Real-Time Top-R Topic Detection on Twitter with Topic Hijack Filtering

Kohei Hayashi[†‡] Takanori Maehara[§] Masashi Toyoda[*] Ken-ichi Kawarabayashi[†‡]
kohei-h@nii.ac.jp maehara.takanori@shizuoka.ac.jp mtoyoda@acm.org
k_keniti@nii.ac.jp

[†]National Institute of Informatics, Japan
[‡]JST, ERATO, Kawarabayashi Large Graph Project, Japan
[§]Shizuoka University, Japan
[*]The University of Tokyo, Japan

ABSTRACT

Twitter is a "what's-happening-right-now" tool that enables interested parties to follow thoughts and commentary of individual users in nearly real-time. While it is a valuable source of information for real-time topic detection and tracking, Twitter data are not clean because of noisy messages and users, which significantly diminish the reliability of obtained results.

In this paper, we integrate both the extraction of meaningful topics and the filtering of messages over the Twitter stream. We develop a streaming algorithm for a sequence of document-frequency tables; our algorithm enables real-time monitoring of the top-10 topics from approximately 25% of all Twitter messages, while automatically filtering noisy and meaningless topics. We apply our proposed streaming algorithm to the Japanese Twitter stream and successfully demonstrate that, compared with other online nonnegative matrix factorization methods, our framework both tracks real-world events with high accuracy in terms of the perplexity and simultaneously eliminates irrelevant topics.

Categories and Subject Descriptors

I.5.3 [**Clustering**]: Algorithms

Keywords

Twitter, topic detection, streaming algorithm, nonnegative matrix factorization, noise filtering

1. INTRODUCTION

Currently, hundreds of millions of users participate in online social networks and forums, subscribe to microblogging services, and maintain blogs. Twitter is currently the major microblogging service with over 41 million users and 1.47 billion social interactions. Twitter is a "what's-happening-right-now" tool that allows interested parties to follow individual user's thoughts and commentary on events in their lives in nearly real-time. Every *tweet* has an explicit timestamp that notes the exact time of its generation. Further, each user can express their thoughts in a maximum of 140 characters. Therefore, Twitter is a potentially valuable source for detecting a variety of trends. As an example, it is possible to predict the geographical movement of natural disasters such as earthquakes and typhoons [28].

A major research direction with Twitter is the dynamic extraction of multiple trending topics[1] such as emerging events and breaking news. Suppose there are underlying topics, as illustrated in Figure 1 (b), that are continuously time-evolving, and Twitter users post tweets influenced by the topics, which we observe as a text stream (Figure 1 (a)). A typical task is to recover the underlying topics from the noisy observation. For automatic extraction of such information, topic detection [5, 19, 27, 31, 36] is commonly used, which represents each topic as a distinct group of terms. On the basis of latent Dirichlet allocation (LDA) [5], one of the most popular probabilistic models for topic detection, several extended studies have addressed dynamic topic detection on Twitter [19, 36]. Nonnegative matrix factorization (NMF) is an alternative approach; in general, NMF is a simplified method of LDA [12], and is more computationally scalable than LDA. Saha and Sindhwani proposed a dynamic extension of NMF for Twitter [27].

A fundamental challenge here is to perform dynamic topic detection on Twitter given the following conditions: (i) detecting topics as quickly as possible while (ii) managing the topics as trending and meaningful for humans.

Real-time topic detection is an extreme case of condition (i), which allows us to be aware of the many events and news at the same time such events occur. This behavior is particularly desirable for Internet advertising [3]; for example, real-time topic detection enables the display of dynamically changing advertisements linking to currently trending topics. In addition, Twitter's instantaneousness and richness of content has a high affinity with the news media. More specifically, it potentially has promising applications such as automatic alert creation and content generation of news. However, the huge volumes of text data on Twitter prevents condition (i). Moreover, currently, the number of tweets has increased exponentially, reaching $277,000$ tweets per minute [13]. Unfortunately, none of the existing topic detection approaches can tolerate such scale. A random sampling approach (rather than using the entire stream) may help, but it significantly degrades the quality of extracted topics.

For condition (ii), we need dynamic adaptation to noise and outliers on the Twitter stream. In particular, anomaly users, advertisements, and automatic messages often create pseudo-topics (Figure 1 (c)) that "hijack" other meaningful topics we want to know

[1]In this paper, we used the term "trends" as highly-ranked topics in terms of the volume that may include periodical topics (e.g., morning greetings), in addition to emerging topics that are entirely new and distinguished from past topics.

KDD'15, August 10-13, 2015, Sydney, NSW, Australia.

ACM 978-1-4503-3664-2/15/08 ...$15.00.
DOI: http://dx.doi.org/10.1145/2783258.2783402.

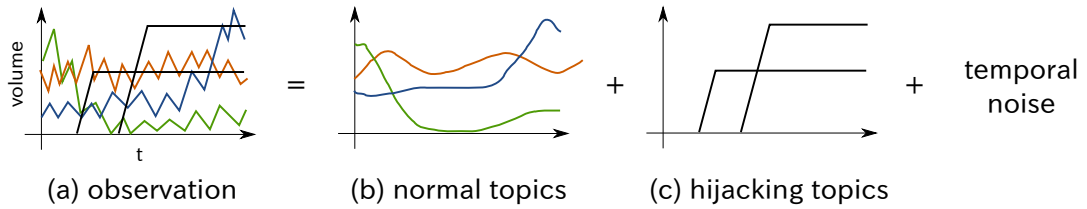

Figure 1: An illustration of Twitter topics. Each line indicates a topic and the y-axis indicates the volume of topics.

about. Our main observation here is that such *topic hijacking* can be triggered by two Twitter-specific phenomena described below.

- An *outsider* is a user who massively posts messages that only relate to a specific theme. Because of the exclusive content, *outsiders* compose a narrow but peaky topic. Twitter-bots and tool-assisted users [10] are typical examples of *outsiders*.
- A *machine-generated-phrase (MG-phrase)* is a combination of terms used simultaneously that is duplicated by many users. Twitter tools such as Foursquare[2] are typical sources, which semi-automatically post template messages and drastically increase the frequency of a specific combination of terms.

Because the number of manageable topics should be limited due to the corresponding computational cost, we would like to remove these hijacked topics to identify more meaningful topics. Unfortunately, detecting topic hijacking is not a trivial problem. Because new variants of *outsiders* and *MG-phrases* can emerge at any time, we need to filter them not just once but continuously in an unsupervised manner. Further, the detection of *MG-phrases* is a very difficult problem even for a static setting. Because an *MG-phrase* appears as a combination of terms, we need to check all combinations of unique terms that could be of arbitrary length, which is computationally infeasible.

Motivated by the abovementioned issues, we investigated an integrated framework for dynamic top-R topic detection, i.e., extracting the R most-trending topics in an online manner. For tracking evolving topics, we propose a simple streaming algorithm of NMF, which we refer to as *streaming NMF* (described in Section 2); the algorithm is derived as a one-step update equation for each data point, which efficiently tracks trending topics while mitigating overfitting. In addition, we have developed a dynamic filtering method of "hijacked" topics (Section 4). By exploiting a nature of NMF, we consider probabilistic models of normal and hijacked topics and discriminate them using a statistical hypothesis test, which enables the detection of emerging *outsiders* and *MG-phrases* in a consistent manner. The abovementioned algorithms are integrated into a single streaming algorithm. Using a Twitter stream, containing more than 10 million tweets, we demonstrate that our proposed algorithm successfully eliminates hijacked topics and extracts meaningful topics in the streaming setting, which outperforms other online NMF methods (Section 6).

Our main contributions are summarized as follows:

- A streaming NMF algorithm based on second-order stochastic gradient descent (SGD) that has theoretical guarantees for the convergence and an efficient computation trick for sparse data.
- A fully-automatic, robust, and fast detection method for topic hijacking.
- An integrated framework that enables real-time top-10 topic detection with more than 25% of the full Twitter stream, which is 10–250 times more scalable than the existing online NMF algorithms.

To the best of our knowledge, our proposed method is the first to consider a simultaneous system of topic detection and noise filtering for a streaming environment.

Our framework works via the following procedure. First, suppose we have a blacklist of users and term combinations that have triggered topic hijacking. Then, for each timestamp t, a bundle of tweets observed within a time window are converted to a *user* $\times$ *term* frequency matrix (with variable size). Within this time window, some tweets are removed if either a user who is posted the tweet or a term combination in the tweet is blacklisted. Next, using the abovementioned input, streaming NMF estimates the current top-R topics. Our framework then checks whether the obtained topics are hijacked or not. If the topics are hijacked, it updates the blacklist such that the same topic hijacking will never occur.

Notation In this paper, we denote the (i,j)-th element, the i-th row vector, and the j-th column vector of $\mathbf{A}$ as a_{ij} or A_{ij}, $\mathbf{a}_i$, and $\mathbf{a}_{\cdot j}$, respectively.

2. STREAMING NMF

NMF [20, 21] is an established method for relational data analysis including topic detection. Given a nonnegative matrix, NMF decomposes the matrix into row-specific and column-specific nonnegative features. Because of the nonnegative constraints, NMF attempts to find distinctive and non- overlapping features, which significantly improves the interpretability of results compared to other linear dimensional reduction methods such as PCA [20].

