Chapter 4

Efficient k-Sketch Discovery in Sequence Data

4.1 Introduction

Today's journalists must pore over large amounts of data to discover interesting newsworthy themes. While such a task has traditionally been done manually, there is an increasing reliance on computational technology to reduce human labor and intervention to a minimum. To support automatic news theme discovery, early research efforts have focused on discovering newsworthy patterns derived from a single event, such as the *situational facts* [51], *one-of-few facts* [62] and FactsWatcher [30]. Recently, Zhang et. al. proposed a new type of news theme pattern, named *prominent streak* [69]. A *streak* corresponds to an event window with a sequence of consecutive events belonging to the same subject. Their goal is to discover all the non-dominated streaks to be news themes. However, the number of non-dominated streaks can be overwhelmingly large.

In this paper, we study the automatic discovery of a novel news theme, named *rank-aware* theme. Compared with previous works, the rank-aware theme is able to

capture the strikingness of an event window. We observe that such a rank-aware theme is an interesting pattern that has been frequently used:

1. (Feb 26, 2003) With 32 points, Kobe Bryant saw his 40+ scoring streak end at **nine** games, tied with Michael Jordan for **fourth** place on the all-time list[1].
2. (April 14, 2014) Stephen Curry has made 602 3-pointer attempts from beyond the arc,... are the **10th** most in NBA history in a season (**82 games**)[2].
3. (May 28, 2015) Stocks gained for the **seventh consecutive day** on Wednesday as the benchmark moved close to the 5,000 mark for **the first** time in seven years[3].
4. (Jun 9, 2014) Delhi has been witnessing a spell of hot weather over the **past month**, with temperature hovering around 45 degrees Celsius, ... **highest** ever since 1952[4].
5. (Jul 22, 2011) Pelican Point recorded a maximum rainfall of 0.32 inches for **12 months**, making it the **9th driest** places on earth[5].

In the above news themes, there are a subject (e.g., Kobe Bryant, Stocks, Delhi), an event window (e.g., nine straight games, seventh consecutive days, past month), an aggregate function on an attribute (e.g., minimum points, count of gains, average of degrees), a rank (e.g., fourth, first time, highest), and a historical dataset (e.g., all time list, seven years, since 1952). These news theme indicators are summarized in Table 4.1. To avoid outputting near-duplicate news themes and control the result size, we further propose a novel concept named *Sketch*. A sketch contains k most representative rank-aware news themes under a scoring function that considers both

[1]http://www.nba.com/features/kobe_40plus_030221.html

[2]http://www.cbssports.com/nba/eye-on-basketball/24525914/stephen-curry-makes-history-with-consecutive-seasons-of-250-3s

[3]http://www.zacks.com/stock/news/176469/china-stock-roundup-ctrip-buys-elong-stake-trina-solar-beats-estimates

[4]http://www.dnaindia.com/delhi/report-delhi-records-highest-temperature-in-62-years-1994332

[5]http://www.livescience.com/30627-10-driest-places-on-earth.html

E.g.	Aggregate function	Event window	Rank
1	min(points)	9 straight games	4
2	sum(shot attempts)	82 games	10
3	count(gains)	7 consecutive days	1
4	average(degree)	past months (30 days)	1
5	max(raindrops)	12 months	9

Table 4.1: News theme summary

strikingness and diversity. Our objective is to discover sketches for each subject in the domain.

We study the sketch discovery problem under both offline and online scenarios. In the offline scenario, the objective is to efficiently discover the sketches for each subject from historical data. The major challenge lies in generating candidate themes from event windows. Since the number of event windows is quadratic, enumerating all of them is not scalable. By leveraging the subadditivity among the upper bounds of event windows, we design two effective pruning techniques to facilitate efficient candidate theme generation. Furthermore, we note that generating exact sketches from candidate themes is expensive. Thus, we design an efficient $(1 - 1/e)$ approximation algorithm by exploiting submodularity of the sketch discovery problem.

In the online scenario, fresh data is continuously fed into the system and our goal is to maintain the sketches for each subject up-to-date. When a new event about subject s arrives, many event windows of various lengths can be derived. For each derived event window, not only the sketch of s but also the sketches of other subjects may be affected. Dealing with such a complex updating pattern is non-trivial. To efficiently support the update while maintaining the quality of sketches, we propose a $1/8$ approximation solution which only examines $2k$ candidate themes for each subject whose sketch is affected.

Our contributions are hereby summarized as follows:

- We study the automatic discovery of a new type of news theme that is common

in real-life reports but has not been addressed in previous works. We formulate it as a sketch discovery problem under a novel scoring function that considers both strikingness and diversity.

- We study the sketch discovery problem in both online and offline scenarios. In the offline scenario, we propose two novel pruning techniques to efficiently generate candidate themes. Then we design a $1 - 1/e$ greedy algorithm to discover the sketches for each subject. In the online scenario, we propose a $1/8$ approximation solution to efficiently support the complex updating pattern as each new event arrives.

- We conduct extensive experiments with four real datasets to evaluate the efficiency of our proposed algorithms. In the offline scenario, our solution is three orders of magnitude faster than baseline algorithms. While in the online scenario, our solution achieves up to a 500 times speedup. In addition, we also perform an anonymous human study via Amazon Mechanical Turk[6] platform, which shows the effectiveness of sketch in news theme generation.

The rest of the paper is organized as follows. In Section 4.2, we summarize the related work. Section 4.3 formulates the sketch discovery problem. Section 4.4 presents the algorithms for offline sketch discovery. Section 4.5 describes the algorithms for processing and maintaining sketches in the online scenario. In Section 4.6, comprehensive experiments studies on both efficiency and effectiveness of our algorithms are conducted. Section A.1 discusses the extension of our methods. Finally, Section 4.7 concludes our paper.

[6]https://requester.mturk.com

4.2 Related Work

In this section, we briefly review three closely related areas: automatic news theme detection, frequent episode mining and top-k diversity query.

4.2.1 Automatic News Theme Detection

Earlier works on automatic news theme generation were focused on finding interesting themes from a single event. For example, Sultana et al. [51] proposed the *Situational Fact* pattern, which is modeled as a skyline point under certain dimensions. Wu et al. [62] proposed the *One-of-the-Few* concept to detect news themes with some rarities. Examples of candidate news themes for the above two patterns are illustrated in Table 4.2.

Table 4.2: Examples of different news themes

Method	**Example news theme**
Situational facts [51]	Damon Stoudamire scored 54 points on January 14, 2005. It is the highest score in history made by any Trail Blazer.
One-of-the-τ facts [62]	Jordan, Chamberlain, James, Baylor and Pettit are the five players with highest points and rebounds in NBA history.
Prominent streaks [69]	1.Kobe scored 40+ in 9 straight games, first in his career! 2.Kobe scored 50+ in 4 straight games, first in his career! 3...
k-Sketch	1.Kobe scored 40+ in 9 straight games, ranked 4th in NBA history! 2.Kobe scored 50+ in 4 straight games, ranked 1st in NBA history. 3....

Zhang et al. [69] proposed using prominent streaks to generate interesting news themes. In [69], a *Prominent Streak* is characterized by a 2D point which represents the window length and the minimum value of all events in the window. The objective

is to discover the non-dominated event windows for each subject, where the dominance relationship is defined among streaks of the same subject. Our model differs from [69] in two aspects. First, we look at the global prominence (quantified by the rank) among all subjects rather than local prominence (quantified by the dominance) within one subject. Second, our model provides the best k-sketch for each subject whereas [69] returns a dominating set which could be potentially large.

4.2.2 Frequent Episode Mining

In time sequenced data mining, an episode [38, 47, 52, 75] is defined as a collection of time sequenced events which occur together within a time window. The uniqueness of an episode is determined by the contained events. The objective is to discover episodes whose occurrences exceeding a support threshold. Our sketch discovery differs from the episode mining in three major aspects. First, an episode is associated with a categorical value while our sketch is defined on numerical values. Second, the episodes are selected based on the occurrence, while in sketch, news themes are generated in a rank-aware manner. Finally, episode mining does not restrict its output size, while sketch only outputs the best k news themes. As such, episode mining techniques cannot be straightforwardly applied to sketch discovery.

4.2.3 Top-k Diversity Query

Top-k diversity queries [1,9,17,25] aim to find a subset of objects to maximize a scoring function. The scoring function normally penalizes a subset if it contains similar elements. Our sketch discovery problem has two important distinctions against the top-k diversity queries. First, the inputs of the scoring function are known in advance in top-k diversity queries; whereas in our problem, the ranks of event windows are unknown. Since their calculations are expensive, we need to devise efficient methods to compute the ranks. Second, existing methods for online diversity queries [9, 17, 25]

only study the updates on a single result set when a new event arrives. However our online sketch maintenance incurs the problem of multiple sketch updates for each new event. Such a complex update pattern has not been studied yet and hence there is a need to develop efficient update scheme.

4.2.4 Event Detection and Tracking

In the information retrieval field, event detection and tracking aim to extract and organize new events from various media sources. Allen et.al [3] first proposed the problem of detecting unseen events from text streams, where they adapted an online clustering algorithm to tackle the problem. Subsequent researches extended the problem to facilitate heterogeneous sources. For example, Brants et.al. [10] proposed a TF-IDF model to detect unseen events from multiple text streams. Ritter et.al. [**?**] proposed a system named TwiCal to categorize events on Twitter streams. Li et.al. [41] proposed a ranking model to detect events on Flickr and Vuurens et.al. [55] described a system on tracking events from web articles.

Despite the usefulness of those works, they differ from the rank-aware news themes proposed in this paper. The major difference is that news detection and tracking focuses on detecting a single event from various sources; While the rank-aware news theme aims at providing insightful derived events from a single source. This crucial difference prevents the above-mentioned techniques directly applying to our problem.

4.3 Problem Formulation

Let S denote a set of subjects of potential interests to journalists. For example, S can refer to all the players or teams in the NBA application. Let $e_s(t)$ denote an event about subject s, where t denotes its timestamp or sequence ID. For example, an event can refer to an NBA game a player participated on a certain day. Note that

we maintain a sequence ID for each subject that is automatically incremented. It is possible that the events of different subjects may have the same sequence IDs. A sequence of consecutive events of the same subject is grouped as an *event window*:

Definition 4.3.1 (Event Window)**.** Let w be a window length, an event window $W_s(t, w)$ refers to w consecutive events of subject s and ending at sequence t, i.e., $W_s(t, w) = \{e_s(t - w + 1), ..., e_s(t)\}$.

In general, if a subject s has $|\mathbb{H}_s|$ events, there are $\binom{|\mathbb{H}_s|}{2}$ possible event windows. Given an aggregate function f, events in an event window can be aggregated to a numerical value $\overline{v}$ as:

$$W_s(t, w).\overline{v} = f(e_s(t - w + 1), ..., e_s(t))$$

We support all the common aggregate functions such as *sum*, *avg*, *count*, *min* and *max*. For simplicity of presentation, we only consider a single aggregate function in our solution. To support multiple aggregation functions, one may simply invoke our solution multiple times to process them independently.

Example 4.3.1. Fig. 4.1 (A) illustrates examples of *event window*s of three NBA players. Each event represents the points scored by a player in a game. In the figure, the event window $W_{s_1}(3, 2)$ refers to two consecutive events about player s_1 ending at $t = 3$, i.e., $W_{s_1}(3, 2) = \{e_{s_1}(2), e_{s_1}(3)\}$. Given an aggregate function $f = avg(points)$, we yield $W_{s_1}(3, 2).\overline{v} = (46 + 10)/2 = 28$.

