alpha, f) \in \mathcal{I} \mid s \leq t \leq f\}.$$



(a)

Id	Sender	Recipients	Time	Topic
a	CEO	[PM]	Mon	progress
b	PM	[T1, T2]	Tue	progress
c	T1	[PM]	Wed	progress
d	PM	[CEO]	Thu	progress
e	T2	[PM]	Wed	suggestion
f	PM	[CEO]	Fri	suggestion
g	T2	[T1]	Thu	football

(b)



(c)

Figure 3.1: A synthetic example of company email network. (a) is the interaction network. Each edge corresponds to one interaction/email, which is labeled by (`interaction id, message topic, timestamp`). (b) lists the individual interactions. All interactions happen within the same week. (c) is the corresponding interaction meta-graph, in which edges that correspond to the true information flow are colored by the corresponding events.

The motivation for defining $G([s, f])$ is that we are interested in interactions that occur within a bounded time interval.

4 Problem formulation

We aim at detecting the top-k events that are (1) topically and temporally coherent individually and (2) together cover interactions as many as possible.

We define an *event* to be a rooted subtree T of the interaction meta-graph G. An event naturally has a source vertex (or interaction) and is spread in the temporal network H by propagation mechanism that corresponds to information flow along the edges of G. We are interested in events that have high volume, which translates to the number of interactions included into tree T. We are also interested in events in which the interactions are temporally close and topically coherent. Finally, we are interested in finding the top-k events that cover as many interactions as possible. Note that events might overlap in their covered interactions.

To simplify the problem of finding the top-k events, we decompose the main task into two sub problems: (1) finding a set of independent candidate events that satisfy topic and time constraints and maximize coverage of interactions, and (2) selecting the top-k events to maximize total coverage.

4.1 Finding the best event

We give the formal definition of the problem of finding the best event:

Problem 3 *Given an interaction meta-graph $G = (\mathcal{I}, E, c)$, a root vertex $r \in \mathcal{I}$, a time budget I, and a dissimilarity budget B, find a directed subtree $T = (V_e, E_e) \subseteq G$, rooted at r, which maximizes the number of vertices*

$$|V_e|$$

while satisfying the constraints

$$\sum_{e \in E_e} c(e) \leq B \quad \textit{and} \quad (\max_{i \in V_e} t_i - \min_{j \in V_e} t_j) \leq I$$

Note that the time constraint can be omitted, if we restrict the input graph to be induced by the time interval $[t_r, t_r + I]$, where t_r is the root node's time-stamp.

We observe that Problem 3 is a special case of *PCST-Budget* when $p(v) = 1$. In addition, the input graph is DAG. Overall, our problem can be written as follows.

Problem 4 *Given a weighted directed acyclic graph $G([s,f]) = (\mathcal{I}([s,f]), E, c)$, a root vertex r, and cost budget B, find a subtree $T = (V_e, E_e) \subseteq G([s,f])$, rooted at r, that maximizing the number of vertices*

$$|V_e|$$

while satisfying

$$\sum_{e \in E_e} c(e) \leq B$$

.

Despite being a special case to *PCST-Budget*, Problem 4 is **NP**-hard. The proof of the following proposition is given in Appendix.

Proposition 1 *Problem 4 is **NP**-hard.*

4.2 Finding the top-k events

As the interaction network is likely to contain more than one event, we are interested in finding k events that describe different aspects of the whole network while covering as many interactions as possible. This is captured in the following.

Problem 5 *Given an interaction meta-graph $G = (\mathcal{I}, E, c)$ and $k \in \mathbb{N}$, find a set of k trees $\mathcal{T} = \{T_1, \ldots T_k\}$, with each event tree $T = (V_e, E_e) \in \mathcal{T}$ to be a subgraph of G rooted in some $r_i \in \mathcal{I}$, such that the total number of spanned interactions $|\cup_{T=(V_e,E_e)\in\mathcal{T}} V_e|$ is maximized.*

It is easy to observe that this problem is equivalent to *maximum k-coverage* problem, which is **NP**-hard.

Problem 6 ***Maximum k-coverage**: Given a set of ground elements, $U = \{u_1, \ldots, u_n\}$, $S = \{S_i \subseteq U \mid i = 1, \ldots, n\}$ and $k \in \mathbb{Z}$, find a set $S' \subseteq S$ that $\left|S'\right| = k$ and $\underset{S_i \in S'}{\cup} S_i$ is maximized.*

Proposition 2 *Problem 5 is **NP**-hard. [Vaz13]*

5 Algorithms

In this section we present a list of algorithms for problems defined previously. For Problem 4, there are no existing α-approximation algorithms. We propose three algorithms: a greedy algorithm, a dynamic-programming algorithm, and an adaptation to an existing approximation algorithm.

5.1 Finding the best event

To find the best tree, as defined by Problem 4, we consider three algorithms.

Greedy tree growing: The greedy algorithm (Algorithm 1) starts from the root and builds the event tree by adding one vertex (interaction) at a time. At each step the algorithm selects the edge with the minimum cost (topic dissimilarity) from the cutset of the current tree. This choice aims to maximize the topical coherence of the event discovered. The running time is $O(|\mathcal{I}|^2)$.

Algorithm 1: Greedy growing tree algorithm

Data: G, B, r

Result: T rooted at r

$T = (V_e, E_e), V_e = \{r\}, E_e = \{\}$;

$(i, j) = \underset{(i',j') \in cutset(T,G)}{\arg\min} \; c(i', j')$;

while $c(T) + c(i, j) \leq B$ **do**

  $V_e = V_e \cup \{j\}$;

  $E_e = E_e \cup \{(i, j)\}$;

  $(i, j) = \underset{(i',j') \in cutset(T,G)}{\arg\min} \; c(i', j')$;

return T;

Directed Steiner-tree algorithm (*DST*): This algorithm (Algorithm 2) directly relates to the background material in Section 2.4. We illustrate the relationship in Figure 5.1. In this case, we approximate *PCST-Budget* via binary search using *k-MST* on k.

It's easy to see that the larger k is for *k-MST*, the larger cost the resulting tree should have (and vice versa). We initially run algorithm for *k-MST* with $k = |G.\mathcal{I}|$, the largest possible value and see if the produced tree cost $c(T)$ exceeds B. If so,

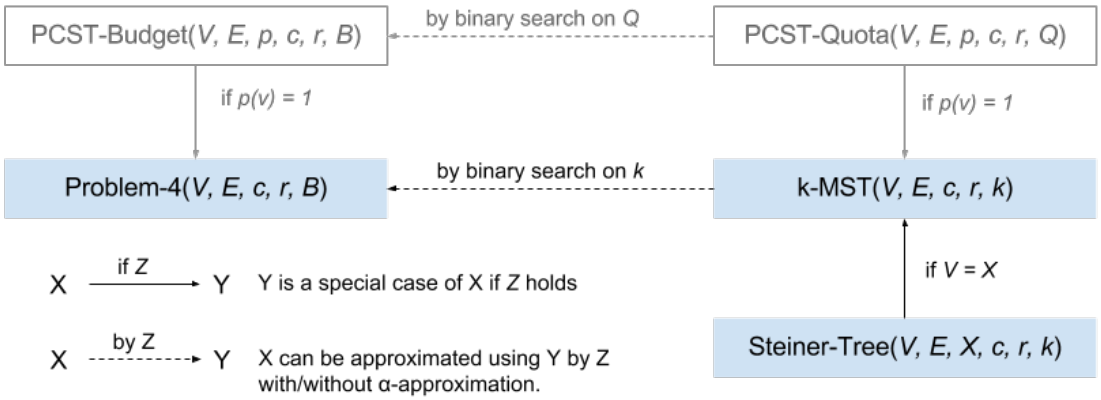


Figure 5.1: Problems related to Problem 4 and their relationships. Problem 4 is added, compared to Figure 2.1. We use algorithm for *Steiner-Tree*, for its special case, *k-MST*, to approximate Problem 4. The approximation is done via binary search.

a new run on *k-MST* with k halved is performed. Otherwise, k is doubled. This process is repeated until the largest possible k which produces a feasible solution for Problem 4 is found.

There is a special treatment as k is integer. In the end, we need to test if setting $k = k_u$ gives feasible solution because the algorithm has not evaluated such case when it quits the while loop.

We use the algorithm by Charikar [CCC+99], $CharikarDST$, for *k-MST* as *k-MST* is a special case of *Steiner-Tree* which $CharikarDST$ approximates. In our case, $CharikarDST$ takes five arguments as input: G, r, X, k and ℓ. X are the terminal node set. k is the minimum number of terminal nodes to cover in the returning tree. ℓ controls the recursion depth of the algorithm. The deeper the recursion, the better approximation guarantee, however, the longer computational time. Thus, ℓ is a parameter that provides a quality-of-approximation vs. efficiency trade-off. The output of $CharikarDST$ is one feasible solution. We assume the edge cost function c is included as attribute in G.

The running time for $CharikarDST$ is $O(|\mathcal{I}|^{\ell}|X|^{2\ell})$. In our case, $X = \mathcal{I}$, thus the running time is $O(|\mathcal{I}|^{3\ell})$. We use $\ell = 1$ but still the algorithm is mainly of theoretical interest and not practical for large datasets.

Unlike [JMP00], which gives approximation ratio for *PCST-Budget* in undirected graph, we are not able to derive any non-trivial approximation guarantee for Algo-

rithm 2, even though $CharikarDST$ provides approximation guarantee.

Algorithm 2: Binary search using Charikar's DST algorithm

Data: G, r, B, ℓ

Result: T rooted at r

```
k_l = 1, k_u = |G.I|;
while k_l < k_u − 1 do
    k = ⌊(k_l + k_u)/2⌋;
    T = CharikarDST(G, r, G.I, k, ℓ);
    if c(T) > B then
        k_u = k − 1;
    else
        k_l = k;
T = CharikarDST(G, r, G.I, k_u, ℓ) ;
if c(T) ≤ B then
    return T ;
else
    return CharikarDST(G, r, G.I, k_l, ℓ);
```

Dynamic programming algorithm (DP): The third algorithm we present is inspired by the idea that when the input DAG is a *tree* with *integer* edge cost, the problem can be solved optimally by using a simple dynamic programming approach.

Each node $u \in V$ has one hash table A_u. $A_u[i] = j$ denotes the total prize of the optimal tree rooted at u using budget i. Initially for each node, $u \in V$, $A_u[0] = p(u)$.

$A_u[i] = j$ is defined recursively in the following ways:

1. If u has only one child v and $c(u, v) \leq B$, then $A_u[k] = A_v[k - c(u, v)] + p(u)$.
2. If u has two children v, w, we can choose the subtree rooted at either of its children or both but using different budgets. This structure can be defined recursively:

$$A_u[k] = max \begin{cases} max\{A_v[i] + A_w[j] + p(u) \mid i + j + c(u, v) + c(u, w) = k\} \\ A_v[k - c(u, v)] + p(u) \\ A_w[k - c(u, w)] + p(u) \end{cases} \quad (1)$$

Without loss of generality, we assume the input tree is binary.