Here, given an observation $\mathbf{X} \in \mathbb{R}_+^{I \times J}$, we consider NMF that minimizes the following squared loss function with ℓ_2-norm regularization under nonnegative constraints of factor matrices $\mathbf{U} \in \mathbb{R}_+^{I \times R}$ and $\mathbf{V} \in \mathbb{R}_+^{J \times R}$:

$$f_\lambda(\mathbf{X}; \mathbf{U}, \mathbf{V}) = \frac{1}{2}\|\mathbf{X} - \mathbf{U}\mathbf{V}^\top\|_{\mathrm{F}}^2 + \frac{\lambda}{2}(\|\mathbf{U}\|_{\mathrm{F}}^2 + \|\mathbf{V}\|_{\mathrm{F}}^2) \quad (1)$$

where $\|\cdot\|_{\mathrm{F}}^2$ denotes the Frobenius norm and R is the rank, which corresponds to the number of topics.

The projected gradient method [22] finds the local minima of NMF, which consists of the following alternating update equations:

$$\mathbf{U}^{\mathrm{new}} = [\mathbf{U} - \eta \nabla_U f_\lambda(\mathbf{X}; \mathbf{U}, \mathbf{V})]_+, \quad (2a)$$
$$\mathbf{V}^{\mathrm{new}} = [\mathbf{V} - \eta \nabla_V f_\lambda(\mathbf{X}; \mathbf{U}^{\mathrm{new}}, \mathbf{V})]_+, \quad (2b)$$

where η denotes the learning rate, $([\mathbf{A}]_+)_{ij} = \max(A_{ij}, 0)$ denotes the projection operator to non-negative space, and

$$\nabla_U f_\lambda = \mathbf{U}(\mathbf{V}^\top\mathbf{V} + \lambda\mathbf{I}) - \mathbf{X}\mathbf{V} \quad \text{and} \quad (3a)$$
$$\nabla_V f_\lambda = \mathbf{V}(\mathbf{U}^\top\mathbf{U} + \lambda\mathbf{I}) - \mathbf{X}^\top\mathbf{U} \quad (3b)$$

denote the gradients of the loss function.

[2] http://foursquare.com

2.1 Problem Formulation

Before introducing our proposed algorithm, we define the streaming environment. For every time step $t = 1, \ldots, T$, instead of complete observation $\mathbf{X}$, we observe time-slice $\mathbf{X}^{(t)}$ having N_t nonzero elements such that $\mathbf{X} = \sum_{t=1}^{T} \mathbf{X}^{(t)}$. Here, we assume that we have $O(N_* + (I+J)R)$ space where N_* is an upperbound of N_t, i.e., $N_* > N_t$ for all t. In this setting, we can store $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{X}^{(t)}$ in main memory.

In this environment, the simplest solution is the decomposition of $\mathbf{X}^{(t)}$, i.e., to solve $f_\lambda(\mathbf{X}^{(t)}; \mathbf{U}, \mathbf{V})$. However, because $\mathbf{X}^{(t)}$ is very sparse, the extracted $\mathbf{U}$ and $\mathbf{V}$ will be degraded by noise. In addition, it does not take into account the temporal smoothness between $\mathbf{X}^{(t)}$ and $\mathbf{X}^{(t-1)}$. Another solution is the decomposition of the accumulated data $\sum_{s=1}^{t} \mathbf{X}^{(s)}$. Although this strategy mitigates the sparsity problem, maintaining $\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(t-1)}$ violates the memory constraint. Furthermore, both solutions require iterating the batch NMF updates (2) until convergence for every $t = 1, \ldots, T$, which is time consuming.

Instead, we minimize the following accumulated loss function:

$$\sum_{s=1}^{t} f_\lambda(\mathbf{X}^{(s)}; \mathbf{U}, \mathbf{V}). \tag{4}$$

We refer to this as *streaming NMF*. Surprisingly, the local minima of streaming NMF at $t = T$ are exactly the same as those of NMF (1) with some scaling.

THEOREM 1. *Suppose $\|\mathbf{X}^{(t)}\|_{\mathrm{F}}^2 < \infty$ for all t. Let $\mathbf{g}_\lambda(\cdot)$ be the derivative of $f_\lambda(\cdot)$ and $\hat{\mathbf{U}}, \hat{\mathbf{V}}$ be solutions of $f_\lambda(\cdot)$, i.e., $\mathbf{g}_\lambda(\mathbf{X}; \mathbf{U}, \mathbf{V}) = \mathbf{0}$. Then, the derivative of Eq.* (4) *is given by $\sum_{t=1}^{T} \mathbf{g}_\lambda(\mathbf{X}^{(t)}; \mathbf{U}, \mathbf{V})$ and is taken zero if and only if $\mathbf{U}\mathbf{V}^\top = \frac{1}{T}\hat{\mathbf{U}}\hat{\mathbf{V}}^\top$.*

Theorem 1 always holds because the difference between streaming NMF (4) and original NMF (1) is a fixed constant.

LEMMA 2. *Suppose $\|\mathbf{X}^{(t)}\|_{\mathrm{F}}^2 < \infty$ for all t. Let $\bar{\mathbf{X}} \equiv \frac{1}{T}\mathbf{X} = \frac{1}{T}\sum_t \mathbf{X}^{(t)}$ and $\bar{f}_\lambda(\mathbf{U}, \mathbf{V}) \equiv \frac{1}{T}\sum_t f_\lambda(\mathbf{X}^{(t)}; \mathbf{U}, \mathbf{V})$. Then, for any $\mathbf{U}$ and $\mathbf{V}$,*

$$\bar{f}_\lambda(\mathbf{U}, \mathbf{V}) - f_\lambda(\bar{\mathbf{X}}; \mathbf{U}, \mathbf{V}) = \frac{1}{T}\sum_t \|\mathbf{X}^{(t)} - \bar{\mathbf{X}}\|_{\mathrm{F}}^2 < \infty. \tag{5}$$

PROOF. Let $\mathbf{Y}_t \equiv \mathbf{X}^{(t)} - \mathbf{U}\mathbf{V}^\top$ and $\bar{\mathbf{Y}} \equiv \bar{\mathbf{X}} - \mathbf{U}\mathbf{V}^\top$. Then, $\bar{f}_\lambda(\mathbf{U}, \mathbf{V}) - f_\lambda(\bar{\mathbf{X}}; \mathbf{U}, \mathbf{V})$ is rewritten by $\frac{1}{T}\sum_t \|\mathbf{Y}_t\|_{\mathrm{F}}^2 - \|\bar{\mathbf{Y}}\|_{\mathrm{F}}^2 = \mathrm{tr}(\frac{1}{T}\sum_t \mathbf{Y}_t^\top \mathbf{Y}_t) - 2\mathrm{tr}(\frac{1}{T}\sum_t \mathbf{Y}_t^\top \bar{\mathbf{Y}}) + \mathrm{tr}(\bar{\mathbf{Y}}^\top \bar{\mathbf{Y}}) = \frac{1}{T}\sum_t \|\mathbf{Y}_t - \bar{\mathbf{Y}}\|_{\mathrm{F}}^2$. Since $\mathbf{Y}_t - \bar{\mathbf{Y}} = \mathbf{X}^{(t)} - \bar{\mathbf{X}}$, Eq. (5) holds. □

Lemma 2 is an analogous of the (sample) variance, which is given by the expectation of the square ($\bar{f}_\lambda(\cdot)$) minus the square of the expectation ($f_\lambda(\bar{\mathbf{X}}; \cdot)$).

Note that the scaling factor $1/T$ is necessary in Theorem 1 because $\mathbf{X}$ is roughly T times larger than $\mathbf{X}^{(t)}$. We emphasize that this is not an essential problem; in any time step t, we can recover the original NMF solution by multiplying $\sqrt{t}$ to the streaming NMF solution.

2.2 The Streaming Algorithm

We see that Eq. (4) is written by the sum of multiple functions, and now applying stochastic optimization makes sense. Specifically, we derive a one-step update algorithm using the SGD method:

$$\mathbf{U}^{(t)} = [\mathbf{U}^{(t-1)} - \eta_t \nabla_U f_\lambda(\mathbf{X}^{(t)}; \mathbf{U}, \mathbf{V}^{(t-1)})\mathbf{A}_{\mathbf{U}}^{(t)}]_+, \tag{6a}$$

$$\mathbf{V}^{(t)} = [\mathbf{V}^{(t-1)} - \eta_t \nabla_V f_\lambda(\mathbf{X}^{(t)}; \mathbf{U}^{(t)}, \mathbf{V})\mathbf{A}_{\mathbf{V}}^{(t)}]_+, \tag{6b}$$

where η_t is the time-dependent learning rate and $\mathbf{A}_{\mathbf{U}}^{(t)}$ and $\mathbf{A}_{\mathbf{V}}^{(t)}$ are positive-definite matrices. When $\mathbf{A}_{\mathbf{U}}^{(t)} = \mathbf{A}_{\mathbf{V}}^{(t)} = \mathbf{I}$, Eq. (6) becomes the first-order SGD. Although this is the simplest choice and the implementation is easy, the convergence speed is relatively slow. Alternatively, we use the second-order information of the loss function, which significantly improves the convergence speed. Since the inversion of the full Hessian matrix $\begin{pmatrix} \nabla_U \nabla_U f_\lambda & \nabla_U \nabla_V f_\lambda \\ \nabla_V \nabla_U f_\lambda & \nabla_V \nabla_V f_\lambda \end{pmatrix}$ is computationally expensive, we alternatively use its block-diagonal parts, which are given as follows:

$$\mathbf{A}_{\mathbf{U}}^{(t)} \equiv (\nabla_U \nabla_U f_\lambda)^{-1} = ((\mathbf{V}^{(t-1)})^\top \mathbf{V}^{(t-1)} + \lambda \mathbf{I})^{-1}, \tag{7a}$$

$$\mathbf{A}_{\mathbf{V}}^{(t)} \equiv (\nabla_V \nabla_V f_\lambda)^{-1} = ((\mathbf{U}^{(t)})^\top \mathbf{U}^{(t)} + \lambda \mathbf{I})^{-1}. \tag{7b}$$

Substituting Eq. (7) to (6) yields the (approximated) second-order SGD:

$$\mathbf{U}^{(t)} = [(1-\eta_t)\mathbf{U}^{(t-1)} + \eta_t \mathbf{X}^{(t)} \mathbf{V}^{(t-1)} \mathbf{A}_{\mathbf{U}}^{(t)}]_+, \tag{8a}$$

$$\mathbf{V}^{(t)} = [(1-\eta_t)\mathbf{V}^{(t-1)} + \eta_t (\mathbf{X}^{(t)})^\top \mathbf{U}^{(t)} \mathbf{A}_{\mathbf{V}}^{(t)}]_+. \tag{8b}$$

We remark that our update equations (8) depend on only the current time-slice $\mathbf{X}^{(t)}$, which satisfies the memory constraint of the streaming environment. Furthermore, in contrast to the batch NMF algorithm (2), we perform Eq. (8) only once for each t.

