Real-time Targeted Influence Maximization for Online Advertisements

Yuchen Li Dongxiang Zhang Kian-Lee Tan
Department of Computer Science
School of Computing, National University of Singapore
{liyuchen,zhangdo,tankl}@comp.nus.edu.sg

ABSTRACT

Advertising in social network has become a multi-billion-dollar industry. A main challenge is to identify key influencers who can effectively contribute to the dissemination of information. Although the influence maximization problem, which finds a seed set of k most influential users based on certain propagation models, has been well studied, it is not target-aware and cannot be directly applied to online advertising. In this paper, we propose a new problem, named Keyword-Based Targeted Influence Maximization (KB-TIM), to find a seed set that maximizes the expected influence over users who are relevant to a given advertisement. To solve the problem, we propose a sampling technique based on weighted reverse influence set and achieve an approximation ratio of $(1-1/e-\varepsilon)$. To meet the instant-speed requirement, we propose two disk-based solutions that improve the query processing time by two orders of magnitude over the state-of-the-art solutions, while keeping the theoretical bound. Experiments conducted on two real social networks confirm our theoretical findings as well as the efficiency. Given an advertisement with 5 keywords, it takes only 2 seconds to find the most influential users in a social network with billions of edges.

1. INTRODUCTION

Many companies have started to use social network as the main advertising tool to precisely target the customers. Such trends brought Facebook a total advertising revenue of 4.28 Billion dollars in 2012[1]. *Influence Maximization (IM)* is a key algorithmic problem behind online viral marketing. By word-of-mouth propagation effect among friends, it finds a seed set of k users to maximize the expected influence among all the users in a social network. Since the problem is NP-Hard, Kempe et al. [15] first proposed a greedy algorithm to solve the IM problem, which returns a seed set with a $(1-1/e-\varepsilon)$ approximation ratio to the optimal solution. However, the greedy solution still takes a prohibitively long time to finish. To address the efficiency issue, a state-of-the-art solution, named Reverse Influence Set (RIS), was proposed to support $(1-1/e-\varepsilon)$ approximation ratio [2, 21]. The method uses random sampling and can support various propagation models that have been proposed. Despite the performance improvement, it still takes nearly an hour to find the most influential users in a social network with millions of nodes.

There have been some efforts to extend the influence maximization problem to topic-aware IM [1, 13, 4, 18, 3] so as to support online advertisements. The propagation model is required to take into account influence probability based on different topics. However, all of the proposed techniques suffer from the efficiency issue. Their models require offline training of the propagation probability w.r.t. different topics, which is not scalable to the graph size and number of topics. The most recent work [3] was reported to handle a graph with 4 million vertices and 10 topics only. In addition, the proposed solutions are all heuristic and none of them provides theoretical guarantee on the quality of the results.

To bridge the gap, we propose a new Keyword-Based Targeted Influence Maximization (KB-TIM) query for online targeted advertising. The query finds a seed set that maximizes the expected influence over users who are relevant to a given advertisement. In other words, the expected influence only incorporates those users interested in the advertisement as targeted customers. To solve the problem, we propose a weighted sampling technique based on RIS and achieve an approximation ratio of $(1-1/e-\varepsilon)$. However, the method needs to generate hundreds of thousands of random sample sets to guarantee the theoretical bound and requires intensive computation overhead. To meet the real-time requirement, we propose two disk-based solutions, RR and IRR, that improve the query processing performance. The idea is to push the sampling procedure from online to offline and build index on the random sample sets for each keyword. During query processing, RR directly loads all the related random sample sets into memory and uses the greedy algorithm on the maximum coverage problem [22] to find the top-k seed users. IRR futher improves over RR by incrementally loading the most promising RR sets into memory and adopts the top-k aggregation strategy to save computation costs. Our contributions can be summarized as follows:

1. We propose a KB-TIM query to support scalable social IM in online advertising platforms.

[1]http://www.engadget.com/2013/01/30/facebook-2012-q4-earnings/

Proceedings of the VLDB Endowment, Vol. 8, No. 10
Copyright 2015 VLDB Endowment 2150-8097/15/06.

2. We propose a weighted sampling technique, i.e. WRIS, based on RIS and show that it has an approximation ratio of $(1 - 1/e - \varepsilon)$ to the optimal solution to a KB-TIM query.
3. To meet the instant-speed requirement, we propose two disk-based solutions that improve the running time by two orders of magnitude over WRIS, while preserving the theoretical bound.
4. We evaluate the performance on real social network with billions of edges and hundreds of topics. The experiment results confirm our theoretical findings and show that the two disk-based methods significantly outperform the weighted online sampling.

We present the preliminaries in Section 2 and the problem definition of KB-TIM query in Section 3. We extend the RIS method and propose WRIS in Section 3.2. Then two disk-based methods, RR and IRR, are presented in Sections 4 and 5 respectively. We report the experimental results in Section 6. Subsequently, various IM techniques are reviewed in Section 7. Finally we conclude the paper in Section 8.

2. PRELIMINARY

In this section, we review the classic influence maximization problem as well as the state-of-the-art solutions to the problem.

2.1 Influence Maximization (IM)

Consider a directed graph $G = (V, E)$ where vertices in V are users and edges in E capture the friendships or follow relationships in a social network. To model the propagation process, a number of methods such as independent cascade (IC) model [9], linear threshold (LT) model [11] and general triggering model [15] have been proposed. In this paper, we adopt IC model because it has been widely used in previous work [21, 5, 6, 19] [2].

Under the IC model, each directed edge $e = (u, v)$ is associated with an *influence probability* $p(e)$ to measure the social impact from user u to user v. This probability is normally set to $p(e) = \frac{1}{N_v}$, where N_v is the in-degree of v [3]. Each user is either in an "active" state or "inactive" state, and an active user can activate his inactive neighbors with probability $p(e)$. Once a user is activated, his (active) state remains unchanged. Initially, a set of seed users S are selected to influence other people and their states are set to be active at time step 0. Then, each active user at time step i will activate the neighbors that are inactive at time step $i+1$ with probability $p(e)$. Note that each user u has only one chance to activate his neighbors. In other words, we flip a coin with $P(head) = p(e)$ and if head occurs, v is activated by u. Otherwise, v can only be activated by other incoming neighbors except u. This procedure terminates when no more users can be activated.

Let $I(S)$ denote the set of nodes activated by the seeds S in an instance of the influence propagation process. Intuitively, the IM problem finds a seed set S^* with k users to maximize the expected number of users influenced by a seed set S, denoted by $\mathbb{E}[I(S)]$, over the social network and can be formally defined as follows:

[2]Note that the methods proposed in this paper can support LT and general triggering model as well.

[3]Our proposed methods are also independent of how $p(e)$ is set.

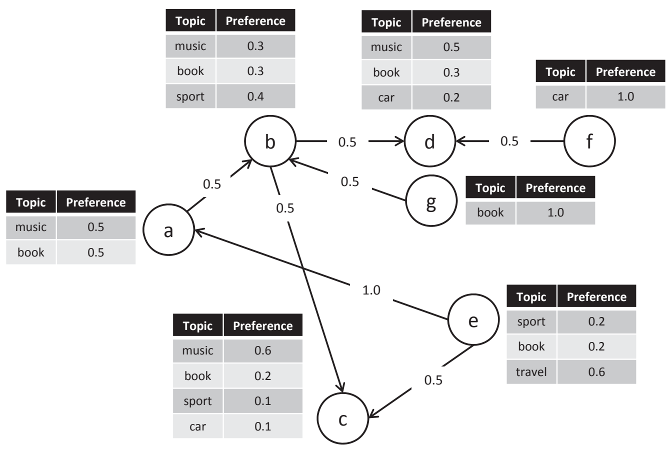


Figure 1: the social network adopted in the paper.

Definition 1 (IM). *Let OPT_k denote the maximum expected influence spread of any node set with size k, i.e. $OPT_k = \max_{S \subseteq V, |S|=k} \mathbb{E}[I(S)]$. The IM problem finds an optimal seed set S^* with k users such that $\mathbb{E}[I(S^*)] = OPT_k$.*

Due to the linearity of expectation, $\mathbb{E}[I(S)]$ can also be expressed as $\mathbb{E}[I(S)] = \sum_{v \in V} p(S \mapsto v)$ where $p(S \mapsto v)$ is the probability with which a user v is activated by seed set S. The following is an example of how to compute the expected influence given a seed set and how to retrieve the optimal influential seed set on a social graph.

Example 1. *In Figure 1, we have a social network with 7 users $\{a, b, c, d, e, f, g\}$ and the edge value is the influence probability from one user to his neighbor. Let us assume, at time step 0, the seed set S contains two nodes e and f, i.e. $S = \{e, f\}$. Suppose after flipping the coin in step 1, nodes (a, c) are activated by e and node d is activated by f, but in step 2, node a fails to activate b. Then, the process terminates because no more nodes can be further activated and $I(S) = \{a, c, d, e, f\}$.*

Although evaluating $p(S \mapsto v)$ has been proven to be #P hard in [5], the calculation of the probability is feasible in this small example graph. Take $p(\{e, g\} \mapsto b)$ as an example: since $p(e \mapsto b) = 0.5$ and $p(g \mapsto b) = 0.5$ and each activated node has only one chance to activate the neighbors, the probability of b being activated by $\{e, g\}$ is $p(\{e, g\} \mapsto b) = 1 - [1 - p(e \mapsto b)] \cdot [1 - p(g \mapsto b)] = 0.75$. We can then find, among all possible seed sets with two nodes, the optimal initial set is $S^* = \{e, g\}$ for the IM problem and $\mathbb{E}[I(S^*)] = \sum_{v \in V} p(S^* \mapsto v) = 1 + 0.75 + 0.6875 + 0.375 + 1 + 0 + 1 = 4.8125$.