Note that the DAG version (Problem 4) is **NP**-hard. We investigate a way to adapt this algorithm for general (non-tree) DAGs in sacrifice of optimality. Specifically, we first transform the input DAG into a tree by using the single-source shortest paths from r of G. The shortest paths can be calculated by Dijkstra's algorithm [Joh73] and then apply the dynamic-programming algorithm. For integer edge weights and a tree input, the running time of the DP algorithm is $O(|\mathcal{I}|B^2)$. In our case, edge weights are real numbers. So we discretize the weights to some decimal points.

Algorithm 3: Dynamic programming algorithm for *PCST-Budget*: preprocessing by Dijkstra and edge-weight discretization is performed. p is the decimal point for weight discretization.

Data: G, B, r, p

Result: T

1 $T' = dijkstra(G, r)$;

2 $T'' = discretize\_edge\_weights(T', p)$;

3 **return** $DP(T'', B, r)$;

5.2 Finding the top-k events

Once we have computed a set of candidate event trees using any algorithm for Problem 4, we need to select k event trees from the candidate set so that vertex coverage is maximized. This is essentially the *maximum k-coverage* problem. A standard greedy algorithm gives approximation ratio $1 - \frac{1}{e}$ [Vaz13]. The running time is $O(|\mathcal{I}|^2)$.

5.3 Root sampling strategy

Ideally, we calculate candidate trees rooted at all nodes and use them as the input to greedy *maximum k-coverage* algorithm discussed above. However for practical purposes, this leads to an inefficient scheme, as finding the best event for a given interaction as the root is itself an expensive computational task and real-world networks consist of millions of interactions. To overcome this issue and speed up the algorithm for finding top-k event trees, we propose a simple root-sampling strategy that ranks roots according to how promising they are with respect to the node-coverage

Algorithm 4: Greedy algorithm for maximum k-coverage problem (Problem 5)

Data: $\mathcal{T}^{'}, k$

Result: $\mathcal{T}$

$\mathcal{T} = \{\}$;

$\mathcal{V} = \{\}$;

while $|\mathcal{T}| < k$ **do**

  $T = \arg\max_{T' \in \mathcal{T}'} |T'.V - \mathcal{V}|$;

  $\mathcal{T} = \mathcal{T} \cup T$;

  $\mathcal{V} = \mathcal{V} \cup T.V$;

return $\mathcal{T}$;

objective function. In practice, roots with very little promise will be excluded from evaluation.

The idea of our sampling scheme is to propose a fast-to-calculate measure that gives an upper bound on the candidate tree size *before* actually calculating the tree (using any of the algorithms in Section 5.1). We use this upper bound as the score to rank the roots and decide which one to evaluate next. The higher the score, the more likely the root is selected for actual calculation.

This upper bound measure basically answers the following question: given $G = (V, E)$, root $r \in V$ and budget B, what's maximum number of nodes that a feasible solution of Problem 4 can cover?

We define the *minimum in-edge* of $u \in V$ as

$$e^*(G, u) = \arg\min_{e' \in \delta^+(G,u)} c(e'),$$

where $\delta^+(G, u) = \{e \in G.E \mid e.i = u\}$.

Ranked minimum in-edges of $V^{'} \subseteq G.V$ is defined as:

$$RMIE(G, V^{'}) = \left\{e^*(G, v_1), \ldots, e^*(G, v_{|G.V|})\right\},$$

where $v_1, \ldots, v_{|V'|} \in V^{'}$ and $c(e^*(G, v_i)) \leq c(e^*(G, v_{i+1}))$. Note that $RMIE(H, V^{'})$ is an ordered set. The top-k elements in $RMIE(G, V^{'})$ is $RMIE(G, V^{'}, k)$.

Given B, the *budgeted ranked minimum in-edges* is defined as:

$$B\text{-}RMIE\ (G, V^{'}, B) = RMIE(G, V^{'}, k),$$

where k satisfies:

$$\sum_{i=1,\ldots,k} c(e^{*}(G, v_i)) \leq B \text{ and } \sum_{i=1,\ldots,k+1} c(e^{*}(G, v_i)) > B.$$

Denote all feasible solutions of Problem 4 as $\mathcal{T}(G, r, B)$, then we have the following:

Proposition 3 $\forall T \in \mathcal{T}(G, r, B), |T.V| \leq |B\text{-}RMIE\ (G, G.V - \{r\}, B)| + 1$

Proof: assume there is a feasible solution $T^{'}$ such that its size is greater than the upper bound: $\left|T^{'}\right| > |B\text{-}RMIE\ (G, B)| + 1$. Then according to the definition of $B\text{-}RMIE$, $c(T^{'}) > B$, which is a contradiction.

As a conclusion, our root sampling strategy first ranks all the vertices by the upper bound (defined by $B\text{-}RMIE$) in descending order. Then it sequentially selects vertices from this list.

6 Experimental evaluation

We evaluate our methods on both one synthetic dataset and 3 types of real datasets, *letter*, *email* and *tweet*. Because of insufficient ground truth data, we mainly use synthetic datasets as the ground truth to evaluate different algorithms' performance. We also evaluate the results on real datasets both qualitatively and quantitatively.

6.1 Datasets and preprocessing

Synthetic data. We generate synthetic datasets in two steps: (1) we generate ground truth event trees; (2) we inject noise interactions. For event generation, we use the tree generation model by Kumar et al. [KMM10]. Initially, the event topic vector $\widetilde{\alpha}$ is generated by randomly setting one of its elements to 1 and the rest to 0. At each iteration, a new interaction, $i = (u, V, \alpha, r)$ is created and linked randomly to one of the nodes in the constructed tree. u and V are randomly selected from a global participant set under the interaction meta-graph requirement (Section 3). We consider the case where there is only one recipient for each interaction: $|V| = 1$. α is generated by adding small amount of random noise to $\widetilde{\alpha}$. For t, we add a small time difference to the timestamp of the interaction that i connects to.

Noise interactions are independently generated by randomly and independently sampling a topic vector, a sender, a recipient and a timestamp.

Real-world data. We evaluated our methods on four real-worlds datasets (statistics given in Table 6.1):

English letter: named as Corpus of Early English Correspondence Extension (CEECE). [4]. This dataset contains 4945 personal letters by 315 unique persons from 1620 to 1800. Each letter maps to an interaction in our model.

Enron: the original dataset contains the email communication records of Enron Corporation. In this experiment, we use a preprocessed version of the original one[5]. This version contains 882 unique messages sent by 133 email addresses during 1999-9-6 to 2002-2-13. Each email maps to an interaction in our model.

Tweets by hashtag: we use three sub datasets, in each of which contains only tweets that contain a specific hashtag. The hashtag for each sub dataset is *#beefban*, *#baltimore* and *#ukraine* respectively. For example, in *#beefban*, all tweets contain

[4] http://ota.ox.ac.uk/desc/2510 Spelling variations have been normalized to modern English.
[5] http://bailando.sims.berkeley.edu/enron_email.html

hashtag #beefban. We include only tweets that contain at least one *mention* [6], which serve as the recipients. Each tweet is considered an interaction.

#beefban contains mainly tweets about Indian president signing the bill to ban beef in state Maharashtra[7]. *#baltimore* is mainly about Baltimore protests[8] in 2015. *#ukraine* is on the Ukrainian crisis [9] and its political relation with Russian Federation. All three datasets are provided by Garimella et al. [GDFMGM16].

Sports tweets: We tracked live Twitter stream on a UEAF football game between Real Madrid and Manchester City (*football*) [10]. The tracking aligns with the start and finish of game, including the intermission period. This dataset differs from the previous one on Twitter as it serves as the ground truth for evaluation.

Table 6.1: Network statistics on real datasets. For interaction network, both sender and recipients are considered nodes. Singleton interactions in the interaction meta-graph are removed.

Datasets	**Interaction networks**		**Interaction meta-graphs**		
	#nodes	#edges	#nodes	#edges	Period
Letter	723	820	3720	6745	1680 - 1800
Enron	1144	2106	812	21297	1998-10-30 - 2002-02-13
#beefban	11895	33584	26317	75870	2015-03-03 - 2015-03-05
#ukraine	16218	59096	46540	142746	2015-02-27 - 2015-03-03
#baltimore	38541	102139	61501	132012	2015-04-26 - 2015-04-28
football	76205	103592	16035	17261	2016-05-04 (21:00 - 22:59)

Preprocessing. We observe the phenomenon that the same person sends the same (or very similar) messages multiple times (e.g, >100). This phenomenon is especially remarkable for controversial topics such as Indian beef ban and Baltimore riots, for which certain people tend to repeatedly state their stances using the same messages.

Our methods are easily misled by the sheer amount of redundant messages. To avoid this problem, we merge similar messages of the same sender into one. We consider

[6] https://support.Twitter.com/articles/14023

[7] http://indianexpress.com/article/india/india-others/beef-banned-in-maharashtra-5-yrs-jail-rs10000-fine-for-possession-or-sale/

[8] https://en.wikipedia.org/wiki/2015_Baltimore_protests

[9] https://en.wikipedia.org/wiki/Ukrainian_crisis

[10] http://www.uefa.com/uefachampionsleague/season=2016/matches/round=2000637/match=2015785/index.html

two messages *similar* if (1) they are sent by the same user, (2) the Levenshtein edit distance [BM00] between their content is below some given threshold (we use 10%) (3) their time distance is relatively small (e.g., one day). In the newly-merged message, the text content and timestamp are copied from the earliest message. Then, recipients are the conjunction of all original messages' recipients.

We take different approaches for vectorizing interaction content in emails/letters and tweets. For emails/letter, whose body are usually much longer than tweets, we train topic model using *Mallet*[11]. We then assign a topic vector for each node in G and use cosine distance to compute edge weight.

Measuring tweet similarity is an open challenge due to its short length, conciseness and spelling variations. We took an ensemble approach where the vector representation comes from several models. Besides topic vectors, we use also (i) bag-of-word (BoW) with tf-idf re-weighting and (ii) hashtags collected for each tweet. For BoW and hashtag representations, we use cosine and Jaccard distance for weight assignment respectively. Last, we sum up the three distances in a weighted manner. The weights for topic vector, BoW and hashtag are 0.4, 0.4, 0.2 respectively for tweet-related experiments.

For topic modeling, for *Enron*, *Letter* and tweet-related datasets, we use 50 topics, run it for 200 iterations.

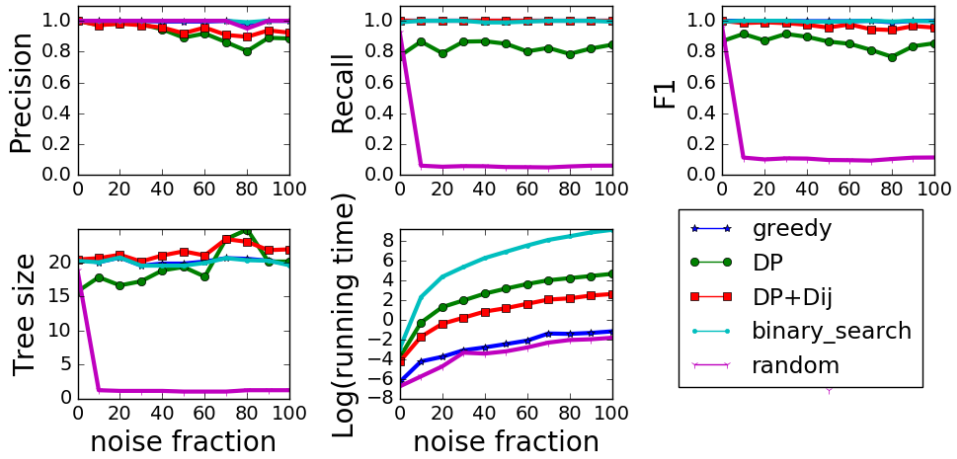
6.2 Evaluation on synthetic datasets

We evaluate five different algorithms for finding the best event (Problem 4): (1) greedy tree growing (*greedy*), (2) binary search using Charikar's DSP algorithm (*binary_search*), (3) dynamic programming without Dijkstra preprocessing (*DP*), (4) dynamic programming with Dijkstra preprocessing (*DP+dij*), and (5) random tree growing (*random*) as a baseline. The *random* algorithm mimics the *greedy* excepts that it selects a random edge to grow at each step.

