Joint Local and Global Sequence Modeling in Temporal Correlation Networks for Trending Topic Detection

Kai Shu
Arizona State University
kai.shu@asu.edu

Liangda Li
Yahoo Research
liangda@yahoo-inc.com

Suhang Wang
Penn State University
szw494@psu.edu

Yunhong Zhou
Yahoo Research
yunhongz@yahoo-inc.com

Huan Liu
Arizona State University
huan.liu@asu.edu

ABSTRACT

Trending topics represent the topics that are becoming increasingly popular and attract a sudden spike in human attention. Trending topics are critical and useful in modern search engines, which can not only enhance user engagements but also improve user search experiences. Large volumes of user *search queries* over time are indicative aggregated user interests and thus provide rich information for detecting trending topics. The topics derived from query logs can be naturally treated as a temporal correlation network, suggesting both *local* and *global* trending signals. The local signals represent the trending/non-trending information within each frequency sequence, and the global correlation signals denote the relationships across frequency sequences. We hypothesize that integrating local and global signals can benefit trending topic detection. In an attempt to jointly exploit the complementary information of local and global signals in temporal correlation networks, we propose a novel framework, *L*ocal-*G*lobal *Rank*ing (*LGRank*), to both capture local temporal sequence representation with adversarial learning and model global sequence correlations simultaneously for trending topic detection. The experimental results on real-world datasets from a commercial search engine demonstrate the effectiveness of LGRank on detecting trending topics.

ACM Reference Format:
Kai Shu, Liangda Li, Suhang Wang, Yunhong Zhou, and Huan Liu. 2020. Joint Local and Global Sequence Modeling in Temporal Correlation Networks for Trending Topic Detection. In *12th ACM Conference on Web Science (WebSci '20), July 6–10, 2020, Southampton, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3394231.3397924

1 INTRODUCTION

Nowadays, web search engines play a major role in people's information seeking and receiving. Due to the huge amounts of information curated by search engines, it becomes increasingly important to detect *trending topics*, which represent the topics that are becoming popular and attract a sudden spike in human attention. Figure 1 gives two examples of trending topic services by Yahoo![1] and Google[2]. Detecting trending topics in search engines are critical for benefiting both service providers and end users. First, trending topics can help improve user search experience. For example, when the world cup was held in Russia, people started searching for more details on news pieces such as what countries those teams are from, how the competition is scheduled, etc. If Yahoo can capture the early trending topics (e.g. "World Cup") and lead users to timely and relevant news contents, the users' experiences are significantly enhanced, which can, in turn, drive better monetization opportunities. Second, trending topics can also be leveraged by end users to gain insights in various domains such as predicting election results [27], stock markets [9], health conditions [23], etc. For example, a study reported that Google Trends data helped predict stock price moves for 30 stocks and increased the portfolio by 326%; while a constant buy-and-hold strategy only yielded just a 16% revenue return.

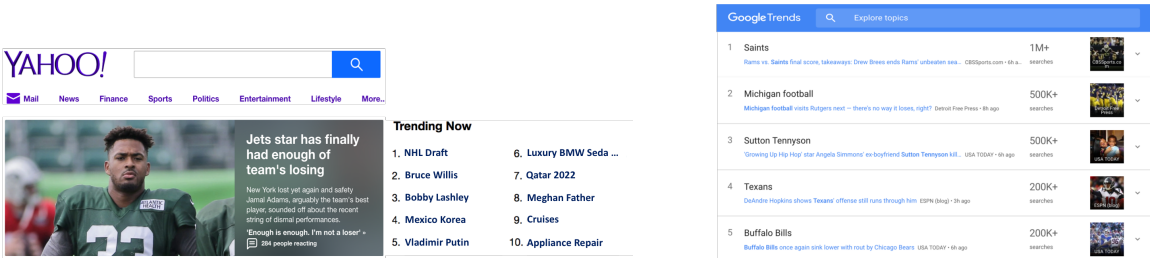
The search queries provide a valuable and rich source of information to discover trending signals [12, 15, 24]. For example, Dong *et al.* [11] built an n-gram language model of queries for the current time, and compare it to several reference language modes of past times (such as 1-day, 1-week ago), and the trendiness is measured by the difference between the current probability of an n-gram and its highest reference probability. Golbandi *et al.* [15] proposed to predict query counts, with an auto-regression model by explicitly assuming the query count sequence follows a normal distribution, to help predict trending topics. Despite the success of utilizing search query sequence for detecting trending topics, the majority of existing works model frequency sequences with explicit assumptions on the sequence generation process.

The topic ecosystem derived from query logs can be naturally treated as a temporal correlation network [20], and provides useful signals from both **local** and **global** perspectives. Figure 2(a) gives three topic examples of their normalized search frequencies over time from the search logs. In Figure 2(a), $\mathbf{y}^1$, $\mathbf{y}^2$, and $\mathbf{y}^3$ are representing the search frequency sequences for topics q_1 ("World Cup"), q_2 ("Curry MVP"), and q_3 ("Meghan Markle"). We can construct a temporal correlation network for these three topics as shown in Figure 2(b). In this correlation network, the **local** temporal signals represent the trending/non-trending information within each frequency sequence, such as $\mathbf{y}^1$ for trending topic q_1; and the **global** correlation signals denote the relationships across frequency sequences, such as $\mathbf{y}^1$ and $\mathbf{y}^2$ with strong positive correlations. As

WebSci '20, July 6–10, 2020, Southampton, United Kingdom

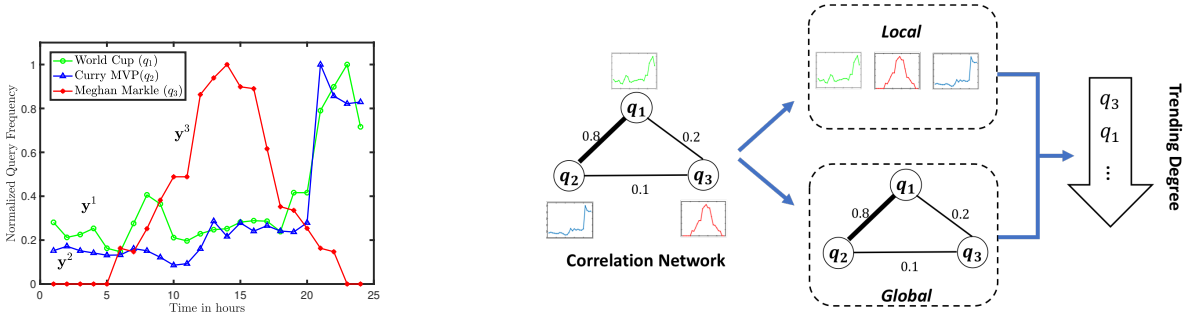
ACM ISBN 978-1-4503-7989-2/20/07.
https://doi.org/10.1145/3394231.3397924

[1] https://www.yahoo.com/
[2] https://trends.google.com/trends/



(a) Yahoo! Trending Now at Yahoo! homepage (b) Google Trends

Figure 1: The screenshots of the trending topic services in major commercial search engines.



(a) An example of topic frequency sequences (b) The flowchart of the proposed framework LGRank

Figure 2: An illustration of frequency sequences for different topics, and the basic idea of proposed framework integrating local and global sequence modeling for trending topic detection.

we will show in the following, the local and global signals contain complementary information to help detect trending topics.