2.2 Reverse Influence Set (RIS)

The state-of-the-art solutions to the IM problem [2, 21] are based on a sampling technique called *Reverse Influence Set (RIS)*. To facilitate the understanding of RIS, we first introduce the concepts of the *Reverse Reachable* (RR) Set and *Random RR Set* [2, 21]:

Definition 2. *Let G' denote a sub-graph of G generated by removing any edge $e \in E$ with probability $1 - p(e)$. The Reverse Reachable (RR) Set for vertex v contains all the vertices in G' that can reach v. Then, a random RR set* *is generated on an instance of G' sampled from G and v is randomly picked from V.*

Intuitively, the random RR set generated from a random user v contains the users who can influence v. By building multiple random RR sets on different random users, if a user u has a great impact on other people, u will have a higher probability to appear in these random RR sets. Similarly, if S^* covers most of the RR sets, S^* is likely to maximize $\mathbb{E}[I(S))]$. Based on this idea, the query processing framework of RIS works as follows:

1. Generate θ random RR sets from G.
2. Use the standard greedy algorithm on the *maximum coverage* problem [22] to select k users to cover the maximum number of RR sets generated above. The result set S^* is a $(1 - 1/e - \varepsilon)$-approximate solution for the IM problem.

In [21], it was proved that when θ is sufficiently large, RIS returns near-optimal results with high probability:

THEOREM 1 ([21]). *If $\theta \geq (8+2\varepsilon)\cdot|V|\cdot\frac{\ln|V|+\ln\binom{|V|}{k}+\ln 2}{OPT_k\cdot\varepsilon^2}$, RIS returns a $(1-1/e-\varepsilon)$-approximate solution with at least $1-|V|^{-1}$ probability.*

The proof sketch in [21] is summarized as follows:

S1: Let $F_\theta(S)$ denote the number of sampled RR sets covered by a seed set S and prove $\mathbb{E}[\frac{F_\theta(S)\cdot|V|}{\theta}] = \mathbb{E}[I(S)]$.

S2: Based on the property in S1, show that if $\theta \geq (8+2\varepsilon)|V|\cdot\frac{\ln|V|+\ln\binom{|V|}{k}+\ln 2}{OPT_k\cdot\varepsilon^2}$, we have $|\frac{F_\theta(S)\cdot|V|}{\theta} - \mathbb{E}[I(S)]| < \frac{\varepsilon}{2}\cdot OPT_k$ holds with probability $1-|V|^{-1}$ simultaneously for all S s.t $|S| = k$.

S3: Utilize the greedy algorithm of maximum coverage problem which produces a $(1-1/e)$ approximation solution.

S4: By combining the two approximation ratios $\frac{\varepsilon}{2}$ and $(1-1/e)$ in S2 and S3, show the final approximation ratio is $(1-1/e-\varepsilon)$ with at least $1-|V|^{-1}$ probability.

EXAMPLE 2. *Suppose $k = 2$, $\theta = 4$ and four random RR sets $G_d = \{b, d, f\}$, $G_e = \{e\}$, $G_d = \{d, f\}$ and $G_b = \{a, b, e\}$ are generated from the social graph in Figure 1 (d is sampled twice). Then $\{e, f\}$ will be selected as the seed set as $\{e, f\}$ intersects (or "covers") all the random RR sets generated.*

3. TARGETED INFLUENCE MAXIMIZATION

In this section, we define Keyword-Based Targeted Influence Maximization (KB-TIM) query and propose a baseline solution.

3.1 Problem Definition

To model a social network as an online advertisement platform, we extend each node v in G to be associated with a user profile represented by a weighted term vector. Each advertisement is modeled as a weighted term vector and the impact of an advertisement to an end user is calculated as the similarity between two term vectors. If a user does not contain any keyword in the advertisement, we consider the user not being impacted. Our goal is to find k seeds in the social network, that generate the maximum impact based on the influence propagation model.

Formally, let $G = (V, E, T)$ be a social network for online advertisements where T is a universal topic space to model user interests. Each user is associated with a weighted term vector to capture the preference in different topics. The term vector can be generated by aggregating all the social activities such as new posts, like and sharing into a single document and then use existing topic modeling techniques [12] to map explicit keywords into the latent topic space T. Let $tf_{w,v}$ denote the preference weight between a user v and a topic w and idf_w denote the inverse document frequency for a topic w. By applying tf-idf model [23] on the topic space, the impact of an advertisement with a keyword set $Q.T \subseteq T^4$ on a user v is defined as:

$$\phi(v, Q) = \sum_{w\in Q.T} tf_{w,v}\cdot idf_w \qquad (1)$$

Then, the impact of k selected seeds w.r.t an advertisement can be calculated by accumulating the impact to each activated user when the influence propagation terminates. Let $I^Q(S)$ denote the influence score of $I(S)$ w.r.t the query keywords $Q.T$, i.e. $I^Q(S) = \sum_{v\in I(S)}\phi(v, Q)$. we have

$$\mathbb{E}[I^Q(S)] = \mathbb{E}[\sum_{v\in I(S)}\phi(v,Q)] = \sum_{v\in V} p(S\mapsto v)\cdot\phi(v,Q) \qquad (2)$$

To this end, we define our keyword based targeted influence maximization problem as follows:

DEFINITION 3. *(Keyword Based Targeted Influence Maximization (KB-TIM)). A KB-TIM query Q on a social graph G is associated with a tuple $(Q.T, Q.k)$, where $Q.T \subseteq T$ is the advertisement keyword set and $Q.k$ is the number of seed users. Let $OPT^{Q.T}_{Q.k}$ denote the maximum expected influence spread of any size-$Q.k$ node w.r.t the weighting function for $Q.T$, i.e. $OPT^{Q.T}_{Q.k} = max_{S\subseteq V,|S|=Q.k}\mathbb{E}[I^{Q.T}(S)]$. A KB-TIM query finds $Q.k$ seed users to achieve $OPT^{Q.T}_{Q.k}$.*

EXAMPLE 3. *In Figure 1, each node is associated with a user profile depicting the user's preferences on different topics. Suppose a KB-TIM query is $Q = (\{music\}, 2)$, the optimal seed set S^* selected by Q should be $\{b, e\}$ and the expected impact is $\mathbb{E}[I^{\{music\}}(S^*)] = \sum_{v\in V} p(S^*\mapsto v)\cdot\phi(v,\{music\}) = 1\cdot 0.5 + 1\cdot 0.3 + 0.75\cdot 0.6 + 0.1875\cdot 0.5 + 0\cdot 0 + 0\cdot 0 + 0\cdot 0 = 1.5$. The result is different from the general IM problem in Example 2 as we are now considering targeted influence maximization.*

3.2 Weighted RIS Sampling

To facilitate our presentation, all frequently used notations are listed in Table 1. In the traditional IM problem, all the users are treated as candidates to be influenced. However, in our KB-TIM query, we only target at the users associated with query keywords in the advertisement. The original uniform sampling technique cannot work in the new scenario because the condition S1 in the proof sketch of Theorem 1 does not hold, i.e., $\mathbb{E}[\frac{F_\theta(S)}{\theta}\cdot|V|] \neq \mathbb{E}[I^Q(S)]$. Therefore, we need to find a new unbiased estimator for $\mathbb{E}[I^Q(S)]$. In this section, we propose a weighted RIS (WRIS) sampling technique for the KB-TIM query processing.

[4]In this paper, we use 'topic' and 'keyword' interchangeably.

Table 1: Frequent notations used across the paper

Notation	meaning
G, V, E, T	the social network, the vertex set, the edge set and the topic space respectively.
$Q(Q.T, Q.k)$	the KB-TIM query Q. $Q.T$ are the topics queried and $Q.k$ is the size of seed set. We use Q to denote $Q.T$ if there is no ambiguity in the context.
$OPT_{Q.k}^{Q.T}$	the maximum expected spread among all seed set with size $Q.k$ for query keywords $Q.T$.
$F_\theta(S)$	the number of RR sets, which is covered by seed set S.
$tf_{w,v}$	the preference weight between a user v and a topic w.
idf_w	the inverse document frequency for a topic w.
ϕ_w	sum of the relevance scores among all users for a given topic w, i.e., $\sum_{v\in V} tf_{w,v} \cdot idf_w$.
$\phi(v, Q)$	the relevance score of a KB-TIM query Q to a user v, i.e., $\sum_{w\in Q.T} tf_{w,v} \cdot idf_w$.
ϕ_Q	sum of the relevance scores among all users to Q, i.e., $\sum_{v\in V} \phi(v, Q)$.
θ	the number of RR sets needed to process Q using WRIS.
p_w	the proportion of RR sets w.r.t a topic w among all RR sets needed to process Q, i.e. ϕ_w/ϕ_Q.
$p_s(v, w)$	the probability to sample a user v w.r.t a topic w in discriminative WRIS sampling.
$I(S)$	an instance of the influence spread of a seed set S.
$I^Q(S)$	influence score of $I(S)$ w.r.t query keywords $Q.T$, $I^Q(S) = \sum_{v\in I(S)} \phi(v, Q)$

To differentiate the sampling probability of targeted users from non-targeted users, we define the sampling probability for a user v w.r.t a KB-TIM query Q as follows:

$$p_s(v, Q) = \frac{\phi(v, Q)}{\sum_{v\in V} \phi(v, Q)} = \frac{\phi(v, Q)}{\phi_Q} \tag{3}$$

where we denote $\phi_Q = \sum_{v\in V} \phi(v, Q)$. Intuitively, the users are sampled based on their relevancies to the query. Users whose profiles are more relevant have a higher probability to be sampled. Under the weighted sampling scheme, we propose a new unbiased estimator for computing the expected influence score:

Lemma 1. $\mathbb{E}[\frac{F_\theta(S)}{\theta}] \cdot \phi_Q = \mathbb{E}[I^Q(S)]$

Proof. Let R_v be the RR set by first sampling v with probability $p_s(v, Q) = \frac{\phi(v,Q)}{\phi_Q}$ and then sampling a RR given v to form R_v, it follows:

$$\mathbb{E}[\frac{F_\theta(S)}{\theta}] = \sum_{v\in V} \frac{\phi(v, Q)}{\phi_Q} \cdot p(R_v \cap S \neq \emptyset) \tag{4}$$

$p(R_v \cap S \neq \emptyset) = p(S \mapsto v)$ where $p(S \mapsto v)$ is the probability that S activates v on G. This leads to:

$$\mathbb{E}[\frac{F_\theta(S)}{\theta}] = \sum_{v\in V} \frac{\phi(v, Q)}{\phi_Q} \cdot p(S \mapsto v) = \frac{\mathbb{E}[I^Q(S)]}{\phi_Q} \tag{5}$$

This means $\frac{F_\theta(S)}{\theta} \cdot \phi_Q$ is an unbiased estimator to the expected influence score of any seed set S for KB-TIM query. □

With the sampling scheme, we propose our WRIS method as follows:

1. Sample θ number of vertices from G with a probability of $p_s(v, Q)$ (in Eqn. 3) for any vertex v.
2. For each vertex v sampled, sample a RR set for v.
3. Follow step 2 of RIS

The steps of WRIS are similar to the RIS algorithm, except that we need to determine a θ value as the minimum number of random RR sets to guarantee that $WRIS$ can also achieve a theoretical bound like Theorem 1 for RIS. Theorem 2 gives a sufficient lower bound for θ such that our WRIS algorithm is a $(1-1/e-\varepsilon)$-approximate solution with high probability.

