REAL-TIME ADVERTISING ON SOCIAL NETWORKS

YUCHEN LI
B.Sci.(Hons.) & B.Comp.(Hons.), NUS

A THESIS SUBMITTED
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
NUS GRADUATE SCHOOL FOR INTEGRATIVE SCIENCES AND ENGINEERING
NATIONAL UNIVERSITY OF SINGAPORE

2016

SUPERVISOR:
PROFESSOR KIAN-LEE TAN

EXAMINERS:
PROFESSOR MONG LI LEE
ASSOCIATE PROFESSOR STÉPHANE BRESSAN
DR. WENJIN ZHANG, UNIVERSITY OF NEW SOUTH WALES
Declaration

I hereby declare that the thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

Yuchen Li
18 September 2016
Acknowledgements

This thesis would not have been completed without the support of many people. I would like to dedicate this section to express my gratitude to all of them.

First and foremost, I would like to express my deepest gratitude and appreciation to my adviser Prof. Tan Kian-Lee. He welcomed me on board when I was still a fresh and ignorant undergraduate student. During the entire period of my doctoral study, Prof. Tan has provided me with his excellent patience and support. Moreover, he has arranged exceptional research environment and helped me develop independent research skills which would be of great benefit for my entire life. I would also like to thank my thesis committee, Prof. Li Mong-Lee and Prof. Roger Zimmermann for their valuable reviews, comments and suggestions to improve the quality of my research.

I would like to express my sincere gratitude to Prof. Bao Zhifeng and Prof. Zhang Dongxiang. Prof. Bao Zhifeng collaborated with me in my first work and helped me with my first publication. This is a good starting point and I am really thankful for working with him. Prof. Zhang Dongxiang, who has collaborated with me in all my thesis works, deserves my special appreciation. His insightful comments and valuable discussions have inspired me in finding research problems and develop novel solutions. I also want to thank my co-authors, Prof. Li Guoliang, Prof. Fan Ju and my friend Lan Ziquan. They spent a lot of their precious time and contributed so much to improve our papers. It was a wonderful experience to work with them and learn from them.

Many friends in Singapore have also helped me a lot during my academic pursuit. First, my senior lab mates who provided helpful advices and support when I first came to the lab. They are Prof. Wang Zhengkui, Prof. Wu Sai, Guo Long, Fan Qi, Wang Guoping, Zheng Yuxin, Zhou Jingbo and Zeng Zhong. I also would like to express my
appreciation to my other lab colleagues and basketball friends as we shared wonderful experiences together.

Furthermore, I would like to thank NUS Graduate School of Integrative of Science and Engineering for providing me the scholarship and learning opportunities during my PhD study.

I would also like to thank my parents Li Wending and Li Xiaohuai. They have been always unconditionally supportive and encouraging for me during my entire life. I would not be who I am now without them. Last but not least, the most special thanks are reserved for my dearest Liu Zhilin for her accompany and love which have sustained me through the otherwise grueling period of my PhD study.
Contents

Acknowledgements ......................................................... i
Summary ................................................................. vii
List of Tables ............................................................ ix
List of Figures ............................................................. xi

1 Introduction ......................................................... 1

1.1 Motivations .......................................................... 1

1.2 Opportunities and Challenges ...................................... 3

1.2.1 Context-Aware Ad Recommendation for DTA ................. 4

1.2.2 Targeted Influence Maximization for CSA .................... 5

1.2.3 Personalized Influential Tags Exploration for SIA ............ 6

1.3 Contributions .......................................................... 7

1.3.1 Context-Aware Ad Recommendation for DTA ................. 8

1.3.2 Targeted Influence Maximization for CSA .................... 8

1.3.3 Personalized Social Influential Tags Mining for SIA .......... 9

1.4 Organization ............................................................ 11

2 Literature Review ................................................. 13

2.1 Top-k Aggregation Query ........................................... 14

2.1.1 Allowing Random Access ...................................... 15
2.1.2 No Random Access ............................................. 17
2.1.3 Immutable Regions ........................................... 19

2.2 Social Search and Recommendation .................................................. 20

2.3 Influence Maximization ................................................................. 22
  2.3.1 Propagation Models ......................................................... 23
  2.3.2 Traditional Influence Maximization ........................................ 25
  2.3.3 Topic-aware Influence Maximization ...................................... 27
  2.3.4 Other Influence Maximization Problems .................................. 28

3 Context-Aware Advertisement Recommendation for DTA .................. 29
  3.1 Problem Statement ............................................................... 31
  3.2 Online Retrieval Algorithm ..................................................... 35
  3.3 Safe Region Algorithm ........................................................... 37
    3.3.1 Safe Region Construction ................................................ 37
    3.3.2 Computing MinS and MaxS ............................................ 39
    3.3.3 Safe Region Based Query Processing .................................. 45
    3.3.4 Optimizations ............................................................... 46
  3.4 Hybrid Algorithm ................................................................. 48
    3.4.1 Variance of Topic Distributions ....................................... 49
    3.4.2 Hybrid Retrieval Strategy ............................................. 53
  3.5 Experimental Study .............................................................. 54
    3.5.1 Tuning $\rho$ ................................................................. 56
    3.5.2 Varying $\alpha$ ............................................................. 57
    3.5.3 Increasing $k$ ............................................................... 59
    3.5.4 Vary Read/Write Ratio .................................................. 60
    3.5.5 Scalability ................................................................. 63
4 Targeted Influence Maximization for CSA

4.1 Preliminary

4.1.1 Influence Maximization (IM)

4.1.2 Reverse Influence Set (RIS)

4.2 Targeted Influence Maximization

4.2.1 Problem Definition

4.2.2 Weighted RIS Sampling

4.3 Disk-based RR Index for KB-TIM Query Processing

4.3.1 Discriminative WRIS Sampling

4.3.2 RR Index Construction

4.3.3 Improved Estimation of $\theta_w$

4.3.4 KB-TIM Query Processing

4.4 Incremental RR Index (IRR)

4.4.1 IRR Index Construction

4.4.2 Incremental KB-TIM Query Processing

4.5 Experimental Study

4.5.1 Experimental Setup

4.5.2 Index Sizes and Construction Time

4.5.3 Vary the Seed Set Size

4.5.4 Vary the Number of Query Keywords

4.5.5 Vary the Graph Size

4.5.6 Effectiveness of Propagation Model

4.6 Summary
Summary

The availability of social networking platforms (SNPs) has led to a rapid growth of social media marketing (SMM) deployments. The advantages of SMM over traditional marketing strategies are two folds. First, user preferences can be quickly extracted from user profiles and activities on SNPs and relevant advertisements are then effectively posted to targeted users. Second, SNPs provide extremely convenient ways to share information between friends or followers. Thus viral marketing approaches would be more effective on SNPs as it facilitates ease of sharing.

Although many efforts have been made to address utilizing each of the aforementioned benefits of SMM to assist effective online advertisements, few works have been conducted to combine the merits from both perspectives to enable more intelligent SMM approaches. Thus, this thesis is devoted to design innovative solutions which leverage both user profiles and sharing behaviors for more advanced SMM strategies. In particular, we propose novel problems with efficient algorithms for each of the three mainstream channels for SMM:

First, we examine the direct targeted advertising (DTA) channel where the advertisements are directly posted to match users’ preferences. However, the slowly evolving personal interests result in repetitious advertisement recommendation. To resolve this issue, we propose a context-aware advertisement recommendation framework that takes into account the relatively static personal interests as well as the dynamic news feed posted or shared by friends to drive growth in the advertisement click-through rate. Subsequently, we design three techniques to efficiently process recommendation requests issued by dynamically changing news feed content to meet the real time requirement.

