Min-d-Occur: Ensuring Future Occurrences in Streaming Sets

Vidit Jain
Yahoo Labs
Bangalore, India

Sainyam Galhotra[*]
Dept. of Comp. Sci. and Engg.
IIT Delhi, India

Abstract

Given a set of n elements and a corresponding stream of its subsets, we consider the problem of selecting k elements that should appear in at least d such subsets arriving in the "near" future with high probability. For this min-d-occur problem, we present an algorithm that provides a solution with the success probability of at least $1 - O\left(\frac{kd\log n}{D} + \frac{1}{n}\right)$, where D is a known constant. Our empirical observations on two streaming data sets show that this algorithm achieves high precision and recall values. We further present a sliding window adaptation of the proposed algorithm to provide a continuous selection of these elements. In contrast to the existing work on predicting trends based on potential increase in popularity, our work focuses on a setting with provable guarantees.

1 Introduction

Consider a set S_n of n elements. When different sets $S^t \subseteq S_n$ are being observed at time t and may not be analyzed at a later time, we refer to these sets as *streaming sets*. This formulation of streaming sets is ubiquitous at least in the analysis of network traffic (Edwards et al., 1997), query logs (Golbandi et al., 2013), and social media (Mathioudakis and Koudas, 2010), where these sets arrive rapidly. Analyzing these streaming sets to identify historical patterns and predict future trends has been extensively studied for diverse applications including detection of intrusions (Lee et al., 2000), disease outbreak (Achrekar et al., 2011), and viral content (Weng et al., 2013). These works primarily focus on the recall and the earliness of these predictions. As a result, they are useful for detecting outliers and arranging for preventive measures (Lamb et al., 2013). However, without any lower bound on the obtained precision values, these approaches are ineffective when the false positives may have significant cost associated with them.

[*]Work done during an internship at Yahoo Labs Bangalore.

Precise predictions are necessary for several planning applications such as resource allocation. For instance, accurate estimates of future traffic may help optimize routing to reduce network latency (Padmanabhan and Mogul, 1996). Similarly, accurate estimates of search query volume may help advertisers optimize marketing budget while bidding for key phrases on a search engine (Jansen and Mullen, 2008). To this end, we study the problem of making these predictions only when we can guarantee a minimum probability of success before the time horizon Δ. We define this problem formally as follows.

Definition 1. Min-d-occur. *Given a stream of sets $S^t \subseteq S_n$ arriving at time t, min-d-occur(S_n,Δ,k) selects at most k elements $\{y_1, y_2, \ldots, y_k\} \in S_n$ at time τ such that each of these y_j appears at least d number of times in $\{S^{\tau+1}, \cdots, S^{\Delta}\}$ with probability close to* 1.

Note that a solution to this problem may select fewer than k elements when the criterion for the probabilistic guarantee is not met. The constant Δ depends on the requirements of the application domain, which can be understood through an illustrative example as follows. Consider a stream of queries issued to a search engine that may be arriving at a rate of one million queries per second. Choosing Δ close to $100K$ would correspond to making the predictions about occurrences in the next 100 milliseconds. The choice of the parameter d depends on the characteristics of the stream of sets and determines the effectiveness of the solution. For the stream of queries, let us assume that we wish to track a set of 50 popular queries (i.e., $n = 50$), which cover approximately 1% of the incoming stream. In other words, 99% of the stream would be composed of empty sets, making the solution likely to be effective only when $d \leq 1\%$ of Δ. Since these domain-specific stream characteristics are external to our problem formulation, we ignore the empty sets hereafter.

In the absence of any other assumptions about the stream characteristics, the min-d-occur problem is ill-posed. In fact, if there is a sudden drop in the frequency of the queries from the set S_n in the query stream (in the above example), there may not exist a solution set for an algorithm to predict. Using an appropriate selection of the set S_n and an assumption about the continuity of the stream statistics in the given time horizon Δ, it might be fair to assume that at least k queries will appear at least $D = 100$ times in the next 100 milliseconds. Under this additional assumption, the above min-d-occur problem becomes more useful for $d \leq D$. Using this constant integer D, we define a constrained variant as follows.

Definition 2. Constrained-min-d-occur. *Given that there exist at least k elements in S_n that appear D number of times in $\{S^1, \cdots, S^{\Delta}\}$, constrained-min-d-occur(S_n,D,Δ,k) solves min-d-occur(S_n,Δ,k).*

Below we present a randomized algorithm for the constrained-min-d-occur problem. We show that the probability of a successful prediction from this algorithm is high when $d < \frac{D}{3 \log n}$ and $k = o(n/\log n)$. This theoretical result is further validated using experiments on two streaming data sets: search query logs and hash-tags in tweets. In both of these data sets, this algorithm achieved high precision and recall values for predictions. Interestingly, this algorithm is empirically effective for larger values of d even when a corresponding lower-bound is not computed. We also present a sliding-window based adaptation of our algorithm to accommodate a practical setting where the predictions need to be updated only at regular intervals of time.

Contributions. There are two main theoretical contributions in this paper. First, we present a novel problem formulation – *constrained min-d-occur* – for making guaranteed predictions of future occurrences in streaming data. Second, we present a randomized algorithm for making predictions under this formulation and derive an effective lower-bound on its performance. These theoretical contributions are supported by empirical observations on two real, streaming data sets i.e., Twitter hash-tags and query logs of a search engine.