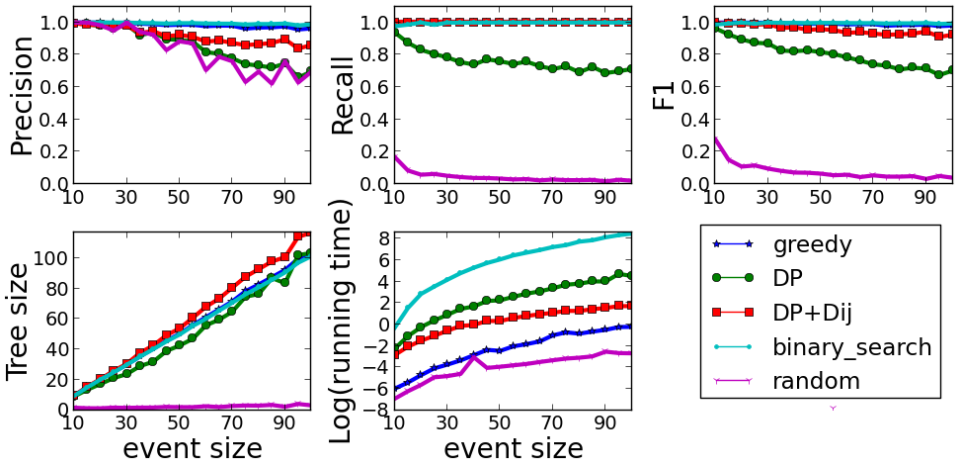
We compare the performance of the algorithms under (1) different noise levels and (2) different event sizes. *Noise level* is defined as the ratio of the number of noise interactions over the total number of interactions of ground truth events. For example, if the noise level is 10.0 with one event of size 10, there are 100 noise interactions. Event size is the number of nodes in the ground truth event tree.

[11] http://mallet.cs.umass.edu/topics.php

For the DSP algorithm we set $\ell = 1$, as we empirically encounter insufficient memory for larger values.



(a)



(b)

Figure 6.1: (a) Performance of algorithms under different noise levels from 0 to 100 at step size 0.5. Results are averaged over 50 experiments. (b) Performance of algorithms under different event sizes from 10 to 100 at step size 10. Noise level is fixed to 20.0. Measurement values are averaged from 50 experiments.

Different noise levels. To compare the capability of the algorithms in finding *one* best event, we generate a sequence of datasets with increasing noise levels and only one event of size 20 (containing 20 nodes). To solve Problem 4, we provide as input the ground truth values of I, B, r. We consider three types of measurements: (1) precision, recall, and F1 over the ground truth even interactions and the extracted

interactions by the algorithms, (2) the objective function value of Problem 4, and (3) running time. Log scale is applied to running time as we observe magnitudes of difference among the algorithms.

In Figure 6.1 (a), we see that all our algorithms beat the trivial *random* baseline. Though *greedy* is a simple heuristic, its performance is among the top. Dijkstra preprocessing proves helpful for *DP* in both improving F1 and saving computation time. *binary_search* seems to be unnecessary as it consumes much more time, even though it is among the best in other measurements. Notice that *random* achieves high precision because it easily selects a wrong edge that violates the budget constraint at the first few steps, causing the algorithm to terminate.

Different event sizes. We also study how the algorithms compare in extracting events of different sizes. The experiment setting is similar to the above except that noise level is held fixed to 20 while the event size varies. In Figure 6.1 (b), *greedy*, *binary_search* are among the best in terms of precision, recall, F1 and set cover objective, whereas *DP+dij* is slightly worse. Again, preprocessing for *DP* proves useful. Running time comparison gives consistent results to the previous case.

6.3 Parameter effects on real datasets

We evaluate the effects of (1) different B values using the algorithms of Problem 4 (2) different sampling schemes on the objective functions in Problem 4 and Problem 5. In addition, we demonstrate the difference of event trees produced by different algorithms of Problem 4.

Effect of B. We evaluate the effect of topic dissimilarity budget B on the tree size objective in Problem 4. We randomly sample 100 roots for each dataset. B varies from 0 to 100 at a step size of 5.0. Time budget I is set to 1 day and 4 weeks for Twitter-related datasets and *Enron* respectively. We take the median size of all trees returned by each algorithm (Figure 6.2).

In Enron, we observe a converging effect on both objectives as the dataset is relatively small, while this is not the case in all Twitter datasets. In practice, *greedy* is the best performing algorithm, as it is both competitive in maximizing the objective function and it is computationally efficient.

Sampling scheme comparison. We compare two sampling schemes in real data setting: (1) random root sampling (*random*) as the baseline, (2) ranking roots by

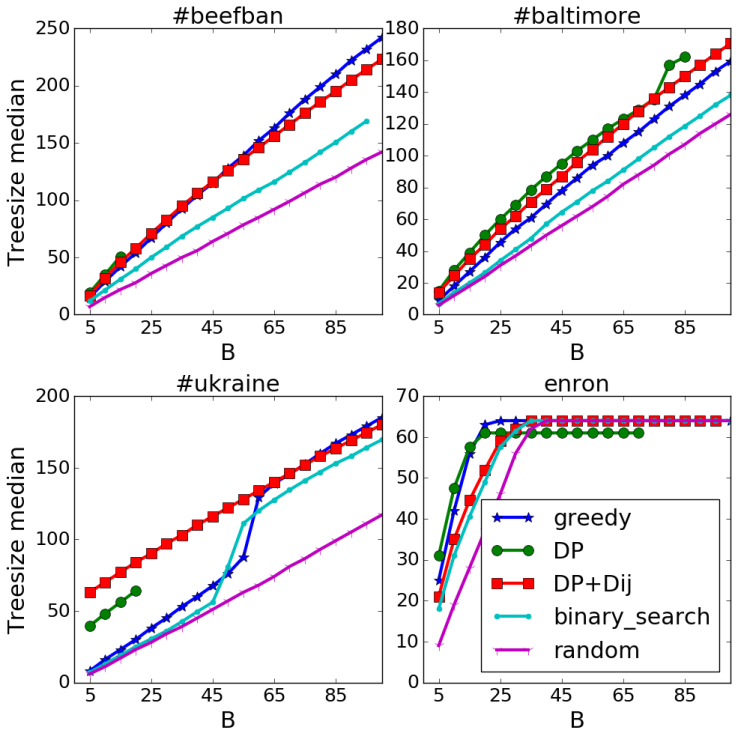


Figure 6.2: Effects of B on the median of tree sizes using different algorithms. For #ukraine, DP fails to complete experiments with $B > 25$ where it consumes excessive memory.

event size upperbound defined by B-$RMIE$ in Section 5. For each scheme, the set cover objective is stored at each iteration.

As we can see in Figure 6.3, the event size upper-bound heuristic helps to discover better solutions, especially for #baltimore and Enron.

Trees by different algorithms. We compare the behaviours of the algorithms for Problem 4 in real-world datasets. In Figure 6.4, the trees are produced by *greedy*, and *DP+dij* are given the same root and budget. The *greedy* algorithm avoids selecting heavy edges with weights larger than 0.8 due to its local search strategy whereas *DP+dij* achieves larger tree by selecting a few heavy edges. Therefore we expect *greedy* to produce more topically-coherent events as the pairwise dissimilarity between nodes tend to be smaller.

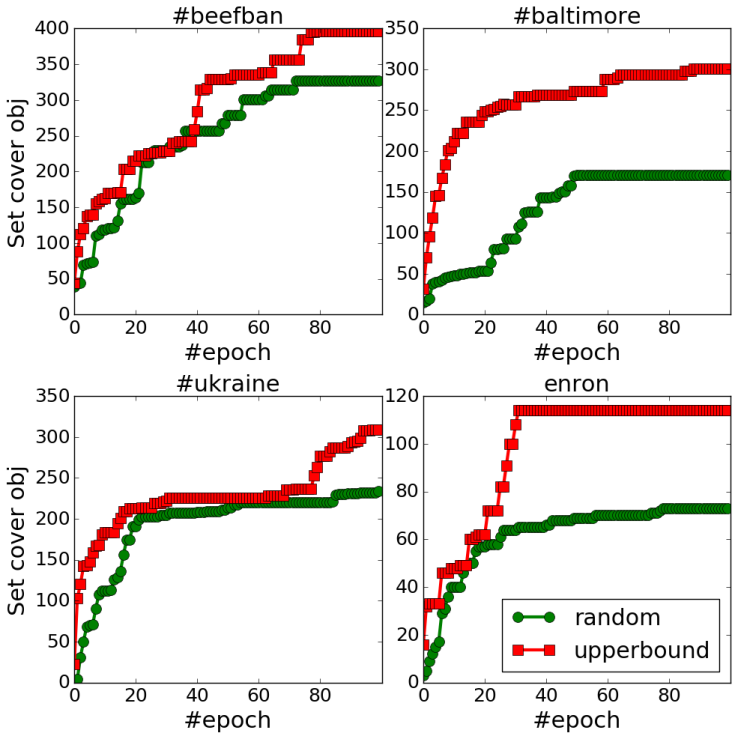


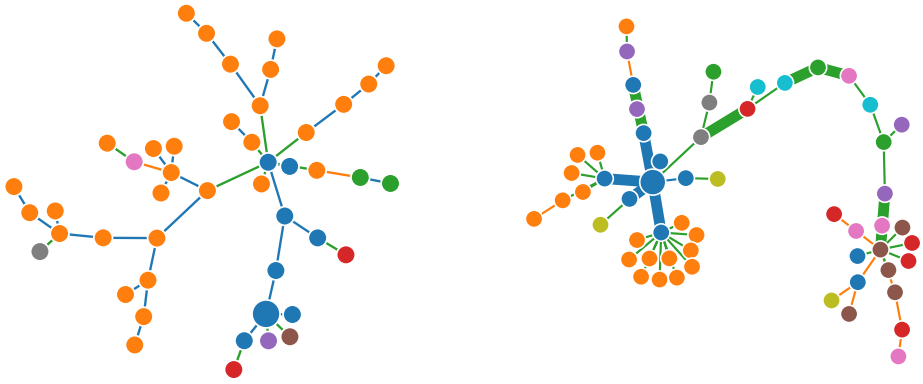
Figure 6.3: Performance of different sampling schemes on real datasets. For Twitter, $B = 15.0, I = 1$ day. For Enron, $B = 10.0, I = 4$ weeks. 100 unique roots are selected based on the sampling scheme. *greedy* is used. Top-10 events are selected.