Second, the celebrity social advertising (CSA) channel is studied. The advertisers find celebrities through this channel and ask them for their endorsements of the promoted product on SNPs. Earlier researches have been focusing on finding a small set of users who
generate the most influence on SNPs without considering the personal preferences of SNP users. Although there have been some progress on topic-aware influential analysis, there is yet no efficient solution that can handle large social networks. Given these motivations, we propose a new keyword-based targeted IM query which finds a set of users that maximize the expected influence over users who are relevant to a given advertisement. As millions of advertisers conduct marketing in real-world SNPs, we further design a series of efficient algorithms to find the targeted influential set to achieve instant query response.

Lastly, we study the self influential advertising (SIA) channel. SIA refers to the kind of campaigns made by the advertiser directly on his personal social homepage/channel. A carefully designed SIA could be effectively propagated to wider audiences through the advertiser’s friends or followers. To generate more attentions of SIA on SNPs, this thesis is the first to propose a personalized social influential tags exploration (PITEX) problem to find a size-k tag set that maximizes the advertisers’ social influences. We show that PITEX is NP-hard to approximate within any constant factor. Due to PITEX’s hardness result, we design a best-effort pruning framework with efficient indices to achieve real-time performance with probabilistic guarantee.

For each solution proposed in this thesis, we conduct extensive experiments over real-world social graphs with millions of users and billions of social links. These experiments reveal the effectiveness and efficiency of the proposed solutions. We believe our research findings demonstrate the immense potential for leveraging user profiles and sharing behavior to support advance SMM strategies.
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Notations frequently used across Chapter 3</td>
</tr>
<tr>
<td>3.2</td>
<td>All parameter settings used in the experiments of Chapter 3</td>
</tr>
<tr>
<td>4.1</td>
<td>Notations frequently used across Chapter 4</td>
</tr>
<tr>
<td>4.2</td>
<td>All parameter settings used in the experiments of Chapter 4</td>
</tr>
<tr>
<td>4.3</td>
<td>Disk space and running time of using $\theta_w$ and $\theta_w$ for constructing indices for news datasets</td>
</tr>
<tr>
<td>4.4</td>
<td>Disk space and running time for constructing indices for various datasets with $\theta_w$</td>
</tr>
<tr>
<td>4.5</td>
<td>Sum of $\theta_w$ and mean rr set size for increasing graph size</td>
</tr>
<tr>
<td>4.6</td>
<td>Number of I/O for IRR when varying $Q.k$</td>
</tr>
<tr>
<td>4.7</td>
<td>Influence spread when varying $Q.k.$</td>
</tr>
<tr>
<td>4.8</td>
<td>Example KB-TIM query results</td>
</tr>
<tr>
<td>5.1</td>
<td>Frequently used notations across Chapter 5</td>
</tr>
<tr>
<td>5.2</td>
<td>Statistics of Datasets</td>
</tr>
<tr>
<td>5.3</td>
<td>Index Sizes (MB) &amp; Construction Time (s)</td>
</tr>
<tr>
<td>5.4</td>
<td>An example case study of PITEX query on dblp dataset</td>
</tr>
</tbody>
</table>
## List of Figures

1.1 Campaign propagation network ..................................... 6

2.1 TA example ................................................................. 16

2.2 NRA example ................................................................. 18

3.1 System Overview of Context-Aware Advertisement Recommendation in Social Networks ...................................................... 34

3.2 Computing $\text{MinS}$ and $\text{MaxS}$ when the safe region sphere lies in the positive quadrant ................................................................. 41

3.3 Computing $\text{MinS}$ and $\text{MaxS}$ when the safe region sphere overlaps with x-y plane ................................................................. 42

3.4 The location of $x_{\text{max}}$ for an ad against a safe region ................................................................. 43

3.5 Example of computing $x_{\text{max}}$ when the safe region sphere overlaps with x-y plane ................................................................. 45

3.6 Example of computing if a SSR contains a query vector ................................................................. 47

3.7 Vary $\rho$ ................................................................. 57

3.8 Vary $\alpha$ ................................................................. 58

3.9 Vary $k$ ................................................................. 60

3.10 Vary Read Write Ratio ................................................................. 61

3.11 Vary Number of Topics ................................................................. 62
Chapter 1

Introduction

1.1 Motivations

The rapid development of social network platforms (SNPs) is driving new forms of social interactions, information dissemination and collaborations [85]. From gentle humour [103] to propagating breaking news [87], from scheduling a date [107] to following election results [107], SNPs are now becoming a ubiquitous host for Internet users. As such, we have witnessed a proliferation of social network usages. This generates tremendous opportunities for social media marketing (SMM) which has two distinct benefits over traditional ways of promotion and business advertising.

1. **Targeted Reach** [67, 73, 8]: Traditional marketing approaches like TV commercials or newspaper advertisements are incapable of identifying the audience interests and often incur high costs to be implemented. Compare to the traditional “shotgun” approaches, intelligent social marketing strategies can effectively target relevant users based on the analytics performed over the users’ profiles and activities on SNPs. Such marketing precision will significantly lower the advertising cost and achieve high returns for the promoted product.
2. **Easy Sharing** [65, 32, 28] As SNPs provide extremely easy ways for people to share information with their friends, viral marketing strategies make use of such channel to disseminate the advertisements on SNPs and generate a huge amount of buzz and brand awareness with much smaller investment comparing to traditional approaches. In a recent marketing study [5], social media turns out to be cheaper than any forms of advertising available today with a cost of less than $3 for a 1000 people advertisement exposure.

The aforementioned advantages have attracted millions of advertisers and billions of investments onto SMM. As the dominator in the market, Facebook made an advertisement revenue of 17.93 billion dollars in 2015, an increase of 58% year-over-year [3]. Moreover, due to complex structures of social networks and rich forms of social interactions, SMM has brought a full spectrum of research opportunities including machine learning [73, 11], information retrieval [94, 96] and database research [51, 58].

The aim of this thesis is to develop innovative approaches to promote SMM by leveraging both user interests and information propagation on SNPs. Hidden knowledge of user preferences can be mined from user profiles and social activities, whereas the model on how information propagates is constructed through analyzing the sharing behaviors observed on SNPs. As prior research has been focusing on developing SMM solutions by utilizing either side of the knowledge, this thesis intends to fill the research gap by combining the merits and develop novel SMM strategies. Specifically, this thesis studies three mainstream channels for SMM:

1. **Direct Targeted Ads (DTA)** [63, 73]: By mining user preferences, DTA channel posts ads directed to the potential relevant customers against the promoted ad. As the ad matches users’ interests, better ad click rates will be achieved than blindly choosing advertising targets.
2. **Celebrity Social Ads (CSA) [65, 32]**: The advertiser who chooses CSA negotiates with celebrities for their endorsements of the promoted product on SNPs. As the endorsements of the celebrities enable active sharings among SNP users, CSA creates more attentions and generates awareness of the product.

3. **Self Influential Ads (SIA) [1, 6]**: SIA refers to the kind of ad posts made by the advertiser directly on his personal social homepage/channel. A carefully designed SIA could be effectively propagated to wider audiences through the advertiser’s friends or followers.

The outcome of this thesis results in innovative solutions for each of the above channels, i.e. “Context-Aware Ad Recommendation” for DTA, “Targeted Influence Maximization” for CSA and “Personalized Social Influential Tags Exploration” for SIA.

In the remaining part of this chapter, we will first present the opportunities and challenges when designing SMM strategies through the above channels. Subsequently, we will summarize the contributions of the proposed solutions. The organizations of this thesis will be shown at the end of this chapter.