With the aggregated value $\overline{v}$, we can derive the rank of an event window, which can be used to measure the strikingness of an event window as a news theme. For instance, "*The total points Kobe has scored is* $32,482$" can be transformed into a rank-aware representation: "*Kobe moved into third place on the NBA's all-time scoring list*", in which the rank is 3. Let W be a length-w event window, and $\gamma_w(W.\overline{v})$ be the

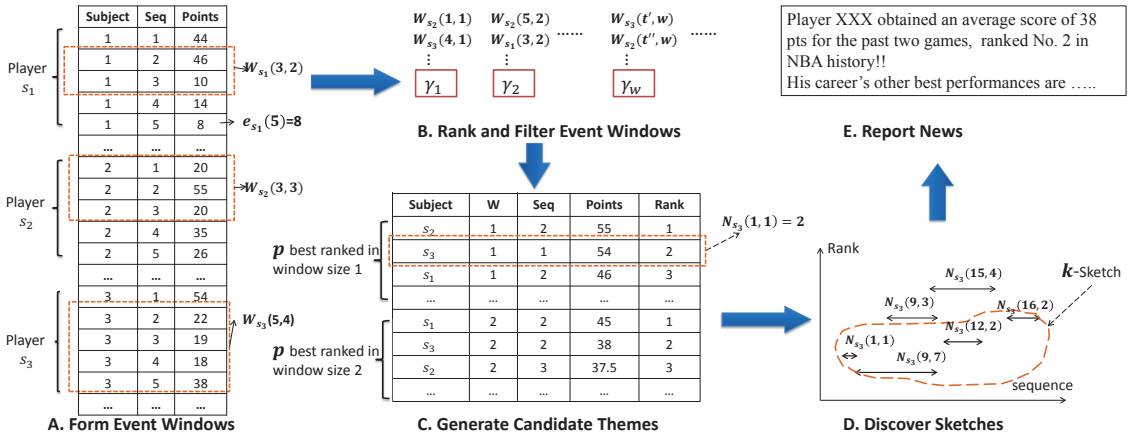


Figure 4.1: An illustration sketch discovery workflow. (A): various event windows are formed based on events' sequence IDs. (B): event windows with rank greater than p are filtered. (C): best ranked event windows form candidate themes. (D): k-sketches are discovered for each subject from their candidate themes. (E): subject related news can be generated from each k-sketch.

rank of W by comparing it with all other length-w event windows on their aggregate values. Generally speaking, an event window is newsworthy if its rank is smaller than a predefined threshold. This leads to our definition of *Candidate Theme*:

Definition 4.3.2 (Candidate Theme)**.** An event window $W_s(t, w)$ is said to be a candidate theme, denoted by $N_s(t, w)$, if its rank $\gamma_w(W_s(t, w).\overline{v}) \leq p$. p is a user-defined threshold.

Example 4.3.2. In the step (B) of Fig. 4.1, we group the event windows based on their window length. Each window is associated with a rank value γ_w. If the rank value is greater than the threshold p, the associated event window is pruned. Otherwise, it is considered as a candidate. In the step (C) of Fig. 4.1, we present the candidate themes in the tabular format. Each candidate contains the rank computed from step (B). For instance, among all candidate themes with window length 1, $N_{s_3}(1, 1)$ (with value 54) is ranked 2nd because its aggregated value is smaller than that of another event window $N_{s_2}(2, 1)$ (with value 55).

Let $\mathbb{N}_s$ be the set of candidate themes of subject s. Since for each possible window

length, there are at most p candidate themes, the size of $\mathbb{N}_s$ can be as large as $p|\mathbb{H}_s|$, which is still overwhelming for news reporting. To control the output size while maintaining the quality of news themes, we aim to find, for each subject s, a small subset of k candidate themes from $\mathbb{N}_s$. We call the subset a k-**Sketch**.

To measure the quality of a sketch, we define a scoring function $g(\cdot)$ over all sketches. The design of g gives rise to two concerns. First, g needs to assign higher scores to the candidate themes with better ranks. This is because better rank indicates good strikingness which implies that the news theme would be more eye-catching. For instance, we can expect people to care more about who the top scorer in NBA history is rather than who ranked 20^{th} in the history. Second, g prefers the sketches containing candidate themes with fewer overlapped events and avoids near-duplicate events. This is because near-duplicate themes generally do not add news values. For example, when Kobe Bryant scored 81 points in a single game (2nd highest in NBA history), all event windows containing that event are likely to have good and similar ranks. Thus, reporting all of them is not newsworthy.

To resolve these two concerns, we define g as follows: given any set $\mathbb{X}_s$ of candidate themes about subject s (i.e., $\mathbb{X}_s \subseteq \mathbb{N}_s$), the score of $\mathbb{X}_s$ is:

$$g(\mathbb{X}_s) = \alpha C(\mathbb{X}_s) + (1-\alpha)R(\mathbb{X}_s), \alpha \in [0,1] \tag{4.1}$$

where $C(\mathbb{X}_s)$ is the ratio between the number of distinct events covered by $\mathbb{X}_s$ over the total number of events about subject s. In this way, duplicated candidate themes contribute to a lower score. $R(\mathbb{X}_s) = \frac{1}{|\mathbb{X}_s|}\sum_{X_s \in \mathbb{X}_s} \frac{p - X_s.\gamma}{p}$ is the strikingness of $\mathbb{X}_s$. Any candidate themes in $\mathbb{X}_s$ changing to a better rank increases $R(\mathbb{X}_s)$. The values of $C(\mathbb{X}_s)$ and $R(\mathbb{X}_s)$ are normalized between $[0,1]$. α is an adjustable coefficient to balance the weights between $C(\mathbb{X}_s)$ and $R(\mathbb{X}_s)$. If α is high, it means users are more interested in finding news themes that cover most of the subject's events. If α is low,

it indicates users prefer news themes whose ranks are small.

With Equation 4.1, we then define the *Sketch Discovery* problem as follows:

Definition 4.3.3 (Sketch Discovery)**.** Given a parameter k, Sketch Discovery aims to, for each subject s, find a subset SK_s from the candidate themes of s (i.e., $SK_s \subseteq \mathbb{N}_s$), s.t. $|SK|_s = k$ and $g(SK_s)$ is maximized.

Example 4.3.3. In the step (D) of Fig. 4.1, we show a collection of candidate themes and a k-sketch derived from them. The y-axis is the rank and the x-axis represents the complete sequence of events of a subject. Each candidate theme is represented by a line segment, covering a window of events. When $k = 4$, four of the candidate themes are selected as a 4-sketch as the most representative news for the subject.

Before we move on to the algorithmic part, we first list the notations in Table 4.3. For the ease of presentation, we present our techniques using *avg* as the default aggregate function in the following sections. Extending our techniques to other aggregate functions will be addressed in Section A.1.

Notation	**Meaning**
S	set of subjects
$\mathbb{H}_s$	set of events associated with subject s
$W_s(t, w)$	length-w event window of s ending at t
$\mathbb{N}_s$	set of candidate themes associated with subject s
$N_s(t, w)$	candidate theme derived from $W_s(t, w)$
$WI(w)$	p best ranked candidate themes of window length w
$\beta(w)$	lower bound of $WI(w)$
SK_s	sketch for subject s
J_s	visiting-window bound for subject s
M_s	unseen-window bound for subject s
P_s	online-window bound for subject s

Table 4.3: Notations used in this chapter

4.4 Offline Sketch Discovery

In the offline scenario, the input is a set of events of all subjects and the output is a k-sketch for each subject s, denoted by SK_s. The sketch discovery process consists of two major steps: first generating the candidate themes of each subject (i.e., $\mathbb{N}_s$), and then discovering the sketches among those candidate themes.

4.4.1 Candidate Theme Generation

Generating candidate themes for each subject is computationally expensive. In order to retain the candidate themes with rank no greater than p, a brute-force approach needs to evaluate all the event windows with every possible length to generate accurate ranks. Since there could be $\binom{\mathbb{H}_s}{2}$ event windows (with different lengths and ending sequence ids) associated with subject s, the total time complexity for the brute-force approach is $O(\sum_{s \in S} |\mathbb{H}_s|^2)$. Such a complexity makes the solution infeasible even for moderate-sized datasets.

To improve the efficiency, we observe that it is not necessary to compute all the event windows to generate $\mathbb{N}_s$. The intuition behind is that the upper bound value of event windows with longer lengths can be estimated from those with shorter lengths. This means that we can compute event windows with increasingly lengths, and as the shorter event windows are computed, the longer event windows not in $\mathbb{N}_s$ can be pruned. To realize such an intuition, we design the candidate generation algorithm by adapting two novel window-based pruning methods which exploit the *subadditivity* property among event windows with different window lengths.

4.4.1.1 Overview of window pruning

For each subject, its event windows can be grouped by window lengths. Our candidate generation algorithm gradually evaluates a subject's event window from shorter length

to longer length. To support efficient pruning, we define two concepts, namely *visiting-window bound* (J_s) and *unseen-window bound* (M_s). In particular, $J_s(w)$ is the upper bound of all the event windows about subject s and with length w, i.e., $\forall w, J_s(w) \geq \max\{W_s(t,w).\overline{v}|t \in (0,w)\}$. $M_s(w)$ is the upper bound of all the event windows about subject s with length larger than w, i.e., $M_s(w) \geq \max\{J_s(t)|t > w\}$. These two bounds will be used for event window pruning and we will present how to estimate these two upper bounds in Sections 4.4.1.2 and 4.4.1.3.

The overview of news theme generation algorithm is presented in Algorithm 6. We maintain two global structures WI and β (Lines 1-2). For each window length w, $WI(w)$ stores the top p event windows with length w among all the subjects and $\beta(w)$ is the p-th score among the candidates in $WI(w)$. In other words, $\beta(w)$ serves as a lower bound score. An event window with length w is a candidate for the k-sketch only if its aggregate value is larger than $\beta(w)$. A priority queue Q (Line 3) is used to provide an access order among the subjects. Each element in the queue is a triple (s, w, q), where s is a subject, w indicates the next window length of s to evaluate, and q denotes the priority. We use the unseen-window bound (i.e., $M_s(w)$) as the priority during every iteration (Lines 13-14). Intuitively, an event window with higher $M_s(w)$ is more likely to spawn new event windows that can increase β and benefit the subsequent pruning. During initialization, we insert, for each subject, an entry with window length 1 and priority infinity into Q.

In each iteration, we pop the event window with the highest priority (Line 4). Then, we compute the visiting-window bound $J_s(w)$ for the subject and determine whether all the length-w windows about subject s can be pruned (Lines 5-6). If the bound $J(s)$ is no greater than $\beta(w)$, then all these event windows can be ignored. Otherwise, all the length-w event windows of s are computed to update $WI(w)$ and $\beta(w)$ accordingly (Lines 7-9). In the next step, we estimate the unseen-window bound M_s and compare it with the minimum $\beta(w')$ for all $w' > w$. If M_s is smaller, then

Algorithm 6 Candidate Theme Generation Overview

```
1: WI() ← {} // p best event windows for each window length
2: β ← {} // smallest scores in WI for each window length
3: Q ← {(s, 1, +∞)|s ∈ S}
4: while (s, w, q) ← Q.pop() do
5:     compute J_s(w)
6:     if J_s(w) > β(w) then
7:         for t ∈ w...|H_s| do
8:             Update WI(w), β(w), and J_s(w)
9:         end for
10:    end if
11:    compute M_s(w) whose value relies on J_s(w)
12:    if M_s(w) > min{β(w')|w < w' ≤ |H_s|} then
13:        q ← M_s(w)
14:        Q.push(s, w + 1, q)
15:    end if
16: end while
17: return WI
```

we would not find a better candidate and all the event windows about subject s and with length larger than w can be pruned (Lines 11-12). Otherwise, we assign M_s to the priority q, and push the triple $(s, w+1, q)$ back into the queue (Lines 13-14). The algorithm terminates when all the candidates have been evaluated and Q becomes empty.

4.4.1.2 Visiting-window pruning

In Algorithm 6, we need to compute the visiting-window bound $J_s(w)$ to facilitate pruning all length-w event windows associated with subject s. Our idea is that suppose we have successfully derived the bounds for windows with smaller lengths, we can use them to estimate the bounds of larger windows. We formulate it as the subadditivity property and use avg[7] as the aggregate function for illustration:

[7]Although here we demonstrate the bound using "avg", the properties and bounds also hold for other aggregate functions. Corresponding properties and bounds for other aggregate functions are listed in Section A.1.