2.3 Convergence and the Learning Rate

Selecting η_t is also an important task because η_t strongly govern the solutions of our SGD algorithm. In Eq. (8a), the t-th solution is given by the η_t-weighted mean of the previous solution $\mathbf{U}^{(t-1)}$ and the current solution $\mathbf{X}^{(t)}\mathbf{V}^{(t-1)}\mathbf{A}_{\mathbf{U}}^{(t)} = \mathrm{argmin}_{\mathbf{U}} f_\lambda(\mathbf{X}^{(t)}; \mathbf{U}, \mathbf{V}^{(t-1)})$. This observation indicates that η_t controls the importance of current and past observations, i.e.,, larger η_t values give much weight to resent observations for learning $\mathbf{U}^{(t)}$ and $\mathbf{V}^{(t)}$.

If $\mathbf{X}^{(t)}$ is not dynamically evolving, a common choice of η_t is $O(1/t)$ [6], which weights all data equally and computes a simple average. In this setting, a standard analysis in SGD [6] yields the following convergence property.

PROPOSITION 3. *If data $\{\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(T)}\}$ are i.i.d, the $O(1/t)$ learning rate guarantees that, for $T \to \infty$, solutions* (6) *converge to critical points of streaming NMF* (4)*.*

Proposition 3 implies that, our SGD algorithm can yield the solutions of streaming NMF (and thus batch NMF) at sufficiently large T.

When $\mathbf{X}^{(t)}$ is dynamically changing, as is the case with Twitter, a constant learning rate $\eta_t = \eta$ ($0 < \eta < 1$) is an alternative choice. Similar to Proposition 3, the convergence property is given below. Note that we omit a proof because of page limitation but it follows the same idea as that of Proposition 2.2 in [25].

PROPOSITION 4. *If data $\{\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(T)}\}$ are i.i.d, constant learning rate $0 < \eta < 1$ guarantees that, for $T \to \infty$, solutions* (6) *are bounded by ϵ-balls of critical points of the loss function* (4) *with probability one, where $\epsilon = O(\eta)$.*

Proposition 4 indicates that the solution with a constant learning rate may slightly drift from the streaming NMF solution, which is less accurate than the case of the $(1/t)$ learning rate. However, because a constant learning rate quickly decays the weight of past observations by $(1-\eta)^t$, the current observation is reflected more strongly to $\mathbf{U}$ and $\mathbf{V}$. Thus, if a stream is rapidly changing, constant η is a reasonable choice to follow that trend. Note that we empirically found that, in general, a reasonably small η_t (such as $\eta_t = 0.1$) works well.

Table 1: Complexity of streaming NMF. R is the rank, I is the number of rows, J is the number of columns, T is the time length, and N_t ($< N_*$) is the number of non-zeros in $\mathbf{X}^{(t)}$.

	Time	Space
per Iteration	$O(N_t R^2 + R^3)$	$O(N_t + (I+J)R)$
Total	$O(\sum_t N_t R^2 + T R^3)$	$O(N_* + (I+J)R)$

We should note that the i.i.d. condition of Propositions 3 and 4 is sometimes too strong, especially for a Twitter stream. However, it is still helpful for understanding the behavior of the algorithm. In addition, for the constant learning rate, the i.i.d. condition is possibly weakened by following [1].

2.4 Further Reduction of Time Complexity

In the update of $\mathbf{U}$ (8a), the first term requires $O(IR)$ computation and the second term requires $O(N_t R^2 + R^3)$. By combining with the update of $\mathbf{V}$, the total computational cost is $O((I+J)R + N_t R^2 + R^3)$, which is more efficient than the batch update (2). However, it is still expensive because it depends on the matrix size (I and J); in real data, I and J are extremely large, and computation of $O(I)$ or $O(J)$ is crucial for real-time processing.

Lazy update [18] is a technique that reduces the computational cost of SGD by skipping unnecessary computations while maintaining the same solutions. Let us consider the update of $\mathbf{U}$. When all the elements of the i-th row of $\mathbf{X}^{(t)}$ are zero, those of the second term of Eq. (8a) are also zero, and the update only affects their scale, i.e., $\mathbf{u}_i^{(t)} = (1-\eta_t)\mathbf{u}_i^{(t-1)}$. This property allows us to skip the update of these elements if we maintain a cumulative scale. For example, suppose $t_i' < t$ is the last time the i-th row was non-zero and $\tilde{\mathbf{u}}_i$ is a parameter that has not been updated since time t_i'. Then, $\mathbf{u}_i^{(t)}$ can be obtained by rescaling $\tilde{\mathbf{u}}$ as follows:

$$\mathbf{u}_i^{(t)} = \Big(\prod_{s=t_i'}^{t}(1-\eta_s)\Big)\tilde{\mathbf{u}}_i = \frac{\prod_{s=1}^{t}(1-\eta_s)}{\prod_{r=1}^{t'}(1-\eta_r)}\tilde{\mathbf{u}}_i = \frac{c_t}{\alpha_{\mathbf{u}i}}\tilde{\mathbf{u}}_i. \quad (9)$$

This allows us to skip the scaling computation of $\mathbf{u}_s^{(t)}$ (the first term of Eq. (8a)) for $s \in \{i | \mathbf{x}_i^{(t)} = \mathbf{0}\}$. Because the numbers of non-zero rows is bounded above by N_t, the complexity of the scaling becomes $O(N_t R)$. The lazy update also helps the computation of $\mathbf{A}_{\mathbf{U}}^{(t)}$, which can only update the non-zero columns of $\mathbf{X}^{(t)}$. Here, let $\mathbf{G}_{\mathbf{U}}^{(0)} = (\mathbf{V}^{(0)})^\top \mathbf{V}^{(0)}$. Then, $\mathbf{A}_{\mathbf{U}}^{(t)}$ is given by $(\mathbf{G}_{\mathbf{U}}^{(t)} + \lambda \mathbf{I})^{-1}$, where

$$\mathbf{G}_{\mathbf{U}}^{(t+1)} = (1-\eta_t)^2 \sum_{j:\mathbf{x}_{\cdot j}^{(t)} \neq \mathbf{0}} \left(\frac{\mathbf{v}_j^{(t+1)}(\mathbf{v}_j^{(t+1)})^\top}{(1-\eta_t)^2} - \mathbf{v}_j^{(t)}(\mathbf{v}_j^{(t)})^\top \right) + (1-\eta_t)^2 \mathbf{G}_{\mathbf{U}}^{(t)}.$$

The above computation is bounded by $O(N_t R^2)$. Consequently, by applying the same reduction for $\mathbf{V}$, the computations depending on I and J are vanished, and the time complexity is now reduced to $O(N_t R^2 + R^3)$. Algorithm 1 describes the entire procedure of our proposed streaming NMF with the lazy update, and Table 1 summarizes its time and space complexity.

3. TOPIC DETECTION ON TWITTER

Thus far, we have addressed the streaming NMF algorithm for a general purpose. Here, by exploiting domain-specific knowledge, we apply our proposed algorithm to topic detection in a Twitter stream.

Algorithm 1 Streaming NMF

input $\{\mathbf{X}^{(t)}\}$, $\{\eta_t\}$, λ, R

1: initialize $\mathbf{U}, \mathbf{V}$ by $[0,1]$-uniform random variables
2: $\mathbf{G}_{\mathbf{U}} \leftarrow \mathbf{V}^\top\mathbf{V}$, $\mathbf{G}_{\mathbf{V}} \leftarrow \mathbf{U}^\top\mathbf{U}$, $\{c, \alpha_{\mathbf{u}i}, \alpha_{\mathbf{v}i}\} \leftarrow 1$
3: **for** $t = 1, \dots, T$ **do**
4: $\quad \mathbf{A}_{\mathbf{U}} \leftarrow (\mathbf{G}_{\mathbf{U}} + \lambda\mathbf{I})^{-1}$
5: $\quad \mathbf{G}_{\mathbf{V}} \leftarrow (1-\eta_t)^2 \mathbf{G}_{\mathbf{V}}$
6: $\quad$ **for** $i \in \{i \mid$ Non-zero rows of $\mathbf{X}^{(t)}\}$ **do**
7: $\quad\quad \mathbf{G}_{\mathbf{V}} \leftarrow \mathbf{G}_{\mathbf{V}} - (\frac{(1-\eta_t)c}{\alpha_{\mathbf{u}i}})^2 \tilde{\mathbf{u}}_i \tilde{\mathbf{u}}_i^\top$
8: $\quad\quad \tilde{\mathbf{u}}_i \leftarrow [\frac{(1-\eta_t)c}{\alpha_{\mathbf{u}i}} \tilde{\mathbf{u}}_i - \mathbf{A}_{\mathbf{U}} \sum_j \frac{\eta_t c}{\alpha_{\mathbf{v}j}} X_{ij}^{(t)} \tilde{\mathbf{v}}_j]_+$
9: $\quad\quad \mathbf{G}_{\mathbf{V}} \leftarrow \mathbf{G}_{\mathbf{V}} + \tilde{\mathbf{u}}_i \tilde{\mathbf{u}}_i^\top$
10: $\quad\quad \alpha_{\mathbf{u}i} \leftarrow c$
11: $\quad$ **end for**
12: $\quad$ do lines 4–11 with exchanging (rows, $\mathbf{U}$) for (columns, $\mathbf{V}$)
13: $\quad c \leftarrow (1-\eta_t)c$
14: **end for**

output $\mathbf{U} = \mathrm{diag}(\boldsymbol{\alpha}_{\mathbf{U}})\tilde{\mathbf{U}}$, $\mathbf{V} = \mathrm{diag}(\boldsymbol{\alpha}_{\mathbf{V}})\tilde{\mathbf{V}}$

3.1 User Aggregation and TF-IDF

The sparseness of word co-occurrences is a notorious problem for topic detection on a Twitter stream [37, 15, 23]. Because of the 140 characters constraint in Twitter, very few word interactions are observed in a single tweet, which significantly reduces the number of effective samples for topic estimation.