### 1.2 Opportunities and Challenges

In general, models built from combining knowledge of user preferences and information propagation are expected to be more effective in predicting the result of SMM strategy deployment. Advertisers can utilize such advance models to adjust their social campaign plans for higher returns. However, social data is gigantic with a highly complex social graph, which brings immense difficulties when designing new frameworks for SMM. In particular, the proposed solutions need to effectively model user preferences and information dynamics while producing efficient and theoretically sounded algorithms to perform the
advertising task. In the followings, we present the detailed opportunities and challenges in each of the thesis work.

### 1.2.1 Context-Aware Ad Recommendation for DTA

With the pay-per-click advertising methodology to assess the cost effectiveness, existing SNPs place great emphasis on delivering matching ads to potentially interested users through the DTA channel. They learn a prediction model for each user based on the personal interests and historical activities. When a user logs in his/her account, the most relevant ads matching the learned model are embedded in the news feed and presented to the user. However, the model only captures the slowly evolving personal interests of a user, resulting in repetitious ad recommendation. In addition, recent research has shown that, people find targeted advertising to be intrusive since the ads are too relevant to their specific areas of interest [102].

To mitigate the issue, we propose a context-aware ad recommendation framework that takes into account the relatively static personal interests as well as the dynamic news feed posted or shared by friends to drive growth in the ad click-through rate. Including the news feed as a dynamic context provides additional clue in the spatial, temporal and social dimensions for ad recommendation. For example, when a friend posts in Facebook the dining photos in a restaurant, relevant promotion coupons can be recommended. When a friend shows the status in hospital, displaying gift delivery ads is a good choice. Such motivation was also supported by a very recent work from Twitter [73] in which the contents in the tweet stream were taken into account to enhance the click-through prediction rate of advertising.

However, it is a rather challenging task to support social ad recommendation in a highly dynamic context. First, the posting rate and login frequency in Facebook and
Twitter are very high. A new post will appear in all the friends’ news feed and may cause their relevant ad sets to be changed. Second, the ad repository is huge, e.g., Facebook has over 1 million advertisers [7], making the query processing of relevant ad set rather expensive when the read frequency is very high. Chapter 3 shows how these challenges are tackled.

1.2.2 Targeted Influence Maximization for CSA

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. Subsequently, the advertiser will contact the users selected by IM for propagating CSAs on SNPs to generate broad brand awareness.

Earlier stage of researches on IM focuses on finding influential users efficiently without considering the personal preferences of each individual SNP user, which is obviously not realistic for real-world applications as users only browse information according their interests. Moreover, to produce a solution with $(1 - 1/e - \varepsilon)$ approximation ratio against the optimum, existing approaches, which based on greedy heuristics, takes a prohibitively long time to analyze a large social networks with millions of users. Recently, there have been some efforts to extend the IM problem to topic-aware IM [11, 58, 30, 74, 28] 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 online 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 [28] reported results on 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. Thus there are many challenges and opportunities to support instant topic aware IM on predominately large social networks with theoretical groundings to ensure solution quality. In Chapter 4 we will present our techniques against the above challenges and opportunities.

### 1.2.3 Personalized Influential Tags Exploration for SIA

SIA is a very fresh area in SMM. Although IM has been extensively studied, recent wide adoption of SNPs has brought a new demand on *personalized social influence exploration* for *individual users* in the SNPs. Unlike IM, which selects a subset of users to maximize the network influence, it aims to help a *target* user explore *how she influences the network*.

![Campaign propagation network](image)

**Figure 1.1: Campaign propagation network**

For example, political campaigns have an urgent demand to take advantage of SNPs as important channels for opinion poll to analyze their “selling points”. Fig. 1.1 shows an exemplary re-tweet network. The tweets containing the candidates’ political standpoints (e.g., *foreign policy*, *social security*) are propagated through their followers to the entire SN. It is highly desirable for any candidate, say *Hillary Clinton*, to eval-
uate the “effect” of her standpoints and judge which ones can influence more people, e.g. infrastructure rebuild and US-China relation. To win more voters, her publicity manager should then spend more time on these issues for Hilary’s subsequent public speeches or tweets. Another useful application is the case where businesses also want to position their marketing strategies by identifying the influential product features (e.g., high-tech, energy-saving) which are propagated to more people in SNPs. Similarly, researchers want to explore their most influential research contributions from academia SNs. Last but not the least, with an emerging trend of “we-media” (aka “self-media”), long-tail users are eager to know which topics to be published would receive more attentions from their potential SNP audiences.

Motivated by the new demand, we propose a novel social influence query known as personalized social influential tag exploration (PITEX): Given a target user, it extracts a size-\(k\) tag set that maximizes the user’s social influence from a set of tags which characterize the content propagated in an SNP. Due to its immense practical potentials, it calls for supporting online PITEX computation, as there would be a huge number of users in SNPs posing PITEX queries and each query needs to be answered efficiently. The solutions proposed for achieving instant PITEX processing will be presented in Chapter 5.

1.3 Contributions

The main contribution of this thesis is that we propose using information from both user preference and social sharing behavior in assisting various SMM applications. The proposed algorithms are not only efficient and scalable against large social data but are also theoretical sounded as we provide approximation guarantees to those intractable problems. The detailed technical contributions of this thesis are presented as follows:
1.3.1 Context-Aware Ad Recommendation for DTA

For DTA, we propose a news feed ad posting framework that considers both static user interests and dynamic news feeds made by the users’ friends. To meet the requirement of real-time advertising, we first propose an online retrieval strategy that adopts existing top-k aggregation algorithms [42, 59] to find the most relevant ads matching the dynamic context when a read operation is triggered.

However, when the context varies little, the online retrieval may retrieve the same set of top-k ads repetitively, which is a waste of CPU resources. Thus, we further propose a safe region method to quickly determine whether the top-k ads of a user are changed. We guarantee that as long as the dynamic context is located within the safe region, the top-k results remain the same and the cost of repetitive retrieval is saved.

Finally, we observed that when the dynamic context vary dramatically, online retrieval is preferred because the safe region can only guarantee the safeness for a short period of time and requires frequent reconstruction. Otherwise, safe region technique is a suitable choice. To combine the merits of both retrieval strategies, we propose a hybrid model that analyzes the dynamism of news feed for each user to determine which strategy should be applied.

Extensive experiments have been conducted on real-world social networks with billions of edges real ad datasets with millions of tuples. The experimental results show that our hybrid method significantly outperforms the other two retrieval strategies up to 30x speedups.

1.3.2 Targeted Influence Maximization for CSA

For CSA, we propose a new keyword-based targeted IM (KB-TIM) query for targeted viral SMM. The query finds a seed set that maximizes the expected influence over users who are
relevant to a given ad. In other words, the expected influence only incorporates those users interested in the ad as targeted customers. To solve the problem, we propose a weighted sampling technique based on RIS and achieve an approximation ratio of \((1 - 1/e - \varepsilon)\).

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 \([104]\) to find the top-\(k\) seed users. IRR further improves over RR by incrementally loading the most promising RR sets into memory and adopts the top-\(k\) aggregation strategy to save computation costs.

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.

### 1.3.3 Personalized Social Influential Tags Mining for SIA

We introduce a novel social tags exploration problem, namely \textit{personalized social influential tags exploration} (PITEX), to support effective SIA applications. In this thesis, we study the research challenges that naturally arise in supporting efficient PITEX processing. The first challenge is to generate a total set of tags from the contents propagated in an SNP and analyze the correlation between the tags and influence spread. Thanks to the existing topic-based social influence models \([11, 82, 58, 84, 28]\), we can generate both the tag set and its correlation to influence spread by analyzing the interaction history of SNP users. The second challenge is to analyze the complexity of answering PITEX query. We prove that it is not only a NP-hard problem but also NP-hard to be approximated within
any constant ratio. We introduce a sampling-based framework and employ Monte Carlo (MC) sampling [65, 71] and Reverse Random set (RR) sampling [17, 101] to estimate the influence spread and find the best size-\(k\) tag set. The third challenge is to achieve high performance, as millions of SNP users may issue PITEX queries to explore their personalized influence tags. Due to PITEX’s hardness, the framework with the existing sampling techniques is both theoretically and experimentally inefficient according to our studies. We are thus motivated to devise efficient algorithms to achieve real-time performance with theoretical guarantee.