6.4 Evaluation on sports tweet dataset

For *football* dataset, we obtain the ground truth events from ESPN[12]. Each ground truth event (different from our "event" definition) has a timestamp, a label from one of the 4 categories: *substitution*, *yellow card*, *red card* or *goal*, and information about the main participants (for example, who made a goal).

We mark a detected event as correct if it aligns with any of the ground truth events in both time and topic terms (to the event label). The result is given in Table 6.2. We observe two of the true events are detected: one substitution and one goal (the only one). Besides the event about goal, which is naturally the most important one, we highlight the detected substitution. In the real event, captain of Manchester City, Vincent Kompany, got injured and was substituted off. In fact, this injury was considered a turning point for the team and was heatedly discussed on Twitter. We also observe mixture of topics in the detected events. For example, the 4th event contains tweet corresponding to the yellow card at 83' and comments on the

[12]http://www.espnfc.us/gamecast/447230/gamecast.html



(a) (b)

Figure 6.4: Algorithm comparison in terms of tree shapes. Trees are computed from #beefban given fixed root by *greedy* (a) and *DP+dij* (b), which achieves tree size 46 and 57 respectively. $B = 30$ and $I = 1$ day are the same for both algorithms. Edges with weight ≥ 0.8 have wider stroke. No edges with weight ≥ 0.8 are selected by *greedy*. Nodes are colored by senders and edges are colored by its type (broadcast: blue, relay: green, reply: orange)

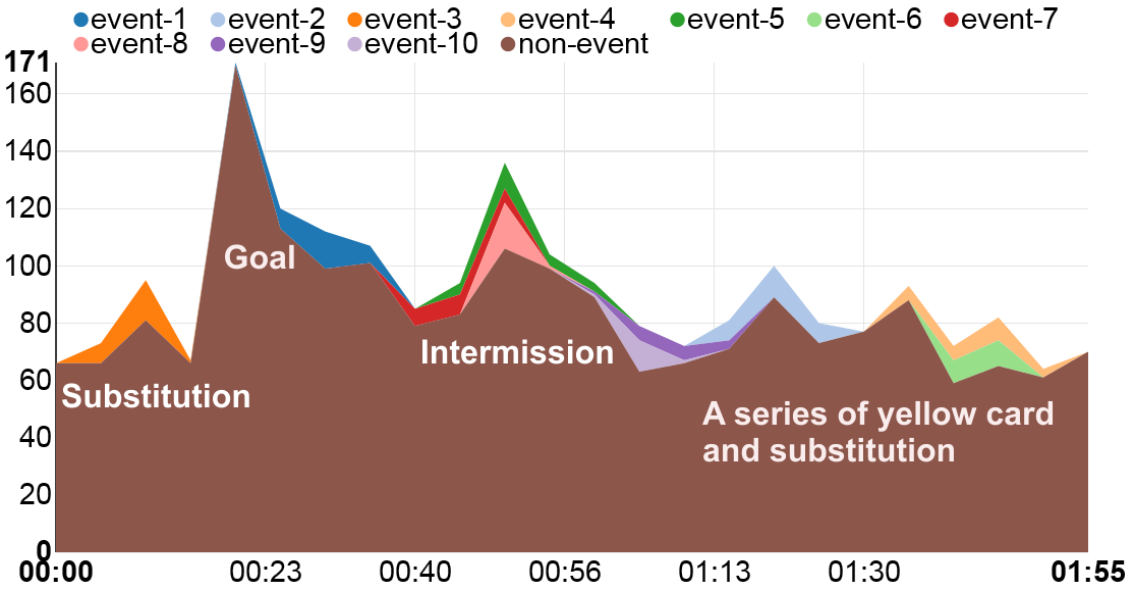


Figure 6.5: Tweet volume against time for *football* dataset. Both the first goal and substitution are detected, while the events by the end of the game are not. During the intermission, heated discussion (event 5, 7, 8) are observed.

previous goal.

In Figure 6.5, we observe that during the intermission there is heated discussion

Table 6.2: Evaluation result for *football* dataset. Successful detected events are commented with the relevant terms. The top-10 detected events are used for evaluation. *mcfc* means "Manchester City Football Club".

Time	Ground truth event	Detected?	Comment
10'	Substitution	✓	*kompany*, *injured*
20'	Goal	✓	*mcfc*, *goal*, *bale*
30'	Yellow card	✗	
55'	Substitution	✗	
61'	Substitution	✗	mixed in event 2
69'	Substitution	✗	
71'	Yellow card	✗	
83'	Yellow card	✗	mixed in event 4
88'	Substitution	✗	
90'	Yellow card	✗	

covering many aspects of the game, such as game status report and detailed statistics of the first half. Also we found the detected events at the tail of the game aligns well with the ground truth in time. However, as those true events are close in time (seen in Table 6.2), our methods are not able to distinguish them apart.

6.5 Case study in English Letter

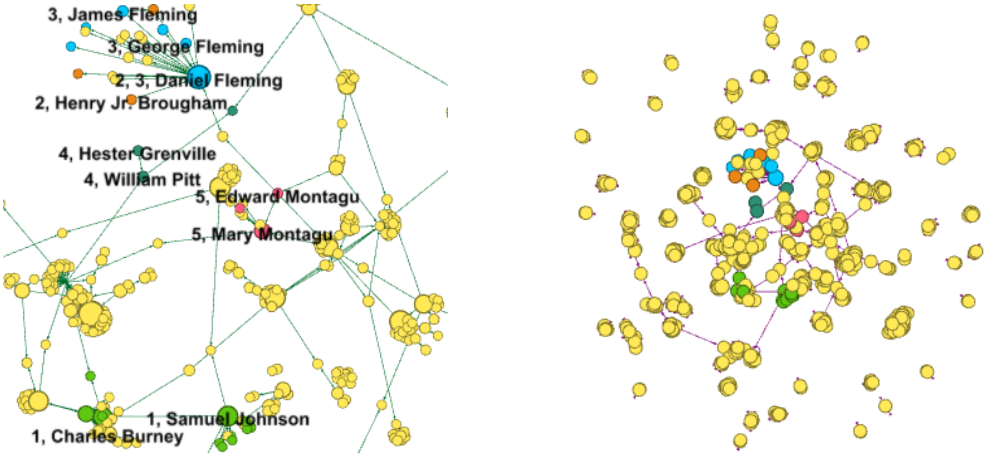
We sample 1000 nodes using *upperbound* scheme and applied *greedy* algorithm with $B = 20, I = 1$ year. Besides the requirements in Section 3, two letters are connected if the time difference is smaller than 90 days. The intuition is two letters are not very likely to relate to each other if they span a long time period.

For the extracted events, we observe difficulty in interpreting the events based solely on terms, therefore we summarize the events by manually checking the actual document content. The extracted events with manual summarization is given in Table 6.3.

For example, the 1st detected event centred around Samuel Johnson, an English writer. The period stars from the beginning of 1784 to the end of this year. Interestingly, Johnson died in December of the same year and one aspect of the event is Johnson's description about his health conditions. However, we observe mixing

Table 6.3: Top-5 events from *Letter* dataset: description is manually compiled.

Id	Main people	Period	Description
1	Samuel Johnson, Charles Burney	1784	Johnson talks about book publishing and his health condition
2	Daniel Fleming, his son, James and his cousin, Henry Brougham	1694 - 1695	Fleming's ask his cousin to help on his sons' education
3	Daniel Fleming, his sons, George and James	1696 - 1697	George's tuition fee for school and James's acceptance into Oxford
4	William Pitt and his future wife Hester Grenville	1754 - 1754	Love letters
5	Mary Montagu, Edward Montagu and Philippa Massingberd	1711 - 1712	Mary's courtship by Edward and Philippa



(a) (b)

Figure 6.6: (a) Part of the interaction network covering the participants of main events. Nodes are colored by the event ids and labeled by the participant names. (b) The whole interaction network.

of topics, where Johnson also had discussion about book writing with his acquaintances. We also noticed that discussion by Fleming family on the children's education are prevalent in the final result (the 2nd and 3rd events). Note that the King

of Britain actually had letters in this dataset, however, according to our problem formulation, those letters are not selected as events.

As an interesting by-product, Figure 6.6 gives the corresponding interaction network with the event participants highlighted.

6.6 Case study in Enron dataset

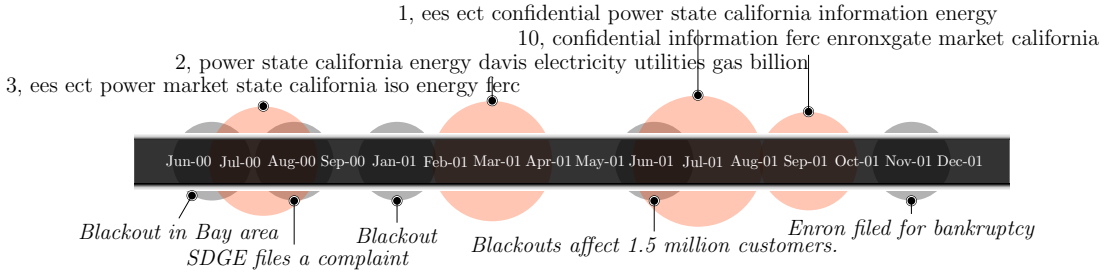


Figure 6.7: Timeline with extracted events (larger red circle) and publicly recognised events (smaller grey circle with italic label) for *Enron*: events on Enron's energy scandal and bankruptcy are highlighted. Event 3, 2, 1 and 10 are displayed. The larger the circle, the larger the event size. For each event, top topic terms are displayed. *ees*: Enron Energy Service, *ect*: Enron Capital and Trade Resources, *iso*: California Independent System Operator, *ferc*: Federal Energy Regulatory Commission. *SDGE*: San Diego Gas & Electri Company.