Therefore, we use an author-based aggregation strategy, inspired from a study by Mehrotra *et al.* [23]. We aggregate tweets by user to obtain a user-term matrix $\mathbf{X}$. This aggregation is expressed mathematically as follows. Suppose $\mathbf{m}^{(s)} \in \mathbb{N}^J$ is a bag-of-words representation of the s-th tweet with a single (hidden) topic (e.g., political news or current weather). Given a topic, assume that each $\mathbf{m}^{(s)}$ is an independent random variable that follows an identical distribution. For example, if the topic of the s-th and s'-th tweets is the same, then $\mathbf{m}^{(s)}$ and $\mathbf{m}^{(s')}$ follow the same distribution. By introducing an indicator vector $\mathbf{e}^{(s)} \in \{0,1\}^I$ that represents the user who tweeted $\mathbf{m}^{(s)}$, an $I \times J$ user-term frequency matrix, in which all past messages are accumulated until the current time, is expressed as $\mathbf{Y} = \sum_s \mathbf{e}^{(s)}(\mathbf{m}^{(s)})^\top$. This representation has the following advantages. First, a single tweet message is very short; however, there are a tremendously large number of tweets. Therefore, many users have many more terms than those found in a single tweet. Thus, such aggregation could extract meaningful topics [15, 23]. Second, in addition to the extraction of meaningful topics, a user's interest toward topics can simultaneously be extracted.

To avoid cases in which commonly used terms are extracted as a strong topic, we rescale $\mathbf{Y}$ by TF-IDF weighting, written as $X_{ij} = g(Y_{ij}) \cdot \mathrm{idf}_i$, where $g(\cdot)$ is either the raw-frequency ($g(Y) = Y$) or the log-frequency ($g(Y) = \max(0, \log(Y)+1)$) and $\mathrm{idf}_i = \log J / \#\mathrm{nonzero}(\mathbf{y}_{\cdot i})$ is the IDF of the i-th document.

3.2 Real-time Tracking with Streaming NMF

To extract topics in the streaming environment, we extend the concept of user aggregation in a streaming manner. With a timewindow Δ, consider a time-sliced user-term matrix $\mathbf{X}^{(t)}$, which has been observed from time Δt to $\Delta(t+1)$ and is defined as

$$X_{ij}^{(t)} = g(Y_{ij}^{(t)}) \cdot \mathrm{idf}_i^{(t)} \quad \text{where} \quad \mathbf{Y}^{(t)} = \sum_{s=\Delta t}^{\Delta(t+1)} \mathbf{e}^{(s)}(\mathbf{m}^{(s)})^\top$$

and $\mathrm{idf}_i^{(t)} = \log J^{(t)}/\#\mathrm{nonzero}(\sum_{s=1}^{t} \mathbf{y}_{\cdot i}^{(s)})$ is the current estimate of IDF ($J^{(t)}$ is the number of unique terms that appeared in $\mathbf{Y}^{(t)}$.) Clearly, $\sum_{t=1}^{T} \mathbf{Y}^{(t)} = \mathbf{Y}$ and $\mathrm{idf}_i^{(T)} = \mathrm{idf}_i$.

Given the sequence of $\mathbf{X}^{(t)}$, our streaming NMF sequentially provides up-to-date topics by $\mathbf{U}^{(t)}$ and $\mathbf{V}^{(t)}$ with moderate flexibility. First, the frequency of topic updates is easily controlled by Δ. For example, if we set Δ to 5 min, our streaming NMF updates topics every 5 min. The time complexity per iteration is $O((N_t + R)R)$ (Table 1); therefore, if we assume that $O(N_t R)$ computation can be performed in real-time, then streaming NMF can also be performed in real-time (unless the time resolution Δ is sufficiently large,) as long as computation $O(R^2)$ can be performed in unit time. Second, our streaming NMF allows changing the size of $\mathbf{X}^{(t)}$ over time, i.e., new users and new terms can be added at any time. For example, suppose the i'-th user is newly added at time t. Because $\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(t-1)}$ do not contain information about the i'-th user, i.e., $\mathbf{x}_{i'}^{(1)}, \ldots, \mathbf{x}_{i'}^{(t-1)}$ are all zero, they do not affect the results of previous $\mathbf{U}$s and $\mathbf{V}$s. Thus, we simply handle the new user by adding a new row to $\mathbf{U}$ and updating it with $\mathbf{x}_{i'}^{t}$.

3.3 A Probabilistic Interpretation of NMF

An advantageous property of NMF is that $\mathbf{U}$ and $\mathbf{V}$ estimated from $\mathbf{Y}$ can be considered to be (unnormalized) topic-wise conditional probability of terms and users, respectively. From the definition, $\mathbf{Y}$ represents a sample estimation of the joint probability of users and terms. If each tweet has a single topic, the joint probability forms a mixture of topic distributions, which can be written as follows (assuming $R < \infty$):

$$\mathbb{E}[Y_{ij}] = h \cdot p(\mathrm{user}_i, \mathrm{term}_j) = \sum_{r=1}^{R} h_r p(\mathrm{user}_i, \mathrm{term}_j \mid \mathrm{topic}_r),$$

where h is the entire volume of Y_{ij} and $h_r = h \cdot p(\mathrm{topic}_r)$ is the topic-wise volume.[3] Ideally, if the joint distribution is decomposable, i.e.,

$$p(\mathrm{user}_i, \mathrm{term}_j|\mathrm{topic}_r) = p(\mathrm{user}_i|\mathrm{topic}_r)p(\mathrm{term}_j|\mathrm{topic}_r),$$

then the rank-R NMF decomposition ($\mathbf{Y} = \mathbf{U}\mathbf{V}^\top$) perfectly recovers the topic-wise probabilities as follows:

$$\frac{u_{ir}}{\sum_i u_{ir}} = p(\mathrm{user}_i|\mathrm{topic}_r) \text{ and } \frac{v_{jr}}{\sum_j v_{jr}} = p(\mathrm{term}_j|\mathrm{topic}_r) \tag{10}$$

This interpretation plays an important role in our filtering method, which is discussed in the next section.

4. TOPIC HIJACKING FILTER

4.1 Probabilistic Models of Topics

A Twitter stream can be seen as a large corpus of the short texts. From this viewpoint, the entire term distribution is expected to follow a heavy-tailed curve, i.e., it follows Zipf's law. More specifically, if we have an infinite vocabulary, then the rank of the j-th term denoting $z_j \in \mathbb{N}$ follows the following discrete power-law distribution [11]:

$$p_{\mathrm{power}}(z_j \mid \alpha) = \frac{z_j^{-\alpha}}{\zeta(\alpha)}, \tag{11}$$

where $\alpha > 1$ denotes the scale parameter and $\zeta(x) = \sum_{n=1}^{\infty} n^{-x}$ is the Riemann zeta function.

[3]We write $p(\mathrm{user}_i, \mathrm{term}_j|\mathrm{topic}_r)$ as $\Pr(\mathrm{user} = i, \mathrm{term} = j \mid \mathrm{topic} = r)$ and so on for notational simplicity.

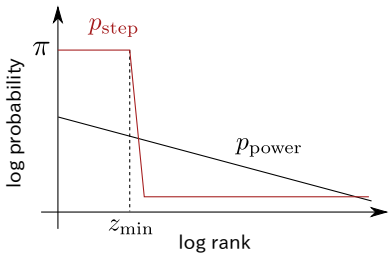

Figure 2: The power-law and the two-step distributions.

By following the above observation, we model each topic-sensitive term distribution $p(\mathrm{term}|\mathrm{topic}_r)$ by the power law (11).[4] This idea is reasonable because, in a normal topic, we expect many users involve and there exists the sufficient diversity of term usage ensuring Zipf's law. However, if a topic is dominated by an *MG-phrase*, there is no diversity and the power law is not appropriate. In this case, the probabilities of the *MG-phrase* terms would be high and roughly the same values and the rests would be very small values, which is represented as the following two-step distribution:

$$p_{\mathrm{step}}(z_j \mid \pi, z_{\min}) = \begin{cases} \pi & (z_j > z_{\min}) \\ \pi_0 \equiv \frac{1 - z_{\min}\pi}{J - z_{\min}} & (z_j \leq z_{\min}), \end{cases} \tag{12}$$

where $\mathcal{S}$ is a set of unique term indices of an *MG-phrase* and $z_{\min} = |\mathcal{S}|$ is the length of the MG-phrase. Figure 2 highlights the difference between these distributions.