On the one hand, the sequence generation process of search counts may not follow the same frequency distribution for different topics. From Figure 2(a), we can see that even though the sequence of query q_3 generally follows a normal distribution, the query q_1 and q_2 have a sudden increase of search counts around 22:00 PM, and their frequency sequences may not follow normal distributions. If the frequency distributions are accurately modeled, we could better capture the sudden spikes of frequency ratios over time to detect trending signals. For example, if we can predict the spike point at hour 21 for topic q_2, this is very likely an appropriate time slot to recommend this topic as a trending topic. Thus, exploiting the unique temporal signals within each frequency sequence is essential in capturing trending signals.

On the other hand, besides its self-distinguished content, the frequency distribution of a topic also depends on certain global factors, including the attribute of this topic, and scenario when it happens. (For instance, the category this topic belongs to, or users' activeness when it arises.) Such global factors could result in similar frequency distribution shared by different topics. By exploring the relationship between the topics with similar distribution, the learning of temporal representation of those topics can mutually benefit each other. Looking back at our previous example in Figure 2(a), topic q_1 and q_2 are strongly correlated in terms of their frequency sequences $\mathbf{y}^1$ and $\mathbf{y}^2$, and the correlation score between them is high. If we can capture the correlations among them, we may group trending topics of similar frequency distribution (such as q_1 and q_2) and pull these topics as high as possible, and similarly, group non-trending topics of similar frequency distribution and pull them as low as possible in the topic ranking list. However, the work on exploring the correlations among sequences for learning query temporal representations is rather limited.

Therefore, in this paper, we study the novel problem to capture both *local* and *global* signals of search frequency sequences and exploit their mutual benefits for trending topic detection (see Figure 2(b)). First, we propose a novel **local sequence adversarial modeling** framework to learn the temporal representations of sequences without explicit statistical model assumptions over sequence generation process. We employ a generative Recurrent Neural Network (RNN) to match the inputs from both the training frequency sequences and self-generated ones, and a discriminator to distinguish the output hidden states of them. Second, we utilize a **global sequence correlation embedding** framework to model the mutual correlations among topics. We assume those topics with similar frequency sequence vectors also should have close distances in the latent representation space. As an attempt to jointly exploit the complementary information of local and global signals in temporal correlation networks, we propose a novel framework called LGRank, which can model local sequence prediction and global sequence correlations simultaneously for trending topic detection. The main contributions of the paper are:

- We provide a principled way to study local and global information in temporal correlation networks;

- We propose a novel framework LGRank, which jointly performs local sequence adversarial learning and global sequence correlation embedding for search trending topic detection;
- We conduct extensive experiments on real-world datasets from a world-wide commercial search engine. The experiments demonstrate the effectiveness of the proposed framework for trending topic ranking.

2 PROBLEM STATEMENT

We aim to extract trending topics from query logs in a batch manner, since the trending service of search engine is required to show trending topics to users in such way (e.g. per hour). However, the queries collected are usually in large-scale, and contains duplicative instances. In order to obtain good topic candidates from query log, it is necessary to perform a **preprocessing** step which can: 1) remove spam queries that are likely to be pushed by non-human users, such as "12232 walmart"; and 2) reduce duplicate queries to ensure the topic diversity and improve the training efficiency of trending topic detection algorithms, such as queries "world cup" and "world cup russia". Therefore, we utilize the well-deployed real-time spam query detection system adapted from [6] to filter spam queries which are unlikely to be trending. In addition, we cluster all non-spam queries into groups so that queries in each group have similar semantics with agglomerative hierarchical clustering[3], which is widely used and essential for many downstream search engine services such as query recommendations [2]. Since we want to detect diverse trending topics, we select the representative queries with the highest search counts from each cluster as a candidate topic. Note that by selecting topics from query clusters, the size of candidate topics is significantly reduced, which can better serve the needs for batch-updating.

Let $\mathcal{Q} = \{q_1, ..., q_m\}$ denotes all the candidate topics in current batch. Each query q_i may be searched by users at different batches, and the search frequency of q_i can be denoted by $\mathbf{y}^i = (\mathbf{y}^i_1, \mathbf{y}^i_2, \cdots, \mathbf{y}^i_t)$, where $\mathbf{y}^i_t$ denotes the frequency of query q_i being searched at batch t. We denote $\mathbf{Y} = [\mathbf{y}^1; \mathbf{y}^2; \cdots; \mathbf{y}^m]$. We also obtain the historical user engagement records such as impressions (views) and clicks of those topics that have been shown to users from the search engine. We can compute the click-through rate (CTR) for each showed topic q_i as $\mathbf{r}_i = \frac{\#clicks}{\#impressions}$. For topics that have not been shown to users, we do not infer user preferences for them. Thus, we can obtain a partial CTR ranking vector at batch t as $\mathbf{r} \in \mathbb{R}^{m \times 1}$ of topics in $\mathcal{Q}$, where $\mathbf{r}_i$ represents the CTR of topic q_i at batch t. We adopt CTR scores as the measure of trendiness degree of topics. Then, in this paper, we study the following problem:

Given a set of candidate topics $\mathcal{Q}$ at batch t, topic search frequency vector $\mathbf{Y}$, and partial topic preference vector $\mathbf{r}$ up to current batch, we aim to rank the topics in $\mathcal{Q}$ as a list such that the higher ranked topic $q_i \in \mathcal{Q}$ is more likely to have a higher CTR score.

3 LOCAL-GLOBAL SEQUENCE MODELING FOR TRENDING TOPIC DETECTION

In this section, we give the details of the local-global ranking framework for detecting trending topics. It mainly consists of three components: i) a local sequence modeling with adversarial neural networks; ii) a global sequence correlation embedding; and iii) a topic preference ranking component.

To be specific, the local sequence adversarial modeling can capture trending signals from the temporal frequency counts within sequences; the global sequence correlation embedding models the correlations of frequency sequences among different topics, and the topic preference ranking component aims to preserve the user preferences among topics from user click information.

3.1 Local Sequence Adversarial Modeling

The search counts of topics are an important indicator of trending topic detection [8, 15]. Previous work on query temporal sequence modeling may specify additional assumptions on the sequence generation process such as Hawks process [25]. Recent advancements show that deep neural networks such as recurrent neural networks (RNNs) can be used to directly learn the temporal representations without introducing further assumptions [38]. Therefore, we utilize deep neural networks to model frequency sequences to learn the within-sequence representation for each topic.

Due to the batch updating manner of trending topics, we construct topic frequency sequences as the synchronized series with evenly spaced interval. For batch t, each topic q_i has the frequency sequence $\mathbf{y}^i_t$, where $\mathbf{y}^i_t$ represents the number of total user search for topic q_i at batch t. Let $\mathbf{z}_i$ denotes the text embedding vector representing the semantic meaning of topic q_i, obtained using Word2Vec [31]. Given $\mathbf{z}_i$, a good frequency sequence generator G should be able to reconstruct the original frequency sequence $\mathbf{y}^i$. However, simply train a generator G to reconstruct tokens in sequences may i) not be able to capture the long-term dependences in sequences; and ii) may be restricted by training sequences and have limited generalization abilities [22].