Theorem 4.4.1 (Subadditivity (for avg))**.**

$$\max_t\{W_s(t,w).\overline{v}\} \leq \frac{w_i J_s(w_i) + (w-w_i)J_s(w-w_i)}{w}, \forall w_i \in (0,w) \tag{4.2}$$

Proof. Given any event window $W = W_s(t,w)$ with length w and a value $w_i \in (0,w)$, we can split the window into two non-overlapping sub-windows with lengths w_i and $w-w_i$, i.e.,$W_1 = W_s(t,w_i)$ and $W_2 = W_s(t-w_i, w-w_i)$. Due to the non-overlapping property of W_1 and W_2, we have $wW.\overline{v} = w_i W_1.\overline{v} + (w-w_i)W_2.\overline{v}$. Since $J_s(w_i)$ and $J_s(w-w_i)$ are two visiting-window bounds, we have $W_1.\overline{v} \leq J_s(w_i)$ and $W_2.\overline{v} \leq J_s(w-w_i)$. Then, we complete the proof and $\frac{w_i J_s(w_i)+(w-w_i)J_s(w-w_i)}{w}$ is a bound for event windows with length w. □

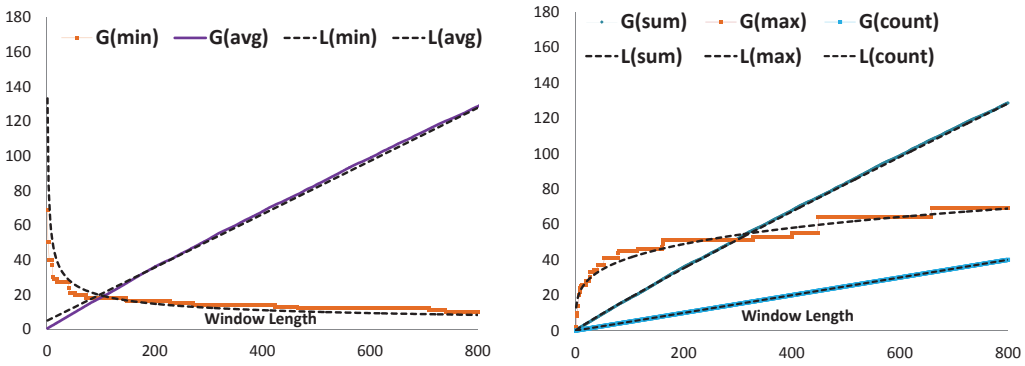
With Theorem 4.4.1, we can estimate $J_s(w)$ by any pair $J_s(w_1)$ and $J_s(w-w_1)$, $\forall w_1 < w$. To derive a tighter bound for $J_s(w)$, we formulate the following optimization problem:

$$w_1 = \operatorname*{argmin}_{t\in(0,w)} \frac{J_s(t) * t + J_s(w-t) * (w-t)}{w} \tag{4.3}$$

A naive solution would enumerate every possible t to get the tightest bound. However, such a solution has a worst time complexity of $O(|\mathbb{H}_s|)$ for subject s. To quickly compute w_1 without compensating the running time, we apply a continuous relaxation to Eqn. 4.3 as follows: Let $G_s(t) = J_s(t) * t \ \ \forall t$, then Eqn. 4.3 is equivalent to:

$$w_1 = \operatorname*{argmin}_{t\in 1..w-1}\{G_s(t) + G_s(w-t)\} \tag{4.4}$$

Let $L_s(t)$ be a continuous and smooth fitting function of $G_s(t)$ for $t \in [1, w-1]$. Eqn. 4.3 can then be relaxed by replacing $G_s(t)$ with $L_s(t)$ to produce the solution at $w_1 = w/2$ if $L_s(t)$ is convex and $w_1 = 1$ if $L_s(t)$ is concave. We have observed, over all aggregate functions, the convexity and concavity for $L_s(\cdot)$ when approximating $G_s(\cdot)$. For example, we use 800 games of a NBA player and plot the function G_s and L_s under various aggregate functions in Figs. 4.2. In Fig. 4.2(a), G_s and L_s for



(a) (b)

Figure 4.2: Illustration of fitting function L on various aggregation functions; solid lines represent G while dotted lines represent L. (a) fitting on min and average (b)fitting on max,sum and count.

min and *avg* are presented. We can see that the fitting L_s for *min* is convex while that for *avg* is concave. Similarly, in Fig. 4.2(b), we can see that L_s is concave for count, sum and max. In the worst-case scenario, even when $L_s(\cdot)$ is neither convex nor concave, $J_s(w) < \frac{J_s(1)+(w-1)J_s(w-1))}{w}$ still holds due to Theorem 4.4.1. Thus we have the visiting-window bound stated as:

Theorem 4.4.2 (Visiting-Window Bound)**.** Given a window length $w > 1$, let $J_s(w)$ be:

$$J_s(w) = \frac{J_s(1) + (w-1)J_s(w-1)}{w} \tag{4.5}$$

then $J_s(w)$ is a visiting-window bound, i.e. $J_s(w) \geq \max_t W_s(t, w).\overline{v}$

By substitute $w_i = 1$ to the r.h.s. of Theorem 4.4.1, we see this theorem holds.

In Algorithm 6, $J_s(w)$ is computed incrementally. Initially, $J_s(1)$ is set as the single event of s with highest value. Then, as the subject s is processed, $J_s(w)$ is computed by Theorem 4.4.2. In the case that $J_s(w)$ is not pruned, we may assign $J_s(w)$ with the maximum length-w event window of s to further tighten the bound.

4.4.1.3 Unseen-window pruning

Unseen-window pruning utilizes $M_s(w)$ to check if it is necessary to evaluate any window $w' \in (w, |\mathbb{H}_s|]$. We observe that $M_s(w)$ can be efficiently estimated from the values of $J_s(w')$, where $w' \leq w$. For example, when avg is used as the aggregate function, $J_s(1)$ is obviously an upper bound for $M_s(w)$ because $J_s(1)$ is essentially the maximum event value. However, such an upper bound is very loose. By utilizing the following theorem, we can derive a tighter bound as follows:

Theorem 4.4.3 (Unseen-Window Bound)**.** Given that $J_s(1)$, ..., $J_s(w-1)$ have already been computed, let $M_s(w)$ be:

$$M_s(w) = J_s(w) + \min\{\frac{1}{2}J_s(1), \frac{w-1}{w+1}J_s(w-1)\} \tag{4.6}$$

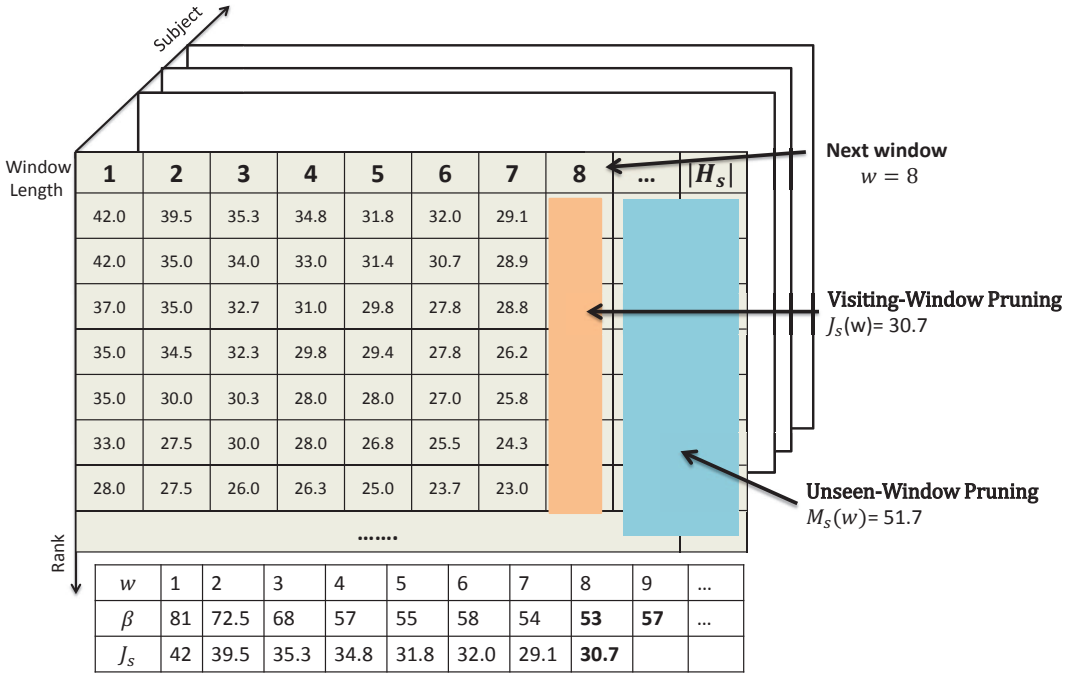
then $M_s(w)$ is an *unseen-window bound*, i.e. $M_s(w) \geq \max\{J_s(t) |\ t \in [w, |\mathbb{H}_s|]\}$

Proof. First, given any integer $k \geq 1$, we see that $J_s(kt) \leq J_s(t)$ by making use of Lemma 4.4.1 in a simple induction proof. Then, for any integer $x > w$, x can be written as $x = \lfloor \frac{x}{w} \rfloor w + x \bmod w$. Based on the subadditivity of $J_s(\cdot)$, we have:

$$\begin{aligned} J_s(x) &\leq \frac{(\lfloor \frac{x}{w} \rfloor * w)J_s(\lfloor \frac{x}{w} \rfloor * w) + (x \bmod w)J_s(x \bmod w)}{x} \\ &\leq J_s(\lfloor \frac{x}{w} \rfloor * w) + \frac{x \bmod w}{x}J_s(x \bmod w),\ \lfloor \frac{x}{w} \rfloor * w \leq x \\ &\leq J_s(w) + \frac{x \bmod w}{x}J_s(x \bmod w),\ J_s(kt) \leq J_s(t) \end{aligned}$$

On one hand, since $t*J_s(t)$ is a monotone increasing function, it follows $(x \bmod w)J_s(x \bmod w) \leq (w-1)J_s(w-1)$. Moreover, since $x \geq w+1$, we have $(x \bmod w)\frac{J_s(x \bmod w)}{x} \leq (w-1)\frac{J_s(w-1)}{w+1}$. On the other hand, $J_s(x \bmod w) \leq J_s(1)$ and $\frac{x \bmod w}{x} \leq \frac{1}{2}$ for $x > w$, we have $(x \bmod w)\frac{J_s(x \bmod w)}{x} \leq \frac{1}{2}J_s(1)$. Then it naturally leads to Theorem 4.5.1. □

To utilize $M_s(w)$, after the derivation, we check if $M_s(w)$ is no greater than any



w	1	2	3	4	5	6	7	8	9	...
β	81	72.5	68	57	55	58	54	**53**	**57**	...
J_s	42	39.5	35.3	34.8	31.8	32.0	29.1	**30.7**		

Figure 4.3: An illustration of window-level pruning techniques. Each square slice represents a set of event windows to be computed for a subject, where the column represents window length and the row represents the rank. The value in each cell is the aggregate result (i.e., $\overline{v}$) for the corresponding event window.

$\beta(w')$ with $w' > w$, i.e. $M_s(w) \leq \min\{\beta(w')|w' \in (w, |\mathbb{H}_s|]\}$. Whenever the condition holds, it is safe to stop further evaluation on subject s. Note that we may maintain an interval tree [**?**] on β to support efficient checking.

Theorem 4.4.4. Each candidate theme returned by Algorithm 6 has a rank no greater than p.

The proof is quite straightforward according to the descriptions for Algorithm 6, visiting-window and unseen-window pruning. Thus, the details are omitted.

Example 4.4.1. We use Fig. 4.3 to illustrate our pruning techniques on a subject s. Each column in the table represents a window length and the cells contain the average values among different event windows with different lengths. For example, the cell in the second row and third column refers to the second largest average value among the event windows with length 3. Algorithm 6 essentially accesses the event

windows in increasing order of the window length. Suppose we are about to estimate the upper bound for event windows with length 8. At this point, the values of β and J_s are depicted in the figure. Then, our first job is to estimate the upper bound for all the average values in the 8-th column. Based on the values of $J_s(w)$ on smaller windows, we estimate $J_s(8) = \frac{42+7*29.1}{8} = 30.7125$. Since $J_s(8) < \beta(8)$, we can safely prune the whole column, as highlighted in the figure. Afterwards, we estimate the upper bound for all the windows with length larger than 8. The value of $M_s(8)$ is then estimated as $M_s(8) = 30.7125 + \min\{21, \frac{7}{9} * 29.1\} = 51.7$. Since all the values of β are greater than $M_s(8)$, it is safe to terminate the event window enumeration.

4.4.2 Sketch Computation

After the candidate themes are obtained, the second step of the sketch discovery process is to, for each subject s, extract k themes to form sketches SK_s to maximize Eqn. 4.1 (i.e., g).

Our goal of optimizing g is related to the Partly Interval Set Cover (PISC) problem. The goal in PISC is to select a set of intervals which covers at least a certain percentage of elements. If no rank value is considered in g (i.e., $\alpha = 1$), the solution in [**?**] for PISC maximizes g in $O(\mathbb{H}_s^3)$ time for each subject s. However, when the rank is considered (i.e., $0 < \alpha < 1$), optimizing g becomes an open problem as stated in [26], where current polynomial time solutions remain unknown. To facilitate scalable sketch discovery, we provide an efficient $(1 - 1/e)$ approximation algorithm by exploiting the submodularity property[8] of the scoring function. Our idea is that since g is not submodular, it is not easy to directly maximize it. However, by applying a relaxation on g, we are able to gain a submodular function g' s.t. maximizing g' would result in the same optimal solution as maximizing g. We design the function

[8]A function I is submodular if and only if given two set $A \subseteq B$ and an element $x \notin B$, then $I(A \cup \{x\}) - I(A) > I(B \cup \{x\}) - I(B))$.

of g' as follows:

$$g'(\mathbb{X}_s) = \eta_1 C'(\mathbb{X}_s) + \eta_2 R'(\mathbb{X}_s) \tag{4.7}$$

where $C'(\mathbb{X}_s)$ is the number of distinct events covered by $\mathbb{X}_s$, $R'(\mathbb{X}_s) = \Sigma_{X_s \in \mathbb{X}_s}(p - X_s.r)$, $\eta_1 = \alpha/|\mathbb{H}_s|$ and $\eta_2 = (1-\alpha)/k$. Given k, s and g, g' is uniquely defined. We have the following theorem which links g' to g:

Theorem 4.4.5. If A^* is the optimal solution wrt. g', then A^* is also the optimal solution wrt. g.