We expand this concept to users' side, i.e., we assume that user distribution follows $p_{\mathrm{step}}(z_j \mid \pi, 1)$ if the topic is hijacked by an *outsider*; otherwise, it follows the power law. Based on these viewpoints, we define a normal topic and a hijacked topic below.

DEFINITION 5 (NORMAL TOPIC). *The r-th topic is* normal *if both $p(\mathrm{user}|\mathrm{topic}_r)$ and $p(\mathrm{term}|\mathrm{topic}_r)$ follow the power-law distribution* (11).[5]

DEFINITION 6 (HIJACKED TOPIC). *The r-th topic is* hijacked *by an* outsider *if $p(\mathrm{user}|\mathrm{topic}_r)$ follows the two-step distribution* (12) *with $z_{\min} = 1$. Similarly, the r-th topic is* hijacked *by a $z_{\min}$-long* MG-phrase *if $p(\mathrm{term}|\mathrm{topic}_r)$ follows the two-step distribution* (12).

4.2 Statistical Testing of Topic Hijacking

According to the abovementioned assumptions, we can distinguish hijacked topics and normal topics using the log-likelihood ratio test [32]. Although this subsection explains the case involving *MG-phrases*, the same approach can be used for *outsiders*.

The basic idea is that we check which distribution can be fit more strongly to an empirical term distribution. We evaluate this using

[4]When we employ the log-frequency as TF, the power law does not hold exactly, but it may hold approximately. Since we use the "online" version of TF, i.e., we decompose $\sum_t \log(\mathbf{Y}^{(t)})$ instead of $\log(\sum_t \mathbf{Y}^{(t)})$ as described in Section 3.2, the effect of TF is weakened and the resulting becomes closer to the raw-frequency matrix.

[5]Note that the the power-law assumption for normal users (Definition 5) is sometime too strong. Because of the Twitter's limitation, the number of words that a user can post per unit time is bounded by constant, which cutoffs the resulting distribution's values of extremely heavily-posting users. However, we empirically confirmed that such heavy users were very rare (< 100) and the rest of the distribution was well explained by the power law. Therefore, we abuse this assumption for the user distribution as an approximation.

the following log-likelihood ratio:

$$\mathcal{L}(z_{\min}) = \sum_j \log \frac{p_{\text{step}}(z_j \mid \hat{\pi}, z_{\min})}{p_{\text{power}}(z_j \mid \hat{\alpha})} \tag{13}$$

where $\hat{\pi}$ is the maximum likelihood estimator (MLE) of π in p_{step} given $z_{\min}$ and $\hat{\alpha}$ is the MLE of α in p_{power}. If the data are more fitted to the two-step distribution with $z_{\min}$, i.e., the terms are likely generated by the $z_{\min}$-long MG-phrase, then $\mathcal{L}(z_{\min})$ takes a positive value. Thus, we use the value of $\mathcal{L}(z_{\min})$ as the criterion to detect topic hijacking.

Prior to computing the log-likelihood ratio, we need to obtain MLEs, which require topic-wise samples of terms and users. By following interpretation (10), we use $\mathbf{U}$ and $\mathbf{V}$ as an empirical estimators of the term probabilities for each topic. Consequently, the log-likelihoods are approximated as follows:

$$\log p_{\text{power}}(\mathbf{z} \mid \hat{\alpha}) \approx -\sum_{j=1}^{J} v_{jr}(\hat{\alpha} \log z_j + \log \zeta(\hat{\alpha})), \tag{14a}$$

$$\log p_{\text{step}}(\mathbf{z} \mid \hat{\pi}, z_{\min}) \approx \sum_{j=1}^{J} v_{jr}(\mathbb{I}_{j \le z_{\min}} \log \hat{\pi} + \mathbb{I}_{j > z_{\min}} \log \hat{\pi}_0), \tag{14b}$$

where $\mathbb{I}_{(\cdot)}$ is the indicator function. Because the maximization of Eq. (14a) has no analytical solution [11], we use numerical maximization for $\hat{\alpha}$. In contrast, the maximum of Eq. (14b) is expressed as $\hat{\pi} = \sum_{j=1}^{z_{\min}} v_{jr} / z_{\min} \sum_{l=1}^{J} v_{lr}$.

We then use a hypothesis test that evaluates whether the power law or the two-step distribution explains the data more convincingly. Here, we consider a situation in which p_{step} and p_{power} equivalently explain the data as null hypothesis H_0, and p_{step} is more likely than p_{power} as the alternative hypothesis H_1, i.e.,

$$H_0 : \mathbb{E}_z[\mathcal{L}(z_{\min})] = 0 \quad \text{and} \quad H_1 : \mathbb{E}_z[\mathcal{L}(z_{\min})] > 0. \tag{15}$$

A key property of $\mathbb{E}_z[\mathcal{L}(z_{\min})]$ is that it can be approximated as the Gaussian distribution if we have a large number of samples.

THEOREM 7 (ASYMPTOTIC NORMALITY OF $\mathcal{L}(z_{\min})$ [32]). *Let z_j be the rank of the j-th term and suppose we observe N terms. Then, under H_0, $\mathcal{L}(z_{\min})/\sqrt{N}$ converges in distribution to a zero-mean Gaussian distribution $N(0, \sigma^2)$ where*

$$\sigma^2 = \frac{1}{N} \sum_{j=1}^{N} \left(\log \frac{p_{\text{step}}(z_j \mid \hat{\pi}, z_{\min})}{p_{\text{power}}(z_j \mid \hat{\alpha})} \right)^2 - \left(\frac{1}{N} \mathcal{L}(z_{\min}) \right)^2.$$

Thus, if the p-value of $\mathcal{L}(z_{\min})/\sqrt{h}$ is greater than 95%, we reject H_0, i.e., we determine that data $\{z_j\}$ are dominated by the $z_{\min}$-long *MG-phrase*. The length of an *MG-phrase* is bound by the 140-character limit; thus, we examine this test for $z_{\min} = 1, \ldots, 140$. The entire procedure is summarized below.

1. Sort $\mathbf{v}_r$ to obtain log-likelihoods (14).
2. Compute $\hat{\alpha}$ and set $j = 1$.
3. Compute $\hat{\pi}$, $\mathcal{L}(z_{\min})$, and σ^2 given $z_{\min} = j$.
4. If $\mathcal{L}(z_{\min})/\sqrt{h_r \sigma^2} > 1.645$[a], reject H_0 and detect the r-th topic as a hijacked topic.
5. Otherwise, $j \leftarrow j + 1$. Then, repeat from step 3 until $j < 140$.

[a] 1.645 is the 95% one-sided confidence interval of the standard Gaussian distribution.

This hijacked topic detection method has three advantages. First, it does not have free parameters and we are not necessary to tune them by using e.g. cross-validation. Second, because it is based on the statistical test, the result is easy to interpret and robustness is high. Third, the computational cost is moderately low. The dominant part of the computation is the look-up of the top-140 elements of $\mathbf{V}$, which requires $O(J)$ time.

4.3 Integration with Streaming NMF

Because new *outsiders* and *MG-phrases* can potentially emerge at each time step, we must continuously apply our detection method. In addition, to ignore the effect of hijacked topics, we must filter such topics from the input. Thus, we simultaneously use our detection method during execution of the streaming NMF algorithm. For *outsiders*, we perform our detection method at interval T_B. If detection occurs, the corresponding user is added to the blacklist $\mathcal{B}_{\mathrm{U}}$. Once blacklisted, the user's contribution is filtered from the input, i.e., the corresponding part of $\mathbf{X}^{(t)}$ is filled by 0, and their information is not used to learn $\mathbf{U}$ and $\mathbf{V}$. The similar procedure is performed for *MG-phrases*. For example, suppose we have two blacklisted *MG-phrases*: (ticket, sale) and (free, movie, preview). Then, we search tweet messages containing "ticket *and* sale" or "free *and* movie *and* preview". These tweets are discarded when we construct $\mathbf{X}^{(t)}$.

4.4 Tradeoff of Topic Hijack Filtering

Detecting topic hijacking by the statistical test is significantly reliable; if the number of samples is sufficient and the data follows the topic distribution models (11) and (12), classification error is perfectly controlled by the confidence interval. Nevertheless, because we cannot know the true models of the data, it possibly causes unexpected false detections of hijacked topics due to the model misspecification. But how do they affect to the extracted topics? Our observation is that the negative effect of our filtering is fairly limited.

We first consider the case of *outsiders*. Because there are tons of users in Twitter, the effect of excluding several users is almost ignorable unless they are not *outsiders*. Therefore, the false detections of *outsiders* are not a critical issue. In contrast, *MG-phrases* are more influential because they affect all users' tweets. However, from the definition of *MG-phrases*, their false detections are supposed to occur rarely. As discussed in Section 4.1, a normal topic is generated by various users, and if a phrase is really meaningful for general users, the phrase should cooccur with other related terms. For example, in January 1st, a Twitter stream filled by the "happy new year" tweets. However, the celebration messages should have a lot of variations, e.g., some users post the "happy new year" phrase with hashtags. Such fluctuation makes the term distribution much closer to the power law than the two step, which prevents from blacklisting "happy new year". Of course the statistical test may falsely detect a normal topic as a hijacked topic when the topic is in the borderline. If we need to decrease such false negative errors, however, we can by setting a higher confidence interval.

5. RELATED WORK

Twitter Topic Detection Many studies based on document-topic models over the Twitter stream have been reported. These studies have primarily focused on the stream's temporal nature by aggregating tweets by terms. Cataldi *et al.* [9] proposed a topic detection system for Twitter that simulates a propagation process of terms among follows/followers in a time-varying setting. TM-LDA [36] is an extension of LDA that handles temporally evolving text in social media. TM-LDA estimates a transition probability of topics, enabling the prediction of future topics from past observations. Lau

et al. [19] proposed an online LDA algorithm that detects emerging Twitter topics.