To this end, we employ an adversarial training scheme to utilize the generator G and a discriminator D (see Figure 3) simultaneously. Besides the reconstruction loss which drives the generator to produce realistic topic frequency sequences, the discriminator D can look at the statistics of the behaviors and not just at single step predictions in topic frequency sequences. The adversarial learning framework provides a better generalization over frequency sequences with flexible sequence length with the help of a *free running* generator component, details as follows.

Generator Learning We use LSTM [18] to model the temporal frequency process due to its capability for efficient long range dependency learning. Note that other RNN variants such as Gated Recurrent Units (GRU) [10] can also be an alternative choice. The LSTM takes $\mathbf{z}_i$ as the initial hidden state to start the generation process for topic frequency sequences. At batch t, the hidden state $\mathbf{h}^i_t$ is updated by,

$$(\mathbf{h}^i_t, \mathbf{c}^i_t) = f_G(\mathbf{h}^i_{t-1}, \mathbf{c}^i_{t-1}, \hat{\mathbf{y}}^i_t) \tag{1}$$

where $\mathbf{h}^i_0 = \mathbf{z}_i$ and $\mathbf{c}^i_t$ is the memory cell which records the history of inputs observed until up to t. The frequency at time step t is

predicted by a linear regression as follows,

$$\hat{\mathbf{y}}_t^i = \mathbf{W}\mathbf{h}_t^i + \mathbf{b} \tag{2}$$

where $\hat{\mathbf{y}}_t^i$ is predicted search frequency counts, and $\mathbf{W} \in \mathbb{R}^{1\times d}$ with d as the dimension of the hidden state in each layer. Thus, the overall probability of generating an output sequence $\hat{\mathbf{y}}^i$ given the input frequency sequence $\mathbf{y}^i$ is defined as follows,

$$p_G(\mathbf{y}^i|\mathbf{z}_i) = \prod_{\tau=1}^{t} p(\hat{\mathbf{y}}_\tau^i|\mathbf{y}_{\tau-1}^i, \mathbf{y}_{\tau-2}^i, \ldots, \mathbf{y}_1^i, \mathbf{z}_i, \boldsymbol{\theta}_G) \tag{3}$$

where t is the current batch, and $\boldsymbol{\theta}_G$ denotes the parameters of the generator. To further alleviate the shortcomings of vanishing gradients for long term sequence modeling and ensure a robust learning process, we also adopt the *free running* generator [22], which generates outputs without the guidance of input sequences $\mathbf{y}^i$. In this case, $\mathbf{o}_\tau^i$ is the self-generated search query counts at step $t\tau$ and will be used as the input for the next time step $\tau + 1$.

Thus, for each query $q_i \in \boldsymbol{Q}$, we have one sequence generated by $G(\mathbf{z}_i)$ teach-forced by the ground-truth sample $\mathbf{y}^i$, and the other free-running one generated by $G(\mathbf{z}_i)$ using the self-generated search counts from previous timestamps. The goal of utilizing two generation strategies are i) ensuring the generative LSTM to match the training search sequences; and ii) constraining the behavior of the generator G to be indistinguishable whether the network is trained with its inputs clamped to a training sequence or whether its inputs are self-generated. The objective function of the generator G is to optimize the reconstruction error of observed search frequency sequences, by minimizing the following negative log likelihood:

$$\mathcal{L}_G(\boldsymbol{\theta}_G) = \mathbb{E}_{q_i\in\boldsymbol{Q}}[-\log p_G(\mathbf{y}^i|\mathbf{z}_i))] \tag{4}$$

Discriminator Learning Now we introduce how to discriminate original data samples with generated data samples using adversarial learning. Following the setting of GANs [16], we add a discriminator D, and the generator parameters $\boldsymbol{\theta}_G$ are trained to fool the discriminator D, so that D can not easily differentiate the representation of sequences generated with the guidance of original sequences and that from the free-running LSTM. Thus, the objective of the discriminator D is to ensure the generated sequences are as realistic and close as possible to original sequences. We aim to minimize the following negative log likelihood,

$$\mathcal{L}_D(\boldsymbol{\theta}_D) = \mathbb{E}[-\log(1 - D(\tilde{\mathbf{h}}_t^i))] + \quad \mathbb{E}[-\log D(\mathbf{h}_t^i)] \tag{5}$$

where $\boldsymbol{\theta}_D$ is the parameters to be inferred for discriminator D. $\mathbf{h}_t^i$ and $\tilde{\mathbf{h}}_t^i$ are the learned representations of a real frequency sequence and a generated sequence for query q_i at batch t.

3.2 Global Sequence Correlation Embedding

In previous section, we have demonstrated how to capture frequency ratio over time within sequence for each topic. Now we aim to learn topic representations that ensure topics with similar temporal frequency patterns should have similar feature representations. Specifically, we want to group trending topics as close as possible in the feature space, so that those actual trending topics missed by local sequence modeling can be pulled up in the topic ranking list. Similarly, we try to group non-trending topics as close as possible in the feature space, such that the actual non-trending topics missed by local sequence modeling can be pulled down in the trending topic ranking list.

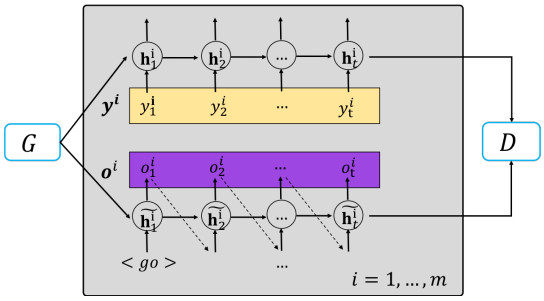


Figure 3: The adversarial learning framework for local prediction of topic frequency sequences. It is a minimax game between a generator G and a discriminator D.

For each pair of topics q_i and q_j at batch t, we have the topic frequency sequence $\mathbf{y}^i = (\mathbf{y}_1^i, \mathbf{y}_2^i, \cdots, \mathbf{y}_t^i)$ and $\mathbf{y}^j = (\mathbf{y}_1^j, \mathbf{y}_2^j \ldots, \mathbf{y}_t^j)$. Following manifold learning setting [4], we hope that the basis frequency vectors can respect the intrinsic Riemannian structure, i.e., if two topics have $\mathbf{y}^i$ and $\mathbf{y}^j$ are close in the intrinsic geometry of the data distribution, then $\mathbf{h}_t^i$ and $\mathbf{h}_t^j$ are also close to each other. This assumption is usually referred to as manifold assumption, which plays an important role in developing clustering algorithms [4].