We sample 50 nodes using *upperbound* scheme and applied *greedy* algorithm with $B = 10$ and $I = 28$ days. First, we observed that the events can be grouped into two topics: (1) California Energy Crisis,[13] (2) investigation into Enron's scandal.[14] In Figure 6.7, we anchored the actual important events (gray circle with italic labels) about the crisis happening during the timespan of the dataset. We found shortly after each major blackout, there is at least one extracted events about the blackout. And before Enron filed bankruptcy, Federal Energy Regulatory Commission (FERC) investigated Enron's illegal operations, which corresponds to the 10th event in the figure. Second, in Figure 6.8, extracted events tend to occur at the peak of the volume plot. Last, we noticed the address `steven.kean@enron.com` (Enron's former chief of staff[15]) sends more than 90% of the emails in the top-5 events. However,

[13]https://en.wikipedia.org/wiki/California_electricity_crisis
[14]https://en.wikipedia.org/wiki/Enron_scandal
[15]http://www.nbcnews.com/id/3606477/ns/business-corporate_scandals/t/lay-skilling-linked-

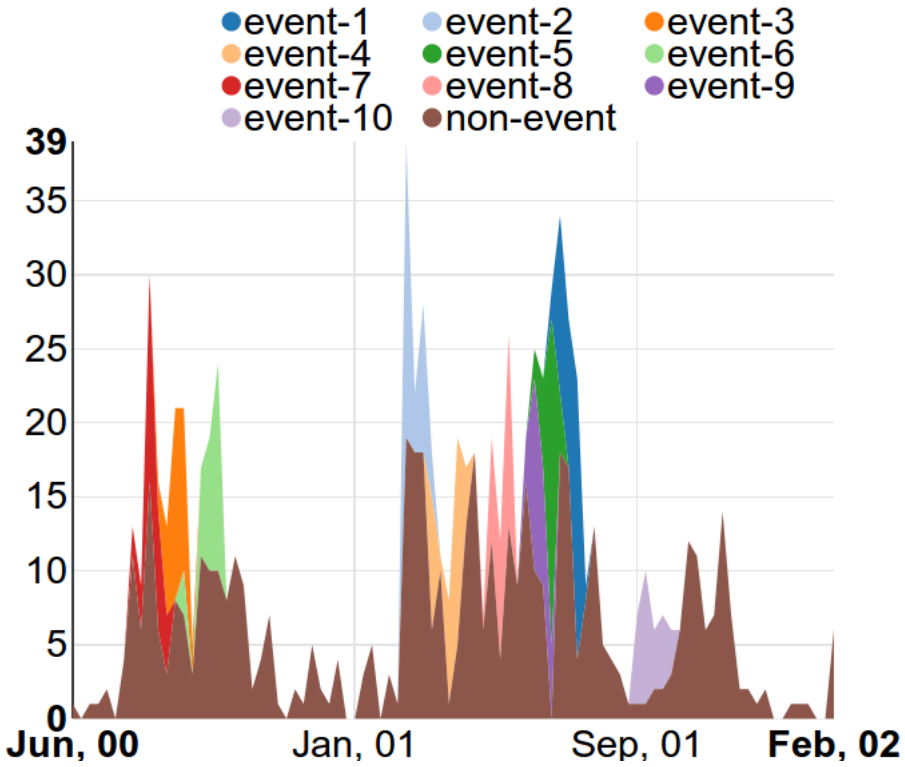


Figure 6.8: Stacked area graph of interaction frequency against time for *Enron*. Top-10 events are visualized.

this is understandable as the same address sends around 55% of emails for the whole dataset.

6.7 Case study in Twitter datasets

We use the same parameters for all three datasets as they have similar size and timespan. Events are extracted by selecting 100 roots using *upperbound* and using *greedy* algorithm with $B = 50$ and $I = 1$ day.

#ukraine. Ukraine crisis arouses media war on Ukraine and Russia.[16] We observe some detected events align well topically and temporally with the actual events in Figure 6.9. We also detected other related events such as argument and discussions on other issues. However, topics are mixed inside events. For example, topics on both freeing Ukrainian pilot and march in memory of murdered Boris Nemstsov are

enron-failure

[16]https://en.wikipedia.org/wiki/Ukrainian_crisis. This dataset spans the time when two important event happens: 1) the continuing detention of Ukrainian pilot Nadiya Savchenko by Russian government, 2) the murder of Boris Nemtsov in Moscow

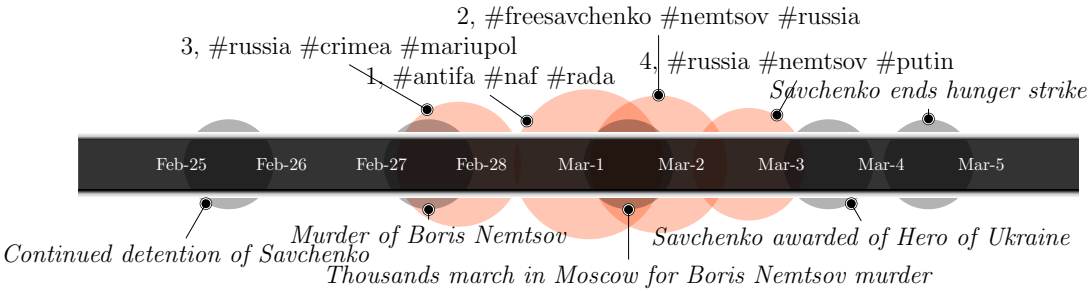


Figure 6.9: Timeline for #ukraine. Top-4 events are displayed with the top-3 hashtags by frequency. Event 2 and 4 map to the murder of Boris Nemstsov (#nemstsov), while event 4 also contains tweets on freeing the Ukrainian pilot, Savchenko (#freesavchenko). Event 1, 3 talks about other related political issues (#antifa: Anti-fascism, #crimea: the annexation of Crimea).

detected in event 2. This is expected due to the local similarity measurement in Problem 4.

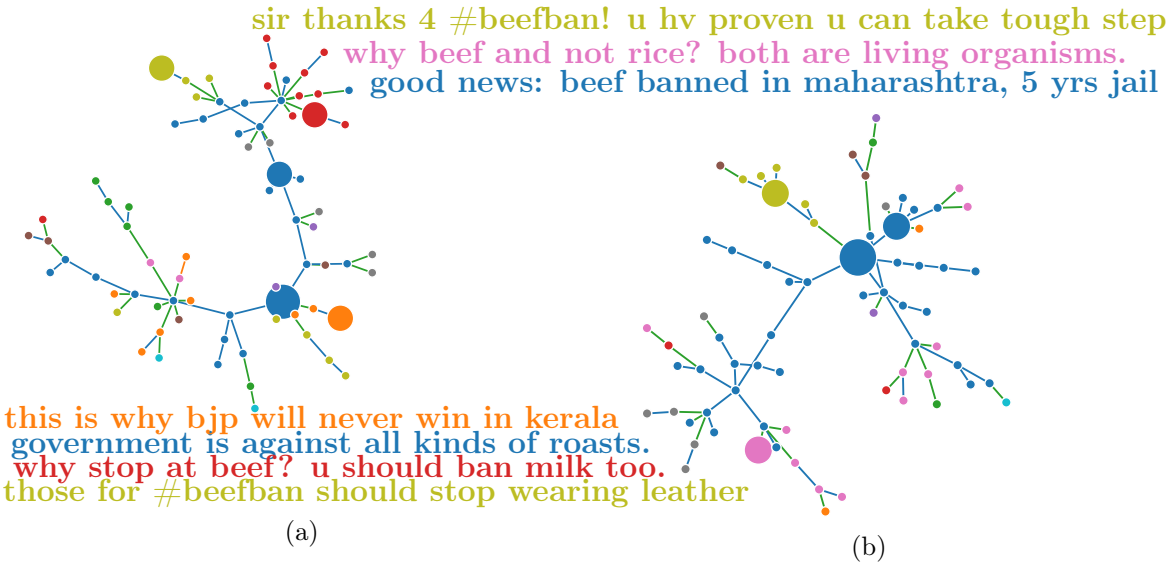


Figure 6.10: 1st and 2nd event tree extracted from Indian #beefban. Nodes (the tweets/interactions) are colored by the sender. The largest node is the root. Example tweets are displayed and colored by the sender and the corresponding tree node (interaction) is enlarged. (a) demonstrates sign of opinion propagation among different users. (b) contains mixing of support and opposition, though support is more dominant.

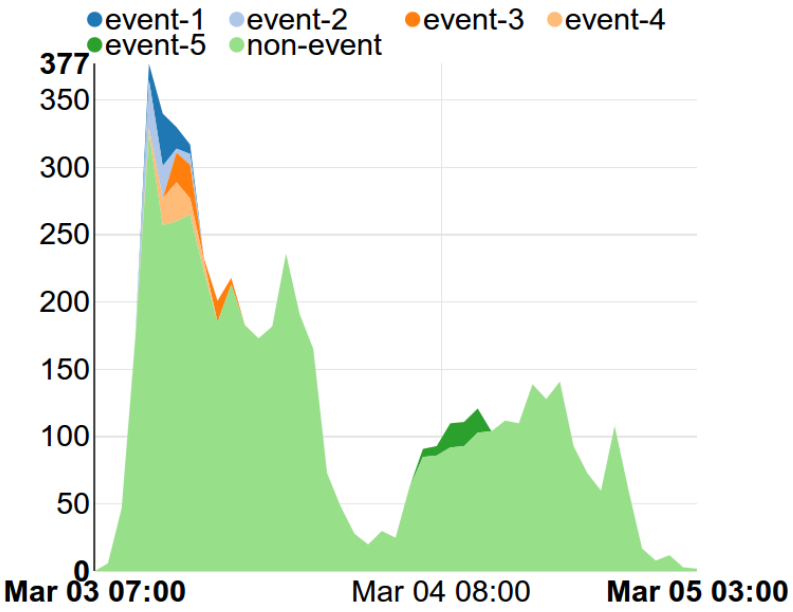


Figure 6.11: Stacked area graph of interaction frequency against time for #beefban. Top-5 events are displayed.

#beefban. India's new law on banning on beef is a controversial topic [GDFMGM16]. [17] Results demonstrate clear separation of opinions among events. For example in Figure 6.10, the 1st and 2nd event represents opinions opposing and supporting the law. However, we are not able to interpret any temporal pattern in the extracted events due to the short timespan (3 days).

We observe the following interesting phenomenon. First, one event (in Figure 6.10 (a)) display evidence of information propagation. For example, opposing opinions spreads along the user network and affected users also express their objection. Second, another event (in Figure 6.10 (b)), there is one dominant user who sent more than half of the tweets. This indicates there are often some enthusiastic who express their opinions frequently for controversial topics. Third, we observe events with mixed opinions (Figure 6.10 (b)), which contains argument between users with distinct opinions. Last, our method tends to discover events at the "peak" as the set cover objective is better than the "bottom" shown in Figure 6.11.

#baltimore. We discovered two types of events. Certain events contain more emotional tweets showing anger towards the riot, while other events are more descriptive (such as news), they tend to report the current situation (Table 6.4).

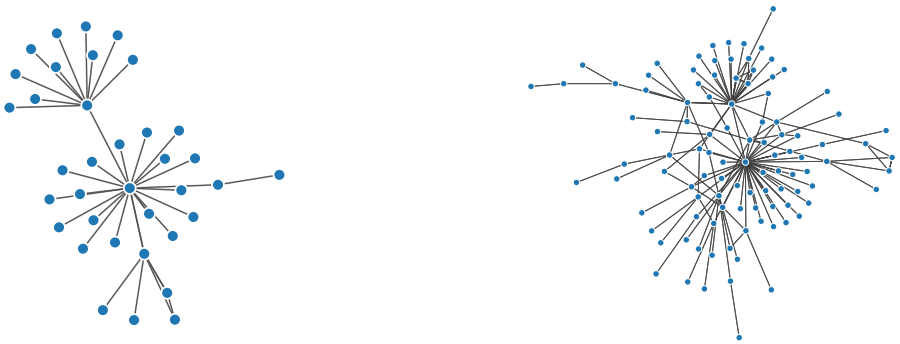
Compared to #beefban, information relay tends to be more extensive as shown in

[17] http://indianexpress.com/article/explained/explained-no-beef-nation/

Table 6.4: Emotional and descriptive events for #baltimore: tweets show different types of information in different events.