Proof. First observe that, for any set $A \subseteq \mathbb{N}_s$ of size k, (i.e., $|A| = k$), $g(A) = g'(A)$. This can be validated by substituting A into Eq. 4.1 and Eq. 4.7. Then, we prove the theorem by contradiction: If A^* is not optimal wrt. g, then $\exists B^*$ s.t. $g(B^*) > g(A^*) = g'(A^*)$. However, since $|B^*| = k$, $g'(B^*) = g(B^*) > g'(A^*)$, which contradicts with A^*'s optimality w.r.t. g'. □

Henceforth, we are able to compute sketches by maximizing g' instead of g. We then prove the *submodularity* on g' as stated below:

Theorem 4.4.6. Given a candidate theme set $\mathbb{X}_s$, $g'(\mathbb{X}_s)$ is submodular.

Proof. Note that C' is a cover function and R' is a scalar function, thus C' and R' are both submodular. Since g' a linear combination of C' and R', it is also submodular. □

By utilizing the submodularity of g', we apply the greedy algorithm [**?**] to efficiently discover the sketches, which guarantees a $(1 - 1/e)$ approximation ratio. The greedy sketch discovery scheme is presented in Algorithm 7. During each step, the algorithm picks the best candidate theme which maximizes g' among all the candidates which have not been selected yet. The algorithm stops at the k^{th} iteration.

Algorithm 7 Greedy Sketch Discovery

1: **for** $s \in S$ **do**
2: $\mathbb{N}_s \leftarrow$ candidate themes of subject s filter by Algorithm 6
3: $SK_s \leftarrow \{\}$
4: **for** $t \in [1, k]$ **do**
5: $x^* \leftarrow \text{argmax}_{x \in \mathbb{N}_s} g'(SK_s \cup \{x\})$
6: $SK_s \leftarrow SK_s \cup \{x^*\}$
7: **end for**
8: **end for**
9: **return** $SK_s \ \forall s \in S$

4.5 Online Sketch Maintenance

In the offline scenario, all the events about every subject are assumed to be available at the time of sketch computation. In contrast, in the online scenario, events arrive incrementally. Given an arrival event, our objective is to keep the sketch for each subject up-to-date. Since the event arrival speed could be very high, such step has to been done very efficiently.

Similar to the offline scenario, we maintain a WI to keep track of the p best event windows (candidate themes) for all the possible window lengths. To handle a newly arrived event $e_s(t)$, a naive solution would generate all the event windows containing $e_s(t)$, (i.e. $W_s(t, w'), w' \in (1, t]$). For each event window, WI is then updated accordingly. Afterwards, Algorithm 7 needs to be invoked to re-compute the sketches. However, there are t associated event windows for each new event $e_s(t)$. Examining all of them is too expensive to support real-time responses. Moreover, Algorithm 7 runs in $O(k|\mathbb{N}_s|)$ time for each affected subject, which imposes further performance challenge.

To achieve instant sketch update, we propose two techniques - *window pruning* and *sketch maintenance*. Window pruning tries to reduce the number of windows evaluated in generating candidate themes. After obtaining the candidate themes, we need to update the sketches affected. As we shall see later, given a candidate theme

$N_s(t, w)$, not only the sketch for subject s but also the sketches for other subjects could be affected. Although in the offline scenario we provided a solution with a $(1 - 1/e)$ approximation, to maintain sketches with the same approximation ratio is difficult in the online scenario [**?**,5]. Instead, we propose a 1/8-approximation solution for sketch maintenance which only takes $O(k)$ time to perform the sketch update.

Algorithm 8 Online Query

Input: $e_s(t) \leftarrow$ arrival event
1: $WI()$ p best event windows for each window length
2: $\beta()$ smallest score in WI for each window length
3: **for** $w \in 1, ..., t$ **do**
4: **if** $W_s(t, w)$ can be added to $WI(w)$ **then**
5: update $\beta(w)$, $J_s(w)$, compute $N_s(t, w)$
6: UpdateSketches($N_s(t, w)$)
7: **end if**
8: $P_s(w) = \frac{w}{w+1} W_s(t, w).\overline{v} + \frac{t-w}{w+1} J_s(t - w)$
9: **break if** $P_s(w) \leq \max\{\beta(w') | w < w' \leq t\}$
10: **end for**

Before we present *window pruning* and *sketch maintenance*, Algorithm 8 demonstrates the overview of our online solution against a new event $e_s(t)$. We iteratively examine event windows which contain $e_s(t)$ (i.e.,$W_s(t, w)$ in Line 4). We then update the sketches which are affected by inserting $W_s(t, w)$ into WI (Lines 5-6). Before continuing to examine the next event window w, we compute the maximum score (i.e., $P_s(w)$) of all event windows which have not been evaluated. If $P_s(w)$ is smaller than all $\beta(w'), w' \in (w, t]$, we can safely stop the process since no further event window could cause any sketches to change.

4.5.1 Online Window Pruning

Since there are t event windows associated with each new event $e_s(t)$, we wish to avoid enumerating all the possible cases. We achieve the window pruning by leveraging the online-window bound $P_s(w)$, which is the upper bound score among event windows with length greater than w. The value of $P_s(w)$ is stated as in the following theorem:

Theorem 4.5.1 (Online-Window Bound)**.** Let $W_s(t,1)$,...,$W_s(t,w)$ be the w event windows computed in Algorithm 8 containing event $e_s(t)$. Let $P_s(w)$ be:

$$P_s(w) = \frac{w}{w+1} W_s(t,w).\overline{v} + min\{\frac{t-w}{w+1} J_s(t-w), \frac{t-w}{t} J_s(1)\} \tag{4.8}$$

Where $J_s(\cdot)$ is the visiting-window bound. Then $P_s(w)$ is the online-window bound, i.e.,$P_s(w) \geq \max\{W_s(t,x).\overline{v} | w < x \leq t\}$.

Proof. First, $\forall x \in (w,t]$, we have:

$$\begin{aligned} W_s(t,x).\overline{v} &= \frac{w * W_s(t,w).\overline{v} + (x-w) * W_s(t-w, x-w).\overline{v}}{x} \\ &\leq \frac{w * W_s(t,w).\overline{v}}{w+1} + \frac{(x-w) * W_s(t-w, x-w).\overline{v}}{x} \end{aligned}$$

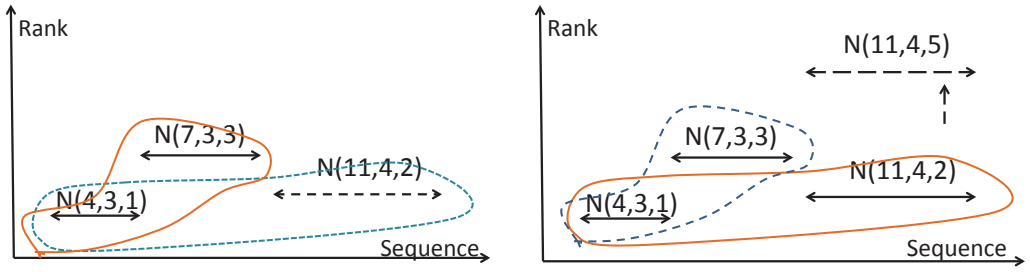
Note that $J_s(x-w) \geq W_s(t-w, x-w).\overline{v}$, and $yJ_s(y)$ monotonically increases wrt. y. It follows that $(x-w)W_s(t-w, x-w).\overline{v}/x \leq (x-w)J_s(x-w)/x \leq (t-w)J_s(t-w)/(w+1)$.

On the other hand, $J_s(1) \geq W_s(\cdot, y).\overline{v}$, for any y. Therefore, $(x-w)W_s(t-w, x-w).\overline{v}/x \leq (x-w)J_s(1)/x \leq (t-w)J_s(1)/t$. Combining the above deductions, it follows that:

$$\frac{(x-w)W_s(t-w, x-w).\overline{v}}{x} \leq min\{\frac{t-w}{w+1} J_s(t-w), \frac{t-w}{t} J_s(1)\}$$

which naturally leads to Theorem 4.5.1. □

When w is small, $\frac{t-w}{w+1} J_s(t-w)$ is too loose as $\frac{t-w}{w+1}$ is large. However, we can leverage $\frac{t-w}{t} J_s(1)$ to obtain a better bound. As w increases, $\frac{t-w}{w+1} J_s(t-w)$ eventually becomes smaller than $\frac{t-w}{t} J_s(1)$. Thus, we leverage $\frac{t-w}{w+1} J_s(t-w)$ to perform efficient pruning.



(a) Active update of new candidate theme $N(11, 4)$

(b) Passive update due to existing candidate theme $N(11, 4)$

Figure 4.4: The illustration of active updates and passive updates, the solid circle represent the original sketch, the dotted circle represent the updated sketch.

4.5.2 Sketch Maintenance

Once we obtain an event window $W_s(t, w)$ which causes changes in the $WI(w)$, two kinds of sketch maintenances may occur. The first is to update the sketch for subject s, which we refer to as *active-update*. The second is to update the sketches for other subjects. This happens when some of their candidate themes' ranks become worse due to $W_s(t, w)$. We refer to this case as *passive-update*. If these updates are not properly handled, sketches maintained for those subjects are not able to obtain an approximate ratio on their qualities. We demonstrate these updates in the following example.

Example 4.5.1. Suppose $k = 2$ and we maintain a 2-Sketch for each subject. As shown in Fig. 4.4(a), when the candidate theme $N(11, 4)$ is generated, the maintained sketch is no longer the best. This is because replacing $N(7, 3)$ would generate a better sketch. This process is referred as *active update*. In Fig. 4.4(b), $N(11, 4)$ is pushed up due to the arrival of the event about another subject; as a result, the quality of the sketch drops. We name this process as *passive update*. If passive update is not handled, the rank of $N(11, 4)$ may continue to be pushing up and may eventually be greater than p, making the entire sketch invalid. Nevertheless, it is evident that when $N(11, 4)$ degrades, replacing it with $N(7, 3)$ would result in a sketch with better quality.

A naive approach to handle online update is to run Algorithm 7 for subjects whose sketches are affected. This maintains a $(1 - 1/e)$ approximation ratio but incurs high computational overheads. In order to support efficient updates, we make a trade-off between the quality of the sketch and the update efficiency by providing a 1/8 approximation solution with only $O(k)$ candidate themes being accessed for each affecting subject.

In particular, we maintain two sets S_1 and S_2 each of size k. S_1 maintains the k best candidate themes which collectively cover the most events whereas S_2 maintains k candidate themes with best ranks. When performing an active update for a candidate theme $N_s(t, w)$, we check if $N_s(t, w)$ could replace any candidates in S_1 to generate a better cover meanwhile we select the new k best ranked candidate themes into S_2. After S_1 is updated, we perform the greedy selection among candidate themes in $S_1 \cup S_2$. During a passive update, S_1 is not affected. We simply update S_2 to be the new k best ranked candidate themes. After that, the new sketch is obtained by performing a greedy selection on $S_1 \cup S_2$. Algorithm 9 summarizes both the active and passive update.

Algorithm 9 UpdateSketches

Input: $N_s(t, w)$

Active update for the subject s
S_1: k candidate themes with best cover
S_2: k candidate themes with best ranks
$N^* \leftarrow \text{argmax}_{N \in S_1} C(S_1 \cup N_s(t, w) \setminus N)$
if $C(S_1) < C(S_1 \cup N_s(t, w) \setminus N^*)$ **then**
    $S_1 \leftarrow S_1 \cup N_s(t, w) \setminus N^*$
end if
$S_2 \leftarrow$ new k best candidates for rank scores
$greedy(S_1 \cup S_2)$

Passive update for an affecting subject s'
$S_2 \leftarrow$ new k best candidates for rank score s'
$S \leftarrow greedy(S_1 \cup S_2)$

Given our sketch update strategies, we are now ready to prove the approximation

ratio for the maintained sketches.

Theorem 4.5.2 (Approximation Ratio for Sketch Update)**.** Each sketch maintained by Algorithms 8 achieves an at least 1/8 approximation to optimal solution.