Therefore, given the m topics with their frequency sequences $\{\mathbf{y}^i|i = 1, \cdots, m\}$, we can construct a weighted nearest neighbor graph $G = (V, E, w)$, where $V = \{q_1, \cdots, q_m\}$ contains the m nodes, where q_i corresponds to the topic with sequence $\mathbf{y}^i$. Let $\mathbf{T} \in \mathbb{R}^{m\times m}$ be the weight matrix of G. If q_i is among the K-nearest neighbors with q_j or q_j is among the K-nearest neighbors with q_i, then $\mathbf{T}_{ij} = \exp\frac{-\|\mathbf{y}^i - \mathbf{y}^j\|_2^2}{\phi}$, where ϕ is the scalar to control the bandwidth; otherwise, $\mathbf{T}_{ij} = 0$. Then the global sequence correlation embedding tries to minimize,

$$\mathcal{L}_C(\boldsymbol{\theta}_C) = \frac{1}{2}\sum_{i,j=1}^{m} \|\mathbf{h}_t^i - \mathbf{h}_t^j\|_2^2 \mathbf{T}_{ij} = \quad Tr(\mathbf{HLH}^T) \tag{6}$$

where $\mathbf{H} = [\mathbf{h}_t^1, \cdots, \mathbf{h}_t^m] \in \mathbb{R}^{d\times m}$, and $\mathbf{L} = \mathbf{S} - \mathbf{T}$ is the Laplacian matrix and $\mathbf{S}$ is a diagonal matrix with $\mathbf{S}_{ii} = \sum_{j=1}^{m} \mathbf{T}_{ij}$.

3.3 Topic Preference Ranking

User engagements such as viewing or clicking on the recommended trending topics provide a direct signal about whether the topics are really trending or not. We utilize the historical click through rates (CTRs) on those showed topics in previous batches to guide the topic representation learning that occurs in the current batch. Now that we have the feature representation for all topics $\{\mathbf{h}_t^i\}_{i=1}^m$, we demonstrate how to learn good topic representations that can also optimize the user engagements. For simplicity, we introduce the modeling process for one batch t. To preserve the CTR ranking of topics in $\mathbf{r} \in \mathbb{R}^{m\times 1}$, we model this probability as a maximum likelihood problem. Intuitively, if we can preserve the relative ranking of all pairs of nodes, we can maintain the whole ranking for all topics. However, it is computationally expensive to optimize all possible pairs as there are $m(m-1)/2$ pairs in total [29].

Algorithm 1 The Learning Process of LGRank model

Input: Q, θ_D, θ_G, $\mathbf{Y}$, $\mathbf{r}$, α and β.
Output: $\mathbf{H}$, $\mathbf{w}$.

1: Pre-compute the embedding vectors for all topics $\{\mathbf{z}_i\}_{i=1}^m$ in the training set Q
2: Initially train the generator G through Eqn. 4
3: **repeat**
4: Train the discriminator D by gradient descent using Eqn. 5
5: Train the overall objective by gradient descent through Eqn. 8 and update weights other than the discriminator
6: **until** Convergence
7: Calculate the ranking score vector $RS = \mathbf{wH}$
8: Top-k topics in highest scores in RS

Thus, we first transform the CTR scores into several levels by splitting the scores into N equal sizes. Now we have N CTR ranking levels, and the goal is to preserve the status of N CTR ranking levels. We generate a set of lists $\mathcal{T} = \{l_b\}_{b=1}^L$, and each list $l_b = \{q_{[1]}, ..., q_{[N]}\}$ consists of N topics, one randomly sampled from each level and the subscript of $q_{[i]}$ represents the level and thus maintains the CTR ranking information. Then we can construct a pair-wise ranking set $\mathcal{S} = \{(q_i, q_j) | q_i, q_j \in l_b \wedge \mathbf{r}_i > \mathbf{r}_j, \forall l_b \in \mathcal{T}\}$, which means we construct CTR pair-wise ranking pairs from all lists in $\mathcal{T}$, and in each pair the first topic has a higher CTR value than the second topic. To compute the predicted preference for the topic , we introduce a projection vector $\mathbf{w} \in \mathbb{R}^{1\times d}$ to map the topic latent representation $\mathbf{h}_t^i \in \mathbb{R}^{d\times 1}$ to the CTR score as $\hat{\mathbf{r}}_i = \mathbf{w}\mathbf{h}_t^i$. The preference difference of topic q_i and q_j can be computed as $\mathbf{r}_i - \mathbf{r}_j$. Then we can denote the probability of topic q_i ranking higher than q_j as $\sigma(\mathbf{r}_i - \mathbf{r}_j)$. Thus, the objective function is to minimize the negative log-likelihood as follows,

$$\mathcal{L}_T(\theta_T) = \frac{1}{|\mathcal{S}|} \sum_{(q_i, q_j)\in\mathcal{S}} -\log \sigma(\mathbf{w}(\mathbf{h}_t^i - \mathbf{h}_t^j)) + \lambda\|\mathbf{w}\|_2^2 \quad (7)$$

where $\lambda\|\mathbf{w}\|_2^2$ is to avoid over-fitting.

3.4 The Proposed Framework - LGRank

In this section, we combine the aforementioned three components together and present the framework LGRank. The proposed framework aims to solve the following objective function,

$$\min_{\theta_G, \theta_T, \theta_C} \max_{\theta_D} \mathcal{L}_T + \alpha(\mathcal{L}_G - \mathcal{L}_D) + \beta\mathcal{L}_C \quad (8)$$

where the parameters of all components of this objective function are learned jointly in an end-to-end fashion. α and β are positive hyper-parameters balancing the importance among different losses.

3.5 Optimization

The optimizing process is illustrated in Algorithm 1. First, we obtain the pre-trained text embedding representations of all topics $\{\mathbf{z}\}_{i=1}^m$ in Line 1. Next, we train the local sequence generator network in Line 2, and obtain the hidden states vectors $\mathbf{h}_t^i$ and $\tilde{\mathbf{h}}_t^i$. Then we train the discriminator D and update parameters θ_D in Line 5. At last, we train the overall objective function as in Eqn. 8 in Line 5.

The generative G is set as a single-layer LSTM with the dimension of hidden states as 20. The search counts are provided as inputs in each timestamp through an embedding layer of size 20. The the generator is initialized by concatenating the query embedding vector and a vector of zeros representing the initial search frequency. The discriminator D is a feed-forward neural network with a single hidden layer and a sigmoid output layer. Specifically, we adopt mini-batch gradient descent with Adam [21] optimizer to learn the parameters. Adam is an adaptive learning rate method which divides the learning rate by an exponentially decaying average of squared gradients. We choose Adam as the optimizer because it is a popular and effective method for determining the learning rate adaptively, which is widely used for training neural networks. In order to capture the global correlation embeddings, we empirically choose neighborhood K = 50. The model shows early convergence in mostly 40 - 50 overall epochs and on average it takes 15 minutes on Linux machine using CPU with 8 cores and 16 GB RAM. This indicate the model can be easily trained in a online setting and can be deployed in real-time setting for predicting trending topics.

4 EXPERIMENTS

In this section, we conduct experiments to validate the effectiveness of the proposed method for trending topic detection, and the factors that could affect the performance of LGRank. Specifically, we aim to answer the following evaluation questions:

- **EQ1** Is LGRank able to improve the trending topic detection performance by jointly modeling local and global ranking signals simultaneously?
- **EQ2** How effective are local frequency modeling and global correlations embedding, respectively, in improving the detecting performance of LGRank?
- **EQ3** Can the adversarial training help the local sequence modeling and improve the performance of trending topic detection?

To answer **EQ1**, we compare the performance of trending topic detection of LGRank with the state-of-the-art algorithms. We investigate the effects of local frequency modeling and global correlation embedding by performing an ablation study to answer **EQ2**. We further explore **EQ3** by removing the discriminator component and compare the performance.