Emotional (1st event)
the lawlessness in #baltimoreriots is definit...
maybe white people should just start burning...
people are justifying the violent looting of ...
only one? #baltimoreriots
alright. don't come crying to me if you get ...
Descriptive (3rd event)
smashing windows at an office and subway and...
#mondawminmall mall now being looted. no polic...
local hospitals have received about 15 injured...
#breaking: crews battling a fire at a cvs...
people are driving up to and running in to ...

Figure 6.12. We suspect because of the protest's high burstiness, people are more likely to share status update or spread their opinions. Meanwhile, we didn't observe the sign of dominant speakers for each events, which is not the case in #beefban.



(a) (b)

Figure 6.12: Participant graph of the 1st event extracted from #beefban (a) and #baltimore (b), where nodes are Twitter users and there is a edge between two users if they interact with each other. Though the corresponding event in (a) and (b) have similar sizes(41 and 48), they show different participant graphs.

7 Discussion

Though meaningful events can be detected at the macro level, we empirically observed several problems in the micro level. These problems are related to (1) how we formulate the problem and (2) how we construct the interaction meta-graph. In this section, we show concrete examples to illustrate those problems and suggest possible improvements. We use results from *Letter* dataset as a demonstration. Note that the problems described in the following also apply to other datasets.

7.1 Problem formulation

We observe two major problems in the *Letter* dataset: *mixed topics* and *broken stories*. For mixed topics, two different topics occur in the same event. For broken stories, two detected events can actually be merged into one. These two observations suggest future improvement for our problem formulation.

7.1.1 Mixed topics

Table 7.1 shows two different topics expressed in the 1st event of the *Letter* dataset. In this case, letters of distinct topics are in the same event for two reasons: First, they are all sent by the same person, Samuel Johnson and within a relatively short time (e.g, 1 year). Thus they are connected in the interaction meta-graph. Second, the way we model topic constraint is two-fold. We assign pair-wise topic distance between two letters to their connecting edge and a event is topically-coherent if the sum of its edge weights is smaller than given threshold B. Hence, when solving Problem 4, off-topics letters can still be included to the current event (even if the edge cost is large) as long as (i) it's connected to the main event (ii) there is extra budget to do so.

This suggests two problems on the way of modeling topic divergence. First, local pair-wise topic distance introduces mixed topic. Even if the drift is small individually, a long path that connect many letters will accumulate the difference. In this case, two ends of the long chain can talk about totally different topics. Second, edge weight sum is inappropriate for measuring topic divergence. As is shown above, off-topic nodes can be added as long as there is enough budget.

Table 7.1: Two different topics, health and book writing, observed in 1st event from *Letter* dataset: typical sentences for each topic from different letters are manually selected. All sentences are written by the same author.

Topic 1: health
My breast is now covered with a blister.
The asthma has been for some time very considerably relieved.
My breath is tolerably easy, and since the remission of asthma...
Topic 2: book writing
I know not in what state Dr. Edwards left his book.
I had ceased to write because, respecting you I had no more to say, and respecting myself could say little good.
I suppose no man but himself, could assign all the parts of the ancient universal history to their proper authors.

7.1.2 Broken stories

A broken story refers to the case where multiple small events can actually be merged into a larger, more complete and more intuitive one. For example, event 2 and 3 in Table 6.3 can be merged because: (1) they are temporally close (event 2 is from Nov, 1694 to Nov, 1695, event 3 is from Mar, 1696 to Mar, 1697). (2) they are topically coherent as they are both about Danial Fleming's concern on his sons' education. Note that extracted events need to satisfy the time constraint (1 year). Thus, both event 2 and 3 span almost (and shorter than) 1 year. As a result, the natural 2-year event is split into two 1-year events by our method.

One straight-forward fix for this problem is increasing I. However, assuming one event with shorter time span and lower topic divergence (relative to the given budget), this fix will include off-topic interactions if there is any. Therefore, we argue that the problem with the time constraint is jointly produced by the topic constraint formulation.

7.1.3 Alternative problem formulations

Based on the above observations, we provide a few alternatives for problem formulation. All alternatives have the same objective function as in Problem 4, however with different constraints.

One way to deal with mixed topic problem is by defining topic coherence based on topic centroid, the mean of topic vectors of event interactions. For example, we can constrain $T = (V_e, E_e)$ with $\frac{1}{|V_e|} \sum_{i \in V_e} c(i.\alpha, \widetilde{\alpha}(V_e)) \leq I$, where the centroid topic $\widetilde{\alpha}(V_e) = \frac{1}{|V_e|} \sum_{i \in V_e} i.\alpha$.

By leveraging the topic centroid, event topic is constrained on the global scale, in contrast to local approach in our current formulation.

On the other hand, mixed topic is a necessary by-product of information propagation, especially in the era of online social network where information exchange is rapid. In other words, we cannot totally get rid of mixed topic if one of our goals is to model information flow.

In the second alternative, we keep the pair-wise edge cost but apply edge weight threshold on *each* edge of the event. More formally, the constraint is: $c(i, j) \leq I$, for $(i, j) \in E_e$. In this way, off-topic interactions (connected by large weight edges) are less likely to be included.

We find this problem optimally solvable by modifying shortest-path problem and Dijkstra algorithm. In the modified problem, the shortest path in the original problem maps to the path in which the weight corresponding to heaviest edge is minimized. Meanwhile, the modified algorithm updates the distance table using the new "shortest path" definition.

For both options, dropping time constraint is reasonable. First, different events have different time span. A fixed maximum time span doesn't apply to all events.

Second, it's intuitive to think that topic coherence implies temporal coherence. Usually, people talk about certain topics only during a certain period. Thus, the two above formulations can automatically narrow down the time span using the topic information.

7.2 Interaction meta-graph construction

During the interaction meta-graph construction process, we use topic dissimilarity as edge weight aiming to model topic coherence as well as information flow. In the ideal case, our method should be able to extract topically coherent events and meanwhile reconstruct information flow in the event tree. However in practice, we fail to consistently observe meaningful information flow patterns in the extracted events.

7.2.1 Similarity = causality?

Information flow can be treated as causality relationship among the interactions. For example, two interactions that correspond to a reply relationship should get connected if the information flow is correctly reconstructed. However in practice, we observe the wrong connection between interactions in the extracted event, which implies similarity doesn't necessarily imply causality.

For example in Figure 7.1 letter B is actually a reply to letter C. However, A incorrectly gets connected to B in the final event. The correct one is not selected because its edge weight is larger than that of A to B.

One way to address this issue is assigning better edge weights. Beside topic dissimilarity, other factors, such as time difference between the two interactions, can be considered. In addition, as demonstrated by the letter content in Figure 7.1, deeper text analysis can be applied to infer whether a letter is a reply or relay to another one. This problem relates to a broader category, network reconstruction, which is studied in various domains [GRLK12, SGR14, LV12, WH14].

7.2.2 *Self-talking* problem

For various datasets, we observe interactions sent by the *same* participant get connected in the final events. We refer this as *self-talking* problem. For example in Figure 7.2, the majority of the messages are sent by Samuel Johnson. Meanwhile, one cannot easily interpret the links in the event tree. Those interactions get linked simply because they share the same sender and they are sent during a short period. Again, the problem is related to the way we construct the interaction meta-graph.

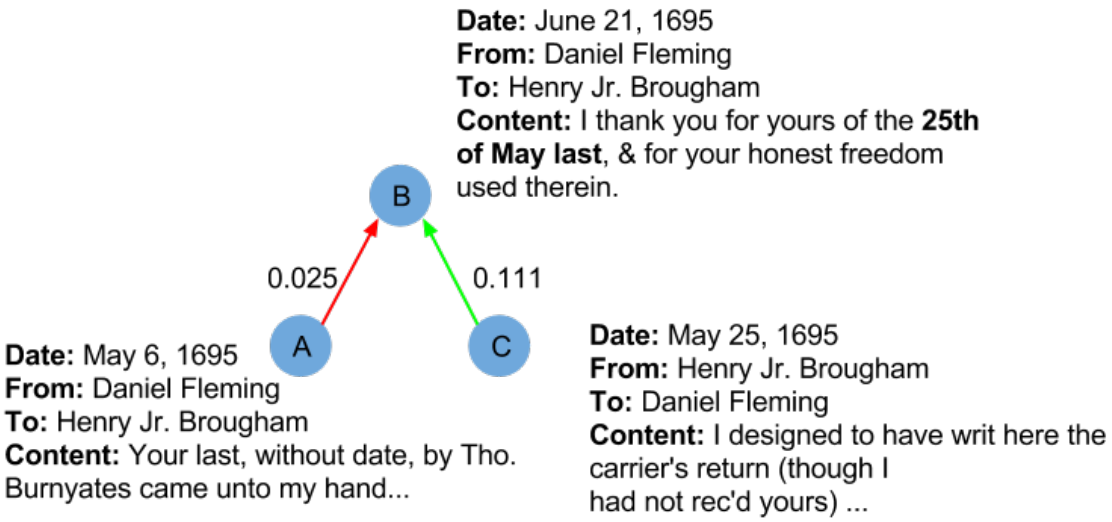


Figure 7.1: Example of incorrect interaction meta-graph construction: three letters are displayed. Letter A gets connected to B incorrectly by our method. However, by inferring from the letter content, the correct edge should be C to B. Edge weights are also displayed.

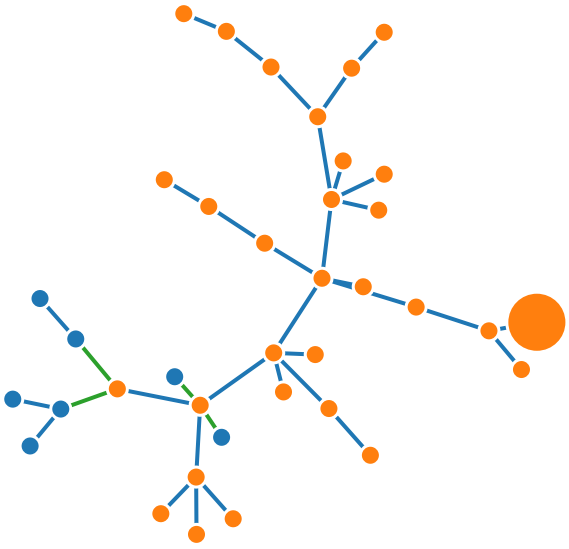


Figure 7.2: Example of self-talking problem: most the messages are sent by the same person. Event tree of 1st-event in *Letter* dataset is displayed, where nodes are colored by the sender. The orange node corresponds to Samuel Johnson.

7.3 Other interesting directions

One interesting direction is assigning different importance scores to messages. Currently, our method assumes interactions are of equal importance, which is not always

true. For example, we find there are letters written by the King of United Kingdom. Intuitively, letters related to him should be more interesting because of his high social rank. Also note in Figure 7.3, some "important" nodes with large size (corresponding to large PageRank [PBMW99] scores) are not selected in the main event.

As another example, tweets are of different importance as they receive different amounts of attention, reflected by the number of likes or retweets. It's reasonable to prefer tweets with larger like/retweet counts.

Figure 7.3: Interaction network of *Letter* dataset. George William Frederick, King of United Kingdom (colored in red) is excluded from the top-5 events (participants in other colors). Node sizes are determined by the PageRank scores.