Theorem 2. *If θ satisfies:*

$$\theta \geq (8 + 2\varepsilon)\phi_Q \cdot \frac{\ln|V| + \ln\binom{|V|}{Q.k} + \ln 2}{OPT_{Q.k}^{Q.T} \cdot \varepsilon^2} \tag{6}$$

the weighted RIS method returns a $(1-1/e-\varepsilon)$-approximate solution with at least $1 - |V|^{-1}$ probability.

With the revised sampling technique in Lemma 1, the proof is similar to that in Theorem 1 (S1-S4). A complete version is presented in [20].

4. DISK-BASED RR INDEX FOR KB-TIM QUERY PROCESSING

Existing methods [2, 21] solve the IM problem by sampling random RR sets online, this takes prohibitively long time to finish on social networks with millions of users which hinders interactive market analysis and decision. To achieve real-time response, we move the sampling procedure offline and propose a disk-based RR index to store the pre-computed sampling sets. In this section, we first introduce how to build an RR index for each keyword and then present our query processing algorithm based on RR index.

4.1 Discriminative WRIS Sampling

We note that the sampling probability $p_s(v, Q)$ appeared in Eqn. 3 relies on the query and cannot be determined in advance. Hence, given a query, we cannot pre-compute the random RR sets using $p_s(v, Q)$. To support offline sampling, we propose discriminative WRIS sampling by rewriting $p_s(v, Q)$ as

$$\begin{aligned} p_s(v, Q) &= \frac{\sum_{w\in Q.T} tf_{v,w} \cdot idf_w}{\phi_Q} \\ &= \sum_{w\in Q.T} \frac{tf_{v,w} \cdot idf_w}{\sum_{v\in V} tf_{v,w} \cdot idf_w} \cdot \frac{\sum_{v\in V} tf_{v,w} \cdot idf_w}{\phi_Q} \\ &= \sum_{w\in Q.T} \frac{tf_{v,w}}{\sum_{v\in V} tf_{v,w}} \cdot \frac{\sum_{v\in V} tf_{v,w} \cdot idf_w}{\phi_Q} \end{aligned}$$

Let $p_s(v, w) = \frac{tf_{v,w}}{\sum_{v\in V} tf_{v,w}}$ and $p_w = \frac{\sum_{v\in V} tf_{v,w} \cdot idf_w}{\phi_Q}$. We have

$$p_s(v, Q) = \sum_{w\in Q.T} p_s(v, w) \cdot p_w \tag{7}$$

This implies that we can sample the RR sets for each keyword w and store them on disk as an offline step. When a query Q is issued, the RR sets associated with the relevant keywords are loaded and merged. The random RR sets generated by our discriminative WRIS sampling still guarantee

Algorithm 1: BuildRR(TopicSet T)

```
1 RRIndex $(R, L) \leftarrow (\emptyset, \emptyset)$
2 for $w \in T$ do
3     Compute $\hat{\theta}_w$ RR sets
4     $R_w \leftarrow$ generate $\theta_w$ RR sets with probability $p_s(v, w)$
5     $L_w \leftarrow$ inverse mapping of $R_w$
6 Store $(R, L)$
```

the same theoretical bound as in Theorem 2 based on the following lemma:

LEMMA 2. *Given a query Q, let θ_w^Q be the number of RR sets sampled by a probability of $p_s(v, w)$ w.r.t each vertex v given a query keyword w. Then θ^Q is the total number of RR sets and $\theta^Q = \sum_{w \in Q.T} \theta_w^Q$. If $\theta^Q \geq \theta$ and $\frac{\theta_w^Q}{\theta^Q} = p_w$, we can achieve the same theoretical bound as in Theorem 2 for the discriminative WRIS sampling.*

PROOF. On one hand, as the RR samples are taken independently, the expected fraction of RR samples w.r.t v among all the RR samples using WRIS should be $p_s(v, Q)$ according to Eqn. 7. On the other hand, the discriminative sampling is expected to sample $\sum_{w \in Q.T} [(\theta^Q \cdot p_w) \cdot p_s(v, w)]$ number of samples w.r.t each user v if we sample θ^Q RR sets. Therefore the expected fraction of RR sets w.r.t v among the θ^Q RR samples is: $\sum_{w \in Q.T} [(\theta^Q \cdot p_w) \cdot p_s(v, w)] / \theta^Q = p_s(v, Q)$ which is the same as the WRIS method. Besides, we take more RR samples than the desired threshold specified in Eqn. 6 for WRIS. Thus we conclude that the discriminative sampling achieves at the same theoretically accuracy as the WRIS method as in Theorem 2. □

4.2 RR Index Construction

As we want to pre-compute the RR sets w.r.t keyword $w \in Q.T$ in an offline step and merge the relevant RR sets for query processing, we need to determine how many RR sets needs to be built for the RR index w.r.t each w. We define K to be a system specified parameter such that $Q.k \leq K$ $\forall Q$. This is because $Q.k$ will not be interesting if $Q.k$ is large since the purpose of IM is to influence a large group of people by a small seed set for budgeted advertisement. Since $\theta \cdot p_w$ is a value dependent on Q, we cannot set $\theta_w^Q = \theta \cdot p_w$ in the offline sampling. Hence, we need to find a θ_w value that is independent of Q and satisfies $\theta_w \geq \theta \cdot p_w$, but as small as possible to save the time and space cost for index construction. Following this intuition, we have the following lemma:

LEMMA 3. *For any keyword w, if we choose*

$$\hat{\theta}_w = (8 + 2\varepsilon)(\sum_{v \in V} tf_{w,v}) \cdot \frac{\ln |V| + \ln \binom{|V|}{K} + \ln 2}{OPT_1^{\{w\}} \cdot \varepsilon^2} \quad (8)$$

we have $\hat{\theta}_w \geq \theta \cdot p_w$.

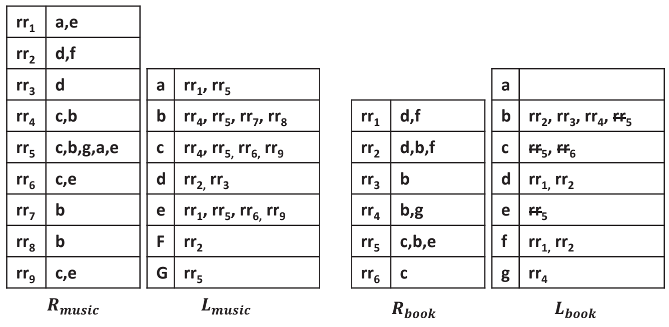
PROOF. According to Eqn. 6 and the definition of p_w, we have:

$$\theta \cdot p_w = (8 + 2\varepsilon)(\sum_{v \in V} tf_{w,v}) \cdot \frac{\ln |V| + \ln \binom{|V|}{Q.k} + \ln 2}{OPT_{Q.k}^{Q.T} / idf_w \cdot \varepsilon^2} \quad (9)$$

Now let us prove $OPT_{Q.k}^{\{w\}} \leq OPT_{Q.k}^{Q.T} / idf_w$. Let S^w be the seed set that achieves influence of $OPT_{Q.k}^{\{w\}}$ w.r.t the weight of w, we have: $\mathbb{E}[I^{Q.T}(S^w)] \leq OPT_{Q.k}^{Q.T}$ according to the definition of $OPT_{Q.k}^{Q.T}$. Then:

$$\begin{aligned}
&\frac{OPT_{Q.k}^{Q.T}}{idf_w} - OPT_{Q.k}^{\{w\}} \\
&\geq \frac{\mathbb{E}[I^{Q.T}(S^w)]}{idf_w} - OPT_{Q.k}^{\{w\}} \\
&\geq \frac{1}{idf_w} \mathbb{E}[\sum_{v \in I(S^w)} \sum_{w^* \in Q.T} tf_{w^*,v} \cdot idf_{w^*}] - OPT_{Q.k}^{\{w\}} \\
&\geq \sum_{w^* \in Q.T \setminus \{w\}} \mathbb{E}[\sum_{v \in I(S^w)} tf_{w^*,v} \frac{idf_{w^*}}{idf_w}] + \mathbb{E}[\sum_{v \in I(S^w)} tf_{w,v}] - OPT_{Q.k}^{\{w\}} \\
&\geq \sum_{w^* \in Q.T \setminus \{w\}} \mathbb{E}[\sum_{v \in I(S^w)} tf_{w^*,v} \frac{idf_{w^*}}{idf_w}] + 0 \geq 0
\end{aligned}$$

Since the influence spread is monotonic w.r.t the size of the seed set [15], $OPT_1^{\{w\}} \leq OPT_{Q.k}^{\{w\}} \leq OPT_{Q.k}^{Q.T} / idf_w$. In addition, as $Q.k \leq K$, $\ln \binom{|V|}{Q.k} \leq \ln \binom{|V|}{K}$ when $K \leq |V|/2$. Thus we can conclude that $\hat{\theta}_w \geq \theta \cdot p_w$. □



rr_1	a,e
rr_2	d,f
rr_3	d
rr_4	c,b
rr_5	c,b,g,a,e
rr_6	c,e
rr_7	b
rr_8	b
rr_9	c,e

R_{music}

a	rr_1, rr_5
b	rr_4, rr_5, rr_7, rr_8
c	rr_4, rr_5, rr_6, rr_9
d	rr_2, rr_3
e	rr_1, rr_5, rr_6, rr_9
F	rr_2
G	rr_5

L_{music}

rr_1	d,f
rr_2	d,b,f
rr_3	b
rr_4	b,g
rr_5	c,b,e
rr_6	c

R_{book}

a	
b	rr_2, rr_3, rr_4, ~~rr_5~~
c	~~rr_5~~, ~~rr_6~~
d	rr_1, rr_2
e	~~rr_5~~
f	rr_1, rr_2
g	rr_4

L_{book}

Figure 2: example of basic RR index structures for keyword "music" and "book".