4.1 Datasets

We use the query logs of a commercial search engine to obtain search count sequences for trending topic detection. Every search query made by a user is logged with the query string and respective timestamps. We also perform the following pre-processing steps: i) every query is normalized by lower casing, trimming, special character removal, etc; ii) the queries are aggregated in batches of short, regular time intervals of length. Due to the frequency of the trending topics service is approximately one hour, we construct the topic frequency time series with interval as one hour.

We collect the user engagements such as impressions and clicks on user behavior logs from the search engine. Each log record indicates the timestamps of when trending topics have been shown in Trending Now module, and the number of page views (impressions) and clicks on specific topics. We can compute the click-through rate (CTR) of these topics in previous batches to construct the pair-wise CTR pairs to train the topic preference ranking. The displayed

Table 1: The statistics of the dataset

# Queries	3,807,238
# Batches	168
# Impressions	344,749,188
# Clicks	454,463
Time Range	July 16th to July 23rd

trending topics are composed of organic trending terms and commercial terms for monetization. To maintain a fair comparison of algorithms, we only consider organic trending topics for our trending topic ranking task. In total, we collect query logs for one week, which covers the time span from July 16th to July 23rd, 2018. The statistics of the dataset is detailed in Table 1.

It is worth mentioning that there is no existing public benchmark dataset that contains both the search count sequences of topics and user engagements (e.g., clicks) for trending topic detection. In particular, representative datasets mainly include the semantic and frequency of posts, as well as auxiliary information of user profiles and social networks, within a time range from social media sites such as Twitter, Weibo [7, 39], however, user engagements are not the focus and not provided. Instead, our goal is to increase user engagements such as clicks, which may be hardly achieved by only considering the post frequencies because users do not necessarily engage in topics with high frequencies. Therefore, it is necessary and important to include user engagement information in our dataset. While we could possibly generate synthetic data of search count sequences of topics, the inherent properties and distributions of user engagements are not fully understood and may not be generated easily, and we would leave it for future work.

4.2 Experimental Settings

In this section, we first discuss how we obtain the ground truth of trending topics, then introduce the evaluations metrics, and summarize the baseline methods for comparison.

4.2.1 Ground Truth. Previous work on trending topic detection either obtains the ground through editorial annotation [1, 11, 24], or through online evaluation with click through rate (CTR) [1]. However, manually annotating the trending queries is usually labor- and time-consuming, which may not be appropriate since trending queries are dynamically changing over time and the editors' domain knowledge may be locally restricted and may not reflect the trending globally. Instead, the online evaluation strategy utilizes the user engagements (such as viewing and clicking) from the homepage of the search engine to obtain the ground truth GT^t of trending queries at batch t. Online evaluation has several advantages: i) *accurate*, meaning a higher CTR of the topic is more likely to indicate it is trending [1]; ii) *efficient*, indicating that we can obtain the ground truth efficiently for each batch by CTR scores. To this end, we empirically specify two parameters η and ϕ, where η is threshold of number of impressions (views) of queries, and ϕ is the threshold of CTR scores to determine trending and non-trending queries. We perform an empirical study to determine the values of $\eta = 10000$ and $\phi = 0.005$ that can maximize the overlap ratio between the resultant trending queries with those provided by editorial evaluations. Thus, we select queries with $\#views > 10000$, and treat queries that satisfy $CTR \geq 0.005$ as trending and those with $CTR < 0.005$ as non-trending. Note that for fair comparison, we only compare those k topics that have been shown to users (provided by the deployed method in the search engine) at batch t rather than all candidate topics.

4.2.2 Evaluation Metrics. To evaluate the effectiveness of different trending topic detection algorithms, we need to compare the ranking list (A^t) provided by a topic trending detection algorithm A with the ground truth (GT^t) of trending queries at batch t. Following previous work on trending topic detection [15, 24], we utilize top-k ranking metrics to evaluate the performance. We measure the performance with precision@k, recall@k, F1@k, defined as follows: $Precision@k = \frac{|A^t \cap GT^t|}{k}$, $Recall@k = \frac{|A^t \cap GT^t|}{|GT^t|}$, $F1@k = 2 \cdot \frac{Precision@k \times Recall@k}{Precision@k + Recall@k}$. The aforementioned metrics all depend on k. To evaluate the overall performances for ranking methods, for each of the three curves we can calculate the Area Under Curve (AUC) metric. These three values are denoted as AUC_P, AUC_R, AUC_F1 for the curves of precision, recall and F1 respectively. Note that it is possible that AUC_F1 is less that AUC_P or AUC_R, and the higher values indicate better performances of a ranking method.

4.2.3 Baseline methods. We compare with the state-of-the-art trending topic detection algorithms on search engines:

- **Random**: Random method ranks all the candidate topics through a uniform randomization process.
- **TTSRank**:[11]: TTSRank is a topic trending scoring algorithm using language modeling based on query counts [11]. This algorithm builds an n-gram language model (LM) of queries for the current time, and compare it to several reference language models of past times (such as 1-day, 1-week, and 1-month ago). The trendiness is measured by the difference between the current probability of an n-gram and its highest reference probability.
- **LRRank** [24]: LRRank is an adaptive logistic regression model with features from both user queries and news space [24]. The query-related features describe the query counts, query length, number of characters, query category, query category score, query cluster size. The news-related features illustrate how likely the query is related to recent news articles.

4.3 Trending Topic Detection Performance

To answer **EQ1**, We first compare LGRank with the representative trending topic detection algorithms introduced in Section 4.2.3.

We use cross-validation to determine all the model parameters for each batch. Through cross-validation, we empirically set the latent dimension $k = 20$, model weights $\alpha = 1, \beta = 1e-6, \lambda = 0.001$ and the learning rate $\gamma = 0.005$. In addition, we find that when we set $N = 2$ (the number of CTR levels in Section 3.3), the detection performance will achieve the best. We provide details of parameter analysis in next subsection. For each ranking method, we compute the AUC_P, AUC_R and AUC_F1 in each batch, and then we take the best and compute the average performance over all batches, reported in Table 2 and Table 3. We have the following observations:

- In general, the proposed ranking method LGRank always outperforms baseline methods. The major reason is that the proposed framework exploits both local sequence modeling and global sequence correlation embedding, which can help to model temporality and user engagements simultaneously for trending topic detection.
- Moreover, we observe that *LRRank* > *TTSRank*. It indicates that the classification-based model with various contextual features from query and news space can help improve the ranking of trending topics than only using a language model to compute the differences of topic occurring likelihood.
- We can see that all methods are significantly better than random ranking method *Random*. It indicates the difficulty of trending topic ranking, and all the ranking methods can achieve improvements for detecting trending topics by considering various signals in query and entity spaces.

4.4 Evaluation of Local and Global Modeling

In addition to local sequence modeling, we also capture information from global sequence correlations. In order to answer **EQ2**, we further investigate the effects of these components by comparing the variants of LGRank:

- LRank: LRank is a variant of LGRank without considering information from global correlation embedding. We eliminate the effect of global correlation embedding by setting $\beta = 0$ in Eqn 8.
- GRank: GRank is a variant of LGRank without considering the local adversarial sequence modeling. We eliminate the effect of local sequence frequency prediction by setting $\alpha = 0$ in Eqn 8.
- LGRank\LG: LGRank\LG is a variant of LGRank, which eliminates both the local and global sequence modeling components. It only takes the topic preference ranking objective function by setting $\alpha = \beta = 0$ in Eqn 8.