Proof. First, we observe that S_2 always keep the candidates with optimal ranks. Second, we note that S_1 maintains the candidates with 1/4 optimality as shown in [**?**].

Let OPT_C be the optimal solution for k best covering of s's events; Let $C()$ be the number of events a set of candidate theme covers. Similarly, let OPT_R be the optimal solution for k best ranked candidate themes of s; Let $R()$ be the summation of ranks of all members in a candidate theme set. Let S_s^* be the optimal sketch of subject s. Intuitively, we have the following:

$$g'(S_s^*) \leq \eta_1 C(OPT_C) + \eta_2 R(OPT_R)$$

Since $C(S_1) \geq 1/4C(OPT_C)$ and $R(S_2) = R(OPT_R)$, we have the following:

$$\begin{aligned} \eta_1 C(S_1) + \eta_2 R(S_2) &\geq 1/4 * \eta_1 C(OPT_C) + \eta_2 R(OPT_R) \\ &\geq 1/4 g'(S_s^*) \end{aligned}$$

, which implies $\max\{\eta_1 C(S_1), \eta_2 R(S_2)\} \geq 1/8 g(S_s^*)$. As $g'(S_1) \geq \eta_1 C(S_1)$ and $g'(S_2) > \eta_2 R(S_2)$, it leads to:

$$\max\{g'(S_1), g'(S_2)\} > 1/8 g'(S_s^*)$$

Let SK_s be one of the sketch maintained by Algorithms 9, since the greedy algorithm is run on $S_1 \cup S_2$, $g'(SK_s) \geq max(g'(S_1), g'(S_2)) \geq 1/8 g'(S_s^*)$. As a result, our algorithm always ensures a 1/8 approximation ratio for each sketch. □

4.6 Experiments

In this section, we study our solutions for sketch discovery in both offline and online scenarios using the following four real datasets with statistics being summarized in Table 4.4.

NBA[9] contains the game records for each NBA player from year 1985 to 2013. Among all the records, we pick $1,000$ players with at least 200 game records. In total, we obtain a dataset with 569K events.

POWER [**?**] contains the electricity usage for 370 households between Dec. 2006 and Nov. 2010. Each household is treated as a subject with the daily power usage as an event. In total, there are 1.4M events.

PEMS [20] contains the occupancy rate of freeway in San Francisco bay area from Jan. 2008 to Mar. 2009. Each freeway is a subject with the daily occupancy rate as an event. The dataset contains 963 freeways with 5.7M events.

STOCK contains the hourly price tick for 318 stocks from Mar. 2013 to Feb. 2015. The dataset is crawled from Yahoo! Finance[10] and contains 2.3M events.

Table 4.4: Statistics of datasets used in experiments

DataSet	Total Events	Total Subjects	Longest Window
NBA	569,253	1,015	1,476
POWER	1,480,000	370	4,000
PEMS	5,798,918	963	6,149
STOCK	2,326,632	318	10,420

In our efficiency study, we evaluate three parameters: 1) $p \in [20, 200]$, which refers to the worst rank for an event window to become a candidate theme, 2) $k \in [20, 100]$, which refers to the number of candidate themes in a sketch and 3) $h \in [20, 100]$, which refers to the percentage of historical events for scalability test, i.e. $|\mathbb{H}| \cdot h\%$ events are used in the experiments. We use $p = 200$, $h = 100$ and $k = 20$ as the default values.

[9] http://www.nba.com

[10] https://finance.yahoo.com/

All the experiments are conducted on a desktop machine equipped with an Intel i7 Dual-Core 3.0GHz CPU, 8GB memory and 160 GB hard drive. All algorithms are developed using Java 7.

4.6.1 Offline Sketch Discovery

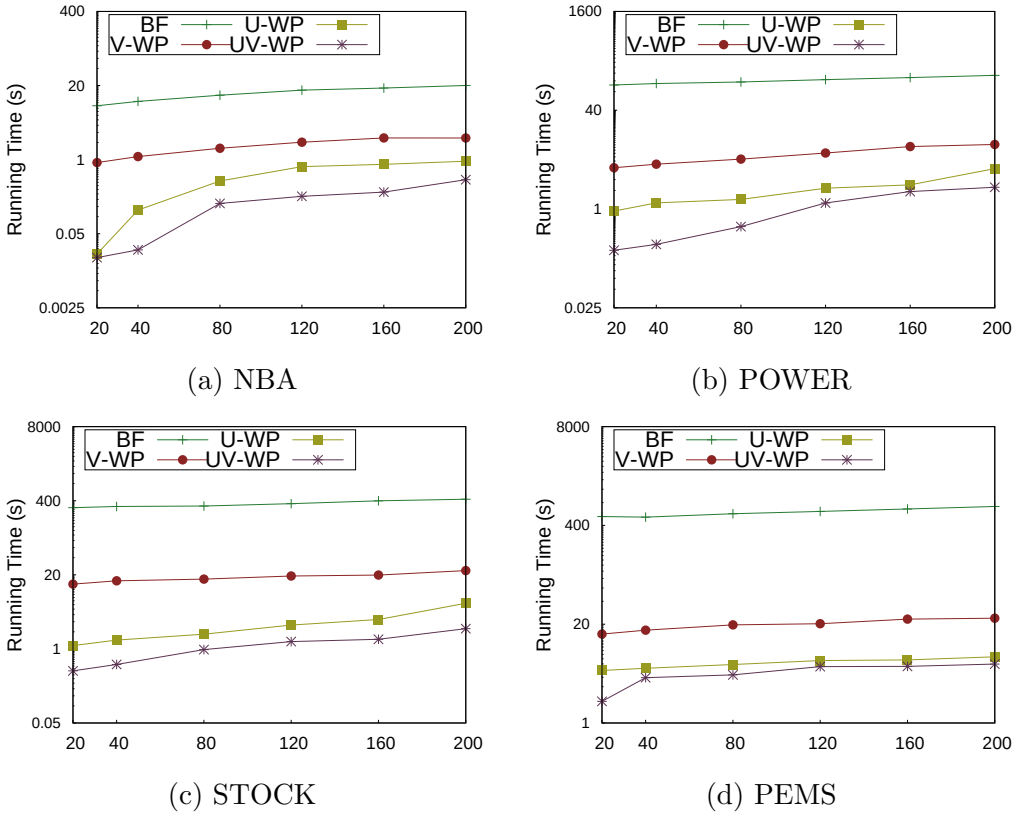
Our offline sketch discovery algorithms consist of two functional components, *Candidate Theme Generation* and *Sketch Discovery*. In the candidate theme generation, we report the performance with varying p and h. In the sketch discovery, we report the performance with varying k.

4.6.1.1 Candidate theme generation algorithms

To evaluate the performance, we design the following four methods for comparison:
Brute-Force (BF): BF exhaustively computes and compares for each subject all the possible window lengths.
Visiting-Window Pruning (V-WP): V-WP only adopts the *visiting-window* bound for pruning.
Unseen-Window Pruning (U-WP): U-WP only adopts the *unseen-window* bound for pruning.
Unseen+Visting Window Pruning (UV-WP): UV-WP adopts both *unseen-window* pruning and *visiting-window* pruning.

4.6.1.2 Candidate themes generation with varying p

The running time of the four algorithms in candidate theme generation wrt. p is shown in Fig. 4.5. It is evident that when p increases, more candidate themes are qualified and thus all four algorithms require more computation time. The effect of the two proposed window-based pruning techniques can also be observed from the figure. The insight is that the unseen-window pruning plays a more important role



(a) NBA (b) POWER

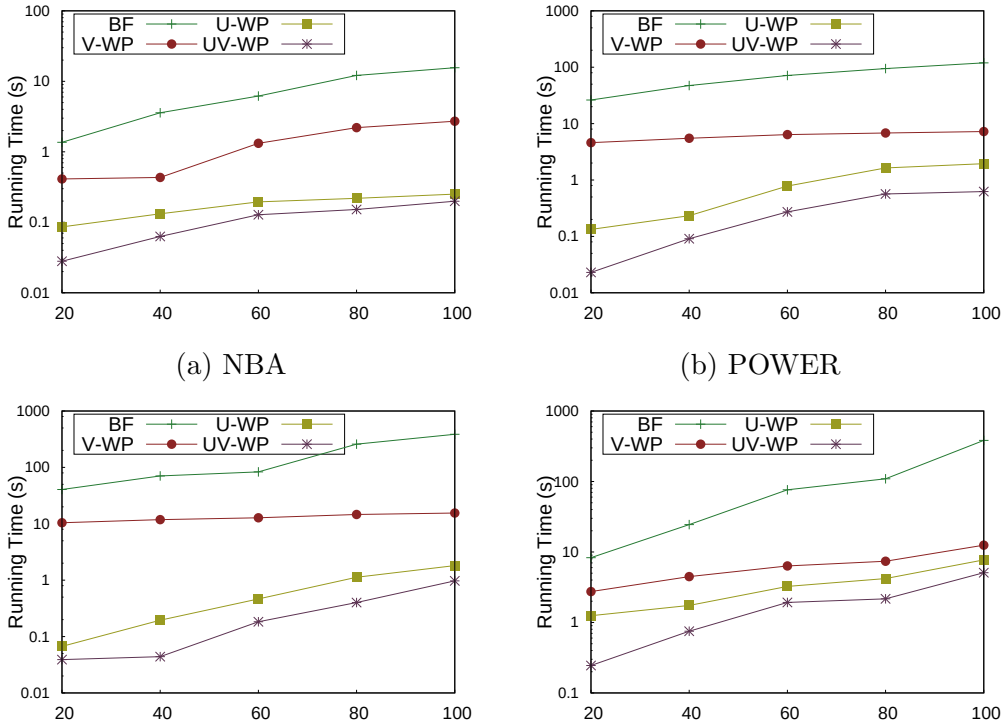
(c) STOCK (d) PEMS

Figure 4.5: Candidate theme generation in the offline scenario with varying p.

in reducing the running time. Furthermore, when both pruning techniques are used, our method achieves at least two orders of magnitude of performance improvement.

4.6.1.3 Candidate theme generation with varying h

We then study the performance of four algorithms wrt. the number of events and report the results in Fig. 4.6. As presented in the figures, when h increases, the running time for all the four algorithms also increases. This is because more event windows need to be evaluated. Again, pruning-based methods are much efficient than the baseline method. When both pruning methods are adapted, our method obtains hundreds of times faster than the baseline method.



(a) NBA (b) POWER

(c) STOCK (d) PEMS

Figure 4.6: Candidate theme generation in the offline scenario with varying h.

4.6.1.4 Sketch discovery with varying k

After candidate themes are generated, we greedily find the k-sketch for each subject. Here, we study the effect of k on the performance of the greedy algorithm. The results on four datasets are presented in Table 4.5. The table indicates that for all four datasets, as k increases, the running time of the greedy algorithm proportionally increases. This is because the complexity of the greedy algorithm is $O(k\Sigma_s|\mathbb{N}_s|)$. Since PEMS is the largest dataset with highest $\Sigma_s|\mathbb{N}_s|$, greedy algorithm performs worst on PEMS. As we observe that, even greedy algorithm takes upto 400 seconds on PEMS, the performance of the exact solution with quadratic complexity is not acceptable. This confirms the necessity of adapting the approximation algorithm.

Table 4.5: Sketch discovery with varying k in (ms)

k	20	40	60	80	100
NBA	9,097	13,345	17,500	21,597	30,769
POWER	36,297	53,513	69,300	86,603	122,856
STOCK	63,679	93,415	122,500	151,179	215,386
PEMS	138,224	206,820	283,190	353,766	491,000

4.6.2 Online Sketch Maintenance

In the online scenario, we evaluate the following four algorithms to make performance comparisons:

Sketch Computing (SC): SC examines all event windows generated from a fresh event. Whenever there is an update in any subject's sketch, Algorithm 7 will be invoked. To improve efficiency, the updates are processed in a batch manner, i.e., multiple updates on the same subject will be batched and processed by calling Algorithm 7 once.

Sketch with Early Termination (SET): SET adopts "Online Window Bound" in Theorem 4.5.1 for early termination.