It would also be useful to apply multi-documents summarization techniques [IK11, Wan07] on top of the extracted events. Currently, our method addresses the summarization problem at a higher level, without looking deeper into the actual sentences or words in documents.

8 Conclusions

We defined the problem of summarizing top-k events in an interaction network. Our approach consists by first transforming the input data into an interaction meta-graph and then defining two optimization problems: budgeted version of PCST and maximum set cover. We offer three algorithms for the former problem. Our experiments show that the greedy approach is more lightweight and performs as good as or even better than other more sophisticated counterparts.

8.1 Contribution

The contributions of this paper are summarized as follows:

- We propose a novel formulation for the problem of discovering events that are temporally and topically coherent in interaction networks, such as, online communication networks.
- We present a transformation of the interaction network to an interaction meta-graph, which captures temporal and topical association of interactions as well as the information flow in the network. This transformation helps to provide a cleaner abstraction to the event-detection problem.
- For the problem of finding high-volume events while satisfying constraints of temporal and topical coherence we present and we evaluate three different algorithms: a greedy approach, a dynamic-programming algorithm, and an adaptation to an existing approximation algorithm.
- We address the problem of finding the top-k events that summarize the network activity. The classic greedy algorithm is the standard way to approach this problem, but here, to speed-up computation, we also propose and evaluate a search strategy that avoids evaluating candidate events at all possible tree roots, but adaptively selects the most promising ones.
- We compare and analyze our algorithms in both synthetic and real datasets, such as twitter and email communication. We show that our methods are able to detect meaningful temporal events.

8.2 Related work

Three lines of work are related to this paper: *event detection*, *text summarization* and *information diffusion*. Our work is closely related to studies on event detection. The main difference between ours and this line lies in (*i*) whether information flow process is considered and (*ii*) the modeling of topical coherence. Also, our work is related to extractive text summarization. First, one of our goals, maximizing node converge, is about the coverage sub task in extractive summarization. Second, both problems select a subset of data. In extractive text summarization, words, phrases or sentences are selected. However in our case, interactions are selected. Last, we attempt to model information flow in the event, which is similar to studies in information diffusion.

8.2.1 Event detection

The definitions of event detection vary in different contexts. In temporal/spatial data analysis, event refers to sub group of data (e.g, spatial area, time duration) with severity/magnitude information (alarm level) [NHkW09]. Usually, the goal is to detect anomalous events that rarely occur, for example disease outbreak [CN14], water pipe breaks [ASA] and anomaly detection for remote sensor data [DD11]. The goal is often early and accurate detection of negative events so that in-time prevention can be taken.

In social media and online communication context, an event often refers to a sudden burst of public discussion on certain topics, for example World Cup game [MNR+15], earthquake outbreak [SOM10] or political campaign [RCM+11]. There has been extensive studies for this context. However, they differ in various dimensions, such as computation demand (real-time or off-line), event representation (words, documents, etc) and types of information being used (text content, location, social network), which is summarized in Table 8.1.

Real-time event detection is actively studied due to the large volume of online personal content produced every day. As real-time computation depends on several practical aspects such as band-width, engineering optimization and machine number for distributed computing, these works are distinguished by whether data is processed in one-pass (so called *stream processing*) or not. In [MK10, CDCS10, SOM10, POL10, BNG10, BNG11, AS12], algorithms are designed in a one-pass fashion. In [RCM+11, ZC14, MNR+15], multiple-pass processing is allowed. In our work,

we aim to understand what is happening for the whole network, therefore one-pass constraint is not imposed.

Event representation varies in different ways. Besides time span information, representative words are used to describe events in [MK10, CDCS10]. On the other hand, [POL10, BNG10, BNG11, ZC14] select representative documents to represent events. The above design choices are combined in [SHM09, ZC14], where event consists of a set of words as well as one or a few representative documents. In some application-specific works, other information is used. In [SOM10] about earthquake detection, an event is a spatial/temporal region where earthquake happened. For online political abuse detection [RCM+11], participant identity, memes and the associated interactions are used. In our case, we use a set of interactions.

Types of information that are utilized in this line of work also differ. Besides textual content, spatial information such as tweets' location is used for either application-specific purpose [MK10] or document similarity measure [SOM10, BNG10]. Social network information such as tweets' author identity is used for feature design in similarity measure [BNG11, AS12] or event classification [RCM+11]. In contrast, social network information is fundamental for both our model and problem formulation.

Information flow is rarely considered except in [RCM+11]. In their work, information flow patterns are used to design features for truthy event identification. In our case, information flow is a natural "by-product" from our problem formulation.

Topic coherence is considered in some of the works, however approached in different ways. Graph-based approach is applied in [SHM09, CDCS10, MNR+15] and interestingly, all operate on notion of word-graph. However, the actual definition on word-graph varies. [SHM09, CDCS10] treat an event as a set of nodes in a word-graph and detect events through community detection algorithms. In [MNR+15], graph degeneracy algorithm is used to detect sets of words of large k-core number as events. Statistical and feature-based methods are used in [MK10, BNG10, AS12]. [MK10] utilizes co-occurrence statistics between bursty keywords. [BNG10, AS12] perform online clustering using feature-based document similarity metric, whereas they differ in the specific features. In both [ZC14] and our work, hard constraints are imposed. The difference is: [ZC14] imposes maximum dissimilarity for *each* pair of documents for one event, while we treat event's topic dissimilarity as the sum of topic dissimilarity from the event's edges.

	Event representation	L	N	S	F	Topic coherence
[SHM09]	words, documents					community detection
[MK10]	words	✓		✓		co-occurrence
[CDCS10]	words			✓		connected component
[SOM10]	region	✓		✓		
[POL10]	document			✓		
[BNG10]	documents	✓		✓		online clustering
[RCM+11]	users, memes, interactions		✓		✓	
[BNG11]	documents		✓	✓		online clustering
[AS12]	interactions		✓	✓		online clustering
[ZC14]	documents	✓	✓			constraint on doc pair
[MNR+15]	words, document					graph degeneracy
Ours	interactions		✓		✓	constraint on sum

Table 8.1: Comparisons of different works on event detection. Acronyms: **L**: use location?, **N**: use social network?, **S**: imposes single-pass processing?, **F**: models information flow?

8.2.2 Text summarization

Text summarization aims at reducing the content of document(s) to a brief summary that concisely covers the most important aspects. Our work is related to it because one of our goals is to cover as much interactions as possible.

Text summarization can be categorized into two approaches: extractive summarization and abstractive summarization. The first approach [Mih04, CHT11, ER04, FdSCL+13] attempts to select a subset of sentences to concisely and coherently summarize articles. Usually, sentences are assigned scores and the higher score sentences/words are included in the summary. For example in *TextRank* [Mih04], sentences are connected into a graph and edges are assigned similarity score. Graph ranking methods such as PageRank [PBMW99] are used to infer the importance scores for each sentence. This approach is purely unsupervised and uses no deep linguistic analysis.

In addition to sentence-level extraction, word-level method is studied in [GZH10]. The author claims that sentence-level method fails to summarize redundant opinions (high overlapping of information). Therefore, they aim at constructing summary

sentences from individual words that cover large portions of the redundancy. In their work, graph approach is used, where graph takes individual words as nodes, an edge from one word to another if they appear adjacently in at least one of the sentences. Under this definition, the goal becomes finding a path that is a valid sentence and have high redundancy score.

Our work is similar to extractive summarization in that both also perform some type of subset selection. We perform the extraction at document level whereas extractive summarization methods usually extract finer units such as sentences and words. Meanwhile, for some of those works, temporal information is not used. As a future direction, we plan to perform deeper text analysis (e.g, at the sentence level).

Using temporal information is gaining more attention in text summarization community. Following the work of *TextRank* [Mih04], [Wan07] proposes *TimedTextRank*, where a time decay term that considers temporal difference between two sentences is added during the graph construction process. [GK12] uses temporal scores on top of scores from traditional summarization methods. In their work, sentences are clustered using temporal event clustering. The score is calculated based on properties of its assigned cluster (such as size)

Our model and problem formulation are conceptually different with this line of works.

8.2.3 Information diffusion

Information diffusion process has been studied in different contexts, e.g, word-of-mouth effect in marketing [KKT05], spread of news and opinions [GGLNT04] and epidemics [Het00]. One important question is given the times when particular nodes are infected, can we infer which nodes infected them?

Gomez-Rodriguez et al. [GRLK12] attempt to infer the propagation flow as a tree among infected nodes in contagion networks. They first give the probability of observed contagion spreading according to a hypothetical tree using a generative model. In this model, each edge in the tree is generated independently and the individual edge generation probability is measured by the temporal closeness. Then the inference problem is defined as finding the tree that maximizes the probability. Compared to this work, our work has two major differences. First, we are dealing with event detection, in which the reconstructed network can be seen as a byproduct. Thus, our problem formulation is conceptually different. Second, edge strength is

measured using topic similarity.

References

AS12 Aggarwal, C. C. and Subbian, K., Event detection in social streams. *Proceedings of the 2012 SIAM International Conference on Data Mining*, 2012, pages 624–635, URL `http://epubs.siam.org/doi/abs/10.1137/1.9781611972825.54`.

ASA Asce, F., Soibelman, L. and Asce, M., Detection of patterns in water distribution pipe breakage using spatial scan statistics for point events in a physical network.

BGHS12 Boden, B., Günnemann, S., Hoffmann, H. and Seidl, T., Mining coherent subgraphs in multi-layer graphs with edge labels. *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, New York, NY, USA, 2012, ACM, pages 1258–1266, URL `http://doi.acm.org/10.1145/2339530.2339726`.

BKNS00 Breunig, M. M., Kriegel, H.-P., Ng, R. T. and Sander, J., Lof: Identifying density-based local outliers. *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD '00, New York, NY, USA, 2000, ACM, pages 93–104, URL `http://doi.acm.org/10.1145/342009.335388`.

BM00 Brill, E. and Moore, R. C., An improved error model for noisy channel spelling correction. *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 2000, pages 286–293.

BNG10 Becker, H., Naaman, M. and Gravano, L., Learning similarity metrics for event identification in social media. *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, New York, NY, USA, 2010, ACM, pages 291–300, URL `http://doi.acm.org/10.1145/1718487.1718524`.

BNG11 Becker, H., Naaman, M. and Gravano, L., Beyond trending topics real-world event identification on twitter. *International AAAI Conference on Web and Social Media*, 2011.

BNJ03 Blei, D. M., Ng, A. Y. and Jordan, M. I., Latent dirichlet allocation. *Journal of Machine Learning Research*, volume 3. JMLR. org, 2003, pages 993–1022.

BV04 Boyd, S. and Vandenberghe, L., *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

CCC+99 Charikar, M., Chekuri, C., Cheung, T.-y., Dai, Z., Goel, A., Guha, S. and Li, M., Approximation algorithms for directed steiner problems. *Journal of Algorithms*, volume 33. Elsevier, 1999, pages 73–91.

CDCS10 Cataldi, M., Di Caro, L. and Schifanella, C., Emerging topic detection on twitter based on temporal and social terms evaluation. *Proceedings of the Tenth International Workshop on Multimedia Data Mining*, MDMKDD '10, New York, NY, USA, 2010, ACM, pages 4:1–4:10, URL `http://doi.acm.org/10.1145/1814245.1814249`.