The parameters in all the variants are determined by cross validation with the average and best performance reported under "variants" category in Table 2 and Table 3. We can observe that:

- When we eliminate the effect of local sequence adversarial modeling, the performance of GRank degrades in comparison with LGRank. For example, the performance reduces 7.53% and 8.29% in terms of best AUC_F1 and average AUC_F1. The results suggest that local sequence modeling in LGRank are important.
- We have a similar observation for LRank when eliminating the effect of global sequence embedding. The results suggest the importance to consider the correlations of topic sequences to guide trending topic detection in LGRank.
- When we eliminate both components in LGRank, the results are further reduced compared to LRank and GRank. It also suggests that components of local sequence modeling and global sequence correlation embedding are complementary to each other.

We conduct t-tests (with the significant level 0.05) on all comparisons and the t-test results suggest that all improvements are significant. With these observations, we can see that: (1) the proposed framework LGRank significantly improves trending topic ranking performances; and (2) information from both local and global provides complementary contributions to learning temporal representations of topics for trending topic detection.

Table 2: The best performance of each method

	Method	AUC		
		AUC_P	*AUC_R*	*AUC_F1*
Baselines	*Random*	125.02	177.32	130.45
	TTSRank	168.52	207.75	157.40
	LRRank	171.99	227.98	170.47
Variants	*LGRank\LG*	173.64	237.43	176.88
	LRank	189.56	239.17	181.06
	GRank	178.25	234.91	174.88
	LGRank	**197.94**	**246.01**	**188.06**

Table 3: The average performance of each method

	Method	AUC		
		AUC_P	*AUC_R*	*AUC_F1*
Baselines	*Random*	95.81	110.53	92.08
	TTSRank	120.68	129.24	110.60
	LRRank	124.70	132.86	114.22
Variants	*LGRank\LG*	127.31	133.23	115.23
	LRank	136.31	143.41	124.08
	GRank	131.11	138.28	119.19
	LGRank	**144.27**	**147.99**	**129.07**

4.5 Assessing Impacts of Adversarial Learning

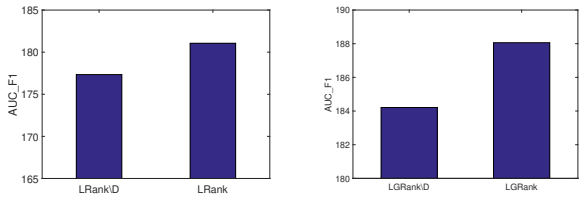
In this subsection, to answer **EQ3**, we focus on assessing the impacts of adversarial learning for local sequence modeling. Specifically, we perform experiments to understand how the adversarial learning strategy can affect the trending topic ranking performance and the optimization process in terms of convergence analysis. To this end, we define the following two variants of the proposed framework:

- **LGRank\D** – We eliminate the effect of discriminator component in LGRank, without considering the loss function as in Eqn. 5 from Eqn.8.
- **LRank\D** – We eliminate the effect of discriminator component in LRank, without considering the loss function as in Eqn. 5 and set $\beta = 0$ from Eqn. 8.

We run the methods in all batches and compare the ranking performances as in Figure 4 and the loss value convergence as in Figure 5. We have the following observations:

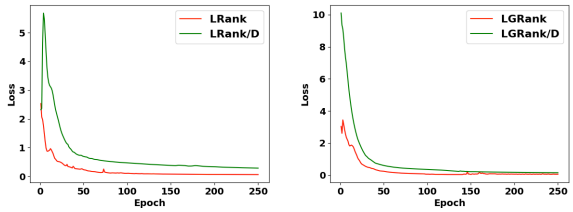
- We can see that without using the discriminator component for adversarial learning, the topic ranking performance degrades, i.e., *LRank\D* < *LRank* and *LGRank\D* < *LGRank*. For example, the performance of *LRank\D* reduces 2.1% in terms of AUC_F1 compared with LRank, and the performance of *LGRank\D* reduces 2.0% compared with LGRank. It demonstrates the importance to consider the adversarial learning to help capture the within-sequence frequency signals for trending topic detection.
- We observe methods with discriminator are consistently converging faster and achieve a lower optimal sequence loss values, in comparison with methods without discriminator. For example, *LRank\D* < *LRank* and *LGRank\D* < *LGRank* hold in terms of optimal loss values. In addition, the convergence speed of *LRank\D* and *LGRank\D* are slower than *LRank* and *LGRank*, respectively. It shows that with adversarial learning, the computational cost of detecting trending topics can be further reduced,

Through the analysis, we can see that by incorporating the discriminator to learn the temporal process within sequences, the



(a) LRank (b) LGRank

Figure 4: The effects of adversarial learning for LRank and LGRank w.r.t. ranking performance.



(a) LRank (b) LGRank

Figure 5: The effects of adversarial learning for LRank and LGRank w.r.t. convergence analysis.

proposed framework can achieve better trending detection performance as well as lower computational cost.

4.6 Case Study

We also perform a case study on a particular batch. We show the top ranked topics detected by LGRank as in Figure 6; and demonstrate the precision, recall, and F1 with respect to the different setting of k as in Figure 7. We have the following observations:

- The propose LGRank can effectively capture trending topics that are related to news events, even associated with different event types. For example, "Andy Samberg" is an actor that made a passionate plea to get Bruce Willis to guest star on the cop comedy, and "Weymouth Police" indicates the societal event of a mass shooting.
- The curves for *LGRank* is the topmost compared with other baselines and the variants of LGRank. As shown in Figure 7(a), the precision of *LGRank* is around 80% for top-50 topics, which is a great achievement compared with other baseline methods.

4.7 Parameter Analysis

In this section, we perform the parameter analysis to explore whether our framework can achieve the above experimental results with different parameter settings. In our methods, we have the following two important parameters α and β, which control the contribution of the local sequence embedding and global correlation embedding components. Due to the space limitation and similar observation for other settings, we only show the results for the batch with the best performance, and the results are shown in Figure 8. We vary the values of α in $\{0, 0.1, 1, 10.100\}$, and β in $\{0, 1e-7, 1e-6, 1e-5\}$. We empirically fix $\lambda = 0.001$ and latent dimension $k = 50$. In general, with the increase of α and β, the performance tends to first increase

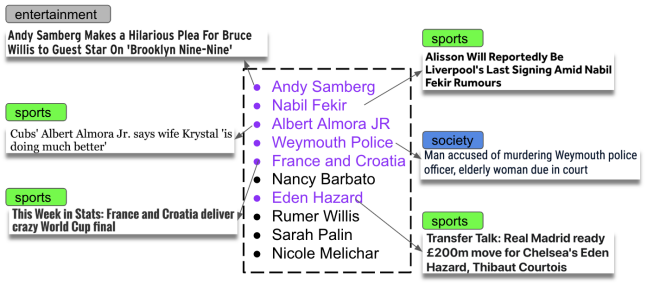


Figure 6: The trending topics captured by LGRank at batch 07/18/2018 3:00 pm UTC. We attach the snippets of news headlines related to the trending topics and the categories of the news events.