To achieve instant performance to answer PITEX queries, we first propose approaches for optimizing online sampling. We devise a lazy propagation sampling algorithm to probe as few edges as possible when estimating the influence spread, which significantly outperforms MC and RR sampling techniques. We also develop a best-effort exploration strategy that effectively estimates bounds of influence spread and thus avoids computing the actual influence spread of tag sets with small influence. Furthermore, to enable real-time influence computation, we devise an effective index structure that materializes the “influencer” of uniformly sampled users, and develop efficient pruning and materialization techniques to support fast influence computation with moderate index size. We prove that all the approaches proposed for PITEX can achieve a \(\frac{1-\varepsilon}{1+\varepsilon}\) approximate solution.

We evaluate the performance on four large-scale real-world datasets. The results of the extensive experimental study demonstrate that PITEX could extract influential-tags effectively and efficiently by using the proposed methods.
1.4 Organization

The rest of the thesis is organized as follows: Chapter 2 review related topics. The literature review fall into three categories: top-\(k\) aggregation query, social search and recommendation and the problem of IM.

Chapter 3 describes our solution on context-aware ad recommendation in high speed news feeding. We introduce a online retrieval method to dynamically search for relevant ads and then propose a safe region methods which the dynamic search is only triggered when the relevant ads changes to save CPU resources. Lastly, we develop a hybrid method by combining online retrieval and safe region so that an customized retrieval strategy is assigned to each SNP user.

Chapter 4 addresses how to efficiently process the keyword-based targeted IM problem. A weighted sampling method is first proposed to solve the problem with a \((1 - 1/e - \varepsilon)\) approximation ratio. Subsequently, two disk-based indices are introduced to achieve the instant processing requirement.

Chapter 5 presents our novel framework for personalized influential social tags mining. As shown in Chapter 5, personalized influential social tags mining is NP-hard to approximate with any constant ratio, we study two sampling approaches and subsequently propose a optimized sampling framework which solves the problem with a \(\frac{1-\varepsilon}{1+\varepsilon}\) approximation ratio. To further enhance the performance, we develop an effective index structure with a filter-and-verification method which enables real-time tag exploration.

Chapter 6 discusses some future working directions and concludes the whole thesis.

The works of Chapter 3 and Chapter 4 have been published in [79] and [80] respectively. The research in Chapter 5 has been submitted for publication and its technical report can be found in [78].
Chapter 2

Literature Review

Before examining the detailed related work, we would like to present some preliminaries about SNPs and SMM. To achieve a successful SMM deployment, we first need to take a deeper look at what are the capabilities of SNPs. SNPs cannot be understood without introducing the concept of Web 2.0: a term that describes a new way in which end users use the World Wide Web, a place where content is continuously altered by all operators in a sharing and collaborative way [64]. Thus, we have seen the evolution of Web 2.0 that people are now continuously creating and consuming knowledge rather than merely performing simple information retrieval tasks [21].

The phrase SNP is often used interchangeably with social media. Kaplan and Haenlein [64] define social media as “a group of Internet based applications that build on the ideological and technological foundations of Web 2.0, and allow the creation and exchange of user generated content.” However, there are some basic features for a website to meet the requirements as a SNP: the site must contain user profile, content, a channel that permits users to connect with each other and post comments on each other’s pages, and join virtual groups based on common propensities such as sports or politics [93].
SNPs have advanced from simply providing a platform for users to stay in touch with their family and friends. Consumers can learn more about their favorite companies and the products they sell. For example, Geo-social networking sites Foursquare provides not only Geo-information services to its users but also restaurant suggestions, shop reviews and price alerts. Marketers and retailers are thus utilizing these SNPs as a new way to reach consumers. Provide shopping services on SNPs can trigger business growth for retailers due to the diversity of consumers who use SNPs. The wide range of SNPs users means that most target markets can be reached [23].

Although SNPs provide excellent environment for social advertisers, massive social data and complex social graphs have also brought huge challenges for effective SMM deployments. Facebook data warehouse stores more than 300 Petabyte data with an incoming daily rate of 600 Terabytes [4]. There are over 1.59 billion facebook users with an average friends of 338 at the end of 2015 [2]. Many research efforts have been focusing on large scale SMM solutions and they are closely related to our works. In the remaining parts of this chapter, we first review existing techniques about top-\(k\) aggregation query (relaed to Chapter 3 and 4) due to its wide range of applications in social recommendations [98, 27, 25, 77]. Then, we survey past works about social search and recommendation (related to Chapter 3 and 5). Lastly, we study the influence propagation models and presents various methods proposed for the IM problem (related to Chapter 4 and 5).

2.1 Top-\(k\) Aggregation Query

Due to its wide range of applications such as web search engines, image retrieval and etc, top-\(k\) aggregation query is a well-studied problem in the literature [42, 59, 41, 52, 52, 53]. Consider a database \(D\) where each object \(o = (p_1, p_2, \ldots, p_n)\) has \(n\) scores, one for each of its \(n\) attributes. Given a monotonic aggregation function \(f\), where \(f(o)\) or \(f(p_1, p_2, \ldots, p_n)\)
denotes the overall score of object \( o \), the top-\( k \) aggregation problem is to find a set of top-\( k \) objects in \( D \) with the highest overall scores.

Many approaches such as term-at-t-time (TAAT) \cite{19, 105}, document-at-a-time (DAAT) \cite{18, 105}, threshold algorithm (TA) \cite{42}, no random access (NRA) \cite{42} and their variants \cite{42, 59, 112, 80} have been proposed. According to the data access assumptions, the proposed solutions can be classified as: (1) methods which allows random access; (2) methods which only allows sorted access and bans random access. In sorted access, objects are accessed sequentially ordered by attribute scores, while for random access, objects are directly accessed by their identifiers. The techniques surveyed in this section assume multiple lists for each attribute that rank the same set of objects based on their attribute scores.

### 2.1.1 Allowing Random Access

Top-k query processing techniques in this category assume that both access methods, sorted and random, are supported. Random access allows for obtaining the overall score of some object right after it appears in one of the sorted list. The Threshold Algorithm (TA), Combined Algorithm (CA) \cite{12} and Document-At-A-Time (DAAT) \cite{18, 105} belongs to this category. Algorithm 1 presents the framework for TA:

\begin{algorithm}
  \caption{TA \cite{42, 59}}
  \begin{algorithmic}
    \STATE 1 Perform a sorted access in parallel to each of the \( n \) sorted lists. For each object accessed, perform a random access to other topics and compute the aggregated score of \( f(o) \). If the computed aggregated score is one of the \( k \) highest we have seen so far, remember the ad and its score.
    \STATE 2 For each list \( L_i \), let \( high[i] \) be the score of the last ad seen under sorted access. Define the threshold value \( B_k \) to be the aggregated score of \( high[i] \) by the aggregation function \( f \). As soon as at least \( k \) objects have been seen whose score is at least equal to \( B_k \), the algorithm terminates.
  \end{algorithmic}
\end{algorithm}
Example 1. TA Example [42]: Consider two sorted lists $L_1$ and $L_2$ holding different rankings for the same set of objects based on two different scoring predicates $p_1$ and $p_2$ respectively. Each of $p_1$ and $p_2$ produces score values in the range $[0, 50]$. Assume each source supports sorted and random access to their ranked lists. Consider a score aggregation function $F = p_1 + p_2$. Figure 2.1 depicts the first two steps of TA. In the first step, retrieving the top object from each list, and probing the value of its other scoring predicate in the other list, result in revealing the exact scores for the top objects. The seen objects are buffered in the order of their scores. A threshold value, $T$, for the scores of unseen objects is computed by applying $F$ to the last seen scores in both lists, which results in $50 + 50 = 100$. Since both seen objects have scores less than $T$, no results can be reported.
In the second step, $T$ drops to 75, and object 3 can be safely reported since its score is above $T$. The algorithm continues until $k$ objects are reported, or sources are exhausted.