2 Related Work

Our problem formulation is related to online coverage problems. In the online set cover problem, Alon et al. (2009) assume that a collection of sets is known beforehand but they arrive in an online fashion. An algorithm then selects k sets from this collection that solves the max-cover problem for the sets that are already observed. Whereas our setup requires predictions to be made for the sets arriving in future. Another related coverage problem is set-streaming maximum coverage proposed by Saha and Getoor (2009). They assume an orthogonal set up where the collection of sets remain static but their elements are streaming. Their algorithm is not applicable in our setting because we need to choose the elements that are present in maximum number of sets, and *not* the sets that cover maximum number of elements. Another similar problem formulation has appeared in the operating systems literature. Awerbuch et al. (1996) studied the problem of allocating jobs to workstations to ensure a timely, successful completion of the given jobs. They presented an algorithm that performs a sequential prediction of these allocations and allows at most one job to be running at any time. That is, the second job is allocated only after the completion (*not* allocation) of the first job. It is non-trivial to adapt their algorithm to perform k simultaneous allocations, although we have borrowed useful concepts from their lower-bound analysis in the development of our solution.

There exists significant literature on statistical trend prediction in streaming data, most notably for the Twitter data. There has also been some work on tracking topics and events in these streams (Ardon et al., 2013) through an appropriate adaptation of statistical topic models (AlSumait et al., 2008), and for detecting bursts or spikes in topics (Diao et al., 2012). Goorha and Ungar (2010) study the spiking behavior of elements in an input set of elements. Their algorithm can only handle a small set of elements, thereby limiting its utility in a general practical setting. Cataldi et al. (2010) modeled the streaming elements as a graph, and employed page-rank algorithm along with an aging theory to predict the spiking elements. Mathioudakis and Koudas (2010) analyzed the sudden change in frequency patterns of these elements in conjunction with a reputation model for the origin of the streaming elements to make these predictions. Becker et al. (2011) discussed an online setting to identify events and the related tweets, but they did not make predictions for future. Similarly, there has also some work on analyzing trends in user comments (Jain and Galbrun, 2013)

Most of these algorithms are optimized for a specific application, and require significant modification to be applicable to a different setting. These algorithms are evaluated empirically using measures such as the perplexity, recall, and earliness of the predictions. Since these approaches do not formally specificy the (implicit) assumptions made about the stream character-

S_n	set of elements $\{e_1, \dots, e_n\}$
n	number of possible elements in the global set
k	number of elements to choose
x_e^t	Score of the element e at time t
y^t	Solution set constructed at time t
N	window size
w_e^t	window data structure for element e at time t

Table 1: Notation used in this paper

.

Algorithm 1 k Min-d-occur Prediction

```
Require: set of elements $S_n$, integer $k$, stream $\{S^1, \cdots, S^\Delta\}$
 1: $n \leftarrow |S_n|$.
 2: $\forall e \in S_n$, $x_e^0 \leftarrow 0$.
 3: $y \leftarrow \{\phi\}$.
 4: $j = 1$
 5: for $t = 1$ to $\Delta$ do
 6:   $\beta^t \leftarrow S^t \setminus y$
 7:   for $e \in \beta^t$ do
 8:     $x_e^t \leftarrow x_e^{t-1} + 1$.
 9:     choose $u \sim Bernoulli\left(\frac{n^{(3x_e^t - 2D)/D}}{d}\right)$.
10:     if $u = 1$ then
11:       $y_j \leftarrow e$.
12:       $j \leftarrow j + 1$.
13:       if j>k then
14:         return $y$.
15:       end if
16:     end if
17:   end for
18: end for
```

istics, it if challenging to assess their utility across different related settings, e.g., our current formulation. Also, lower-bound analyses for the prediction accuracy even for the original problem settings do not exist. In this work, we address both of these issues. We formally stated the assumptions of our prediction setting in the previous section. In the next section, we present a randomized algorithm to obtain a solution for this setting. A lower-bound analysis of the performance of this algorithm is described in the subsequent section.

3 Algorithm

Here we present a randomized algorithm that provides the constrained-min-d-occur (as defined above). In other words, this algorithm selects k elements (from S_n) that are likely to appear at least d number of times in the given future Δ occurrences. As we show below, the probability of a successful prediction from this algorithm is high when: $D > 3d \log n$ and $k = o(n/\log n)$. Later, in Section 6, we demonstrate that these assumptions hold in several practical settings for streaming data.

Algorithm 1 describes the different steps of our approach. We maintain a score x_e^t for each element $e \in S_n$ at every time time t. At time t, a new set S^t is presented to the algorithm. For each element e present in the S^t that has already not been selected, we toss a coin with probability of getting heads $= \frac{n^{\frac{3x_i^t}{D} - 2}}{d}$. If a head is observed, we add e to the selection set y. We update the score x_e^t and continue till k elements have been selected. The set y represents the selected k elements.

4 Analysis

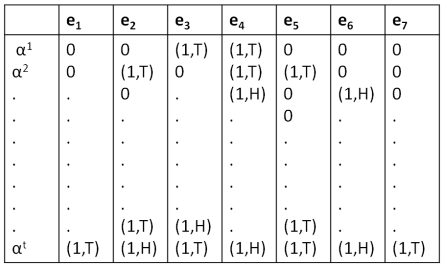
Here we prove that each of the k elements $\{y_1, y_2, \dots, y_k\}$ selected by our algorithm has a high probability of occurring at least d times in future (governed by the horizon Δ). We show this probability to be greater than $1 - O(\frac{kd \log n}{D})$.