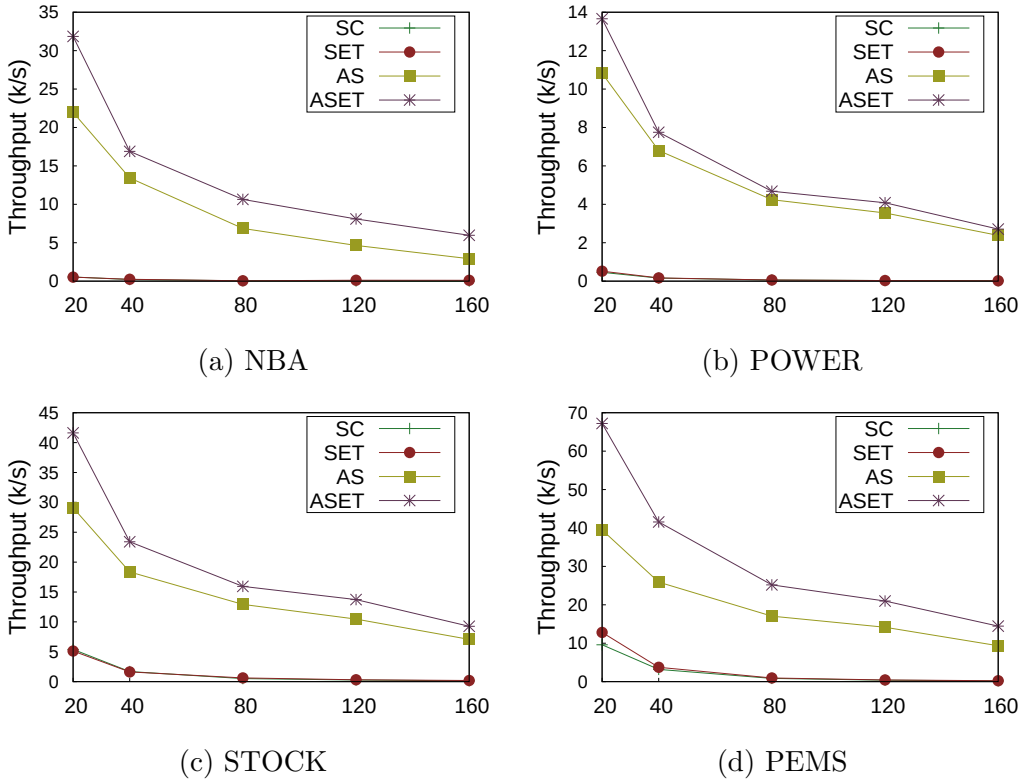
Approx. Sketch (AS): AS is similar to *SC* except that it only computes the approximate sketches.

Approx. Sketch with Early Termination (ASET): ASET computes the approximate sketches with early termination, as shown in Algorithm 8.

In the online setting, we are more interested in evaluating the throughput of algorithms. We report the performance wrt. p, k and h.

4.6.2.1 Query throughput with varying p

We increase p from 10 to 200 and the throughput results are shown in Fig. 4.7. Figs. 4.7 (a)-(d) show that all four datasets demonstrate similar patterns. As p increases, the throughput of the four algorithms drops. This is because as p increases,



(a) NBA (b) POWER

(c) STOCK (d) PEMS

Figure 4.7: Throughput in online scenario with varying p.

the time to maintain the p best candidate themes as well as to update the sketch increases. However, algorithms adapting *online window bound* have higher throughput than their counterparts. This is because with early termination, fewer candidate themes are generated. We can also see that SC and ASC run very slow in the online setting. This is because they need to invoke Algorithm 7 upon every update. This confirms the necessity of our approximation method as $ASET$ achieves up to 500x boosts as compared to SC.

4.6.2.2 Query throughput with varying k

We then evaluate how the throughput varies wrt. k. The results are presented in Fig. 4.8. The figures tell similar patterns as Figs. 4.7. First, as k increases, the throughput of all the four algorithms decreases. This is because as k becomes large, more operations are needed for maintaining the sketch. Second, the throughput of

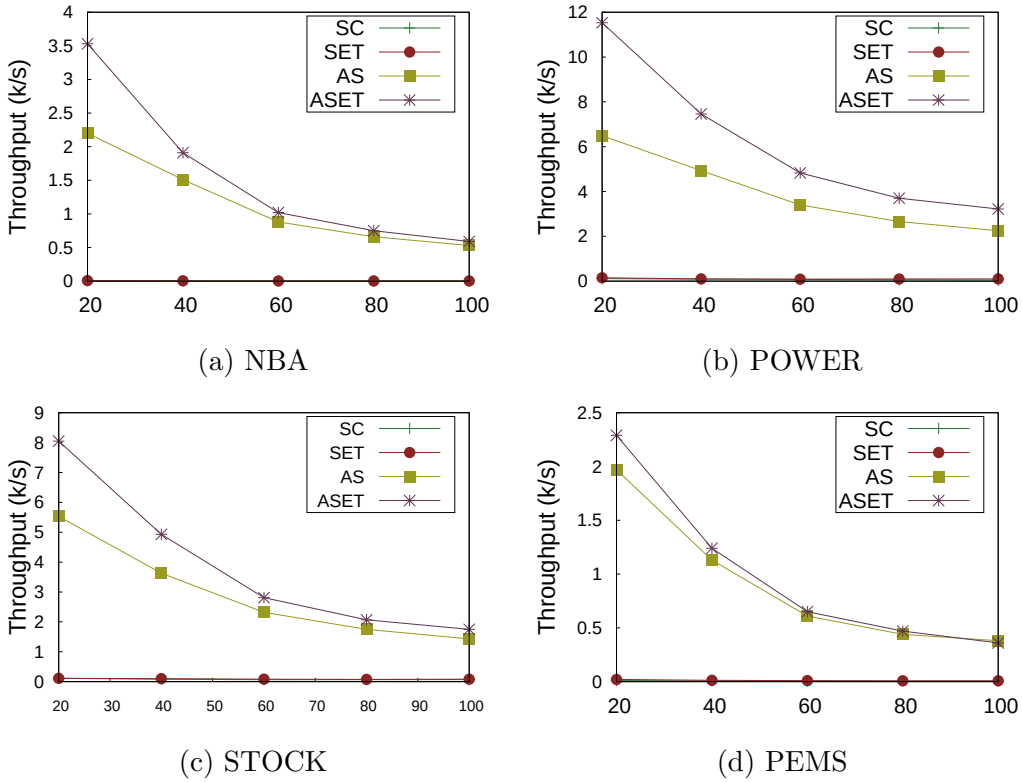
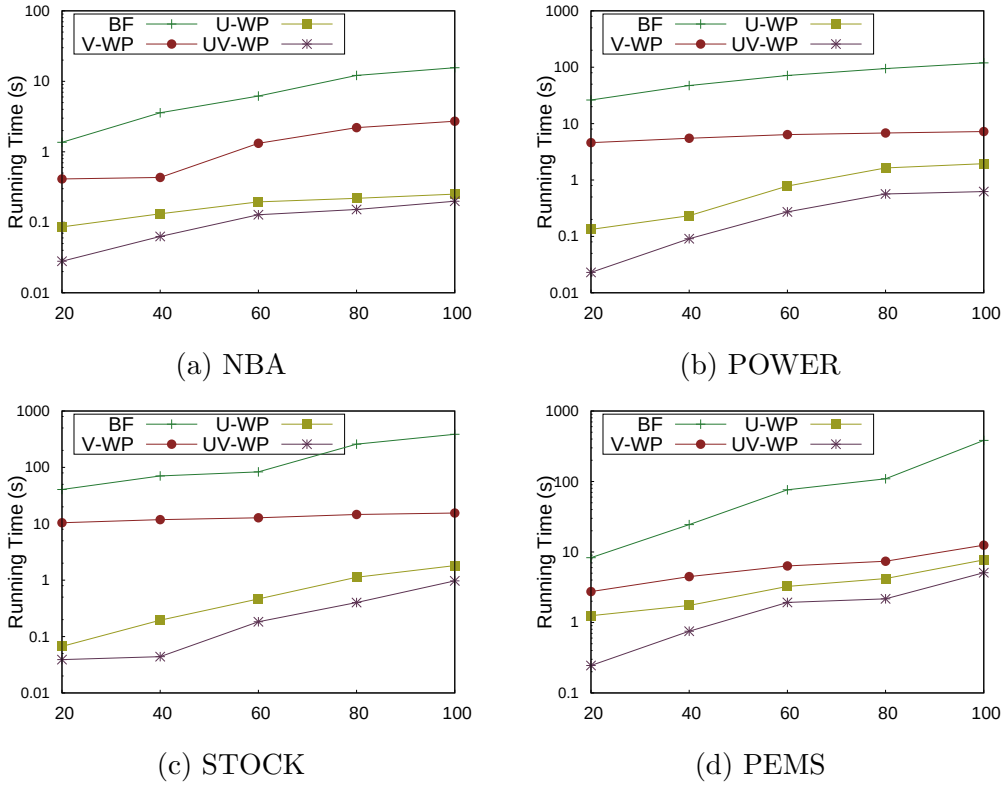


Figure 4.8: Throughput in online scenario with varying k.

SC and SET are order of magnitudes smaller than AS and $ASET$. This is because of the repetitive calling of Algorithm 7 for each arrival event, which heavily depends on k. We observe that in some datasets (e.g., Fig. 4.8 (a)) there is a hundreds time boost for $ASET$ as compared to SC.

4.6.2.3 Query throughput with varying h

Finally we study the effect of h in affecting algorithm throughput online. We change h from 20 to 100, and the results are in Fig. 4.9. As shown in the figures, when h increases, the throughput of the four algorithms steady drops. This is because as h increases, $|\mathbb{H}_s|$ for each subject increases. Therefore, in Algorithm 8, more time is needed to process each event window. We notice that $ASET$ has a flatter slope than AS; this dues to the benefit from the prunings of *online window bound*.



(a) NBA (b) POWER

(c) STOCK (d) PEMS

Figure 4.9: Throughput in online scenario with varying h.

4.6.3 Comparison with Other Techniques

We also compare both the performance and effectiveness of our sketch discovery with prominent streak technique [69] (denoted by skyline method). Due to the mismatched goal of sketch and skyline, these experiments only serve as references.

To study the efficiency, we first modify the skyline technique in [69] to consider ranks. In the offline scenario, the rank is computed using BF method, while in the online scenario, we directly keep the WI and the rank can be easily maintained. We use the default parameters settings for both scenarios. Figs. 4.10 (a)(b) show the running time comparisons under all four datasets. We can see that, when trying to adapt for rank-awareness, naively computing the rank on skyline method incurs high overheads. For instance, as presented in Fig. 4.10 (a), in the offline scenario, the running time of skyline method is order of magnitude slower than sketch method.

This supports the use of pruning techniques in computing the ranks. Similarly, as shown in Fig. 4.10 (b), in the online scenario, our sketch method still has a much better throughput. Although skyline method is free from passive update, as it is computationally expensive to maintain online skyline points, we still observe an upto 5 times boosts of sketch method.

To study the effectiveness, we conduct a user study over Amazon Mechanical Turk [11] to evaluate the attractiveness of the k-sketch. For our method, we set $p = 200$ to allow more candidate themes and set $\alpha = 0.5$ to pay equal attention to strikingness and diversity. For the skyline method, due to the overwhelming skyline points generated for each subject, we propose three augmented methods to pick k of them. In total, we have the following five algorithms to compare with:

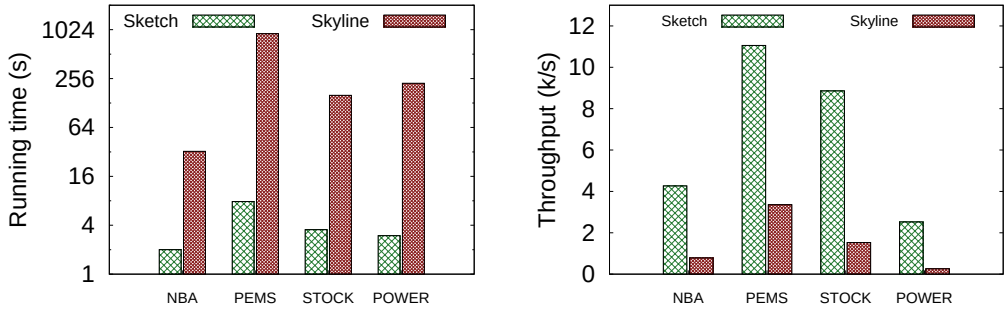
1. SK: selects the k-sketch for each player generated by offline sketch discovery method.
2. SK_a: selects the k-sketch for each player generated by online approximate sketch method;
3. SY_s: randomly selects k streaks for each player from the bunch of skylines generated by [69];
4. SY_m: selects k streaks sorted by freshness of the events;
5. SY_r: selects k rank-attached streaks sorted by the scoring function proposed in this paper;

In our experimental setting, we use the NBA dataset and set $k = 20$ to generate 20 news themes for each player. Each news theme is in the following format:

[2003-04-14]: Michael-Jordan scored an average of 30.30 pts for 989 straight games,which is No. 1 in NBA history!!