The index construction procedure based on the offline sampling is shown in Algorithm 1. For each keyword w, we build an RR index with two components (R_w, L_w). R_w is the RR sets sampled with probability $p_s(v, w)$ for the vertices in the graph. There are $\hat{\theta}_w$ RR sets for each keyword. For each vertex $v \in R_w$, we maintain an inverted list in L_w to indicate which RR sets contain v.

Note that in Line 3 of Algorithm 1, $OPT_1^{\{w\}}$ is unknown and needs to be estimated in advance. We adopt the weighted iterative estimation method in [21] to estimate $OPT_1^{\{w\}}$. After $OPT_1^{\{w\}}$ and $\hat{\theta}_w$ are determined, we construct R_w by sampling $\hat{\theta}_w$ number of RR sets followed by computing its inverse mapping L_w. Finally, we can store the RR index (R, L) in disk for query processing.

EXAMPLE 4. *Figure 2 shows the RR index built for keywords "music" and "book" for the running example in Figure 1. In this example, we estimate $\theta_{music} = 9$ and $\theta_{book} = 6$. Then, we sample 9 random RR sets for keyword "music" and 6 for "book". Based on the RR sets, we construct L_{music} and L_{book} to store the inverse mapping between vertex and RR set.*

Algorithm 2: QueryRR(KB-TIM Q)

```
1  $\theta^Q \leftarrow \min\{\frac{\theta_w}{p_w} | w \in Q.T\}$
2  $S^Q \leftarrow \emptyset, (R^Q, L^Q) \leftarrow (\emptyset, \emptyset)$
3  for $w \in Q.T$ do
4      $R^Q_w \leftarrow \theta^Q \cdot p_w$ number of RR sets from $R_w$.
5      $L^Q_w \leftarrow L_w$
6  for $i = 1$ to $Q.k$ do
7      $v_i \leftarrow$ the user that covers the most RR sets in $R^Q$.
8      $S^Q \leftarrow S^Q \cup \{v_i\}$
9      for $w \in Q.T$ do
10         for $rr \leftarrow L^Q_w[v_i]$ do
11             if $rr$ is not covered then
12                 mark $rr$ as covered
13                 if $rr \in R^Q_w$ then
14                     remove $rr$ from $R^Q_w$
15 return $S^Q$
```

4.3 Improved Estimation of θ_w

Based on Lemma 3, we know that as long as we set $\theta_w = \hat{\theta}_w$, the number of sampled RR sets is sufficient to guarantee the theoretical bound with high probability. However, $\hat{\theta}_w$ could be such a large value that it takes huge amounts of disk storage to build the index. To reduce the index size, we propose a compact estimation of θ_w based on the observation that $\frac{\ln|V|+\ln\binom{|V|}{K}+\ln 2}{\ln|V|+\ln\binom{|V|}{Q.k}+\ln 2}$ can be approximated to $\frac{K}{Q.k}$ when $|V|$ is significantly larger than K and $Q.k$.

LEMMA 4. *For any keyword w, if we choose*

$$\theta_w = (8+2\varepsilon)(\sum_{v\in V} tf_{w,v}) \cdot \frac{\ln|V| + \ln\binom{|V|}{K} + \ln 2}{OPT^{\{w\}}_K \cdot \varepsilon^2} \quad (10)$$

we have $\theta_w \geq \theta \cdot p_w$.

PROOF. As $OPT^{\{w\}}_{Q.k} \leq OPT^{Q.T}_{Q.k}/idf_w$ according to Lemma 3, $\theta \cdot p_w \leq (8+2\varepsilon)(\sum_{v\in V} tf_{w,v})\frac{\ln|V|+\ln\binom{|V|}{Q.k}+\ln 2}{OPT^{\{w\}}_{Q.k}\cdot\varepsilon^2}$.

To prove $\theta_w \geq \theta \cdot p_w$, we need to justify: $\frac{\ln|V|+\ln\binom{|V|}{K}+\ln 2}{OPT^{\{w\}}_K\cdot\varepsilon^2} \geq \frac{\ln|V|+\ln\binom{|V|}{Q.k}+\ln 2}{OPT^{\{w\}}_{Q.k}\cdot\varepsilon^2}$. Let $r = \frac{\ln|V|+\ln\binom{|V|}{K}+\ln 2}{\ln|V|+\ln\binom{|V|}{Q.k}+\ln 2} \cdot \frac{OPT^{\{w\}}_{Q.k}}{OPT^{\{w\}}_K}$. Since $\frac{\ln|V|+\ln\binom{|V|}{K}+\ln 2}{\ln|V|+\ln\binom{|V|}{Q.k}+\ln 2}$ can be approximated to $\frac{K}{Q.k}$, $r = \frac{K}{Q.k} \cdot \frac{OPT^{\{w\}}_{Q.k}}{OPT^{\{w\}}_K}$. Let S^K be the seed set to achieve $OPT^{\{w\}}_K$ with $|S^K| = K$, according to the submodular property of the influence spread function [15], there exists a set $S \subset S^K$ with $|S| = Q.k$ s.t. $\frac{\mathbb{E}[I^{\{w\}}I(S)]}{Q.k} \geq \frac{OPT^{\{w\}}_K}{K}$. According to the definition of $OPT^{\{w\}}$, $\frac{OPT^{\{w\}}}{Q.k} \geq \frac{\mathbb{E}[I^{\{w\}}I(S)]}{Q.k} \geq \frac{OPT^{\{w\}}_K}{K}$. This means $r \geq 1$. Lastly, we can conclude that $\theta_w \geq \theta \cdot p_w$ whenever θ_w satisfies Eqn. 10. □

To utilized the improved θ_w for constructing the index, we simply replace $\hat{\theta}_w$ with θ_w in Line 3 of Algorithm 1. $OPT^{\{w\}}_K$ will be estimated similarly as how we estimated $OPT^{\{w\}}_1$.

4.4 KB-TIM Query Processing

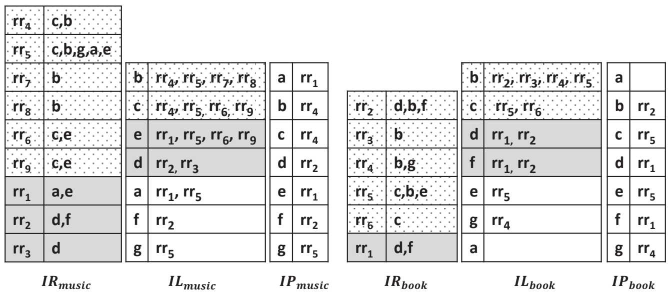
The algorithm for KB-TIM query processing based on RR index is shown in Algorithm 2. To process a query, we need to retrieve θ^Q number of RR sets from the RR index from all the query keywords. To determine θ^Q, we need to ensure the conditions in Lemma 2 so that the algorithm has the result guaranteed as in Theorem 2. Note that we cannot set θ^Q to θ via Eqn. 10 because the optimal influence spread $OPT^{Q.T}_{Q.k}$ is unknown. To determine a proper θ^Q, we set

$$\theta^Q = \min\{\frac{\theta_w}{p_w} \mid w \in Q.T\} \quad (11)$$

When θ^Q is determined (line 1), we retrieve $\theta^Q \cdot p_w$ number of RR sets from each query keyword w to ensure the sampled RR sets are not biased towards any query keyword according to Lemma 2 (line 4). Since $\theta^Q \cdot p_w \leq \theta_w$, we can guarantee that there are at least $\theta^Q \cdot p_w$ RR sets for each keyword w in the index. Finally, the result seed set S^Q is identified by running a greedy algorithm for the maximum coverage problem [22] on (R^Q, L^Q) (lines 6 - 14).

EXAMPLE 5. *Suppose $Q.T = \{$"music", "book"$\}$ and $k = 2$. If the ratio of RR sets for keyword "music" to "book" is 9 : 4, we can determine θ^Q as 13 because* $\min\{\frac{13\cdot\theta_{music}}{9}, \frac{13\cdot\theta_{book}}{4}\} = \min\{\frac{13\cdot 9}{9}, \frac{13\cdot 6}{4}\} = 13$. *This means we need 9 RR sets from "music" and 4 from "book". Thus, we load rr_1-rr_9 in R_{music} and rr_1-rr_4 in R_{book} into memory. The whole index of L_{music} and L_{book} are also loaded. Then, we run the greedy maximum coverage algorithm [22] on the 13 RR sets in memory: in the first iteration, b is selected as the first seed because it appears in the most number of RR sets. In other words, b has the highest probability to influence other people. In the next step, we remove all the RR sets containing b and find that e to be the most frequent user in the remaining RR sets. Therefore, we return $\{b, e\}$ as two most influential users for query keywords $\{$"music", "book"$\}$.*

Finally, we can show that our algorithm returns a $(1 - 1/e - \varepsilon)$-approximate solution to any KB-TIM query with a probability of at least $1 - |V|^{-1}$. Due to space limit, the proof is presented in [20].



rr_4	c,b
rr_5	c,b,g,a,e
rr_7	b
rr_8	b
rr_6	c,e
rr_9	c,e
rr_1	a,e
rr_2	d,f
rr_3	d

IR_{music}

b	rr_4, rr_5, rr_7, rr_8
c	rr_4, rr_5, rr_6, rr_9
e	rr_1, rr_5, rr_6, rr_9
d	rr_2, rr_3
a	rr_1, rr_5
f	rr_2
g	rr_5

IL_{music}

a	rr_1
b	rr_4
c	rr_4
d	rr_2
e	rr_1
f	rr_2
g	rr_5

IP_{music}

rr_2	d,b,f
rr_3	b
rr_4	b,g
rr_5	c,b,e
rr_6	c
rr_1	d,f

IR_{book}

b	rr_2, rr_3, rr_4, rr_5
c	rr_5, rr_6
d	rr_1, rr_2
f	rr_1, rr_2
e	rr_5
g	rr_4
a	

IL_{book}

a	
b	rr_2
c	rr_5
d	rr_1
e	rr_5
f	rr_1
g	rr_4

IP_{book}

Figure 3: Example of incremental RR index structures for keyword "music" and "book".