In our context-aware ad recommendation problem (Chapter 3), as we consider in-memory recommendation without disk I/O, we adopt the TA framework for top-$k$ retrieval as it has been proved to be instance optimal [42].

2.1.2 No Random Access

The techniques we discuss in this category assume random access is not supported. The No Random Access (NRA) algorithm [42] and the Stream-Combine algorithm [53] are two examples of the techniques that belong to this category.

Algorithm 2: NRA [42] 59

1. Let $p_1^{\text{min}}, ..., p_n^{\text{min}}$ be the smallest possible values in lists $L_1, ..., L_n$.
2. Do sorted access in parallel to lists $L_1, ..., L_n$ and maintain the last seen attribute values $\overline{p}_1, ..., \overline{p}_n$ in the $n$ lists. For every object $o$ with some unknown attributes values, compute a lower bound for $f(o)$, denoted by $\overline{f}(o)$, by substituting each unknown attribute $p_i$ with $p_i^{\text{min}}$. Similarly, compute an upper bound $\underline{f}(o)$ by substituting each unknown attribute $p_i$ with $\overline{p}_i$.
3. Let $A_k$ be the set of $k$ objects with the largest lower bound values $\underline{f}(.)$ seen so far and $M_k$ be the $k^{\text{th}}$ largest $\underline{f}(.)$ value in $A_k$. The algorithm terminates and return $A_k$ when (a) at least $k$ distinct objects have been seen, and (b) $\overline{f}(o) \leq M_k$ for all $o \notin A_k$.

Example 2. NRA Example [42]: Consider the same example as we described in Example 1. Assume both sources support only sorted access to their ranked lists. Figure 2.2 depicts the first three steps of the NRA algorithm. In the first step, retrieving the first object in each list gives lower and upper bounds for objects’ scores. For example, object 5 has a score range of $[50, 100]$, since the value of its known scoring predicate $p_1$ is 50, while the value of its unknown scoring predicate $p_2$ cannot exceed 50. An upper bound for the scores of unseen objects is computed as $50 + 50 = 100$. The seen objects are
buffered in the order of their score lower bounds. Since the score lower bound of object 5, the top buffered object, does not exceed the score upper bound of other objects, nothing can be reported. The second step adds two more objects to the buffer, and updates the score bounds of other buffered objects. In the third step, the scores of objects 1 and 3 are completely known. However, since the score lower bound of object 3 is not below the score upper bound of any other object (including the unseen ones), object 3 can be reported as the top-1 object. Note that at this step object 1 cannot be additionally reported, since the score upper bound of object 5 is 80, which is larger than the score lower bound of object 1.
Other than the aforementioned algorithms, TAAT (term-at-a-time) \cite{19,105} is a popular method for top-k query processing without random access in the information retrieval community. In the TAAT search framework, the inverted lists relevant to a query are accessed in descending order of query weight. The objects in each list are retrieved from disk and their partial scores are aggregated in an accumulator. Various early termination techniques \cite{19,105} were employed to save computational cost.

Our proposed efficient disk-based index for processing keyword-based targeted IM queries is motivated by these NRA frameworks as it efficiently supports disk-based applications since random access is rather expensive in such scenarios.

2.1.3 Immutable Regions

Local immutable region (LIR) \cite{90} and global immutable region (GIR) \cite{113} are another two relevant top-k query processing works to our context-aware ad recommendation problem. For a given query vector with the respective top-k entities, LIR searches for a valid interval for a given dimension in the query vector such that the top-k entities remain the same, while all other dimension weights are kept constant. However, in our problem, the weights for different dimensions change simultaneously which LIR is unable to handle because of the local nature of LIRs. More specifically, if the query weight $w_i$ on a simple dimension $i$ is updated, the immutable regions constructed for other dimensions are invalidated, even if the new value of $w_i$ remains within its LIR. GIR is able to support simultaneous adjustments to multiple dimensions. Unfortunately, GIR is computationally expensive as it takes minutes to get the valid region for a given query vector with only 5-8 dimensions. This makes GIR infeasible to handle the dynamic nature of social news feeds as user often refreshes in a few seconds. To overcome this issue, we design a series of
techniques in Chapter 3 to quickly compute a subspace of GIR so that the maintenance cost is greatly reduced.

2.2 Social Search and Recommendation

Social search and recommendation are important problems for improving the user experiences on SNPs. In social search context, users issue keyword queries similarly as normal web searches. However, social search has three distinct features from the web searches. Firstly, time freshness of the query results are essential for social media search as outdated information means nothing to user in a highly dynamic SNP. Secondly, SNP users tend to receive updates from those who are socially close to them. Lastly, the search results should not only match the query keywords, but also match the interests of users based on their historical activities on SNPs. Thus these distinct characteristics must be considered when designing search engines on SNPs.

There has been much efforts made to address the problem of microblog search in social networks. Facebook developed Unicorn to handle large-scale query processing. Uniform supports socially related search and is built on list operators like $LIST\text{AND}$ and $LIST\text{OR}$ to merge search results from different sources. Twitter’s real time query engine, Earlybird, is developed to handle high throughput query evaluation for fast rate of incoming tweets workload. The above social search engines are designed for general purpose social searches. Meanwhile, many researchers have proposed various social search solutions to improve the quality and efficiency for social query processing. Chen et al. introduced a partial index named TI to enable instant keyword search for twitter. TI takes consideration of time freshness, text similarity and pagerank popularity of the tweets in ranking search results. Tao et al. proposed an index to search for the microblogs which are ranked by their provenance in the network. Li et al. devised a
3D inverted index to enable efficient microblog search by considering content similarity, time freshness and social relevance to the query user [77]. However these indices are designed to search the microblogs whereas the microblogs (or new feed posts) in our context-aware ad recommendation problem (Chapter 3) are used as queries to retrieve relevant ads. This means existing work cannot be applied to since the dynamism of the query is not considered. Therefore our solution proposed takes into consideration both the property of social graph and the dynamism of microblogs in the news feeds to deliver highspeed ad recommendation.

Other than social search, social recommendation is another popular area in recommendation systems research [94, 96, 44, 8, 97, 99] and is closely related to our problem studied in Chapter 3. There are various types of recommendation techniques on SNPs, which can be broadly classified as content recommendation, tag recommendation, people recommendation and community recommendation [55]. Content and tag recommendation focus on finding relevant and eye-attracting results for SNP users [97, 96], whereas people and community recommendation aim to performing recommendation based on social relationship and common interests to build new social ties between SNP users [108, 114]. As the personalized influential tags exploration (PITEX) proposed in Chapter 5 is most related to content and tag recommendation, we will omit the discussion for people and community recommendation.

Existing approaches can be classified into two categories: 1) recommend content that match users’ interests [94, 96, 44, 86, 68]; 2) recommend prevalent content with high social popularity [8, 97, 99, 69]. In the first category, previous works often model the relevancy between the recommended content and users by various similarity functions [68, 94] or probabilistic models on which users would view such contents [86, 96, 44]. In the second category, existing works make use of global network ranking functions like pagerank [97] or HITS [69] to measure the popularity of recommended contents. Indeed, many researches
have combined the features from these two categories to perform hybrid recommendation.