CHT11 Celikyilmaz, A. and Hakkani-Tür, D., Discovery of topically coherent sentences for extractive summarization. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, volume 1. Association for Computational Linguistics, 2011, pages 491–499.

CN14 Chen, F. and Neill, D. B., Non-parametric scan statistics for disease outbreak detection on twitter. *Online Journal of Public Health Informatics*, 6,1(2014).

DD11 Dereszynski, E. W. and Dietterich, T. G., Spatiotemporal models for data-anomaly detection in dynamic environmental monitoring campaigns. *ACM Trans. Sen. Netw.*, 8,1(2011), pages 3:1–3:36. URL `http://doi.acm.org/10.1145/1993042.1993045`.

ER04 Erkan, G. and Radev, D. R., Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Intell. Res. (JAIR)*, volume 22, 2004, pages 457–479.

FdSCL+13 Ferreira, R., de Souza Cabral, L., Lins, R. D., e Silva, G. P., Freitas, F., Cavalcanti, G. D., Lima, R., Simske, S. J. and Favaro, L., Assessing sentence scoring techniques for extractive text summarization. *Expert Systems with Applications*, 40,14(2013), pages 5755–5764.

FPS00 Feigenbaum, J., Papadimitriou, C. and Shenker, S., Sharing the cost of muliticast transmissions (preliminary version). *Proceedings of the thirty-second annual ACM Symposium on Theory of Computing*. ACM, 2000, pages 218–227.

GDFMGM16 Garimella, K., De Francisci Morales, G., Gionis, A. and Mathioudakis, M., Quantifying controversy in social media. *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, WSDM '16, New York, NY, USA, 2016, ACM, pages 33–42, URL `http://doi.acm.org/10.1145/2835776.2835792`.

GGLNT04 Gruhl, D., Guha, R., Liben-Nowell, D. and Tomkins, A., Information diffusion through blogspace. *Proceedings of the 13th International Conference on World Wide Web*. ACM, 2004, pages 491–501.

GK12 Gung, J. and Kalita, J., Summarization of historical articles using temporal event clustering. *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 2012, pages 631–635.

GRLK12 Gomez-Rodriguez, M., Leskovec, J. and Krause, A., Inferring networks of diffusion and influence. *ACM Trans. Knowl. Discov. Data*, 5,4(2012), pages 21:1–21:37. URL `http://doi.acm.org/10.1145/2086737.2086741`.

GS99 Guralnik, V. and Srivastava, J., Event detection from time series data. *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '99, New York, NY, USA, 1999, ACM, pages 33–42, URL `http://doi.acm.org/10.1145/312129.312190`.

GW95 Goemans, M. X. and Williamson, D. P., A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, volume 24. SIAM, 1995, pages 296–317.

GZH10 Ganesan, K., Zhai, C. and Han, J., Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions. *Proceedings of the 23rd International Conference on Computational Linguistics*. Association for Computational Linguistics, 2010, pages 340–348.

Het00 Hethcote, H. W., The mathematics of infectious diseases. *SIAM review*, 42,4(2000), pages 599–653.

IK11 Inouye, D. and Kalita, J. K., Comparing twitter summarization algorithms for multiple post summaries. *Inernational Conference on Social Computing (SocialCom)*. IEEE, 2011, pages 298–306.

JMP00 Johnson, D. S., Minkoff, M. and Phillips, S., The prize collecting steiner tree problem: theory and practice. *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 2000, pages 760–769.

Joh73 Johnson, D. B., A note on dijkstra's shortest path algorithm. *Journal of the ACM (JACM)*, 20,3(1973), pages 385–388.

KKT05 Kempe, D., Kleinberg, J. and Tardos, É., Influential nodes in a diffusion model for social networks. In *Automata, Languages and Programming*, Springer, 2005, pages 1127–1138.

KMM10 Kumar, R., Mahdian, M. and McGlohon, M., Dynamics of conversations. *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2010, pages 553–562.

LV12 Liemhetcharat, S. and Veloso, M., Modeling and learning synergy for team formation with heterogeneous agents. *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, volume 1. International Foundation for Autonomous Agents and Multiagent Systems, 2012, pages 365–374.

Mih04 Mihalcea, R., Graph-based ranking algorithms for sentence extraction, applied to text summarization. *Proceedings of the ACL 2004 on Interactive Poster and Demonstration Sessions*. Association for Computational Linguistics, 2004, page 20.

MK10 Mathioudakis, M. and Koudas, N., Twittermonitor: Trend detection over the twitter stream. *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, New York, NY, USA, 2010, ACM, pages 1155–1158, URL `http://doi.acm.org/10.1145/1807167.1807306`.

MNR+15 Meladianos, P., Nikolentzos, G., Rousseau, F., Stavrakas, Y. and Vazirgiannis, M., Degeneracy-based real-time sub-event detection in twitter stream. *International AAAI Conference on Weblogs and Social Media*, 2015.

NHkW09 Neill, D. B., Heinz, H. J. and keen Wong, W., Tutorial on event detection kdd 2009, 2009.

PBMW99 Page, L., Brin, S., Motwani, R. and Winograd, T., The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. URL `http://ilpubs.stanford.edu:8090/422/`. Previous number = SIDL-WP-1999-0120.

POL10 Petrović, S., Osborne, M. and Lavrenko, V., Streaming first story detection with application to twitter. *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, Stroudsburg, PA, USA, 2010, Association for Computational Linguistics, pages 181–189, URL `http://dl.acm.org/citation.cfm?id=1857999.1858020`.

RCM+11 Ratkiewicz, J., Conover, M., Meiss, M., GonÃ§alves, B., Flammini, A. and Menczer, F., Detecting and tracking political abuse in social media. *International AAAI Conference on Web and Social Media*, 2011.

SGR14 Su, H., Gionis, A. and Rousu, J., Structured prediction of network response. *Proceedings of the 31st International Conference on Machine Learning*, 2014, pages 442–450.

SHM09 Sayyadi, H., Hurst, M. and Maykov, A., Event detection and tracking in social streams. *In Proceedings of the International Conference on Weblogs and Social Media (ICWSM 2009). AAAI*, 2009.

SOM10 Sakaki, T., Okazaki, M. and Matsuo, Y., Earthquake shakes twitter users: Real-time event detection by social sensors. *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, New York, NY, USA, 2010, ACM, pages 851–860, URL `http://doi.acm.org/10.1145/1772690.1772777`.

Vaz13 Vazirani, V. V., *Approximation algorithms*. Springer Science & Business Media, 2013.

Wan07 Wan, X., Timedtextrank: adding the temporal dimension to multi-document summarization. *Proceedings of the 30th annual international ACM SIGIR Conference on research and Development in Information Retrieval*. ACM, 2007, pages 867–868.

WH14 Wang, Y. R. and Huang, H., Review on statistical methods for gene network reconstruction using expression data. *Journal of Theoretical Biology*, volume 362. Elsevier, 2014, pages 53–61.

WS11 Williamson, D. P. and Shmoys, D. B., *The design of approximation algorithms*. Cambridge university press, 2011.

ZC14 Zhou, X. and Chen, L., Event detection over twitter social media streams. *The VLDB Journal*, 23,3(2014), pages 381–400. URL `http://dx.doi.org/10.1007/s00778-013-0320-3`.

Appendix 1. NP-hardness of Problem 4

We want to prove Problem 4 is **NP**-hard. We induce Problem 4 from *minimum set cover* problem.

We define *minimum set cover* as a decision problem, $SC(S, \mathcal{C}, k)$. Given $S = \{s_1, \ldots, s_n\}, \mathcal{C} = \{C_1, \ldots, C_m\}$, $C_i \subseteq S, i = 1, \ldots, m$, $k \in \mathbb{Z}$, we are asked if there is $\mathcal{C}' \subseteq \mathcal{C}$ such that $\forall s \in S, s \in \cup_{C \in \mathcal{C}'} C$ and $\left|\mathcal{C}'\right| \leq k$.

We define the decision problem of $P4$ as $P4(G, r, B, W)$. Given directed graph $G = (V, E)$, $c : E \to \mathbb{R}$, $p : V \to 1$, $r \in V$, budget $B \in \mathbb{R}$ and $W \in \mathbb{R}$, we are asked if there exists a subtree $T = (V', E') \subseteq G$ rooted at r such that $c(T) = \sum_{e \in E'} c(e) \leq B$ and $p(T) = \sum_{v \in V'} p(e) \geq W$.

Consider $SC(S, \mathcal{C}, k)$, where $S = \{s_1, \ldots, s_n\}, \mathcal{C} = \{C_1, \ldots, C_m\}$. Construct a directed graph $G = (V, E)$ where:

- $V = \{r, C_1, \ldots C_m, s_1, \ldots s_n\}$
- $E = \{(r, C_i) \mid C_i \in \mathcal{C}\} \cup \{(C_i, s_j) \mid s_j \in C_i, C_i \in \mathcal{C}\}$
- $\forall C_i \in \mathcal{C}, c(r, C_i) = N$
- $\forall (C_i, s_j) \in \mathcal{C} \times S, c(C_i, s_j) = 1$
- $N > n$

Lemma 1 *If $SC(S, \mathcal{C}, k)$ holds, $P4(G, r, kN + n, k + n + 1)$ holds.*

Without loss of generality, let $\mathcal{C}' = \{C_{q_1}, \ldots, C_{q_k}\}$ be the set cover for S. We can construct a tree $T = (V', E')$ such that $p(T) > k + n + 1$ and $c(T) \leq kN + n$ in the following way:

- $V' = \{r\} \cup \mathcal{C}' \cup S$
- $(r, C_i) \in E'$ if $C_i \in \mathcal{C}'$
- $(C_1, s_j) \in E'$ if $s_j \in C_1$
- $(C_i, s_j) \in E'$ if $s_j \in C_i \setminus \bigcup_{i=1 \ldots j-1} C_{q_i}$

It is easy to see T makes $P4(G, r, kN + n, k + n + 1)$ hold.

Lemma 2 *If* $P4(G, r, kN + n, k + n + 1)$ *holds,* $SC(S, \mathcal{C}, k)$ *holds.*

Denote the tree corresponding to $P4(G, r, kN+n, k+n+1)$ as $T = (V^{'}, E^{'})$, then one set cover for $SC(S, \mathcal{C}, k)$ to hold is $\mathcal{C}^{'} = V^{'} \cap \mathcal{C}$. This can be proved by contradiction.

On on hand, if $\mathcal{C}^{'}$ is not a set cover, then $\exists s \in S, s \notin V^{'}$. Then some $C^{'} \in \mathcal{C}$ needs to be included in $\mathcal{C}^{'}$ to satisfy $p(T) \geq k+n+1$, . In this case $c(T) \geq (k+1)N > kN+n$, which is a contradiction.

On the other hand, if $\left|\mathcal{C}^{'}\right| > k$, then $c(T) \geq (k+1)N > kN + n$, which is a contradiction.

Therefore, Problem 4 is **NP**-hard.

Appendix 2. Code and scripts

We host all the code and scripts for this work on Github: `https://github.com/xiaohan2012/lst`