5. INCREMENTAL RR INDEX (IRR)

Although the RR index significantly improves the KB-TIM query processing compared to the WRIS method, it has to load θ^Q_w RR sets in R_w and all the inverted lists in L_w for each query keyword, which still incurs high disk I/O. In addition, we observe that a large number of RR sets do not contain the seed users. It is thus a waste of disk I/O to load these RR sets in memory because they are not accessed by the query processing algorithm. These motivate us to

Algorithm 3: BuildIRR(TopicSet T, BlockSize δ)

```
1  IRRIndex $(IR, IL, IP) \leftarrow (\emptyset, \emptyset)$
2  for $w \in T$ do
3      Compute $\theta_w$ according to Lemma 4
4      $R_w \leftarrow$ generate $\theta_w$ RR sets with probability $p_s(v, w)$
5      $L_w \leftarrow$ inverse mapping of $R_w$
6      for each user $v$ s.t $L_w[v]$ exists do
7          $IP_w[v] \leftarrow$ the RR set with the smallest ID in $L_w[v]$
8      Sort rows of $L_w$ by descending order of $L_w(v)$.size
9      $p \leftarrow 1$
10     while $L_w \neq \emptyset$ do
11         $IL_w^p \leftarrow \delta$ rows $L_w$
12         $IR_w^p \leftarrow \{rr | rr \cap IL_w^p \neq \emptyset \wedge rr \notin \bigcup_{1 \leq j < p} IR_w^j\}$
13         $p \leftarrow p + 1$
14         Remove the first $\delta$ rows from $L_w$
15 Store $(IR, IL, IP)$
```

design an index that incrementally loads relevant RR sets into memory for query processing.

5.1 IRR Index Construction

We observe that if a user has a very high impact in the social network, e.g. followed by millions of other users in Twitter, he has a good chance to be frequently sampled in the RR sets of different keywords. Hence, we sort the inverted lists in L_w in decreasing order of the list length. In this way, by incrementally loading the inverted lists, the more impactful users will be loaded first.

Our IRR index consists of three components (IR_w, IL_w, IP_w), which can be derived from (R_w, L_w) in the RR index. IL_w sorts L_w in decreasing order of list length. Then, we further split IL_w into m equi-size partitions $IL_w^1, IL_w^2, \ldots, IL_w^m$. IR_w also contains m partitions $IR_w^1, IR_w^2, \ldots, IR_w^m$ generated from the partitions in IL_w.

$$IR_w^i = \begin{cases} \{rr | rr \in R_w \ \wedge \ rr \cap IL_w^i \neq \emptyset\} & \text{if } i = 1 \\ \{rr | rr \in R_w \ \wedge \ rr \cap IL_w^i \neq \emptyset \\ \qquad \wedge \ rr \notin \bigcup_{1 \leq j < i} IR_w^j\} & \text{if } i > 1 \end{cases}$$

In other words, IR_w^i picks from the remaining RR sets in R_w that contain users in partition IL_w^i. IP_w preserves the mapping between each vertex in IL_w and its first occurrence in R_w of RR index. Whenever we process a query Q and θ_w^Q RR sets need to be retrieved w.r.t $w \in Q.T$, IP_w determines whether a vertex covers at least one RR set which is among the θ_w^Q samples.

EXAMPLE 6. *Figure 3 shows the three components of the IRR index for keywords "music" and "book". We can see that the inverted lists in IL_w are now sorted by the list length, compared to the L_w in Figure 2. In this example, we set the partition size in IL_w to be 2. The first partition of IL_{music} contain users $\{b, c\}$ and all the RR sets with user b or c are organized in the first partition in IR_{music}. The second partition of IL_{music} contains $\{d, e\}$, which appear in all the remaining RR sets. Thus, all of them are put into the second partition in IR_{music}. Since all the RR sets have been processed, the following partitions in IL_{music} correspond to empty partitions in IR_{music}. The IP_w preserves the first occurrence in the original R_w sets. For example, user d first appears in rr_2 in the R_{music} in Figure 2. Hence, it has an entry in IP_{music} mapping to rr_2.*

Algorithm 4: QueryIRR(KB-TIM Q)

```
1  $\theta^Q \leftarrow \min\{\frac{\theta_w}{p_w} | w \in Q.T\}$
2  for each keyword $w \in Q.T$ do
3      $kb[w] \leftarrow \infty$    // upper bound score for $w$
4      $\theta_w^Q \leftarrow \theta^Q \cdot p_w$ // number of RR sets for $w$
5  $pq \leftarrow \emptyset$   // priority queue sorted by upper bound score
6  $S^Q \leftarrow \emptyset$ // the result set
7  while $|S^Q| < Q.k$ do
8      $(s, tp) \leftarrow pq.top()$ // $s$ is the score for user $tp$ in $pq$
9      $s' \leftarrow$ re-compute upper bound score of $tp$
10     if $s \neq s'$ then
11         $pq.pop()$
12         push $(s', tp)$ to $pq$
13         continue
14     if $tp$.status = COMPLETE $\wedge s \geq \sum_{w \in Q.T} kb[w]$ then
15         $pq.pop()$
16         $S^Q \leftarrow S^Q \cup tp$
17         for each keyword $w \in Q.T$ do
18             for each RR set $rr$ containing $tp$ in $IR_w$ do
19                 if $rr$ is not covered $\wedge$ $rr < \theta_w^Q$ then
20                     mark $rr$ as covered
21                     for each user $v \in IR_w[rr]$ do
22                         update upper bound for $v$ w.r.t $w$
23     else
24         for each keyword $w \in Q.T$ do
25             load the next partition $IR'_w$ and $IL'_w$ from the
               IRR index of $w$
26             for each user $v \in IL'_w$ do
27                 update upper bound for $v$ w.r.t $w$
28                 push $v$ with new upper bound into $pq$
29                 set $tp$.status to COMPLETE if all the partial
                   scores have been determined
30             update $kb[w]$
31 return $S^Q$
```

The construction of IRR index is presented in Algorithm 3. For each keyword w, we build the RR sets R_w and the reverse mapping L_w as the basic RR index. Then IP is computed in Lines 6-7. Lastly, we divide the users, into partitions and each of the partitions has δ users, to form IL_w^p where p is the partition ID. The matching RR sets partition IR_w^p is built against IL_w^p. The construction terminates when all the users are visited.

5.2 Incremental KB-TIM Query Processing

In our IRR index, the users in IL_w are sorted for each keyword. This motivates us to model the KB-TIM query as a top-k aggregation problem and employ an algorithm similar to NRA [8]. The NRA algorithm maintains an accumulation table for candidates with incomplete results and incrementally loads blocks of items in the sorted lists for aggregation. If a candidate contains partial scores from all the keywords, its status is set to `COMPLETE` and pushed into a heap storing top-k results. The algorithm terminates when the upper bound score for the partial or unvisited results is smaller than the k-th best score ever found.

KB-TIM query processing based on IRR index brings two new issues when employing NRA algorithm: 1) how to determine the status of a candidate user is `COMPLETE` when there are missing partial scores; 2) how to find and update all the affected users when a new seed user is confirmed and added to the result set.

For the first issue in NRA, a candidate is said to be `COMPLETE` if the partial scores of all the attributes have been accessed and aggregated. However, we only access $(\theta^Q \cdot p_w)$ RR sets for keyword w in query processing. If a user is not contained in the RR sets of a query keyword, there is a missing partial score for the keyword and the complete score can only be determined when all the RR sets have been scanned. To avoid the case, we check whether a user appears in the RR sets of a keyword before the query processing. If $IP_w[v] \geq \theta^Q \cdot p_w$, where $IP_w[v]$ indicates the first occurrence in the RR sets, we set the partial score of user v on keyword w to be 0. The status of a user becomes `COMPLETE` when the partial scores for all query keywords are set. We use $score(u)$ to denote the complete score of user u.

The second issue arises because each time a new seed user is confirmed by the greedy maximum coverage algorithm [22], we need to remove the seed user from the RR sets and update the list length for the all the affected candidate users. To save computation cost, we propose a lazy evaluation strategy. We mark the RR sets containing previous seed users as `covered` and refine the score of a candidate only when it is the top element in the priority queue.

We present our incremental KB-TIM query processing solution in Algorithm 4. Given a query, we first calculate θ^Q to determine the number of RR sets that should be loaded for each keyword. *pq* is a priority queue in which candidate users are sorted by their upper bound score and $kb[w]$ stores the maximum list length in IL_w for the unvisited user candidates. The algorithm iterates until k seed users have been found. In each iteration, we access the user *tp* with the highest upper bound score. If u has been loaded into memory, we know the accurate number of RR sets for that keyword. Otherwise, the upper bound score on that keyword is $kb[w]$. The upper bound score is the sum of the scores from all the query keywords. If the score of the user is affected by previous seed users, we further refine the score by removing those RR sets containing previous seed users. The user with the new candidate upper bound is pushed back into the priority queue. Otherwise, we check whether the user is a new seed user. If his status is `COMPLETE`, the upper bound score is an exact score. If the score is larger than the upper bound score of unseen users, we can guarantee that there is no other user with a better score than *tp*. So we insert *tp* to the result set and the RR sets covered by *tp* are marked (lines 14-22) such that the candidate users contained in any of these RR set are affected. If *tp* cannot be determined as a new seed, we load more partitions from each query keyword's IRR index. Then we update the upper bound scores for all the users in the loaded partition (lines 23-30). The algorithm terminates when $Q.k$ seed users are found. An example is included in the technical report [20] for more details.