Thus, each algorithm results in 20 news themes for each player. We design each job in AMT to contain 20 questions. For each question, we randomly pick one unchosen

[11] https://requester.mturk.com



(a) Offline Scenario (b) Online Scenario

Figure 4.10: Efficiency comparison with existing technique

news theme from each algorithm. So each question contains 5 news and we require users to pick the most attractive one in each question. We received responses from 202 participants who have knowledge in NBA[12]. Then, for each algorithm, we count the frequency of being selected as the most attractive and report the percentage results in the pie chart in Fig. 4.11.

The charts clearly shows that SK is the most effective method as 45% of its generated news are considered as the most attractive. Interestingly, we observe that there are respondents who choose other comparing methods as their favorites. Besides human noise, we note that a portion of the news themes, which get votes and are generated by the comparing methods, overlap with news themes produced by our sketch based approach. In fact, SK_a overlaps 52% of event windows with SK. Thus it ranked as the second most attractive. Whereas SY_s overlaps only 10% event windows with SK, it is ranked last. The chart also shows that the ranking based on freshness SY_m can slightly improve the attractiveness compared with SY_s. However, when applied with our proposed ranking function(i.e., SY_r), the number of respondents preferring the ranked skyline results increases dramatically, nearly two times of the original number of respondents. This also implies the effectiveness of our scoring function.

[12]In AMT, we are able to request respondents with certain qualifications, i.e. knowledgeable in NBA.

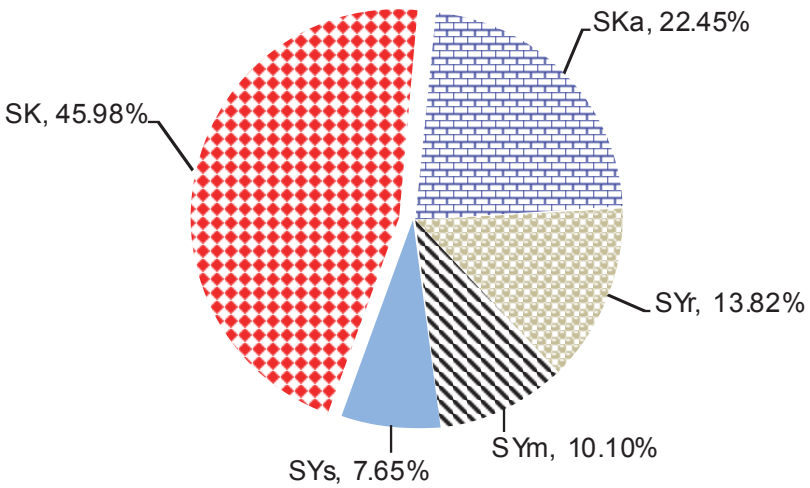


Figure 4.11: Percentage of news considered as the most attractive.

4.7 Conclusion and Future works

In this paper, we look at the problem of automatically discovering striking news themes from sequenced data. We group neighbourhood events into windows and propose a novel idea of *ranking* to represent candidate news themes. We then formulate the *sketch discovery* problem which aims to select the k best new themes as the sketch for each subject. We study the sketch discovery problem in both offline and online scenarios, and propose efficient solutions to cope each scenario. In particular, we design novel window-level pruning techniques and a (1-1/e) greedy algorithm to achieve efficient sketch discovery in offline. Then we design a 1/8 approximation algorithm for the online sketch discovery. Our comprehensive experiments demonstrate the efficiency of our solutions and a human study confirms the effectiveness of sketch discovery in news reporting.

Our work opens a wide area of research in computational journalism. In our next step, we aim to extend the event sources from sensor data to non-schema data such as tweets in social networks. We also aim to investigate multi-subject models to automatically generate news themes across different subjects.

Bibliography

[1] Rakesh Agrawal, Sreenivas Gollapudi, Alan Halverson, and Samuel Ieong. Diversifying search results. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, pages 5–14. ACM, 2009.

[2] Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499, 1994.

[3] James Allan, Ron Papka, and Victor Lavrenko. On-line new event detection and tracking. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 37–45. ACM, 1998.

[4] Htoo Htet Aung and Kian-Lee Tan. Discovery of evolving convoys. In *International Conference on Scientific and Statistical Database Management*, pages 196–213. Springer, 2010.

[5] Baruch Awerbuch, Yossi Azar, Amos Fiat, and Tom Leighton. Making commitments in the face of uncertainty: How to pick a winner almost every time. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 519–530. ACM, 1996.

[6] Jie Bao, Yu Zheng, David Wilkie, and Mohamed F Mokbel. A survey on recommendations in location-based social networks. *ACM Transaction on Intelligent Systems and Technology*, 2013.

[7] Srikanth Bellamkonda, Hua-Gang Li, Unmesh Jagtap, Yali Zhu, Vince Liang, and Thierry Cruanes. Adaptive and big data scale parallel execution in oracle. *Proceedings of the VLDB Endowment*, 6(11):1102–1113, 2013.

[8] Michael A Bender, Martín Farach-Colton, Giridhar Pemmasani, Steven Skiena, and Pavel Sumazin. Lowest common ancestors in trees and directed acyclic graphs. *Journal of Algorithms*, 57(2):75–94, 2005.

[9] Allan Borodin, Hyun Chul Lee, and Yuli Ye. Max-sum diversification, monotone submodular functions and dynamic updates. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI symposium on Principles of Database Systems*, pages 155–166. ACM, 2012.

[10] Thorsten Brants, Francine Chen, and Ayman Farahat. A system for new event detection. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, SIGIR '03, pages 330–337, New York, NY, USA, 2003. ACM.

[11] Erica J Briscoe, D Scott Appling, Rudolph L Mappus IV, and Heather Hayes. Determining credibility from social network structure. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 1418–1424. ACM, 2013.

[12] Andrei Z Broder, Steven C Glassman, Mark S Manasse, and Geoffrey Zweig. Syntactic clustering of the web. *Computer Networks and ISDN Systems*, 29(8):1157–1166, 1997.

[13] Ronald S Burt. *Structural holes: The social structure of competition*. Harvard university press, 2009.

[14] Juan Miguel Campanario. Empirical study of journal impact factors obtained using the classical two-year citation window versus a five-year citation window. *Scientometrics*, 87(1):189–204, 2011.

[15] Yu Cao, Chee-Yong Chan, Jie Li, and Kian-Lee Tan. Optimization of analytic window functions. *Proceedings of the VLDB Endowment*, 5(11):1244–1255, 2012.

[16] Chen Chen, Xifeng Yan, Feida Zhu, Jiawei Han, and S Yu Philip. Graph olap: Towards online analytical processing on graphs. In *2008 Eighth IEEE International Conference on Data Mining*, pages 103–112. IEEE, 2008.

[17] Lisi Chen and Gao Cong. Diversity-aware top-k publish/subscribe for text stream. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 347–362. ACM, 2015.

[18] James Cheng, Silu Huang, Huanhuan Wu, and Ada Wai-Chee Fu. Tf-label: a topological-folding labeling scheme for reachability querying in a large graph. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 193–204. ACM, 2013.

[19] James Cheng, Zechao Shang, Hong Cheng, Haixun Wang, and Jeffrey Xu Yu. K-reach: who is in your small world. *Proceedings of the VLDB Endowment*, 5(11):1292–1303, 2012.

[20] Tom Choe, Alexander Skabardonis, and Pravin Varaiya. Freeway performance measurement system: operational analysis tool. *Transportation Research Record: Journal of the Transportation Research Board*, (1811):67–75, 2002.

[21] Sarah Cohen, Chengkai Li, Jun Yang, and Cong Yu. Computational journalism: A call to arms to database researchers. In *CIDR*, volume 2011, pages 148–151, 2011.

[22] Artur Czumaj, Mirosław Kowaluk, and Andrzej Lingas. Faster algorithms for finding lowest common ancestors in directed acyclic graphs. *Theoretical Computer Science*, 380(1):37–46, 2007.

[23] Lianghao Dai, Jar-der Luo, Xiaoming Fu, and Zhichao Li. Predicting offline behaviors from online features: an ego-centric dynamical network approach. In *Proceedings of the First ACM International Workshop on Hot Topics on Interdisciplinary Social Networks Research*, pages 17–24. ACM, 2012.

[24] David H Douglas and Thomas K Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, 1973.

[25] Marina Drosou and Evaggelia Pitoura. Diverse set selection over dynamic data. *IEEE Transactions on Knowledge and Data Engineering*, 26(5):1102–1116, 2014.

[26] Katherine Edwards, Simon Griffiths, and William Sean Kennedy. Partial interval set cover–trade-offs between scalability and optimality. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 110–125. Springer, 2013.

[27] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[28] Joachim Gudmundsson and Marc van Kreveld. Computing longest duration flocks in trajectory data. In *Proceedings of the 14th annual ACM international symposium on Advances in geographic information systems*, pages 35–42. ACM, 2006.

[29] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *ACM Sigmod Record*, volume 29, pages 1–12. ACM, 2000.

[30] Naeemul Hassan, Afroza Sultana, You Wu, Gensheng Zhang, Chengkai Li, Jun Yang, and Cong Yu. Data in, fact out: automated monitoring of facts by factwatcher. *Proceedings of the VLDB Endowment*, 7(13):1557–1560, 2014.

[31] Clyde W Holsapple and Wenhong Luo. A citation analysis of influences on collaborative computing research. *Computer Supported Cooperative Work (CSCW)*, 12(3):351–366, 2003.

[32] Hoyoung Jeung, Man Lung Yiu, Xiaofang Zhou, Christian S Jensen, and Heng Tao Shen. Discovery of convoys in trajectory databases. *Proceedings of the VLDB Endowment*, 1(1):1068–1080, 2008.

[33] Ryota Jinno, Kazuhiro Seki, and Kuniaki Uehara. Parallel distributed trajectory pattern mining using mapreduce. In *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on*, pages 269–273. IEEE, 2012.

[34] Panos Kalnis, Nikos Mamoulis, and Spiridon Bakiras. On discovering moving clusters in spatio-temporal data. In *International Symposium on Spatial and Temporal Databases*, pages 364–381. Springer, 2005.

[35] Ingrid M Keseler, Julio Collado-Vides, Socorro Gama-Castro, John Ingraham, Suzanne Paley, Ian T Paulsen, Martín Peralta-Gil, and Peter D Karp. Ecocyc: a comprehensive database resource for escherichia coli. *Nucleic acids research*, 33(suppl 1):D334–D337, 2005.

[36] YongChul Kwon, Magdalena Balazinska, Bill Howe, and Jerome Rolia. Skewtune: mitigating skew in mapreduce applications. In *Proceedings of the 2012 ACM*

SIGMOD International Conference on Management of Data, pages 25–36. ACM, 2012.

[37] Patrick Laube, Marc van Kreveld, and Stephan Imfeld. Finding remodetecting relative motion patterns in geospatial lifelines. In *Developments in spatial data handling*, pages 201–215. Springer, 2005.

[38] Srivatsan Laxman, PS Sastry, and KP Unnikrishnan. A fast algorithm for finding frequent episodes in event streams. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 410–419. ACM, 2007.

[39] Xiaohui Li, Vaida Ceikute, Christian S Jensen, and Kian-Lee Tan. Effective online group discovery in trajectory databases. *IEEE Transactions on Knowledge and Data Engineering*, 25(12):2752–2766, 2013.

[40] Xiaohui Li and Tan Kian-Lee. *Managing moving objects and their trajectories*. PhD thesis, National University of Singapore, 2013.

[41] Xuefei Li, Hongyun Cai, Zi Huang, Yang Yang, and Xiaofang Zhou. Social event identification and ranking on flickr. *World Wide Web*, 18(5):1219–1245, 2015.

[42] Yuxuan Li, James Bailey, and Lars Kulik. Efficient mining of platoon patterns in trajectory databases. *Data & Knowledge Engineering*, 100:167–187, 2015.

[43] Zhenhui Li, Bolin Ding, Jiawei Han, and Roland Kays. Swarm: Mining relaxed temporal moving object clusters. *Proceedings of the VLDB Endowment*, 3(1-2):723–734, 2010.

[44] Zhenhui Li, Bolin Ding, Jiawei Han, Roland Kays, and Peter Nye. Mining periodic behaviors for moving objects. In *Proceedings of the 16th ACM SIGKDD*

international conference on Knowledge discovery and data mining, pages 1099–1108. ACM, 2010.

[45] Huaiyu Harry Ma, Steven Gustafson, Abha Moitra, and David Bracewell. Ego-centric network sampling in viral marketing applications. In *Mining and Analyzing Social Networks*, pages 35–51. Springer, 2010.

[46] Nan Ma, Jiancheng Guan, and Yi Zhao. Bringing pagerank to the citation analysis. *Information Processing & Management*, 44(2):800–810, 2008.