We approach this proof by constructing a subspace S_u of probabilistic outcomes of coin-toss experiments (as illustrated in Figure 1). In this subspace, 0 denotes the absence of an element in the set arriving at a given time. Whereas the presence of a particular element is denoted by the probabilistic outcome, i.e., the toss of the coin as H (head) or T (tail). In other words, there will be one coin toss for each pair (t, i) if and only if $e_i \in S^t \setminus y^t$. The element e_{i^*} is selected as the j^{th} prediction at time t^* if and only if:

- (t^*, i^*) toss is H,
- for $t < t^*$ and $e_i \notin y^t$, (t, i) toss is T,
- for $t = t^*$ and $i < i^*$, (t, i) toss is T.

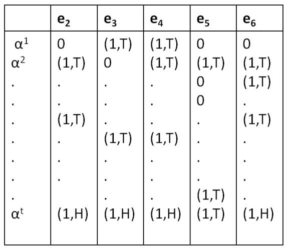
Let $S_o \subseteq S_u$ be the subspace where each of the k selected elements occur in d sets after choosing them. We refer to this subspace as the solution space. In this subspace, one H appears in each of the k different values of i (in different columns) and there are at least d such sets. Each element would appear in at least d sets after their respective coin tosses are heads.

To show that $Pr[S_o]$ is close to 1, we first choose an intermediate subspace $S_i \subseteq S$ for which the corresponding probability, i.e., $Pr[S_i]$, is close to 1. This intermediate subspace is selected such that we can construct a useful injection from S_i to S_o. To this end, we define S_i to consist of sample points for which there are at least k heads for different elements and for which the first d flips for each element are tails.



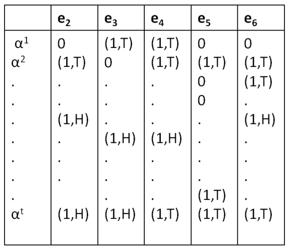
	e_1	e_2	e_3	e_4	e_5	e_6	e_7
α^1	0	0	(1,T)	(1,T)	0	0	0
α^2	0	(1,T)	0	(1,T)	(1,T)	0	0
.	.	0	.	(1,H)	0	(1,H)	0
.	.	.	.	.	0	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	(1,T)	(1,H)	.	(1,T)	.	.
α^t	(1,T)	(1,H)	(1,T)	(1,H)	(1,T)	(1,H)	(1,T)

(a) Universal space S_u



	e_2	e_3	e_4	e_5	e_6
α^1	0	(1,T)	(1,T)	0	0
α^2	(1,T)	0	(1,T)	(1,T)	(1,T)
.	.	.	.	0	(1,T)
.	.	.	.	0	.
.	(1,T)	.	.	.	(1,T)
.	.	(1,T)	(1,T)	.	.
.	.	.	.	.	.
.	.	.	.	.	.
.		.	.	(1,T)	.
α^t	(1,H)	(1,H)	(1,H)	(1,T)	(1,H)

(b) Intermediate space S_i



	e_2	e_3	e_4	e_5	e_6
α^1	0	(1,T)	(1,T)	0	0
α^2	(1,T)	0	(1,T)	(1,T)	(1,T)
.	.	.	.	0	(1,T)
.	.	.	.	0	.
.	(1,H)	.	.	.	(1,H)
.	.	(1,H)	(1,H)	.	.
.	.	.	.	.	.
.	.	.	.	.	.
.		.	.	(1,T)	.
α^t	(1,H)	(1,H)	(1,T)	(1,T)	(1,T)

(c) Solution space S_o

Figure 1: *Illustration of the probabilistic subspace for* $n = 7$. Each row represents a set α, where the absence of an element is denoted by 0; the outcome of the coin-toss for each of the other elements is shown as a tuple $(1, H)$ or $(1, T)$.

From the columns of S_u, only those elements are kept for which the first d tosses are tails to form the subspace S_i. The mapping from S_i to S_o has also been shown where the $(x_e^t - d)^{th}$ and x_e^t outcomes of coin toss are flipped depending upon the factors explained later. We start by proving a basic result about the universal space S_u.

Lemma 1. *The probability of getting tails for all coin tosses in* $S_u \leq e^{-n/2}$.

Proof. The probability of getting tails for all elements is the product of probability of getting tails for each element. The latter probability is less than the probability of getting tails for an element that is present in at least D sets. This upper bound probability is the same as the probability that there are no heads among the last d flips for the element that is available for D steps, which is at most

$$\left(1 - \frac{n^{\frac{3(D-d)}{D}-2}}{d}\right)^d \leq \left(1 - \frac{n}{2d}\right)^d \leq e^{-n/2} \quad (1)$$

because $D \geq 3d \log n$. □

The probabilistic subspace S_i consists of observation sets such that there are at least k heads for different elements and for which the first d flips for each element are tails. Now we lower bound the probability associated with this subspace.