THEOREM 3. *The impact scores of top-k seed users returned by Algorithm 4 and Algorithm 2 are the same.*

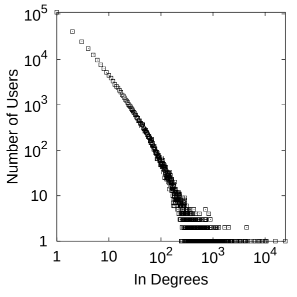
PROOF. Suppose Algorithm 4 iteratively returns k seed users $\{u_1, u_2, \ldots, u_k\}$, Algorithm 2 returns $\{v_1, v_2, \ldots, v_k\}$. If the top-$k$ impact scores are not the same, since Algorithm 2 picks the user with the maximum score in each iteration, we can find an index i such that u_i is the first user with $score(u_i) < score(v_i)$ and $score(u_j) = score(v_j)$ for $j < i$.

case 1): $v_i \neq u_i$. In Algorithm 4, u_i is selected as a new seed only when it has the maximum upper bound score in the priority queue or its score is larger than $\sum_{w \in Q.T} kb[w]$. Therefore, if v_i is in the priority queue, we have $score(v_i) \leq score(u_i)$. Otherwise, v_i has not been loaded into memory and its upper bound score is $\sum_{w \in Q.T} kb[w] \leq score(u_i)$. Both cases contradict with $score(u_i) < score(v_i)$.

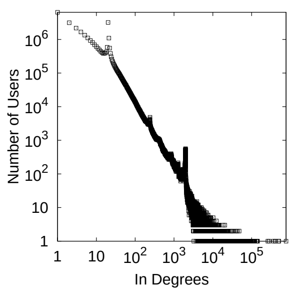
case 2): $v_i = u_i$. When u_i is selected as a seed user, we know that its score is complete, i.e., all the related RR sets have been loaded into memory. Since Algorithm 4 finds the same $i-1$ seed users in previous iterations, after removing all these $i-1$ seed users from the RR sets of u_i, we know that its inverted list length is the same with that of v_i in Algorithm 2. Thus, $score(v_i) = score(u_i)$, which contradicts with $score(u_i) < score(v_i)$. $\Box$

Table 2: All parameter settings used in the experiments. The default values are highlighted.

Datasets	Twitter dataset	News dataset
#Users	10M,20M,30M,**40M**	0.2,0.6M,1M,**1.4M**
#Edges	0.7B,1.1B,1.2B,**1.3B**	1.0,1.9M,2.6M,**3.1M**
AveDegree	76.4,56.8,46.1,**38.9**	5.2,3.1,2.6,**2.2**
#QWords	1,2,...,**5**,6	
k	10,15,...,**30**,...,50	



(a) News



(b) Twitter

Figure 4: In-degrees distributions for both datasets

6. EXPERIMENTAL STUDY

In this section, we study the performance of KB-TIM query processing on two real datasets. We use **WRIS** (in Section 3.2) as a baseline solution because it uses online sampling and can be considered as a variant of the state-of-the-art RIS methods [21, 2]. We compare the methods based on offline sampling (**RR** and **IRR** index in Sections 4 and 5 respectively) with **WRIS** and evaluate the efficiency by average running time and effectiveness by expected influence. We did not evaluate the existing topic-aware IM solutions since they cannot handle the social networks used in this paper due to the incapability to scale for large graph sizes and number of topics (detailed discussion in Section 7). All the methods are implemented with C++ and run on a CentOS server (Intel i7-3820 3.6GHz CPU with 8 cores and 60GB RAM).

6.1 Experimental Setup

Datasets. We use two real world datasets, *Twitter* and *News*, from SNAP [5] for performance evaluation. The *Twitter* dataset contains 41.6 million users with 476 million tweets and the news dataset contains 1.42 million media extracted from a collection of 96 million online news corpus. For the

[5]http://snap.stanford.edu/

Table 3: Disk space and running time of using $\hat{\theta}_w$ and θ_w for constructing indices for news datasets

Data	Disk Size (GB)				Time (Secs)			
	RR		**IRR**		**RR**		**IRR**	
	$\hat{\theta}_w$	θ_w	$\hat{\theta}_w$	θ_w	$\hat{\theta}_w$	θ_w	$\hat{\theta}_w$	θ_w
n0.2M	37	4.1	37	4.1	662	67.3	700	69.9
n0.6M	47	5.7	47	5.8	982	121	994	122
n1.0M	54	6.7	54	6.9	1238	157	1270	164
n1.4M	62	7.6	63	7.8	1487	167	1497	188

news dataset, the vertex in the graph denotes an online media whereas the edge means that there is a link from one online media to another. We extract 200 topics from each dataset and user profile is represented by a term vector in the topic space. To test the scalability with increasing number of users, we sampled 10M, 20M, 30M, 40M users for the *Twitter* dataset and 0.2M, 0.6M, 1M, 1.4M for the news dataset. We use t10M, t20M, t30M, t10M to denote the respective *Twitter* datasets and n0.2M,n0.6M,n1.0M,n1.6M for new datasets. The statistics of the social networks are shown in Table 2. We also plot the frequency distribution of incoming degrees in Figure 4, which shows *Twitter* is a much denser graph than news social network and many nodes are followed by a large number of users.

Queries. We use real keyword queries from AOL search engine[6]. Given the 200 topics defined in advance, we first filter the keyword queries and retain those only containing our topic keywords. To evaluate the performance in terms of increasing number of topics (or keywords) in a query, we vary the number of query keywords from 1 to 6 and extract 100 queries for each length.

Parameters. In our algorithms, two parameters, ε and K, need to be determined to evaluate θ in Eqn. 6, $\hat{\theta}_w$ in Eqn. 8 and θ_w in Eqn. 10. ε is set to 0.1 for all experiments as it is the most accurate setting adopted in [21] (no experiment is done in [2]). K is set to 100 since the largest $Q.k$ in our experiment is 50. For **IRR**, the partition size δ is set to 100 for all experiments. In the following experiments, we evaluate the scalability w.r.t. increasing seed users $Q.k$, query keywords $|Q.T|$ and social network size $|V|$ as shown in Table 2.

6.2 Index Sizes and Construction Time

For **RR** and **IRR**, both methods require offline sampling of RR sets to build respective indices. All indices are constructed by running 8 threads in parallel. We first study the difference between using $\hat{\theta}_w$(Eqn. 8) and θ_w(Eqn. 10) for index construction. The disk space and running time of using $\hat{\theta}_w$ and θ_w for index construction of news datasets are presented in Table 3. It is obvious that using $\hat{\theta}_w$ is not scalable against large graphs. Besides, in subsequent experiments, we will see that indices constructed by θ_w have equivalent approximation power as compared to that by using $\hat{\theta}_w$. Therefore we will not present results for indices built by $\hat{\theta}_w$ for *Twitter* datasets.

Since **RR** and **IRR** use inverted lists, we can apply FastPFOR[7] compression (adopted in Apache Lucene 4.6.x) to reduce disk storage. We report disk space and running time for uncompressed and compressed indices in Table 4. For

[6]http://www.gregsadetsky.com/aol-data/
[7]https://github.com/lemire/FastPFOR

Table 4: Disk space and running time for constructing indices for various datasets with θ_w

Data	Disk Size (GB)				Time (Secs)			
	uncompress		compress		uncompress		compress	
	RR	**IRR**	**RR**	**IRR**	**RR**	**IRR**	**RR**	**IRR**
n0.2M	4.1	4.1	2.0	2.0	67.3	69.9	70.1	73.8
n0.6M	5.7	5.8	3.0	3.1	121	122	113	119
n1.0M	6.7	6.9	3.7	3.9	157	164	148	157
n1.4M	7.6	7.8	4.3	4.6	167	188	164	167
t10M	88	94	52	58	5.9h	6.0h	5.8h	6.0h
t20M	84	93	56	65	4.3h	4.5h	4.4h	4.4h
t30M	77	87	54	64	3.7h	3.8h	3.7h	3.7h
t40M	55	63	41	50	2.4h	2.5h	2.4h	2.5h

Table 5: Sum of θ_w and mean RR set size for increasing graph size

(News) $\lvert V\rvert$	Sum of θ_w	Mean of RR size	Twitter $\lvert V\rvert$	Sum θ_w	Mean RR size
0.2M	385M	2.7	10M	218M	94.9
0.6M	640M	2.3	20M	247M	71.9
1M	798M	2.1	30M	276M	55.0
1.4M	926M	2.0	40M	374M	26.7

uncompressed indices, we can see that the index size in the news dataset grows with the graph size. However, to our surprise, such pattern does not apply to Twitter datasets. The reason is that the index size depends on two factors: the number of RR sets, i.e. θ_w in Lemma 4, and the average size of a RR set. On one hand, θ_w increases for large graphs since θ_w depends on $|V|$. We report the sum of θ_w among all keywords and average RR set size for both datasets in Table 5. On the other hand, as a RR set is constructed by first randomly picking a starting vertex and then performing a random breath first search on the social graph, the size of a RR set will be larger if the graph is denser. As shown in Table 2, the average degree of both the news and the Twitter dataset decrease as the graph becomes larger. This lead to a decrease in the average RR set size for both datasets. Hence, θ_w and average RR set size are two conflicting factors as $|V|$ varies. In the news dataset, θ_w takes the major role in determining the index size while in the Twitter dataset, the average RR set size is more critical. For compressed indices, there are approximately 50% and 40% space reductions for news and Twitter datasets respectively. Besides the construction time for the compressed indices does not increase significantly compared to that of the uncompressed indices. Thus, in the following experiments, we will adopt the compressed scheme for both **RR** and **IRR** indices.

We also obverse that Twitter consumes much more disk space than the news dataset. This is because Twitter is a much larger and denser social network than news. For each user in Twitter, it can be affected by a large number of users. The construction time of the index grows linearly with the index size. It requires hours to build the index for Twitter datasets, even when we used 8 threads in parallel. This is because we need to build the index for 200 keywords and each keyword requires hundreds of thousand of random walks to generate RR sets.