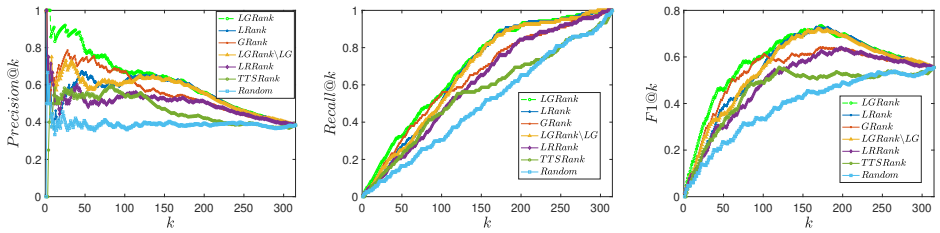
and then decrease. When α is too small, we may lose much information on local sequence adversarial modeling, and the frequency sequence prediction component does not have enough representation capacity. When α is large, the local sequence embeddings tends to overfit the training data. We have similar observations for parameter β. In certain regions, the performance of LGRank is relatively stable. For example, a value of $\alpha = 1$ and $\beta = 1e-6$ gives a relatively good performance. This observation eases the parameter selection process for trending topic ranking in practical online settings. In addition, we can see that when $\alpha = 0$ and $\beta = 0$, the method only utilize the topic preference ranking component to detect trending topics. We can see that performance has significantly drop compared with the other parameter settings. This result indicates that: (i) the performance is limited when only considering the topic preference ranking of CTRs from previous batches, which may fail to capture the change tendencies of topic trendiness; (ii) it is necessary to incorporate information from both local sequence adversarial learning and global correlation embedding for improving the performance of trending topic detection.

5 RELATED WORK

In this section, we briefly review the related work on i) trending topic detection; ii) neural temporal point process; iii) temporal correlation network analysis.

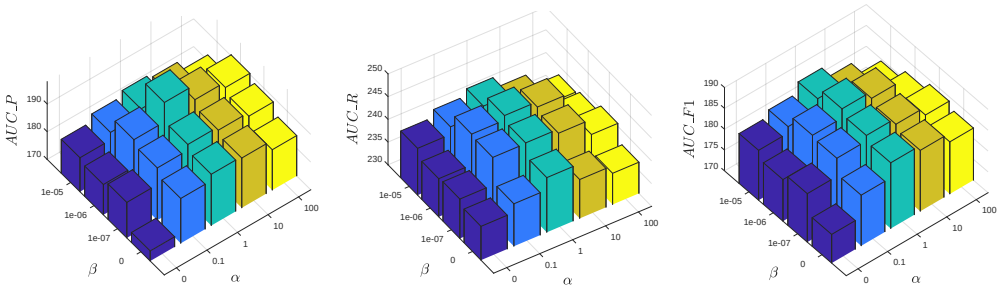
5.1 Trending Topic Detection

Trending topic detection aims to rank trending topics within a broad interest in time. In search engines, *query counts* provide a strong signal of crowd interests. Dong *et al.* [11] proposed to detect buzzing queries using historical query counts, and further generate an optimized document ranking list by extracting various recency related features. Golbandi *et al.* [15] developed an auto-regression model to predict query counts in the future to help predict potential trending queries at an early stage. Lee *et al.* [24] proposed to classify trending queries using features extracted from query intensities over time. Chen *et al.* [8] proposed to first classify queries into different categories by looking through historical query counts such as stable queries (perpetually popular queries such as "Apple"), one time burst query ("Texas shooting"), multiple times burst ("earthquake"), and periodic ("Christmas present"), and these results are utilized to improve document ranking results. Personalized trend



(a) Precision@k (b) Recall@k (c) F1@k

Figure 7: The evaluation curve of the best performance of each method at batch 07/18/2018 3:00 pm UTC.



(a) AUC_P (b) AUC_R (c) AUC_F1

Figure 8: The impact of α and β on LGRank for trending topic detection.

topic detection focuses on exploiting search contexts to provide personalized search trends for different users [17, 35]. Hashavit *et al.* [17] utilized the dynamic search context reflected by user search filters to optimize the search ranking results in real time.

It is worth mentioning that a related line of research is trending topic detection on social media, which considers a different scenario from that in search engines. For example, Chen *et al.* [7] developed an incremental clustering method to utilize several content and temporal social features to detect trending topics. Rao *et al.* proposed to predict the temporal statistics of tweet volume trends on social media [32]. Xie *et al.* focused on obtaining tweets in real-time and extract topics from them to detect trends [39]. Most of these work does not consider user engagements (e.g., clicks), which is necessary and important in our problem scenario.

In this paper, we propose a framework that consists of local adversarial sequence modeling, global sequence correlation embedding, and topic preference embedding simultaneously for trending topics detection for search engines.

5.2 Neural Temporal Point Process

Temporal point process is widely studied in various applications such as event sequence prediction [14] and network diffusion tracking [40], search behavior modeling [19]. Different from traditional temporal process models which usually require explicit parametric models whereby the conditional intensity function's form is manually pre-specified, such as Hawks process [25], neural temporal point process mostly does not require the intensity functions to be explicitly defined. Du *et al.* proposed a recurrent temporal point process framework to model the event timings and markers simultaneously through a LSTM network structure [14]. Based on that, Xiao *et al.* proposed an end-to-end framework modeling the intensity function of temporal point process by using RNNs for both background and history effects [38]. Several approaches utilize adversarial deep learning models such as GANs to better predict temporal sequences [36, 37]. Due to the batch updating scenario of trending topics, we treat the sequences of query frequencies as the synchronized series, and build a novel adversarial learning framework to better predict the frequency sequences of topics to capture trending signals.

5.3 Temporal Correlation Network

Temporal correlation network has been widely used to solve the real-life problems rising from different domains, such as financial marketing [28, 33, 34], climate science [13, 26], etc. The correlation based networks have been first introduced in [30] for financial market studies. [5] showed the correlation graph moving from a structured and clustered graph to a simple star-like graph with the decrease of the time interval between any two successive entries in the time series. [20] studied the dynamics of stock market correlations by visualizing the correlation graph in a 3D space over time. In climate science, [13] identified the *El Niño basin*, an area in the Pacific Ocean, where the surface temperatures are highly correlated to the temperatures at many other locations. They also found that the coming of El Niño climate actually disrupts correlations between temperatures at different locations worldwide. Furthermore, [26] leveraged this climate network for the prediction of El Niño

arrival more than 6 months ahead of time. In this paper, we propose a novel sequence representation learning framework to capture both local adversarial learning and global correlation embedding to better exploit the capacity of correlation network analysis.

6 CONCLUSION AND FUTURE WORK

In this paper, we investigate a novel problem of exploring the local and global information in temporal correlation network for trending topic detection. We propose a new topic ranking framework LGRank, which is composed of i) a local sequence modeling with adversarial neural networks; and ii) a correlation network embedding for global sequence learning; and iii) a topic preference ranking to optimize user engagements on different trending topics. Experiments on real-world datasets from a major commercial search engine show that the proposed framework outperforms state-of-the-art methods for trending topic detection. There are several interesting directions that need further investigation. First, we can consider the auxiliary information other than search logs to help detect trending topics, such as trending news, since trending news can contain additional trending signals. Second, we can explore how to consider the life cycles to capture the tendency such as increasing or decreasing, for predicting trending topics. Third, we can explore how to consider both the semantic and temporal information for improving trending topics detection performance.