Compared with prior studies, our personalized influential tags exploration (PITEX) problem proposed in Chapter 5 acts as the first effort to study social recommendation for maximizing users’ personalized influence in SNPs. Rather than retrieving contents to attract the recommended user, PITEX is aiming to generate more attentions from the recommended the user’s friends, friends’ friends and even further relationships, through the channel where the user posts the contents recommended by PITEX on his personal social homepage. Moreover, PITEX can also be integrated into existing works by recommending social contents (tags) which not only capture users’ interests or global popularity scores but also improve social influence.

2.3 Influence Maximization

Viral marketing strategies is one of the major advantages of using SNPs to promote business. To understand the extent to which the promoted information are adopted, we need to examine how the dynamics of adoption are likely to unfold within the underlying SNP. In particular, the extent to which people are likely to be affected by decisions or actions of their friends, or the extent to which “word-of-mouth” effects will take hold must be scrutinized so that these phenomena could be leveraged for designing effective SMM strategies [65]. The most popular models used to model the aforementioned network diffusion processes are independent cascade [46], linear threshold [50], triggering models [65] and their variants [11, 47]. Based on these propagation models, Influence Maximization (IM) is an algorithmic problem which finds a size-$k$ set of users who generate the most influence within the underlying social network among all possible size-$k$ user set. IM receives a lot of attentions due to both its practical usages and the theoretical challenges.
In the following sections, we first review the diffusion models followed by survey existing methods for solving the IM problem.

2.3.1 Propagation Models

When considering models for the spread of an idea through a directed social network \( G \), we say each individual node as being either active (an adopter of the idea) or inactive. Based on work in Interacting Particle Systems (IPS) \cite{40, 81} from probability theory, Goldenberg, Libai, and Muller \cite{46, 45} investigated the simplest model of IPS, namely Independent Cascade Model, in the context of viral marketing. Let us denote \( A_0 \) to be an initial set of active nodes (with all other nodes inactive) and the influence process unfolds in discrete steps according to the following randomized rule. When a node \( v \) first becomes active in step \( t \), it has only one change to activate each of its inactive neighbor, say \( w \), with a success probability of \( p_{v,w} \). If \( v \) succeeds, then \( w \) become active in step \( t+1 \). This process continues until no more activations are possible.

Linear threshold model is another well-studied method to capture the influence progress, which makes use of the node-specific thresholds \cite{50, 95} to determine if a node becomes active or not. In this model, a node \( v \) is influenced by each neighbor \( w \) according to a weight \( b_{v,w} \) such that \( \sum_{w: \text{neighbor of } v} b_{v,w} \leq 1 \). The dynamics of the process then proceed as follows. Each node \( v \) chooses a threshold \( \theta_v \) uniform randomly in \([0, 1]\), which represents the weighted fraction of \( v \)'s neighbors that must become active in order for \( v \) to become active. Given a random choice of thresholds, and an initial set of active nodes \( A_0 \), the diffusion process also unfolds in discrete steps: in step \( t \), we activate any node \( v \) for which the total weight of its active neighbors is at least \( \theta_v \):

\[
\sum_{w: \text{neighbor of } v} b_{v,w} \geq \theta_v \tag{2.1}
\]
Thus $\theta_v$ intuitively represent the different latent tendencies of nodes to idea the innovation when their neighbors do; Since we are lack of knowledge for the true values of $\theta_v$, we need to average all possible $\theta_v$ values in $[0, 1]$ for all nodes to complete the linear threshold model.

To generalize these models and avoid their limitations, Kempe et al. [65] proposed the general cascade model and the general threshold model as follows:

- General cascade model: The independent cascade model considers the activation from an active node $u$ to an inactive node $v$ independently of $v$’s other active neighbors. Thus the generalization lies in considering dependencies between the activation processes. Let an incremental function $p_v(u, S) \in [0, 1]$ where $S$ and $\{u\}$ are disjoint subsets of $v$’s neighbor set. In the general cascade model, $u$ will active $v$ with a success probability of $p_v(u, S)$ where $S$ is the set of neighbors that have already tried and failed to activate $v$.

- General threshold model: The generalization of linear threshold model lies in making a node $v$’s decision to be come active based on an arbitrary monotone function of the set of $v$’s neighbor who are already active. Thus, a monotone threshold function $f_v$ is defined for $v$ to map subsets of $v$’s neighbors to $[0, 1]$ and $f_v(\emptyset) = 0$. Then $v$ becomes active in step $t$ if $f_v(S) \geq \theta_v$, where $S$ is the set of $v$’s neighbors that are active in step $t - 1$. Note that $\theta_v$ is still chosen uniform randomly in $[0, 1]$ as the linear threshold model.

However, these generalized models are computational intractable for problems like IM [65]. Thus a slightly restricted triggering models was proposed. In such a model, each node $v$ independently chooses a random “triggering set” $T_v$ according to some distribution over subsets of its neighbors. After the initialization, an inactive node $v$ becomes active in step $t$ if it has a neighbor in its chosen triggering set $T_v$ that is active at time $t - 1$. 

24
Algorithm 3: Greedy(Social Graph \( G(V,E) \), Influence Set Size \( k \))

1. \( S \leftarrow \emptyset \)
2. for \( i \leftarrow 1 \ldots k \) do
3. \( u \leftarrow \arg\max_{v \in V \setminus S} (\sigma(f(S \cup \{v\}) - f(S))) \)
4. \( S \leftarrow S \cup \{u\} \)
5. return \( S \)

Although there are other complex models \([47, 92]\), the independent cascade, linear threshold and trigger models are the most popular ones due to the sub-modular property of the induced influence function \( \sigma(S) \) \([65]\), which denote the expected influence of the node set \( S \) under these models. This is particularly useful when dealing with problems like IM as we will discuss in the next section.

### 2.3.2 Traditional Influence Maximization

IM problem is a key algorithm in the era of viral marketing. Although the problem looks simple, which finds a set \( k \) set of users who have the largest expected influence, it is notoriously a NP-Hard problem \([65]\). Moreover, the computation of the influence function \( \sigma(S) \) for any fixed node set \( S \) is often intractable for most models, i.e. \( \#P \) hard for the independent cascade model \([31]\). To overcome these challenges, earlier stage of researches leverages the sub-modular property of \( \sigma(S) \) and propose the simple greedy algorithm which achieves an approximation ratio of \( (1 - 1/e - \varepsilon) \) to the optimal influence \([65, 71]\). The outline of the greedy approaches is presented in Algorithm 3. The algorithm iteratively selects a new initial active user \( u \) that maximizes the incremental change of \( \sigma \) into the influential set \( S \) until \( k \) seeds are selected.

Subsequently, there has been a large body of research work devoted to improve the efficiency while keeping the theoretical bound \([71, 17, 49, 101, 62, 66]\). Although CELF \([71]\) and its variant CELF++ \([49]\) have significantly improved the running time, the meth-
ods were only examined in small graphs with thousands of vertices. The bottleneck of these proposed methods is to estimate the influence score. To overcome such a problem, RIS [17] is proposed and shown to be the first method scalable enough to handle graphs with millions of vertices. Instead of performing Monte Carlo samples from the active set $S$ to estimate $\sigma(S)$, RIS first uniformly samples a node $u$ and then reverse samples a “hyper node set” $G(u)$ where all nodes in $G(u)$ active $u$ in a single propagation instance. By generating many hyper node sets, RIS solves the IM problem by applying greedy maximum coverage algorithm to find the set which intersects with the most number of the hyper node sets sampled. In this way, the samples generated can be used for estimating multiple initial active sets to significantly reduce the influence computation time. A detailed example will be presented in Example 4. The method was further improved in [101, 100] in terms of sampling efficiency.