Several other studies have also employed user-wise aggregation of tweets; however, these studies have focused on a static setting rather than a streaming environment. Michelson and Macskassy [24] examined a method to find Twitter users' interests as topics. Their method considers a map between a term that has appeared in tweets and a category in Wikipedia, which indicates a topic. A user is assigned to a category that corresponds to the most frequently used term. Weng *et al.* [37] investigated user interests in tweets employing LDA and estimating latent topics for each user. Xu *et al.* [38] extended this concept and developed an unsupervised topic model, i.e., they assumed that "no topic" tweets exist and ignored such tweets from topic extraction.

Cleaning and Filtering on Twitter Compared to the considerable number of studies regarding topic extraction for Twitter, only a few studies have considered data prepossessing and cleaning. Chu *et al.* [10] investigated the behavior of Twitter-bots and Twitter-cyborgs. They proposed several methods to distinguish such entities from human users. Furthermore, other studies have attempted to detect spam and spammers on Twitter; however, their motivation was Internet security and the detection of malicious URLs [30, 2] and spam related to pharmaceuticals [29].

Several studies addressed the identification of fully or nearly duplicated tweets. Wang [33] used the Levenshtein edit distance to identify duplicate tweets. O'Connor *et al.* [26] measured Jaccard similarity for a set of trigrams in all pairs of tweets. Unfortunately, their computational cost is quadratic with respect to the number of tweets, which makes these techniques practically infeasible for the current Twitter stream.

Online NMF Algorithms Several studies have proposed online algorithms for NMF that consider the case wherein rows increase with each time step. Cao *et al.* [8] proposed an online algorithm (ONMF). Specifically, at the t-th iteration, suppose we observe row vector $\mathbf{x}^{(t)}$. Their algorithm decomposes the concatenation of $\mathbf{x}^{(t)}$ and $\mathbf{V}^{(t-1)}$ (the previous estimation of $\mathbf{V}$) to obtain the current estimations $\mathbf{U}^{(t)}$ and $\mathbf{V}^{(t)}$. Wang *et al.* [35] and Guan *et al.* [14] solved this problem using stochastic approximation and derived a similar algorithm. Bucak *et al.* [7] and Wang and Lu [34] investigated NMF algorithms for a similar setting.

Dynamic NMF (DNMF) [27] is the most relevant method to our study, which considers the case wherein a new matrix is observed at every time step. DNMF defines a new objective function consisting of the squared error and a temporal regularization, which smooths $\mathbf{V}$ for a previous temporal direction. The major difference is that, to capture emerging topics, DNMF constantly increases the number of topics (i.e., R) with time, which allows more flexible modeling of matrix streams; however, both $\mathbf{U}$ and $\mathbf{V}$ require $O((I+J)R)$ space, and such an approach could potentially exhaust available memory space in the streaming environment. Another difference is that DNMF solves the update with respect to $\mathbf{V}$ as a projection onto a simplex space (i.e., $\sum_j v_{jr} = 1$) with a box constraint. This requires $O(J)$ computation per iteration, which may be a bottleneck when applied to large-scale data.

It is worth noting the key differences between ONMF, DNMF, and our streaming NMF. First, the problem setting of ONMF is essentially different. In DNMF and our case, we observe an entirely new matrix in every time step (the *streaming matrix* environment) whereas ONMF assume new rows are added (the *incremental* environment). Practically, the ONMF can handle the streaming matrix environment by considering the new matrix as additional rows; however, this strategy corresponds to decompose an extremely sparse yet large matrix $[\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(T)}]^\top$, which significantly degrade the quality of its solution. Alternatively, we can employ a *tweet* $\times$ *term* matrix for the ONMF rather than the *user* $\times$ *term* matrix. In this approach, however, we no longer extract the user information, and we cannot detect *outsiders*. Second, our streaming NMF has theoretical guarantees that the solution converges to the original NMF in the streaming matrix environment (Theorem 1, Propositions 3 and 4). Guan *et al.* [14] proved the convergence of ONMF, but it is for the incremental environment. DNMF has a convergence guarantee to the solution of the temporal loss (e.g., $f_\lambda(\mathbf{X}^{(t)}; \mathbf{U}, \mathbf{V})$), but not to that of the collective loss ($f_\lambda(\mathbf{X}; \mathbf{U}, \mathbf{V})$). Third, for every time t, ONMF and DNMF require an inner loop to to update $\mathbf{U}$ and $\mathbf{V}$ until convergence. In contrast, our streaming NMF just requires to compute the one-step update equations. This efficiency provides a huge advantage for the real-time processing.

Table 2: Summary of the network data sets.

Data	I	J	T	#Non-zeros
enron	87K	87K	114	1.1M
citeulike	22K	153K	242	2.4M
dblp	1.2M	1.2M	1763	18M
lastfm	0.9K	174K	1916	19M

6. EXPERIMENTS

We evaluate our proposed algorithm using streaming data generated from real web services. All the experiments were conducted on an Intel Xeon 2.90 GHz CPU with 256 GB memory.

6.1 Comparative Methods and Settings

In our experiments, we compared the performance of our approach with the methods described below.

NMF We implemented a standard NMF with alternating update equations (2) that minimizes $f_\lambda(\mathbf{X}^{(t)}; \mathbf{U}, \mathbf{V})$ for $t = 1, \ldots, T$. We checked the convergence by the relative root-mean-square-error (RMSE) over non-zero elements Ω_t with tolerance $\epsilon = 0.001$, i.e., we iterated the alternating update while $|E_{t-1} - E_t|/E_t > \epsilon$, where $E_t^2 = \frac{1}{|\Omega_t|} \sum_{(i,j)\in\Omega_t} \|X_{ij}^{(t)} - (\mathbf{u}_i^{(t)})^\top \mathbf{v}_j^{(t)}\|_{\mathrm{F}}^2$. We set $\lambda = 0.001$.

ONMF We implemented ONMF [8] by following the explanation of Wang *et al.* [35]. We used $\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(100)}$ for parameter initialization. For handling a Twitter stream, we marginalized out user information, i.e., we used $\sum_i \mathbf{x}_i^{(t)}$ as a new row to be decomposed.

DNMF We implemented DNMF [27]; however, as we have discussed in Section 5, the computational complexity of DNMF is significantly higher than the others. Therefore, we made the following modifications to improve the scalability. First, we assumed that R is fixed over time, i.e., there are no emerging topics. Second, we updated $\mathbf{U}$ and $\mathbf{V}$ only once for each $\mathbf{X}^{(t)}$ instead of iterating the updates until convergence. For other parameters, we used the same settings of the original paper [27].[6]

SNMF Streaming NMF *without* filtering. We set $\lambda = 0.001$ and $\eta_t = 0.1$.

SNMFF Streaming NMF *with* filtering. We set $T_B = 30$, i.e., detect topic hijacking for every 30 timestamps. For other parameters, we used the same setting as SNMF.

[6]We solved the quadratic programming problem using CVXOPT (`http://cvxopt.org`), which was used for the update of $\mathbf{V}$.

Table 3: RMSEs on the network data sets at $h_{\text{noise}} = 0$.

Data	NMF	DNMF	SNMF	SNMFF
enron	1.24070	1.15336	1.09194	1.08773
citeulike	17.32235	3.71833	3.70089	3.70067
lastfm	16.53502	N/A	1.89107	1.89806
dblp	1.10294	N/A	1.00066	1.00090

Note that we refer to SNMF and SNMFF collectively as SNMFs. All the methods were implemented in Python. Throughout our experiments, we fixed the number of topics (i.e., R) at 10 for all the methods. Finally, we emphasize that the main focus of our paper is scalability. Therefore, we do not present results if the runtime of the method exceeded 24 h.

6.2 Temporal Network Data

In the first experiment, we examined streaming NMF in terms of data fitting for temporally-evolving adjacency matrices. Note that we excluded ONMF from this experiment because ONMF is not designed to handle multiple matrices.

Data Set We used the following four data streams: Enron email *sender* $\times$ *recipient* data; CiteULike *user* $\times$ *tag* data; DBLP *author* $\times$ *author* data; and Last.fm *user* $\times$ *band* data [17]. Note that these networks have edges with timestamps. Table 2 describes the details. We constructed $\mathbf{X}^{(1)}$ from the first 10,000 edges ordered by timestamp, and we constructed $\mathbf{X}^{(2)}$ from the next 10,000 edges, and so on. Therefore, $\mathbf{X}^{(t)}$ and $\mathbf{X}^{(t+1)}$ might share similar properties because some events may continue from time t to $t+1$ (or beyond).

Because these data sets are not generated from a Twitter stream, there are no *outsiders* or *MG-phrases*. To explain how *outsiders* and *MG-phrases* affect NMF solutions, we synthetically generated noise that mimicked *MG-phrases*. For this, we randomly picked 10 rows $\{i_1, \ldots, i_{10}\}$ and 10 columns $\{j_1, \ldots, j_{10}\}$ and added them by volume $h_{\text{noise}} \in \{0, 10, 100\}$ to $X_{ij}^{(t)}$ for $i = i_1, \ldots, i_{10}$ and $j = j_1, \ldots, j_{10}$. Note that $h_{\text{noise}} = 0$ is the noiseless case.

Evaluation We measured generalization error (i.e., testing error) through missing-values prediction. We dropped 10% of the elements of $\{\mathbf{X}^{(t)}\}$ and considered them as missing values. Note that these missing values were ignored during training (i.e., they did not contribute to the loss function (1).) After decomposing the matrices on the basis of the remaining 90% of the elements, we made predictions and measured the RMSE against the missing values. After computing the RMSE for each timestamp, we summarized them by taking the mean over timestamps for each data set.