Lemma 2. *$Pr[S_i] \geq$ 1- O(1/n).*

Proof. We found the probability of the complement of S_i, denoted by $\bar{S_i}$, to be easier to compute than for the subspace S_i. This probability is decomposed as a sum of two terms P_1 and P_2 defined below. The term P_1 denotes the probability of getting a head among the first (at most) $d \times n$ flips for each $x_i^t \leq d$. Since the probability of observing an H is at most $\frac{n^{\frac{3d}{D}-2}}{d}$,

$$P_1 \leq dn \left(\frac{n^{\frac{3d}{D}-2}}{d}\right) \leq 2/n. \quad (2)$$

Let P_2 denote the probability of observing at most $k-1$ heads, i.e.,

$$P_2 = \sum_{i=0}^{k-1} Pr_i[H], \quad (3)$$

where Pr_i[H] is the probability of observing a total i occurrences of H. Assuming i.i.d. observations for coin-tosses, we have

$$\begin{aligned} P_2 &= \sum_{i=0}^{k-1} \binom{n}{i} Pr_1[H]^i Pr_1[T]^{n-1} && (4) \\ &\leq \sum_{i=0}^{k-1} \binom{n}{i} Pr_1[T]^{n-i}, && (5) \end{aligned}$$

since $Pr_1[H] \leq 1$. Keeping only those elements for which we get tails and they appear in at least D sets, we have

$$\begin{aligned} P_2 &\leq \sum_{i=0}^{k-1} \binom{n}{i} Pr_1[T]^{k-i} && (6) \\ &\leq k \times \max\left(\binom{n}{i} \times (e^{-n/2})^{k-i}\right). && (7) \end{aligned}$$

For $k < n/2$, $\binom{n}{i} \times (e^{-n/2})^{k-i}$ is an increasing function with respect to i; the maximum value is obtained for $i = k-1$. Therefore

$$\begin{aligned} P_2 &\leq k\binom{n}{k-1} \times \left(e^{-n/2}\right)^{k-(k-1)} && (8) \\ &= k\left(\binom{n}{k-1} \times e^{-n/2}\right). && (9) \end{aligned}$$

Also, since $\binom{n}{k} \leq n^k$,

$$P_2 \leq k \times n^{k-1} e^{-n/2} \leq 1/n, \quad (10)$$

if $k \leq \left(\frac{n}{2\log n} - 1\right)$. Finally, combining Equation 2 and 10

$$\begin{aligned} Pr[S_i] &= 1 - (P_1 + P_2) && (11)\\ &= 1 - (O(2/n) + O(1/n)) && (12)\\ &= 1 - O(1/n). && (13) \end{aligned}$$

□

For each of the members of the above intermediate subspace, we construct a member of the solution space. We ensure that the probabilities of occurrences of both of these instances are similar. The next lemma provides the similarity in the probabilities of these two subspaces.

Lemma 3. $Pr[S_o] \geq (1 - O(\frac{kd\log n}{D}))Pr[S_i]$.

Proof. We construct an injection $f : S_i \rightarrow S_o$ such that $\forall s' \in S_i$,

$$Pr[f(s')] \geq \left(1 - O\left(\frac{kd\log n}{D}\right)\right) Pr[s']. \quad (14)$$

Consider an instance $s' \in S_i$. Let $\{e_{i_1}, e_{i_2}, \ldots, e_{i_k}\}$ be the elements for which the flips are the first heads in s'. Let x_{i_j} denote the number of sets in which e_{i_j} has been present and includes the set where the j^{th} heads occurred for $j \in \{1, 2, \ldots, k\}$. By definition of S_i, x_{i_j} $> d$. Let us define

$$z'_{i_j} = n^{\frac{3x_{i_j}}{D} - 2}/d \quad (15)$$

and

$$\begin{aligned} z_{i_j} &= \frac{n^{\frac{3(x_{i_j} - d)}{D} - 2}}{d} = n^{\frac{-3d}{D}} z'_{i_j} && (16)\\ &= \left(1 - O\left(\frac{d\log n}{D}\right)\right) z'_{i_j}. && (17) \end{aligned}$$

Let p refers to the sets considered in a sample space. Using these notation, we define the injection $f(s')$ as follows. If $z'_{i_j} \geq 1/2$ then all of the (p, i_j) pairs of $f(s')$ are made identical to the respective (p, i_j) of s'. Next, the $(x_{i_j} - d)^{th}$ flip of e_{i_j} is changed from tails to heads.

If $z'_{i_j} \leq 1/2$ then all of the (p, i_j) pairs of $f(s')$ are similar to (p, i_j) of s' except that the $(x_{i_j} - d)^{th}$ flip of e_{i_j} is changed from tails to heads and $x_{i_j}^{th}$ flip of e_{i_j} is changed from heads to tails.

The above process generates $f(s') \in S_o$. Without loss of generality, let us assume that for $j \in \{1, 2, \ldots, r\}$ the value of $z'_{i_j} \geq 1/2$. For $j \in \{r+1, r+2, \ldots, k\}$, let us assume $z'_{i_j} \leq 1/2$ where r is an integer $1 \leq r \leq k$. Using this notation, the ratios of probabilities in the two subspaces is given by:

$$\frac{Pr[f(s')]}{Pr[s']} = \prod_{i_j=1}^{r} \frac{z_{i_j}}{1 - z_{i_j}} \cdot \prod_{i_j=r+1}^{k} \frac{z_{i_j}}{1 - z_{i_j}} \frac{1 - z'_{i_j}}{z'_{i_j}} \quad (18)$$

For $z'_{i_j} \geq 1/2$, using Equation 17, we get

$$\begin{aligned} \frac{z_{i_j}}{1 - z_{i_j}} &= \frac{\left(1 - O\left(\frac{d\log n}{D}\right)\right) z'_{i_j}}{1 - \left(1 - O\left(\frac{d\log n}{D}\right)\right) z'_{i_j}} && (19)\\ &\geq \frac{1/2 - O\left(\frac{d\log n}{D}\right)}{1/2 + O\left(\frac{d\log n}{D}\right)} && (20)\\ &= 1 - O\left(\frac{d\log n}{D}\right). && (21) \end{aligned}$$