6.3 Vary the Seed Set Size

We first examine the performance with increasing seed users $Q.k$. The running time and number of RR sets accessed are shown in Figure 5. It is obvious that the methods based on offline sampling are significantly faster than

Table 6: Number of I/O for IRR when varying $Q.k$

$Q.k$	10	15	20	25	30	35	40	45	50
News	6.10	7.34	8.75	10.2	14.0	19.3	33.6	78.1	170
Twitter	8.00	14.5	23.0	34.0	40.0	51.0	58.5	69.0	81.0

Table 7: Influence spread when varying $Q.k$.

	News dataset				Twitter dataset		
$Q.k$	**WRIS**	**RR**($\hat{\theta}_w$)	**RR**	**IRR**	**WRIS**	**RR**	**IRR**
10	289.6	289.4	289.4	289.4	8673.9	8672.6	8672.6
15	356.5	356.3	356.4	356.4	10666	10666	10666
20	408.9	409.0	409.0	409.0	12088	12092	12092
25	448.7	448.6	448.6	448.6	13100	13099	13099
30	480.3	480.3	480.2	480.2	13929	13925	13925
35	506.7	506.8	506.7	506.7	14618	14616	14616
40	528.6	528.8	528.8	528.8	15209	15209	15209
45	548.6	548.0	548.1	548.1	15731	15735	15735
50	564.9	564.9	564.9	564.9	16211	16214	16214

the online sampling method. In the Twitter dataset, the average response time to a KB-TIM query using **RR** and **IRR** indices are 160x and 434x times smaller than **WRIS** respectively. **IRR** runs faster than **RR** for two reasons. First, **RR** method needs to load θ_w RR sets and this number is invariant to $Q.k$. **IRR** method incrementally loads partitions for top-k aggregation. The number of RR sets loaded by the two algorithms is plotted in Figure 5. Second, in **RR** method, after a seed is found, we need to eliminate its impact on the remaining candidate users. In other words, we scan the inverted lists L_w and update the length by removing those RR sets containing the seed user. The operation is expensive. In **IRR**, our proposed lazy update mechanism only requires updating the score of the user at the top of the priority queue in each iteration.

As $Q.k$ increases, it takes a slightly longer time for **RR** and **IRR** methods to answer a query. This is because both **RR** and **IRR** require more iterations to find the seed users, causing more CPU cost and disk I/O. But the performance of **WRIS** is slightly faster as $Q.k$ increases. The reason is that the performance of **WRIS** is mainly dependent on the number of RR sets generated, i.e. θ in Theorem 2, which is inversely proportional to the optimal spread. Due to the monotonicity of the IM problem, the optimal spread becomes larger when $Q.k$ increases, resulting in smaller number of RR sets sampled.

We also note that the performance of **IRR** degrades to be close to **RR** in the news dataset. This is because the I/O efficiency of **RR** is better than **IRR**. **RR** method incurs a sequential disk I/Os for each query keyword (default number of query keywords is 5 which lead to 5 I/Os for each query). As shown in Table 6, **IRR** method relies on incremental loading of partitions into memory, which causes increasing disk I/Os for larger $Q.k$. However, in the Twitter dataset, although **RR** still has the advantages in I/Os, the number of RR sets loaded for **IRR** is significantly smaller than that of **RR**, which explains why **IRR** does not degrade to **RR**.

In addition to the result that **RR** and **IRR** are much faster than **WRIS**, the expected influence of the top-$Q.k$ seed users generated by them is also no worse than **WRIS**. We report the expected influence of the seed users returned by all the methods in Table 7. To justify our improved estimation of θ_w in Sec. 4.3, we also report the news dataset result for running **RR** method on indices constructed by using $\hat{\theta}_w$. As shown in Table 7, there are almost no difference between all the methods. The expected influence spread for all three methods will not be presented in the rest of the experimental study as the results show similar patterns.

6.4 Vary the Number of Query Keywords

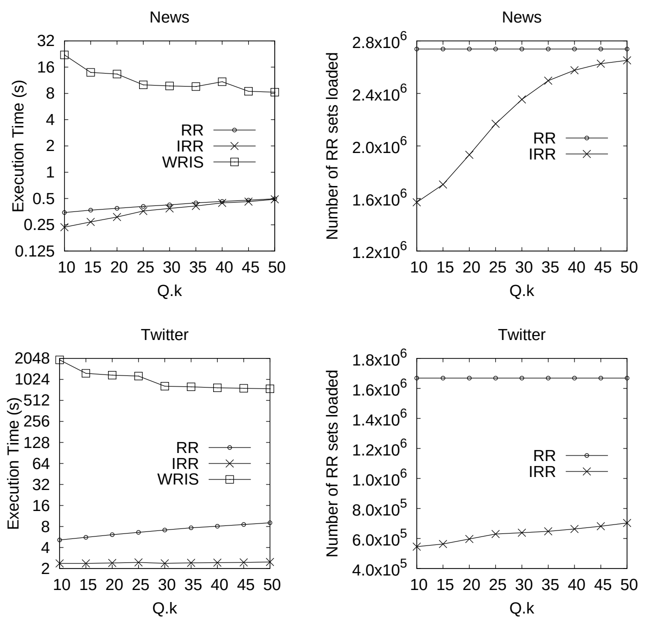


Figure 5: Varying the seed set size: Q.k

When we increase the number of query keywords $|Q.T|$ from 1 to 6, as shown in Figure 6, the results demonstrate similar patterns: **RR** and **IRR** are at least two orders of magnitude better than **WRIS** in a social network with billions of edges. The running time of **IRR** outperforms **RR** in the Twitter dataset but degrades to be close to **RR** in the news dataset. This is because in Twitter, there are a large number of users with dominating number of followers. After we sort the users based on their overall influence, we only need to access a small portion of RR sets. In the news dataset, the number of accessed RR sets for the two methods grows linearly and **IRR** degrades because it requires more random disk I/O.

6.5 Vary the Graph Size

We also vary the graph size, i.e. $|V|$, to test the scalability of our proposed solutions. The results are shown in Figure 7. **RR** and **IRR** clearly outperform **WRIS** by great margins in all scenarios. In the news datasets, **RR** can outperform **IRR** in certain cases because its I/O is more efficient when examining similar number of RR sets. Nevertheless, **IRR** has dominating performance against **RR** when the size of Twitter datasets increases. It shows that **IRR** is more effective for larger graphs without compromising its performance superiority.

6.6 Effectiveness of Propagation Model

In the last experiment, we discuss the effectiveness of the propagation model applied for KB-TIM query processing using the two real datasets. In this section, we present results for both independent cascade (IC) model [21, 5, 6, 19] and linear threshold (LT) model [21, 19, 15]. For IC model, the propagation probability set to $p(e) = \frac{1}{N_v}$ which is widely adopted in [21, 5, 19]. For LT model, following existing work [21, 15, 7], we assign a random value in $[0, 1]$ to each user's incoming neighbours and normalize the values so that the sum of all neighbours' influence probabilities equals 1. In Table 8, we illustrate the top-8 most influential users for

Table 8: Example KB-TIM query results

Method	Keyword	News dataset			
WRIS(IC)	“software”	kb.vmware.com	en.wikipedia.org	davidstef9.wordpress.com	codeplex.com
		suse.ehelp.pl	virtualgeek.typepad.com	softwarelivre.net	linux.pl
WRIS(LT)	“software”	kb.vmware.com	en.wikipedia.org	orums.asp.net	communities.vmware.com
		suse.ehelp.pl	virtualgeek.typepad.com	ntwizard.spaces.live.com	linux.pl
WRIS(IC)	“journal”	www.biblegateway.com	bookology.wordpress.com	hugh.journalspace.com	journals.aol.com
		journal.peishan.org	earticle.wordpress.com	signefavor17.spaces.live.com	jazzitalia.net
WRIS(LT)	“journal”	journals.aol.com	bookology.wordpress.com	bizjournals.com	www.biblegateway.com
		journal.peishan.org	journaldugeek.com	bookalytics.com	jazzitalia.net
RIS	N.A	en.wikipedia.org	blogger.com	youtube.com	myweb.yahoo.com
		wordpress.com	wrzuta.pl	match.seesaa.jp	2.bp.blogspot.com
Method	**Keyword**	**Twitter dataset**			
WRIS(IC)	“software”	BarackObama	britneyspears	cnnbrk	aplusk
		THE_REAL_SHAQ	kevinrose	jimmyfallon	twitter
WRIS(LT)	“software”	biz	cnnbrk	kevinrose	twitter
		BarackObama	mashable	jimmyfallon	britneyspears
WRIS(IC)	“journal”	kevinrose	twitter	BarackObama	TheEllenShow
		RyanSeacrest	britneyspears	THE_REAL_SHAQ	taylorswift13
WRIS(LT)	“journal”	ev	cnnbrk	twitter	BarackObama
		TheOnion	NotTinaFey	jimmyfallon	britneyspears
RIS	N.A	ev	cnnbrk	kevinrose	BarackObama
		mashable	jimmyfallon	TheEllenShow	britneyspears