Results We first focus on the noiseless scenario. Table 3 summarizes the RMSEs and Figure 4 summarizes the runtimes. As can be seen, SNMFs clearly outperformed NMF and DNMF in terms of runtime; more specifically, they were 6–9 times faster than NMF and approximately 150 times faster than DNMF. Note that DNMF could not handle `dblp` and `lastfm` in the 24 h limit. For the accuracy of missing-values prediction, SNMFs are slightly but consistently better than DNMF. Note that NMF was significantly worse than other methods. This might be because NMF perform each time-slice independently and it cannot exploit the long continuous information over the time series. Figure 3 shows the behavior of RMSEs when the synthetic noise was added. The RMSEs of NMF and SNMF increase with h_{noise}. In contrast, the RMSEs of SNMFF remain the same error for varying h_{noise}. These results indicate that SNMFF successfully eliminated the added noise, whereas NMF and SNMF did not.

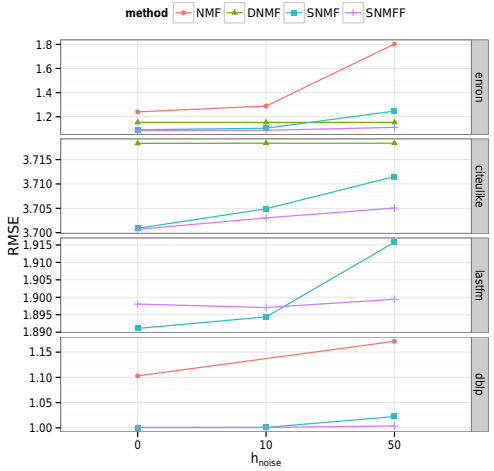

Figure 3: RMSEs on the network data sets varying the noise level h_{noise}. Note that, for clarity, we omit the results of NMF in `citeulike` and `lastfm`, which are significantly worse than the others.

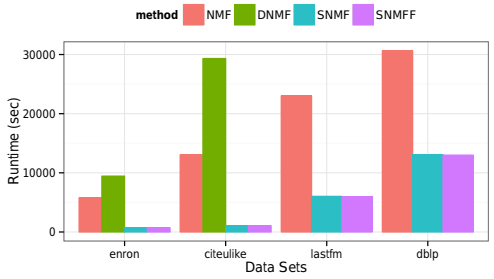

Figure 4: Runtime on the network data sets. Note that the results of DNMF at `dblp` and `lastfm` are not included because their runtime exceeded 24 h.

6.3 Quantitative Evaluation for a Twitter Stream

We examined the applicability of a real Twitter stream by evaluating the quality of extracted topics.

Data Set We prepared a collection of Japanese Twitter streams covering two days (i.e., April 15th (00:00) to 16th (23:59) JST, 2013). Further, we tokenized tweets using a Japanese morphological analyzer [16] and selected only nouns.[7] During preprocessing, we removed one-letter terms, which mostly came from misspecifications of the morphological analyzer. In addition, we removed approximately 50 stop words that were handpicked from the 100 most frequently-used terms from the data sets. Moreover, for fair comparison against the methods that have no filtering mechanism, we eliminated the 300 most frequently-tweeted users as a basic filtering of *outsiders*. We generated the log-frequency version of $\mathbf{X}^{(t)}$ for every $10,000$ non-zero elements. To evaluate scalability, we randomly sampled the tweets and generated additional data sets with 1% and 10% sampling ratios, in addition to the original data (100%). Table 4 shows detailed information about the data sets.

[7]The morphological analyzer handles 5,000 sentences per second, which is sufficient for real-time processing of tweets.

Table 4: Summary of the Twitter stream.

Sampling Ratio	#Tweets	#Non-zeros	#Users	#Terms
100%	15.3M	69.4M	417K	1.98M
10%	1.53M	6.94M	268K	0.45M
1%	0.15M	0.69M	87.1K	0.10M

Table 5: The median of the perplexity and the runtime at the Twitter stream.

	Perplexity			Runtime		
S. ratio	1%	10%	100%	1%	10%	100%
NMF	9.01E+09	6.32E+06	4.51E+04	27.0m	1.9h	16.4h
ONMF	1.06E+04	3.25E+04	1.83E+04	5.6h	9.2h	17.8h
DNMF	6.53E+04	N/A	N/A	16.7h	>24h	>24h
SNMF	5.65E+07	3.25E+04	8.71E+03	4.0m	21.7m	3.6h
SNMFF	5.25E+09	2.40E+04	7.90E+03	5.3m	24.1m	3.8h

Evaluation To evaluate the quality of extracted topics, we investigated how the topics were close to the actually-occurring events. As a source of those events, we collected Yahoo! news headlines[8] during the same periods of the Twitter stream. From this collection, we selected news categorized as "International" or "Domestic," which consisted of 127 headlines. After that, we applied the morphological analyzer to convert these into a set of noun terms $\mathcal{S}_k$. Finally, for the k-th news headline, we computed the perplexity $\exp(-\frac{1}{|\mathcal{S}_k|}\sum_{j\in\mathcal{S}_k}\log p(\mathrm{term}_j))$. In general, the perplexity measures the similarity between the term frequency of target document $\mathcal{S}_k$ and the estimated term distribution given by Eq. (10). A lower perplexity indicates a higher or better quality of the topic. We computed the perplexity with t within 1 h before and 1 h after the headline timestamp and selected the lowest one. Note that some terms of the headlines were not included in the vocabulary of the Twitter stream; in such case, the perplexity diverged to infinite. To avoid such issue, we computed the median over the headlines instead of the mean.

Results Table 5 shows the runtimes and perplexity corresponding to each method. SNMFs computed the 100% sampling data set in less than 4 h, i.e., they handled approximately $67K$ tweets per minute, which is sufficient for the real-time processing of a raw Japanese Twitter stream[9], the second most popular language on Twitter. In addition, the difference between the runtime of SNMF and that of SNMFF is less than 10 minutes, which shows that the detection and filtering of topic hijacking can be performed with nearly no expense. In contrast, other methods were 5–250 times slower than SNMFs. These large gap of runtime can be explained by the following two reasons. First, NMF and ONMF iterate parameter updates until convergence for every time step, where SNMFs update only once. In addition, ONMF and DNMF cannot exploit the sparsity of $\mathbf{X}^{(t)}$ due to algorithmic constraints, i.e., computational cost depends on I and/or J but not non-zeros of $\mathbf{X}^{(t)}$. Table 5 also demonstrates that, except for ONMF, increase of the number of samples significantly improves perplexity. Therefore, SNMFs have a clear advantage owing to their scalability. At larger sampling ratios (10% and 100%), the perplexity of SNMFs is consistently better than those of the others. In addition, SNMFF achieved the lowest perplexity. This result convincingly indicates that both the streaming NMF algorithm and the filtering method significantly improved the quality of extracted topics.

[8] `http://news.yahoo.co.jp/list`

[9] The fraction of Japanese tweets is 16% [4] and the number of Japanese tweets per minute is approximately 44.3K.

Table 6: The numbers of correctly detected events and ratios of hijacked topics.

	# Detected Events			Hijacking Rates		
S/N	0%	2%	5%	0%	2%	5%
NMF	2	3	3	0	0.154	0.304
ONMF	0	0	0	0	0.093	0.229
DNMF	0	0	2	0	0.362	0.390
SNMF	12	9	4	0	0.482	0.822
SNMFF	14	14	13	0	0.256	0.011

6.4 Evaluation of Topic Hijacking

To examine our filtering method, we added artificially-generated *MG-phrases* to the Twitter data used in the previous experiment; each *MG-phrase* consisted of 5 terms ("MGPXXa", ..., "MGPXXe") and we replaced 2% of terms in the data with 4 different *MG-phrases*, i.e., the data were contaminated by at least 4 unnecessary topics. We also conducted the same experiment with 5% S/N ratio and 10 different *MG-phrases*. From the 100% sample data[10], we evaluated how many extracted topics were relevant to the 127 news headlines. For this, we prepared the extracted topics at 1 h later of the time that each event happened, and checked whether the top 20 terms contained the terms used in the event's headline. Similarly, we checked how many topics were hijacked by the synthetic *MG-phrases*; we counted that a topic is hijacked if the top 20 terms contained at least one of the "MGPXXX" terms.

Results Table 6 shows the numbers of event-specific topics (larger is better; the maximum is 127) and the ratios of hijacked topics (lower is better; 1 means all the 127×10 topics were hijacked.) It clearly shows that SNMFF successfully detected event-specific topics, even in the noisy environment. In the noiseless case, SNMF also obtained the event-specific topics; however, SNMF failed to capture them if the noise ratio was high. This result implies that, in SNMFF, the proposed filtering method successfully eliminated irrelevant topics that hided some meaningful topics in SNMF. Note that, at the 2% S/N ratio, the filtering of SNMFF relatively failed compared with the case of the 5% S/N ratio. A plausible reason is that, because the noise level was low, the statistical test misjudged some weakly-hijacked topics as normal topics.

Finally, we directly analyzed the extracted topics of the following major incident.

- `bombing`: During the Boston Marathon (April 15, 2013), two pressure cooker bombs exploded at 2:49 pm EDT (18:49 UTC), killing three people and injuring an estimated 264 others.[11]

Table 7 shows the obtained topics. At first glance, we find that only SNMFF could capture event-specific topics, which are highlighted by underline. In contrast, SNMF suffered from the artificial *MG-phrases*, which are highlighted in italic.

7. SUMMARY

In this paper, we presented a streaming algorithm of NMF that can handle a Twitter stream. Our proposed algorithm can simultaneously perform real-time topic detection and filtering of *outsiders* and *MG-phrases*. Despite our naive implementation in Python, the proposed algorithm handled $70K$ tweets per minute, which amounts to 25% of all the tweets in a non-distributed non-parallelized setting. Our experimental results confirm that our proposed algorithm is superior in terms of scalability and that our filtering method works correctly.