For $z'_{i_j} \leq 1/2$, using Equation 17, we get

$$\begin{aligned} &\left(\frac{z_{i_j}}{1 - z_{i_j}}\right)\left(\frac{1 - z'_{i_j}}{z'_{i_j}}\right) && (22)\\ \geq\ & \left(1 - O\left(\frac{d\log n}{D}\right)\right) \frac{1 - z'_{i_j}}{1 - z'_{i_j} + O\left(\frac{d\log n}{D} z'_{i_j}\right)} \\ \geq\ & 1 - O\left(\frac{d\log n}{D}\right). && (23) \end{aligned}$$

Using Equation 21 and Equation 23, the Equation 18 reduces to the following.

$$\begin{aligned} \frac{Pr[f(s')]}{Pr[s']} &\geq \prod_{i_j=1}^{k} \left(1 - O\left(\frac{d\log n}{D}\right)\right) && (24)\\ &\geq 1 - O\left(\frac{kd\log n}{D}\right). && (25) \end{aligned}$$

□

Theorem 3. *$Pr[S_o] \geq 1 - O\left(\frac{kd\log n}{D} + \frac{1}{n}\right)$.*

Proof. Using Lemma 2 and Lemma 3. □

Corollary 4. *For $k = 1$, we obtain a $2/3$ approximation, i.e., $Pr[S_o] \geq 2/3$ since $D > 3d\log n$.*

5 Sliding Window Approach

In a practical setting, the predictions about future occurrences are made continuously as the streaming sets continue to arrive. Alternatively, the predictions are updated at regular intervals of time using the sets arriving within a sliding time window of size W. The algorithm described above would require a recomputation of the frequency counts of the elements as the window is updated. To reduce the computational cost of this update, we employ a data structure proposed by Datar et al. (2002) that maintains count statistics for a stream using buckets of different sizes. Using the binary representation for the presence of an element in a streaming set, we use this data structure to calculate the approximate number of 1's in a given window. For each of these buckets, the time-stamp of the most recent 1 and the total count of 1's appearing in that bucket are maintained. As a new binary element arrives the following steps are taken.

- The time stamp of each bucket is increased by one.
- If the time stamp of a bucket exceeds the window size, the bucket is dropped.
- If the arriving element is 0, we proceed to the next element.
- If the arriving element is 1, then we create a new bucket with size 1 (this operation is referred to as add_bucket operation).
- if there exists $m/2 + 2$ buckets of same size, the oldest two buckets are merged. Here m is a predefined integer that is inversely proportional to the maximum permissible relative error ϵ, i.e., $m = \lceil 1/\epsilon \rceil$.

In this data structure, the total number of elements in each bucket is equal to the number of 1's in the window (referred to as window_score). Datar et al. (2002) show that this data structure gives an estimate of the number of 1's in the window of size W with a relative error of at most ϵ using at most $\left(\frac{m}{2}+1\right)\left(\log\left(\frac{2W}{m}+1\right)+1\right)$ buckets. $\log W + \log\log W$ bits are used per bucket and each new element is processed in O($\log W$) worst case time. At each instant, this data structure provides a count estimate in O(1) time.

Algorithm 2 use add_bucket and window_score to present the sliding window adaptation of Algorithm 1. In our problem formulation, sets $\{S^1, S^2, \ldots, S^W\}$ arrive in the window W. Let there be x elements in the union of these sets, then the input stream for the above data structure would be a permutation of x 1's and $(n-x)$ 0's. For this stream, we create n buckets at the beginning of the stream and update them as above when a new element arrives.

Algorithm 2 k Min-d-occur Prediction over a sliding window

```
Require: $S_n$, integer $k$, window size $W$.
 1: $\forall e \in S_n$, $x_e^0 \leftarrow 0$.
 2: $j = 1$
 3: for $t = 1$ to $\Delta$ do
 4:   $\beta^t \leftarrow S^t \setminus y$
 5:   for $e \in \beta^t$ do
 6:     $w_e^t \leftarrow add\_bucket(w_e^{t-1})$
 7:     $x_e^t \leftarrow window\_score(x_e^{t-1}, w_e^t)$.
 8:     choose $u \sim Bernoulli\left(\frac{n^{(3x_e^t - 2D)/D}}{d}\right)$.
 9:     if $u = 1$ then
10:       $y_j \leftarrow e$.
11:       $j \leftarrow j + 1$.
12:       if $j > k$ then
13:         return $y$.
14:       end if
15:     end if
16:   end for
17: end for
```

6 Experiments

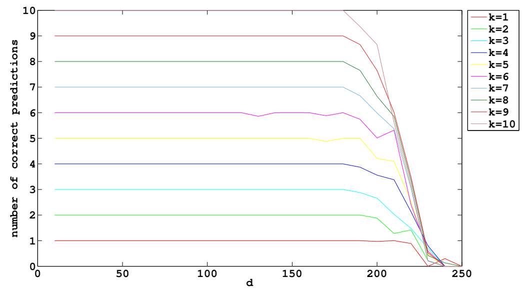
The above analysis ensures a high probability of successful predictions from our algorithm. To understand its effectiveness in practical settings, we performed experiments on two real-world collections of streaming data: query logs from a commercial search engine and hash-tags appearing in tweets.

A naïve approach based on selecting the most frequent elements could be considered as a baseline approach. However, such comparisons would be unfair to the baseline approaches because they would not be directly optimizing the criterion our problem setup mandates. More importantly, such comparisons would risk the clarity in the distinction of the proposed problem formulation and the traditional setup for statistical trend prediction. That said, the results for a naive approach were indeed empirically inferior to ours.