[47] Heikki Mannila, Hannu Toivonen, and A Inkeri Verkamo. Discovery of frequent episodes in event sequences. *Data mining and knowledge discovery*, 1(3):259–289, 1997.

[48] Peter V Marsden. Egocentric and sociocentric measures of network centrality. *Social networks*, 24(4):407–422, 2002.

[49] Jayanta Mondal and Amol Deshpande. Eagr: Supporting continuous ego-centric aggregate queries over large dynamic graphs. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, pages 1335–1346. ACM, 2014.

[50] Jian Pei, Jiawei Han, Runying Mao, et al. Closet: An efficient algorithm for mining frequent closed itemsets. In *ACM SIGMOD workshop on research issues in data mining and knowledge discovery*, volume 4, pages 21–30, 2000.

[51] Afroza Sultana, Naeemul Hassan, Chengkai Li, Jun Yang, and Cong Yu. Incremental discovery of prominent situational facts. In *2014 IEEE 30th International Conference on Data Engineering*, pages 112–123. IEEE, 2014.

[52] Nikolaj Tatti and Boris Cule. Mining closed strict episodes. *Data Mining and Knowledge Discovery*, 25(1):34–66, 2012.

[53] Yuanyuan Tian, Richard A Hankins, and Jignesh M Patel. Efficient aggregation for graph summarization. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 567–580. ACM, 2008.

[54] Virginia Vassilevska and Ali Pinar. Finding nonoverlapping dense blocks of a sparse matrix. *Lawrence Berkeley National Laboratory*, 2004.

[55] Jeroen B.P. Vuurens, Arjen P. de Vries, Roi Blanco, and Peter Mika. Online news tracking for ad-hoc information needs. In *Proceedings of the 2015 International Conference on The Theory of Information Retrieval*, ICTIR '15, pages 221–230, New York, NY, USA, 2015. ACM.

[56] Brett Walenz, You Will Wu, Seokhyun Alex Song, Emre Sonmez, Eric Wu, Kevin Wu, Pankaj K Agarwal, Jun Yang, Naeemul Hassan, Afroza Sultana, et al. Finding, monitoring, and checking claims computationally based on structured data. In *Computation+ Journalism Symposium*, 2014.

[57] Brett Walenz and Jun Yang. Perturbation analysis of database queries. *Proc. VLDB Endow.*, 9(14):1635–1646, October 2016.

[58] Jianyong Wang, Jiawei Han, and Jian Pei. Closet+: Searching for the best strategies for mining frequent closed itemsets. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 236–245. ACM, 2003.

[59] Yida Wang, Ee-Peng Lim, and San-Yih Hwang. Efficient mining of group patterns from user movement data. *Data & Knowledge Engineering*, 57(3):240–282, 2006.

[60] Zhengkui Wang, Qi Fan, Huiju Wang, Kian-Lee Tan, Divyakant Agrawal, and Amr El Abbadi. Pagrol: parallel graph olap over large-scale attributed graphs. In

2014 IEEE 30th International Conference on Data Engineering, pages 496–507. IEEE, 2014.

[61] Hao Wei, Jeffrey Xu Yu, Can Lu, and Ruoming Jin. Reachability querying: An independent permutation labeling approach. *Proceedings of the VLDB Endowment*, 7(12):1191–1202, 2014.

[62] You Wu, Pankaj K Agarwal, Chengkai Li, Jun Yang, and Cong Yu. On one of the few objects. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1487–1495. ACM, 2012.

[63] Xifeng Yan, Bin He, Feida Zhu, and Jiawei Han. Top-k aggregation queries over large networks. In *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*, pages 377–380. IEEE, 2010.

[64] Hilmi Yildirim, Vineet Chaoji, and Mohammed J Zaki. Dagger: A scalable index for reachability queries in large dynamic graphs. *arXiv preprint arXiv:1301.0977*, 2013.

[65] Jin Soung Yoo and Shashi Shekhar. A joinless approach for mining spatial colocation patterns. *IEEE Transactions on Knowledge and Data Engineering*, 18(10):1323–1337, 2006.

[66] Jeffrey Xu Yu and Jiefeng Cheng. Graph reachability queries: A survey. In *Managing and Mining Graph Data*, pages 181–215. Springer, 2010.

[67] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J Franklin, Scott Shenker, and Ion Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pages 2–2. USENIX Association, 2012.

[68] Fred Zemke. What. s new in sql: 2011. *ACM SIGMOD Record*, 41(1):67–73, 2012.

[69] Gensheng Zhang, Xiao Jiang, Ping Luo, Min Wang, and Chengkai Li. Discovering general prominent streaks in sequence data. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 8(2):9, 2014.

[70] Peixiang Zhao, Xiaolei Li, Dong Xin, and Jiawei Han. Graph cube: on warehousing and olap multidimensional networks. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 853–864. ACM, 2011.

[71] Kai Zheng, Yu Zheng, Nicholas Jing Yuan, and Shuo Shang. On discovery of gathering patterns from trajectories. In *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, pages 242–253. IEEE, 2013.

[72] Yu Zheng. Trajectory data mining: an overview. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 6(3):29, 2015.

[73] Yu Zheng. Trajectory data mining: an overview. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 6(3):29, 2015.

[74] Yu Zheng, Yanchi Liu, Jing Yuan, and Xing Xie. Urban computing with taxicabs. In *Proceedings of the 13th international conference on Ubiquitous computing*, pages 89–98. ACM, 2011.

[75] Wenzhi Zhou, Hongyan Liu, and Hong Cheng. Mining closed episodes from event sequences efficiently. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 310–318. Springer, 2010.

Appendix A

Appendix

A.1 Discussions on Other Aggregate Functions

First, we shall see that supporting *sum* is equivalent to supporting *avg*. A candidate theme which has a rank under *avg* will have the same rank under *sum* as the ranking is derived by comparing all candidates with the same window length. Second, supporting *count* is equivalent to supporting *sum*. By assigning each event with value of either 1 or 0, we can apply the same pruning bounds for *sum* to support *count*. Third, supporting *max* is equivalent to supporting *min*. This is because when *max* is used as the aggregate function, we are more interested to find news themes which have smaller aggregation values. For example, "XXX stock has a maximum of $0.2 price in consecutive 10 days, which is the lowest ever". Then finding the sketches according to *max* can be derived from *min* directly by negating the event values. Therefore, we only provide bounds for *sum* and *min*, which are shown as in Table A.1:

Table A.1: Bounds for other aggregate functions

Aggregate Function	**Subadditivity**
sum	$J_s(w) \leq J_s(w_1) + J_s(w - w_1)$
min	$J_s(w) \leq max(J_s(w_1), J_s(w - w_1))$
Aggregate Function	**Visting-Window Bound**
sum	$J_s(w) = J_s(w-1) + J_s(1)$
min	$J_s(w) = J_s(w/2)$
Aggregate Function	**Unseen-Window Bound**
sum	$M_s(w) = W_s(t,w).\overline{v} + J_s(t-w)$
min	$M_s(w) = max\{W_s(t,w).\overline{v}, J(1)\}$
Aggregate Function	**Online-Window Bound**
sum	$M_s(w) = W_s(t,w).\overline{v} + J_s(t-w)$
min	$M_s(w) = max\{W_s(t,w).\overline{v}, J(1)\}$

A.2 Proofs of Theorems in Chapter 5

A.2.1 Proofs of Theorem 5.4.1 and 5.4.2

Proof. Γ can be formalized using linear algebra: Let G_A be an aggregated graph, with a $n \times n$ adjacent matrix J. Since a vertex order is a permutation of J, the adjacent matrices of any reordered graphs can be represented as PJP^T where $P \in \mathbb{P}$ is a $n \times n$ *permutation matrix* [1]. In star partitioning, we assign each edge $e(i,j)$ in G_A to the lower vertex, then the matrix $B = \text{triu}(PJP^T)$ [2] represents the assignment matrix wrt. P (i.e., $b_{i,j} = 1$ if vertex j is in star Sr_i). Let vector $\vec{b}$ be the *one*[3] vector with size n. Let $\vec{c} = B\vec{b}$, then each c_i denotes the number of edges in star Sr_i. Thus, Γ can be represented as the infinity norm of $B\vec{b}$. Let Γ^* be the minimum Γ among all vertex orderings, that is

$$\Gamma^* = \min_{P \in \mathbb{P}} ||B\vec{b}||_\infty \text{ ,where } ||B\vec{b}||_\infty = \max_{1 \leq j \leq n} (c_j) \tag{A.1}$$

[1] an identity matrix with rows shuffled
[2] triu is the upper triangle part of a matrix
[3] every element in $\vec{b}$ is 1

Let B^* be the assignment matrix wrt. the optimal vertex ordering. Since we have a star for each object, by the degree-sum formula and pigeon-hole theorem, $\Gamma^* = ||B^*\vec{b}||_\infty \geq d/2$. Next, for a vertex ordering P, let $e_{i,j}$ be an entry in PAP^T. Since edges in graph G are independent, then $e_{i,j}$s are independent. Let d_i denote the degree of vertex i, since a vertex ordering does not affect the average degree, then $E[d_i] = E[\Sigma_{1\leq j\leq n} e_{i,j}] = d$. Therefore, entries in B can be written as :

$$b_{i,j} = \begin{cases} e_{i,j}, i > j \\ 0, otherwise \end{cases}$$

There are two observations. First, since $e_{i,j}$s are independent, $b_{i,j}$s are independent. Second, since $i > j$ and $e_{i,j}$s are independent. $E[b_{i,j}] = E[e_{i,j}]E[i > j] = E[e_{i,j}]/2$. As c_i is a sum of n independent 0-1 variables ($b_{i.j}$s). By linearity of expectations, we get: $E[c_i] = E[\Sigma_{1\leq j\leq n} b_{i,j}] = E[\Sigma_{1\leq j\leq n} e_{i,j}]/2 = d/2$. Let $\mu = E[c_i] = d/2$, $t = \sqrt{n \log n}$, by Hoeffding's Inequality, the following holds:

$$Pr(c_i \geq \mu + t) \leq \exp(\frac{-2t^2}{n}) = \exp(-2\log n) = n^{-2}$$

The first step holds since all $b_{i,j}$ are 0-1 variables. Next, the event $(\max_{1\leq j\leq n}(c_j) \geq \mu + t)$ can be viewed as $\cup_{c_i}(c_i \geq \mu + t)$, by Union Bound, the following holds:

$$\begin{aligned} Pr(\Gamma \geq \mu + t) &= Pr(\max_{1\leq j\leq n}(c_j) \geq \mu + t) \\ &= Pr(\cup_{c_i}(c_i \geq \mu + t)) \\ &\leq \Sigma_{1\leq i\leq n} Pr(c_i \geq \mu + t) = n^{-1} = 1/n \end{aligned}$$

Substitute back t and μ, we achieve the following concise form:

$$Pr(\Gamma \geq (d/2 + \sqrt{n \log n})) \leq 1/n$$

This indicates the probability of $(\Gamma - d/2)$ being no greater than $O(\sqrt{n \log n})$ is $(1 - 1/n)$. Since $\Gamma^* \geq d/2$, it follows with probability greater than $(1 - 1/n)$, the $\Gamma - \Gamma^*$ is no greater than $O(\sqrt{n \log n})$. When the aggregated graph is *dense* (i.e., $d \geq \sqrt{12 \log n}$), the Chernoff Bound can be used to derive a tighter bound of $O(\sqrt{\log n})$ following the similar reasoning. □

A.2.2 Proof of Theorem 5.4.8

Proof. For soundness, let P be a pattern enumerated by SPARE. For any two objects $o_1, o_2 \in P.O$, the edge $e(o_1, o_2)$ is a superset of $P.T$. By the definition of star, o_1 o_2 belong to the same cluster at every timestamps in $P.T$. As $P.T$ is a valid sequence, by the definition of GCMP, P is a true pattern. For completeness, let P be a true pattern. Let s be the object with smallest ID in $P.O$. We prove that P must be output by Algorithm 12 form Sr_s. First, based on the definition of star, every object in $P.O$ appears in Sr_s. Since $P.T$ is decomposable, then by Lemma 3 $\forall O' \subseteq O$, the time sequence of O' would not be eliminated by any `sim` operations. Next, we prove at every iteration $level \leq |P.O|$, $P.O \subset O_u$, where O_u is the forward closure. We prove by induction. When $level = 2$, it obviously holds. If $P.O \subset O_u$ at *level* i, then any subsets of $P.O$ with size i are in the candidate set. In *level* $i+1$, these subsets are able to grow to a bigger subset (in last iteration, they grow to $P.O$). This suggests that no subsets are removed by Lines 16-29. Then, $P.O \subset U_{i+1}$ holds. In summary, $P.O$ does not pruned by simplification, monotonicity and forward closure, therefore P must be returned by SPARE. □