[10] We used 1% sample data for DNMF because of its computational cost.

[11] `http://goo.gl/tGsGFB`

Table 7: The top-5 extracted topics for the `bombing` event. Each row shows a topic represented by top-5 relevant terms translated from Japanese to English using Google translation, ordered by the topic volumes. Each column shows the result with an S/N ratio of the artificial *MG-phrases* ("0%" means the noiseless result.) We highlight event-specific topics by underline that contain at least one keyword that appeared in the corresponding Yahoo! news headlines. We also highlight hijacked topics *in italic* that contain at least one *MG-phrase* term. Note that the Twitter stream has numerous (Japanese) Internet slangs, which often incurred mistranslation. Topics that the translation was completely failed are shown as "N/A" (most of them are emoticons.)

S/N	0%	2%	5%
NMF	Lucky fortune love my job Rein-gu plus energy suzume231 Nashiomone Ikue Plus energy energy House sash from ChokuAkira energy path Oregouna Irea Sewell Anne Marie Avu en Oki Cleanliness Domu Namaaji basis	Lucky fortune love work Oidon Oregouna Irea Sewell Anne Marie Avu en The Hia Ah Mme ee Ohyi tsu out to roux E ee Rein-gu suzume231 Nashiomone Ikue Tenpyo Oki basis Cleanliness and Domu Namaaji	Fortune lucky love work Oidon Anne Marie Sewell Irea Oregouna Loen Stein The Hia Ah De to roux E ee Affua Ohyi tsu Pawa To-kun steam pot rack Sash from the positive energy Poland wooden Iwaki
ONMF	Word Mio guy Good night 04-1 04-14 live today After regular day before Ho love place tricks et al 's Yoruho date next week era this month	Post 04-14 today live every time Routine post Ho 04-14 Times Regular night Japan work lucky items Good night Osaka post train participation N/A	Tweets file today live animal 04-14 Love after regular automatic previous day Birthday Arioka goodnight Japan videos Post 04-1 Na O o ... Broadcasting era this month next week program
DNMF	Good night _namonaki 117174 rusa .oO Good night) Bruno severe or Buza Dekin Lucky fortune love goodnight work Regular night) Bruno Yajima unrequited love Work love fortune lucky items digmeout	Lucky fortune work love 15 *The day before after 15 4:06 MGP03d* *The day before after 15 MGP03d MGP03e* Lucky fortune. Tsu ęØ- hero title Since the day before 4:06 buffaloes Contact Tendou	Work fortune love Lucky 15 Since the day before 15 4:06minoson *15 MGP09e MGP09a MGP09c MGP09d* Fortune lucky love my clients Free sama I Contact Tendou fortune
SNMF	Lucky fortune love my job N/A Periodic regulatory wishes descriptor block I lucky items fortune love work N/A	Lucky fortune love my job N/A Regular descriptor regulation wishes block *MGP06a MGP05e oriented Tuesday own* *MGP06d MGP06e press each Nfunfu*	Fortune lucky love my job *MGP18c MGP18a MGP18b MGP18d MGP18e* *MGP17c MGP17a MGP17b MGP17e MGP17d* *MGP16a MGP16b MGP16c MGP16d MGP16e* *Regular descriptor MGP11c MGP11e MGP11d*
SNMFF	Explosion Boston Marathon goal Boston Marathon Regular descriptor wish block Skype N/A Anime Lieutenant giant ranking points Good night ' ęØ birds ęØ) Bruno	Regular descriptor wish regulation block Japan Boston Marathon explosion goal News ... ... Good propensities Anime reverse dream human giant press Automatic regulation post 04-16 magica	Regular descriptor wish block Skype Explosion Boston Marathon goal Boston Boston Japan-oriented reverse dream Tuesday own Proclivity ..... press each ąë

Acknowledgment

We thank S. Xiang, H. Sawada, and anonymous reviewers for helpful comments. KH was supported by MEXT Kakenhi 25880028. MT was supported by MEXT Kakenhi 25280111.

8. REFERENCES

[1] A. Agarwal and J. C. Duchi. The generalization ability of online algorithms for dependent data. *IEEE Transactions on Information Theory*, 59:573–587, 2013.

[2] P. Anantharam, K. Thirunarayan, and A. Sheth. Topical anomaly detection from twitter stream. In *WebSci*, 2012.

[3] S. Asur and B. A. Huberman. Predicting the future with social media. In *WI-IAT*, 2010.

[4] W. Baumann. Mit technology review. `http://goo.gl/2KvLV4`, 2013.

[5] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *JMLR*, 3:993–1022, 2003.

[6] L. Bottou. Stochastic learning. In *Advanced Lectures on Machine Learning*, pages 146–168. 2004.

[7] S. S. Bucak, B. Gunsel, and O. Gursoy. Incremental non-negative matrix factorization for dynamic background modelling. In *PRIS*, 2007.

[8] B. Cao, D. Shen, J.-T. Sun, X. Wang, Q. Yang, and Z. Chen. Detect and track latent factors with online nonnegative matrix factorization. In *IJCAI*, 2007.

[9] M. Cataldi, L. Di Caro, and C. Schifanella. Emerging topic detection on twitter based on temporal and social terms evaluation. In *MDMKDD*, 2010.

[10] Z. Chu, S. Gianvecchio, H. Wang, and S. Jajodia. Who is tweeting on twitter: human, bot, or cyborg? In *ACSAC*, 2010.

[11] A. Clauset, C. R. Shalizi, and M. E. J. Newman. Power-law distributions in empirical data. *SIAM Review*, 51(4):661–703, 2009.

[12] C. Ding, T. Li, and W. Peng. On the equivalence between non-negative matrix factorization and probabilistic latent semantic indexing. *Comput. Stat. Data Anal.*, 52(8):3913–3927, 2008.

[13] DOMO, INC. Data never sleeps 2.0. `http://goo.gl/293Bnq`, 2014.

[14] N. Guan, D. Tao, Z. Luo, and B. Yuan. Online nonnegative matrix factorization with robust stochastic approximation. *IEEE Trans. Neural Netw. Learning Syst.*, 23(7):1087–1099, 2012.

[15] L. Hong and B. D. Davison. Empirical study of topic modeling in twitter. In *SOMA*, 2010.

[16] N. Kaji and M. Kitsuregawa. Efficient word lattice generation for joint word segmentation and pos tagging in japaneses. In *IJCNLP*, 2013.

[17] J. Kunegis. Konect: the koblenz network collection. In *WWW (Companion Volume)*, 2013.

[18] J. Langford, L. Li, and T. Zhang. Sparse online learning via truncated gradient. *JMLR*, 10:777–801, 2009.

[19] J. Lau, N. Collier, and T. Baldwin. On-line trend analysis with topic models: #twitter trends detection topic model. In *COLING*, 2012.

[20] D. D. Lee and H. S. Seung. Learning the parts of objects by nonnegative matrix factorization. *Nature*, 401:788–791, 1999.

[21] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *NIPS*, 2000.

[22] C.-J. Lin. Projected gradient methods for nonnegative matrix factorization. *Neural Computation*, 19(10):2756–2779, 2007.

[23] R. Mehrotra, S. Sanner, W. Buntine, and L. Xie. Improving lda topic models for microblogs via tweet pooling and automatic labeling. In *SIGIR*, 2013.

[24] M. Michelson and S. A. Macskassy. Discovering users' topics of interest on twitter: a first look. In *AND*, 2010.

[25] A. Nedic and D. Bertsekas. Convergence rate of incremental subgradient algorithms. In *Stochastic Optimization: Algorithms and Applications*, pages 263–304. Kluwer, 2000.

[26] B. O'Connor, M. Krieger, and D. Ahn. Tweetmotif: Exploratory search and topic summarization for twitter. In *ICWSM*, 2010.

[27] A. Saha and V. Sindhwani. Learning evolving and emerging topics in social media: a dynamic nmf approach with temporal regularization. In *WSDM*, 2012.

[28] T. Sakaki, M. Okazaki, and Y. Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In *WWW*, 2010.

[29] C. Shekar, S. Wakade, K. Liszka, and C.-C. Chan. Mining pharmaceutical spam from twitter. In *ISDA*, 2010.

[30] J. Song, S. Lee, and J. Kim. Spam filtering in twitter using sender-receiver relationship. In *RAID*, 2011.

[31] C. K. Vaca, A. Mantrach, A. Jaimes, and M. Saerens. A time-based collective factorization for topic discovery and monitoring in news. In *WWW*, 2014.

[32] Q. H. Vuong. Likelihood ratio tests for model selection and non-nested hypotheses. *Econometrica*, 57(2):307–333, 1989.

[33] A. H. Wang. Don't follow me - spam detection in twitter. In *SECRYPT*, 2010.

[34] D. Wang and H. Lu. On-line learning parts-based representation via incremental orthogonal projective non-negative matrix factorization. *Signal Processing*, 93(6):1608–1623, 2013.

[35] F. Wang, P. Li, and A. C. König. Efficient document clustering via online nonnegative matrix factorizations. In *WSDM*, 2011.

[36] Y. Wang, E. Agichtein, and M. Benzi. Tm-lda: efficient online modeling of latent topic transitions in social media. In *KDD*, 2012.

[37] J. Weng, E.-P. Lim, J. Jiang, and Q. He. Twitterrank: finding topic-sensitive influential twitterers. In *WSDM*, 2010.

[38] Z. Xu, L. Ru, L. Xiang, and Q. Yang. Discovering user interest on twitter with a modified author-topic model. In *WI-IAT*, 2011.