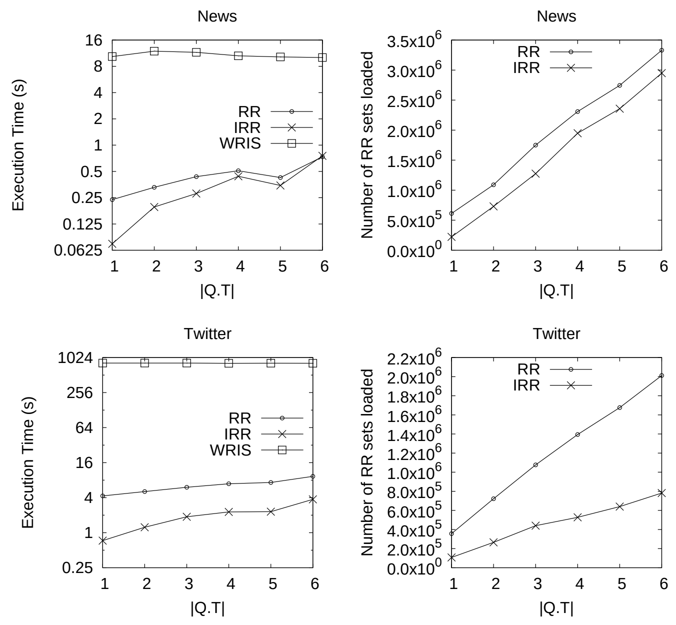


Figure 6: Varying query keywords length: $|Q.T|$

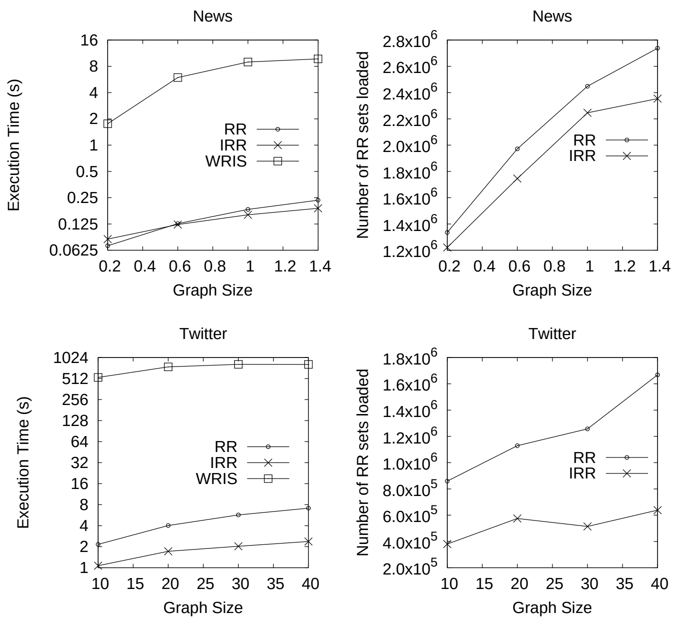


Figure 7: Varying the graph size: $|V|$

keywords “software” and “journal” in both news dataset and Twitter dataset for both models. We can see that it is more effective for the news dataset than the Twitter dataset. In the news dataset, we can see that many websites in the top-10 results are highly relevant to the keyword “software” and “journal”. It shows that 1) our proposed KB-TIM query can be applied for targeted advertising and disseminate the advertisement in the more relevant communities or clusters in the social network; 2) The propagation model used in this paper is effective for topic-aware advertising in the news dataset. However, in the Twitter dataset, the results for different topics are similar because the reported users have a huge number of followers with diverse background. In other words, they are very influential in different topics. We also note that RIS method always returns the same results for different query keywords. There is no clue between its top-10 seed users and query keywords.

Note that how to determine a proper propagation model is beyond the scope of this paper. In this paper, we focus on the efficiency issue and our method can be easily applied to other propagation models like the general triggering model [15, 19]. This is because our proposed **WRIS** is an extension of RIS and RIS has been shown to incorporate all propagation models in [21]. The only difference is that RIS uniformly samples vertices to create RR sets whereas **WRIS** conducts a weighted sampling approach. Since influence propagation models is independent of the vertex sampling methods, **WRIS** can directly adopt other propagation models supported by RIS.

7. RELATED WORK

IM problem is a NP-Hard problem that has been extensively studied. Since Kempe et al. [15] proposed the first simple greedy algorithm with an approximation ratio of $(1 - 1/e - \varepsilon)$, there has been a large body of research work devoted to improving the efficiency while keeping the theoretical bound [17, 10, 5, 7, 6, 16, 2, 14, 21]. Although CELF [17] and its variant CELF++ [10] have significantly improved the running time, the methods were only examined in small graphs with thousands of vertices. RIS [2] is the first method scalable enough to handle graphs with millions of vertices. It uses random sampling and can handle various propagation models that have been proposed. The method was further improved in [21] in terms of sampling efficiency. However, RIS still requires nearly one hour to process a graph with millions of vertices.

Another branch of work on IM improved the efficiency by discarding the theoretical bound. In other words, the expected influence returned does not have any approximation ratio to the optimal results. Chen et al. [6] used vertex degree as a quick selection criterion. In [5, 7, 16, 14], the propagation behaviour is simplified by removing social paths that have low propagation probabilities. Although these heuristic solutions are more efficient, none of them have been examined in large social networks. The state-of-the-art solutions (PMIA [5] and IRIE [14]) require more than 10 minutes to run a graph with less than 1 million edges.

The above methods cannot be directly applied in online advertisements because the same seed users are returned for different advertisements. To solve the issue, topic-aware IM problem was proposed [1, 13, 4, 18, 3]. In [1], the influence probabilities that a user influences his neighbors are different for different topics. Subsequently, Chen et al. [4] proposed a solution for the topic-ware IM by extending PMIA. As mentioned, PMIA does not have any theoretical bound and is not applicable to very large graphs. In [13], Inflex was proposed to achieve real-time performance. Inflex first precomputes a number of top-k seed sets offline and the online query is processed by finding nearest neighbours among the pre-computed seed sets w.r.t the query topics. Thus, Inflex does not have a theoretical bound due to the nearest neighbour approximation. Very recently, Shuo et al. [3] proposed another heuristic solution to improve the efficiency of Inflex, but still failed to provide a theoretical bound. In addition, existing works are not scalable to large number of topics as they take prohibitively long time to train the propagation probabilities and huge storage spaces for different topics. No study was reported in [1, 13, 4] on handling graphs with more than 1 million vertices and [3] is only capable of handling 10 topics for a graph with 4 million users. Obviously, these solutions are infeasible for real-world social IM applications.

Compared with existing topic-aware IM solutions, our KB-TIM query takes into account the influence on the targeted users relevant to the advertisement instead of assigning different influence probabilities between directly connected users for different advertisements. Our solution not only achieves an approximation ratio of $(1 - 1/e - \varepsilon)$, but also is scalable to retrieve the seed users with a few seconds in a graph with billions of edges and hundreds of topics.

8. CONCLUSION

In this work, we studied the Keyword-Based Targeted Influence Maximization (KB-TIM) query on social networks. We first proposed an online sampling RIS method **WRIS** that returns a solution with a $(1 - 1/e - \varepsilon)$ approximation ratio. Then a disk based index **RR** is developed so that the query processing can be done in real time. Subsequently, an incremental index and query processing technique, i.e. **IRR**, is presented to further boost the performance of **RR** method. Extensive experiments on real world social network data have verified the theoretical findings and efficiency of our solutions.

9. ACKNOWLEDGEMENT

We would like to thank Prof. Xiaokui Xiao (NTU) for his insightful comments. This work was supported by NUS-Tsinghua Extreme Search project under the grant number: R-252-300-001-490.

10. REFERENCES

[1] N. Barbieri, F. Bonchi, and G. Manco. Topic-aware social influence propagation models. In *ICDM*, pages 81–90, 2012.

[2] C. Borgs, M. Brautbar, J. T. Chayes, and B. Lucier. Influence maximization in social networks: Towards an optimal algorithmic solution. *CoRR*, abs/1212.0884, 2012.

[3] S. Chen, J. Fan, G. Li, J. Feng, K.-l. Tan, and J. Tang. Online topic-aware influence maximization. *Proc. VLDB Endow.*, 8(6):666–677, 2015.

[4] W. Chen, T. Lin, and C. Yang. Efficient topic-aware influence maximization using preprocessing. *CoRR*, abs/1403.0057, 2014.

[5] W. Chen, C. Wang, and Y. Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *KDD*, pages 1029–1038, 2010.

[6] W. Chen, Y. Wang, and S. Yang. Efficient influence maximization in social networks. In *KDD*, pages 199–208, 2009.

[7] W. Chen, Y. Yuan, and L. Zhang. Scalable influence maximization in social networks under the linear threshold model. In *ICDM*, pages 88–97, 2010.

[8] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. *J. Comput. Syst. Sci.*, 66(4):614–656, 2003.

[9] J. Goldenberg, B. Libai, and Muller. Using complex systems analysis to advance marketing theory development. *Academy of Marketing Science Review*, 2001.

[10] A. Goyal, W. Lu, and L. V. Lakshmanan. Celf++: Optimizing the greedy algorithm for influence maximization in social networks. In *WWW*, pages 47–48, 2011.

[11] M. Granovetter. Threshold models of collective behavior. *The American Journal of Sociology*, 83(6):1420–1443, 1978.

[12] L. Hong and B. D. Davison. Empirical study of topic modeling in twitter. In *SOMA*, pages 80–88, 2010.

[13] Çigdem Aslay, N. Barbieri, F. Bonchi, and R. A. Baeza-Yates. Online topic-aware influence maximization queries. In *EDBT*, pages 295–306, 2014.

[14] K. Jung, W. Heo, and W. Chen. IRIE: scalable and robust influence maximization in social networks. In *ICDM*, pages 918–923, 2012.

[15] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *KDD*, pages 137–146, 2003.

[16] M. Kimura and K. Saito. Tractable models for information diffusion in social networks. In *PKDD*, pages 259–271, 2006.

[17] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. M. VanBriesen, and N. S. Glance. Cost-effective outbreak detection in networks. In *KDD*, pages 420–429, 2007.

[18] F.-H. Li, C.-T. Li, and M.-K. Shan. Labeled influence maximization in social networks for target marketing. In *SocialCom/PASSAT*, pages 560–563, 2011.

[19] G. Li, S. Chen, J. Feng, K. Tan, and W. Li. Efficient location-aware influence maximization. In *SIGMOD*, pages 87–98, 2014.

[20] Y. Li, D. Zhang, and K.-L. Tan. Real-time targeted influence maximization for online advertisements. Technical report, 2015. `http://www.comp.nus.edu.sg/~a0047194/`.

[21] Y. Tang, X. Xiao, and Y. Shi. Influence maximization: Near-optimal time complexity meets practical efficiency. In *SIGMOD*, pages 75–86, 2014.

[22] V. V. Vazirani. *Approximation algorithms.* Springer, 2001.

[23] J. Zobel and A. Moffat. Inverted files for text search engines. *ACM Comput. Surv.*, 38(2), 2006.