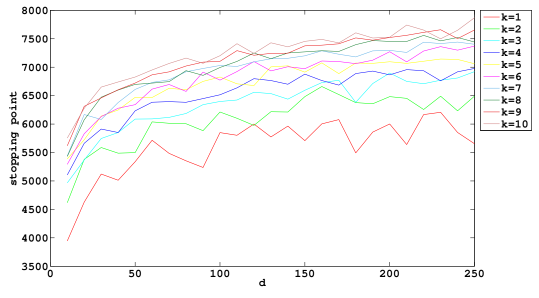
6.1 Search queries

We considered a data set comprising of one month of anonymized search logs that is made public under the Yahoo Webscope program[1]. We only consider the anonymized query identifiers in our experiments, ignoring other information – i.e., document identifiers, relevance judgments, etc. – present in this data set. We randomly sample 100K queries and use their respective timestamp to simulate the arrival of these queries as a stream. Each of these queries corresponds to a singleton, streaming set in our formulation. Figure 3 shows the frequency distribution of the resulting 603 unique queries in this data set. We select all of these unique queries to construct the set S_n (i.e.,

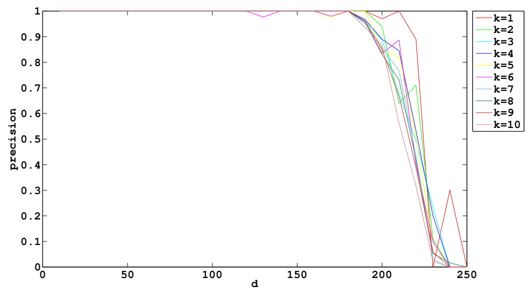
[1]The L18 data set available from http://webscope.sandbox.yahoo.com



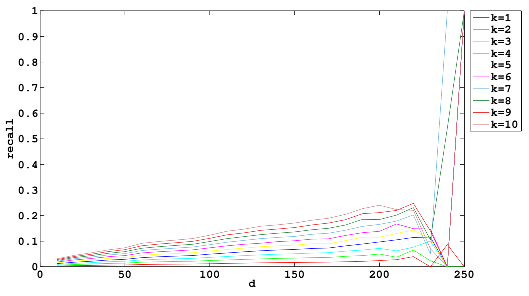
(a) Average Number of correct predictions



(b) Number of streaming subsets observed before making a prediction



(c) Precision



(d) Recall

Figure 2: *Results on search queries.* Each of the four plots show different evaluation metrics for different values of k and d and $D = 200$. These metrics are averaged over 200 iterations of our algorithm on random sub-streams of the original data stream. (Best seen in color).

$n = 603$). There are 140 queries that appear at least 200 times in the entire stream, therefore the assumption required for the constrained-min-d-occur problem clearly holds for $D = 200$ and $k \leq 10$. We ran our algorithm for different values of d and report our observations in Figure 2. Even though we do not have a valid lower-bound on the prediction accuracy for $d > D$, we examine if the algorithm is still able to make predict queries with larger number of future occurrences.

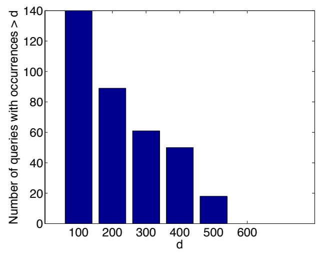


Figure 3: *Frequency distribution of search queries.*

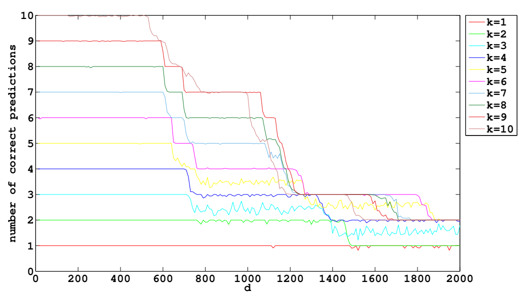
The precision and recall metrics shown in these plots are computed by comparing the set of predicted queries against the queries that actually occur d times in the stream remaining after the prediction is made. For $d = 150$, the predicted queries were 100% accurate for $k \leq 10$. On average, the algorithm was able to make predictions for $k = 10$ after observing about 7500 queries. In the remaining stream, about 60 unique queries appeared more that 150 times. Therefore, the recall of our algorithm is around 0.16.

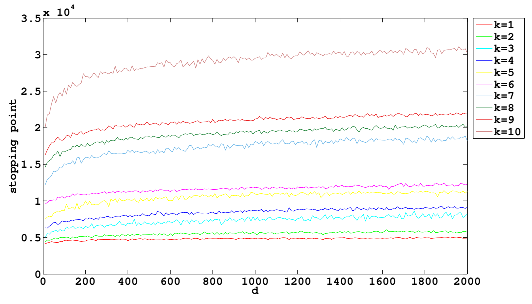
As we increase d, the probability of observing heads in a coin-toss decreases (see Section 4), making the algorithm take longer to make the predictions. Outside the provably correct range $d < \frac{D}{3 \log n}$, the precision is also observed to be adversely affected in a similar manner. For larger values of k, the algorithm takes more time to make predictions, and the performance degrades outside the provably correct range. In the provably correct range, the recall values for $k = 2$ is nearly twice the value for $k = 1$, because the number of predictions made are twice in the former case and all of the predictions are accurate; the minor deviation is attributed to the difference in the stopping point in the two experiments.