ACKOWLEDGMENTS

This material is in part supported by the NSF award # 1614576.

REFERENCES

[1] Ziad Al Bawab, George H Mills, and Jean-Francois Crespo. 2012. Finding trending local topics in search queries for personalization of a recommendation system. In *KDD*.
[2] Ricardo Baeza-Yates, Carlos Hurtado, and Marcelo Mendoza. 2004. Query recommendation using query logs in search engines. In *International Conference on Extending Database Technology*. Springer, 588–596.
[3] Doug Beeferman and Adam Berger. 2000. Agglomerative clustering of a search engine query log. In *KDD*.
[4] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *NIPS*.
[5] Giovanni Bonanno, Guido Caldarelli, Fabrizio Lillo, Salvatore Miccichè, Nicolas Vandewalle, and Rosario Nunzio Mantegna. 2004. Networks of equities in financial markets. *The European Physical J. B-Condensed Matter and Complex Systems* 38, 2 (2004), 363–371.
[6] Carlos Castillo, Claudio Corsi, Debora Donato, Paolo Ferragina, and Aristides Gionis. 2008. Query-log mining for detecting spam. In *Proceedings of the 4th international workshop on Adversarial information retrieval on the web*. ACM, 17–20.
[7] Yan Chen, Hadi Amiri, Zhoujun Li, and Tat-Seng Chua. 2013. Emerging topic detection for organizations from microblogs. In *SIGIR*. ACM.
[8] Zhumin Chen, Haichun Yang, J Ma, J Lei, and H Gao. 2011. Time-based query classification and its application for page rank. *J Comput Info Sys* 7 (2011), 3149–3156.
[9] Hyunyoung Choi and Hal Varian. 2012. Predicting the present with Google Trends. *Economic Record* 88 (2012), 2–9.
[10] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).
[11] Anlei Dong, Yi Chang, Zhaohui Zheng, Gilad Mishne, Jing Bai, Ruiqiang Zhang, Karolina Buchner, Ciya Liao, and Fernando Diaz. 2010. Towards recency ranking in web search. In *WSDM*.
[12] Anlei Dong, Ruiqiang Zhang, Pranam Kolari, Jing Bai, Fernando Diaz, Yi Chang, Zhaohui Zheng, and Hongyuan Zha. 2010. Time is of the essence: improving recency ranking using twitter data. In *WWW*.
[13] J. F. Donges, Y. Zou, N. Marwan, and J. Kurths. 2009. Complex networks in climate dynamics - comparing linear and nonlinear network construction methods. *European Physical J. - Special Topics* 174 (2009), 157–179.
[14] Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. 2016. Recurrent marked temporal point processes: Embedding event history to vector. In *KDD*.
[15] Nadav Golbandi Golbandi, Liran Katzir Katzir, Yehuda Koren Koren, and Ronny Lempel Lempel. 2013. Expediting search trend detection via prediction of query counts. In *WSDM*.
[16] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *NIPS*.
[17] Anat Hashavit, Roy Levin, Ido Guy, and Gilad Kutiel. 2016. Effective Trend Detection within a Dynamic Search Context. In *SIGIR*.
[18] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
[19] Shubhra Kanti Karmaker Santu, Liangda Li, Dae Hoon Park, Yi Chang, and ChengXiang Zhai. 2017. Modeling the influence of popular trending events on user search behavior. In *WWW*.
[20] Dror Y Kenett, Yoash Shapira, Asaf Madi, Sharron Bransburg-Zabary, Gitit Gur-Gershgoren, and Eshel Ben-Jacob. 2010. Dynamics of stock market correlations. *AUCO Czech Economic Review* 4, 3 (2010), 330–341.
[21] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[22] Alex M Lamb, Anirudh Goyal ALIAS PARTH GOYAL, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. 2016. Professor forcing: A new algorithm for training recurrent networks. In *NIPS*.
[23] Vasileios Lampos, Andrew C Miller, Steve Crossan, and Christian Stefansen. 2015. Advances in nowcasting influenza-like illness rates using search query logs. *Scientific reports* 5 (2015), 12760.
[24] Chi-Hoon Lee, HengShuai Yao, Xu He, Su Han Chan, JieYang Chang, and Farzin Maghoul. 2014. Learning to predict trending queries: classification-based. In *WWW*.
[25] Liangda Li, Hongbo Deng, Jianhui Chen, and Yi Chang. 2017. Learning parametric models for context-aware query auto-completion via hawkes processes. In *WSDM*.
[26] Josef Ludescher, Avi Gozolchiani, Mikhail I Bogachev, Armin Bunde, Shlomo Havlin, and Hans Joachim Schellnhuber. 2014. Very early warning of next El Niño. *PNAS* (2014).
[27] Catherine Lui, P Takis Metaxas, and Eni Mustafaraj. 2011. On the predictability of the US elections through search volume activity. (2011).
[28] Ping Luo, Kai Shu, Junjie Wu, Li Wan, and Yong Tan. 2020. Exploring Correlation Network for Cheating Detection. *ACM Transactions on Intelligent Systems and Technology (TIST)* 11, 1 (2020), 1–23.
[29] Yao Ma, Suhang Wang, ZhaoChun Ren, Dawei Yin, and Jiliang Tang. 2017. Preserving Local and Global Information for Network Embedding. *arXiv preprint arXiv:1710.07266* (2017).
[30] Rosario N Mantegna. 1999. Hierarchical structure in financial markets. *The European Physical J. B-Condensed Matter and Complex Systems* 11, 1 (1999), 193–197.
[31] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.
[32] Jinfeng Rao, Ferhan Ture, Xing Niu, and Jimmy Lin. 2017. Mining the temporal statistics of query terms for searching social media posts. In *ICTIR*. ACM.
[33] Kai Shu, Ping Luo, Wan Li, Peifeng Yin, and Linpeng Tang. 2014. Deal or deceit: detecting cheating in distribution channels. In *CIKM*. ACM.
[34] Michele Tumminello, Fabrizio Lillo, and Rosario N Mantegna. 2010. Correlation, hierarchies, and networks in financial markets. *J. of Economic Behavior & Organization* 75, 1 (2010), 40–58.
[35] Chun-Che Wu, Tao Mei, Winston H Hsu, and Yong Rui. 2014. Learning to personalize trending image search suggestion. In *SIGIR*.
[36] Shuai Xiao, Mehrdad Farajtabar, Xiaojing Ye, Junchi Yan, Le Song, and Hongyuan Zha. 2017. Wasserstein learning of deep generative point process models. In *NIPS*.
[37] Shuai Xiao, Hongtent Xu, Junchi Yan, Mehrdad Farajtabar, Xiaokang Yang, Le Song, and Hongyuan Zha. 2018. learning conditional generative models for temporal point processes. In *AAAI*.
[38] Shuai Xiao, Junchi Yan, Xiaokang Yang, Hongyuan Zha, and Stephen M Chu. 2017. Modeling the Intensity Function of Point Process Via Recurrent Neural Networks.
[39] Wei Xie, Feida Zhu, Jing Jiang, Ee-Peng Lim, and Ke Wang. 2016. Topicsketch: Real-time bursty topic detection from twitter. *TKDE* (2016).
[40] Shuang-Hong Yang and Hongyuan Zha. 2013. Mixture of mutually exciting processes for viral diffusion. In *ICML*.