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. [32] used vertex degree as a quick selection criterion. In [31, 33, 62, 66], the propagation behaviour is simplified by removing social paths that have low propagation probabilities. The most representative simplification of this type is the Maximum Influence Path (MIP) approach [31]. Instead of considering multiple influence paths from a node $u$ to another node $v$, MIP only considers the path which has the highest probability for $u$ to activate $v$. After discarding all MIPs which have a smaller influence probability than a system defined threshold $\theta$, the influence of a node $u$ is by aggregating all probabilities of MIPs starting from $u$. In this way, MIP approach reduces the complexity of influence computation. However, such reduction does not have any approximation to the original propagation models. Moreover, although heuristic solutions are more efficient, none of them have been examined in large social
networks. The state-of-the-art solutions (PMIA [31] and IRIE [62]) require more than 10 minutes to run a graph with less than 1 million edges.

2.3.3 Topic-aware Influence Maximization

The traditional IM problem 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 [11, 58, 30, 74, 28]. In [11], the influence probabilities that a user influences his neighbors are different for different topics. The topic influence probabilities are extracted from the historical action logs recorded for different users on SNPs. Subsequently, Chen et al. [30] proposed a solution for the topic-aware IM by extending PMIA. As mentioned, PMIA does not have any theoretical bound and is not applicable to very large graphs. In [58], Inflex was proposed to achieve real-time performance. Inflex first pre-computes 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. [28] 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 [11, 58, 30] on handling graphs with more than 1 million vertices and [28] 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 keyword-based IM proposed in Chapter 4 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.

### 2.3.4 Other Influence Maximization Problems

There are many other types of IM problem proposed. Bo et al. [83] studied the time constrained IM problem which finds the most influential set given the number of allowed propagation steps. Li et al. [28] proposed a location-aware IM problem where the selected initial active set will influence the largest number of nodes in a spatial range. The competitive IM problem was introduced in [14]. The competitive IM studies influence propagation of more than one competing parties. For example, when a twitter user likes a post of Samsung smart phone releases, it is unlikely that this person will be affected by other smart phone posts, e.g. iPhone. This is because the smart phone brands are competing relationships and users are often tied to one brand. There are many follow-up works to refine the competitive IM problem so that the model becomes more realistic [29, 91, 76, 10, 37]. Although this thesis does not consider these more advanced models, they could be interesting directions to explore.
Chapter 3

Context-Aware Advertisement Recommendation for DTA

In this chapter, we first study the direct targeted ads (DTA) channel where the social advertisements are distributed to match the users’ interests. With the pay-per-click advertising methodology to assess the cost effectiveness, existing social network platforms place great emphasis on delivering matching ads to potentially interested users. They learn a prediction model for each user based on the personal interests and historical activities. When a user logsins his/her account, the most relevant ads matching the learned model are embedded in the news feed and presented to the user. However, the model only captures the slowly evolving personal interests of a user, resulting in repetitious ad recommendation. In addition, recent research has shown that, people find targeted advertising to be intrusive since the ads are too relevant to their specific areas of interest [102].

To mitigate the issues, we propose a context-aware ad recommendation framework that takes into account the relatively static personal interests as well as the dynamic news feed from friends to drive growth in the ad click-through rate. We treat the news
feed as a dynamic context that provides additional clue in the spatial, temporal and social dimensions for ad recommendation. For example, when a friend posts in Facebook the dining photos in a restaurant, relevant promotion coupons can be recommended. When a friend shows the status in hospital, displaying gift delivery ads is a good choice. Such motivation was also supported by a very recent work from Twitter \[73\] in which the contents in the tweet stream were taken into account to enhance the click-through prediction rate of advertising.

It is a challenging task to support social ad recommendation in a highly dynamic context. First, the posting rate and login frequency in Facebook and Twitter are very high. A new post will appear in all the friends’ news feed and may cause their top-\(k\) relevant ads to be changed. Second, the ad repository is huge, e.g., Facebook has over 1 million advertisers \[7\], making the top-\(k\) query processing rather expensive when the read frequency is very high. To meet the real-time requirement, we first propose an online retrieval strategy that adopts existing top-\(k\) aggregation algorithms \[42, 59\] to find the most relevant ads matching the dynamic context when a read operation is triggered. However, when the context varies little, the online retrieval may retrieve the same set of top-\(k\) ads repetitively, which is a waste of CPU resources. Thus, we further propose a safe region method to quickly determine whether the top-\(k\) ads of a user are changed. We guarantee that as long as the dynamic context is located within the safe region, the top-\(k\) results remain the same and the cost of repetitive retrieval is saved. Finally, we observed that when the dynamic context vary dramatically, online retrieval is preferred because the safe region can only guarantee the safeness for a short period of time and requires frequent re-construction. Otherwise, safe region technique is a suitable choice. To combine the merits of both retrieval strategies, we propose a hybrid model that analyzes the dynamism of news feed for each user to determine which strategy should be applied.

To sum up, the contributions of this chapter include:
• We propose a new context-aware advertisement recommendation framework on social networks by considering both long-term user interests and highly dynamic contents in the news feed.

• We present an online retrieval strategy that obtains $k$ most relevant ads when a read operation is triggered.

• We devise a safe region technique to avoid repetitive retrieval when the context varies little.

• We propose a hybrid model to seamlessly combine the merits of the two retrieval strategies.

• We conduct extensive experiments on real social networks with billions of edges and real ad datasets with millions of tuples. The experimental results show that our hybrid method significantly outperforms the other two retrieval strategies up to 30x speedups.

The preliminaries of the context-aware ad recommendation are presented in Section 3.1 and we devise the online retrieval algorithm in Section 3.2. The safe region method is introduced in Section 3.3. Subsequently, we propose the hybrid method in Section 3.4. The experimental results are reported in Section 3.5. Finally we summarize the chapter in Section 3.6.

### 3.1 Problem Statement

We study the problem of context-aware advertisement recommendation for users in a directed social graph $G = (V, E)$. All notations frequently used in this chapter are listed in Table 3.1 for ease of reference. Each user is associated with a personal profile that
<table>
<thead>
<tr>
<th>Notation</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>( G, V, E, T )</td>
<td>the social network, the vertex set, the edge set and the topic space respectively.</td>
</tr>
<tr>
<td>( A )</td>
<td>the set of all ads</td>
</tr>
<tr>
<td>( u, v )</td>
<td>users in social networks.</td>
</tr>
<tr>
<td>( m )</td>
<td>number of posts that appeared in the current window of a user’s news feed.</td>
</tr>
<tr>
<td>( k )</td>
<td>number of ads posted in a user’s news feed.</td>
</tr>
<tr>
<td>( \text{rel}(u, d, w) )</td>
<td>the relevance of a user ( u ), a post ( d ) and an advertisement ( a ) against a given topic ( w ) respectively.</td>
</tr>
<tr>
<td>( \alpha )</td>
<td>the weight parameter between ([0, 1]) to balance the importance between personal interests and dynamic context.</td>
</tr>
<tr>
<td>( \theta(x, y) )</td>
<td>the angle between two vectors ( x ) and ( y ).</td>
</tr>
<tr>
<td>( Q_u )</td>
<td>the context aware query vector for user ( u ).</td>
</tr>
<tr>
<td>( Q^{ub}_u, Q^{ub}_u )</td>
<td>upper and lower bound vectors for the safe region of user ( u ).</td>
</tr>
</tbody>
</table>

evolves slowly and a pool of unread messages from friends that update dynamically. Let \( T \) denote the topic space with a fixed number of pre-defined topics. We represent the relatively static profile of user \( u \) as a weighted term vector \( H_u \) with \(|T|\) dimensions. \( H_u \) can be generated by aggregating all the social activities such as historical posts, likes and sharing into a single document and use \textit{latent Dirichlet Allocation (LDA)} \cite{15} to map the explicit information into the latent topic space \( T \). We use \( \text{rel}(u, w) \in [0, 1] \) to denote the relevance between a user \( u \) and a topic in \( T \).