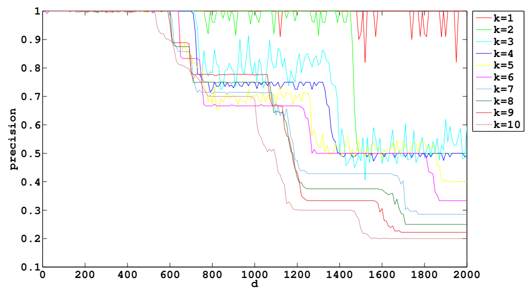
6.2 Twitter hash tags

We used the Twitter's public REST API[2] to obtain $150K$ tweets from the month of November 2013. We

[2]https://dev.twitter.com/docs/api/1.1


(a) Average Number of correct predictions


(b) Number of streaming subsets observed before making a prediction


(c) Precision

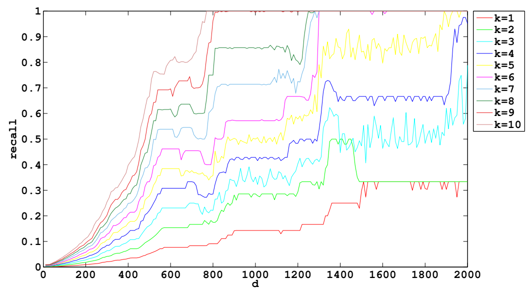
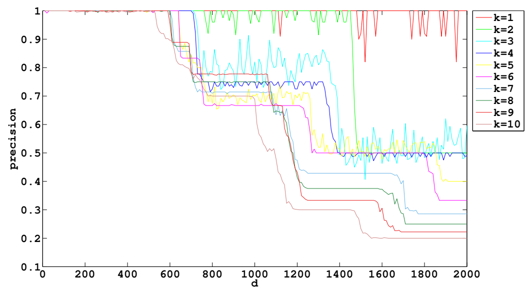

(d) Recall

Figure 5: *Results on Twitter hash-tags.* Each of the four plots show different evaluation metrics for different values of k and d and $D = 200$. These metrics are averaged over 200 iterations of our algorithm on random sub-streams of the original data stream. (Best seen in color).


(a) D=200

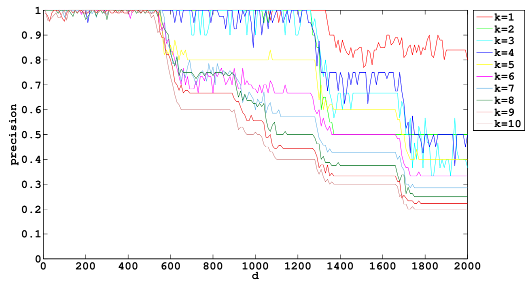
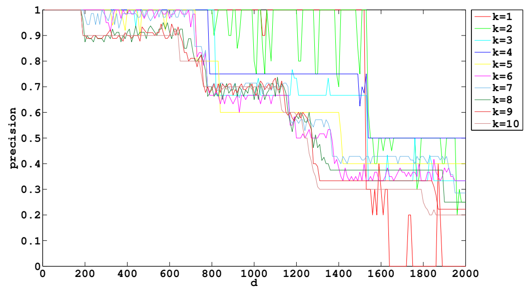

(b) D=400

Figure 6: *Comparison of precision curves for different choices of D on Twitter hash-tags.*

consider all of the hash-tags appearing in each of these tweets to compose the respective streaming set. We selected the $1K$ most frequent hash-tags in this collection to construct S_n. Figure 4 shows the frequency distribution of the unique hash-tags in this data set. Considering the skew in this distribution, we selected a larger value of $D = 200$ for this data set.

The overall observations for this data sets are similar to those for the previous data set. However, the drop in performance is gradual over a bigger range of values of d. For $k = 1$, the algorithm continued to make almost accurate predictions till $d = 10*D = 2000$. For $k = 10$, around 70% precision was observed while predicting 1000 future occurrences. The algorithm only observed $30K$ tweets to make these predictions. Assuming an estimated arrival rate of $150K$ tweets per minute, our algorithm can start predicting these ten tags with 70% accuracy in less than a minute.

The parameter D has a trade-off associated with it. On one hand, a larger value of D reduces the time taken to make a prediction and increases the provably useful range of values of d. On the other hand,


(a) D=100

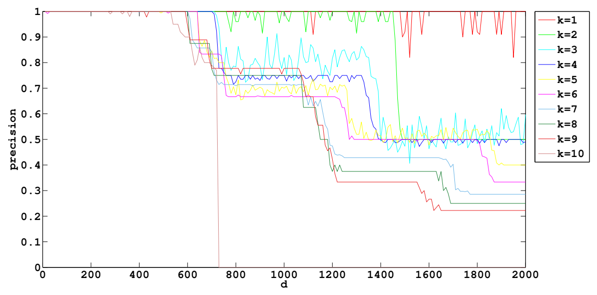

(b) D=200

Figure 7: *Results for the sliding-window approach on Twitter hash-tags.* The window size is set of $20K$. The average number of correct predictions(Accuracy) made for $D = 100$ and 200 for different values of k and d.

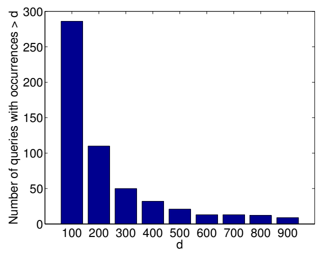


Figure 4: *Frequency distribution of unique hash-tags*

we may have to increase the size of the stream to ensure that the related assumption for constrained-min-d-occur holds. Figure 6 shows a comparison of the observed precision values for two different choices of $D = \{200, 400\}$. In both of these plots for most values of k, the precision is close to 1 for almost identical range of d, but the precision drops faster for smaller value of D. Also, the graphs for $D = 400$ are noisy, which suggests a large variance to be associated with their precision estimates. For this reason, we postulate that useful conclusions could not be derived for $D > 400$ in this data set.

6.3 Sliding window

Figure 7 shows the precision curves obtained using the sliding window adaptation of our algorithm (described in Section 5) with window size $W = 20K$. For $D = 200$, the graphs validate two hypothesis about the correctness of our window-based algorithm. First, for $k = 10$, the stopping point (see Figure 5) is greater than the window size. Therefore, the window-based algorithm should fail to make predictions for most values of d. Second, the stopping points for $k < 10$ are less than the window size for almost all of the values of d. Hence, the window-based algorithm should obtain similar performance as the original algorithm without the window. Similar observations were made when a window size $W = 10K$ was considered. Similar observations were made for $D = 100$. The relative difference between the performances for $D = 100$ and $D = 200$ can be explained similar to the discussion for Figure 6. These observations confirm that sliding window adaptation of the original algorithm well approximates the desired solution. This approximate algorithm is better suited for a practical setting where predictions need to be made at regular intervals of time.

7 Discussion

We presented a new problem formulation, *constrained min-d-occur*, for studying algorithms that guarantee a minimum number of future occurrences in streaming data. For this formulation, we presented a randomized algorithm and derive a lower-bound on the probability of successful predictions obtained from this algorithm. To our knowledge, any prior theoretical results for this problem does not exist. The theoretical result is further validated using experiments on two real-world data sets: search query logs and hash-tags in tweets. In both of these data sets, the proposed algorithm achieved high precision and recall values for predictions. We studied the performance of this algorithm for different choices of parameters. We also presented a sliding-window based adaptation of our algorithm to accommodate a practical setting where the predictions need to be updated only at regular intervals of time. Interestingly, these algorithms were found to be effective for larger values of d for which a useful lower-bound is not computed. Studying the tightness of our current lower-bound would be a useful extension of this work.

References

Harshavardhan Achrekar, Avinash Gandhe, Ross Lazarus, Ssu-Hsin Yu, and Benyuan Liu. Predicting flu trends using twitter data. 2011.

Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Joseph Seffi Naor. The online set cover problem. *SIAM J. Comput.*, 39(2):361–370, June 2009.

Loulwah AlSumait, Daniel Barbará, and Carlotta Domeniconi. On-line lda: Adaptive topic models for mining text streams with applications to topic detection and tracking. In *ICDM*, pages 3–12, 2008.

Sebastien Ardon, Amitabha Bagchi, and Anirban et al. Mahanti. Spatio-temporal and events based analysis of topic popularity in twitter. In *CIKM*, pages 219–228. ACM, 2013.

Baruch Awerbuch, Yossi Azar, Amos Fiat, and Tom Leighton. Making commitments in the face of uncertainty: how to pick a winner almost every time (extended abstract). In *STOC*, pages 519–530. ACM, 1996.

Hila Becker, Feiyang Chen, Dan Iter, Mor Naaman, and Luis Gravano. Automatic identification and presentation of twitter content for planned events. In *ICWSM*, 2011.

Mario Cataldi, Luigi Di Caro, and Claudio Schifanella. Emerging topic detection on twitter based on temporal and social terms evaluation. In *MDMKDD*, 2010.

Mayur Datar, Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Maintaining stream statistics over sliding windows. In *SODA*, pages 635–644, 2002.

Qiming Diao, Jing Jiang, Feida Zhu, and Ee-Peng Lim. Finding bursty topics from microblogs. In *ACL*, pages 536–544, 2012.

T. Edwards, D. S. W. Tansley, R. J. Frank, N. Davey, and Northern Telecom (nortel Limited. Traffic trends analysis using neural networks. In *Proceedings of the International Workshop on Applications of Neural Networks to Telecommuncations*, pages 157–164, 1997.

Nadav Golbandi, Liran Katzir, Yehuda Koren, and Ronny Lempel. Expediting search trend detection via prediction of query counts. In *WSDM*, pages 295–304. ACM, 2013.

Saurabh Goorha and Lyle Ungar. Discovery of significant emerging trends. In *KDD*, pages 57–64. ACM, 2010.

Vidit Jain and Esther Galbrun. Topical organization of user comments and application to content recommendation. In *WWW*, 2013.

Bernard J. Jansen and Tracy Mullen. Sponsored search: an overview of the concept, history, and technology. *IJEB*, 6(2):114–131, 2008.

Alex Lamb, Michael J. Paul, and Mark Dredze. Separating fact from fear: Tracking flu infections on twitter. In *HLT-NAACL*, pages 789–795, 2013.

Wenke Lee, Salvatore J. Stolfo, and Kui W. Mok. Adaptive intrusion detection: a data mining approach. *Artificial Intelligence Review*, 14:533–567, 2000.

Michael Mathioudakis and Nick Koudas. Twittermonitor: trend detection over the twitter stream. In *SIGMOD*, pages 1155–1158. ACM, 2010.

Venkata N. Padmanabhan and Jeffrey C. Mogul. Using predictive prefetching to improve world wide web latency. *Computer Communication Review*, 26:22–36, 1996.

Barna Saha and Lise Getoor. On maximum coverage in the streaming model & application to multi-topic blogwatch. In *SDM'09*, pages 697–708, 2009.

L. Weng, F. Menczer, and Y.-Y. Ahn. Virality prediction and community structure in social networks. *Sci. Rep.*, 3(2522), 2013.