We assume that there is an ad database \( A \) in the system and our goal is to recommend \( k \) most relevant ads when a user logsins or requests for the news feed. Suppose all the ads have been projected to the same latent topic space as the user profile by LDA. We measure the relevance between an ad and the static user profile as

\[
\phi_s(u, a) = \sum_{w \in T} \text{rel}(u, w) \cdot \text{rel}(a, w) \tag{3.1}
\]
Our context-aware advertisement recommendation also takes into account the dynamic news feed when measuring the relevance score between a user and an ad. We use a sliding window $W_u$ to store $m$ most recent posts disseminated to user $u$. The contents in the window are used as the dynamic context for ad recommendation. We apply LDA to project each post in the window to the latent topic space and use $\text{rel}(d, w) \in [0, 1]$ to measure the relevance between a post and a topic. These scores are aggregated and normalized as follows to measure the contextual relevance w.r.t an ad.

$$
\phi_d(u, a) = \frac{1}{m} \sum_{d \in W_u} \sum_{w \in T} \text{rel}(d, w) \cdot \text{rel}(a, w) 
$$

Finally, the total relevance of an ad $a$ and a user $u$ is modeled as a linear combination of $u$’s static interests and dynamic context:

$$
\phi(u, a) = \alpha \cdot \phi_s(u, a) + (1 - \alpha) \cdot \phi_d(u, a)
$$

$\alpha \in [0, 1]$ is a system parameter to balance the importance between personal interests and dynamic context. Based on the ranking function in Eqn. 3.3, we formally define our problem as follows:

**Definition 1.** For any given user $u$ in the social network, the context-aware advertisement recommendation finds a set of ads, i.e. $R$, which has a size of $k$ and satisfies $\phi(u, a) \geq \phi(u, a') \ \forall a \in R \land \forall a' \in A \setminus R$.

Figure 3.1 illustrates the system overview. Each user in the social network is considered as both a subscriber and a publisher. When a user composes, shares or likes a post, we say the user, as a publisher, triggers a write operation. The new post is first sent for topic analysis, stored in the posts database and later may be retrieved to appear in the news feed of his social friends. When a user logins or refresh his/her news feed, we say
the user, as a subscriber, triggers a read operation. Then, the posts from friends are retrieved and sorted chronologically and a sliding window containing $m$ recent unread posts is returned. The topic distributions in the dynamic news feed are aggregated with the static personal profile vector as in the ranking function in Eqn. 3.3 to query the ad database. We call the aggregated vector context-aware query vector, denoted by $Q_u$. In the following section, we show how to handle such top-$k$ query in a large ad database.
3.2 Online Retrieval Algorithm

Existing social ad recommendation systems learn a model for personal interests offline. Since the model is relatively static, the top-k relevant ads for each user can be computed offline and returned together with the news feed when a read operation is triggered. However, when the dynamic context is taken into account in the ranking function, we are unable to pre-compute the ads for each user because each write operation will cause the news feed of all the friends to vary and the incurred pre-computation cost is unaffordable. In this section, we introduce how to efficiently retrieve the top-k relevant ads on the fly.

When a user $u$ triggers a read operation, we need to retrieve the unread posts from the neighbors of $u$, sort them in chronological order and obtain a window of $m$ posts as the dynamic context. Then, a straightforward solution is to construct the query vector by combining the topic distributions in the dynamic window and static personal interests and issue a top-k query against the ad database. Without proper indexes, it needs to scan all the ads in order to find $k$ of them with the highest relevance scores, incurring very high computation cost. To efficiently handle the top-k query processing, we propose to rewrite the ranking function in Eqn. 3.3 and apply existing aggregation methods. In particular, we have

$$
\phi(u, a) = \alpha \cdot \phi_s(u, a) + (1 - \alpha) \cdot \phi_d(u, a)
$$

(3.4)

$$
= \sum_{w \in T} \left[ \alpha \cdot \text{rel}(u, w) + \frac{1 - \alpha}{m} \sum_{d \in W_u} \text{rel}(d, w) \right] \cdot \text{rel}(a, w)
$$

(3.5)

where $Q_u(w)$ is the aggregated relevance between user $u$ and topic $w$ and is set to $\alpha \cdot \text{rel}(u, w) + \frac{1 - \alpha}{m} \sum_{d \in W_u} \text{rel}(d, w)$. Now our ranking function $\phi(u, a)$ becomes an aggregation function among the partial relevance in each topic dimension. It consists of two terms $Q_u(w)$ and $\text{rel}(a, w)$. $\text{rel}(a, w)$
is independent of the dynamic context and can be computed and sorted offline. On the
other hand, when the query user $u$ is determined, $Q_u(w)$ becomes a constant and will
not affect the order of $\text{rel}(a, w)$. Therefore, we can maintain $|T|$ inverted lists for each
user, each sorted by $\text{rel}(a, w)$. When a read operation is triggered, we can retrieve the
sorted lists and directly apply standard top-$k$ aggregation techniques such as Threshold
Algorithm (TA) [42]. It consists of two main steps

- Perform a sorted access in parallel to each of the $|T|$ sorted lists. For each document
  accessed, perform a random access to other topics and compute the aggregated score
  of $\phi(u, a)$. If the computed aggregated score is one of the $k$ highest we have seen so
  far, remember the ad and its score.

- For each list $L_i$, let $\text{high}[i]$ be the score of the last ad seen under sorted access. Define
  the threshold value $B_k$ to be the aggregated score of $\text{high}[i]$ by the aggregation
  function $\phi(u, a)$. As soon as at least $k$ ads have been seen whose score is at least
equal to $B_k$, the algorithm terminates.

Example 3. Let the window size $m = 3$, the weighting parameter $\alpha = 0.25$ and the
number of topics $|T| = 2$. Given a user $u$, let $H_u = (0.4, 0.6)$ be the topic distributions of
his/her static interests. Suppose the topic distributions of the three posts in the window are
$(0.2, 0.8), (0.1, 0.9)$ and $(1.0, 0)$ respectively. When $u$ triggers a read operation, the context-
aware query vector $Q_u$ is calculated as $Q_u = 0.25 \cdot (0.4, 0.6) + \frac{1 - 0.25}{3} [(0.2, 0.8) + (0.1, 0.9) +
(1.0, 0)] = (0.55, 0.45) = (0.425, 0.575)$. Suppose $Q_u$ is used to query an ad database
with four tuples $\{a_1 = (0.3, 0.9), a_2 = (0.4, 0.7), a_3 = (0.5, 0.8) and a_4 = (1.0, 0)\}$. To
support top-$k$ aggregation, we pre-compute two inverted lists $l_{w_1}$ and $l_{w_2}$ for the topics and
get $l_{w_1} = \{(a_4, 1.0), (a_3, 0.5), (a_2, 0.4), (a_1, 0.3)\}$ and $l_{w_2} = \{(a_1, 0.9), (a_3, 0.8), (a_2, 0.7),
(a_1, 0.0)\}$. By calling the TA algorithm presented above, $a_3$ will be returned as the most
relevant ad if $k$ is set to 1.
3.3 Safe Region Algorithm

In a social network, the frequency of the read operations is normally much higher than the write operations. The famous 1-9-90 rule of Internet culture states that 90% of the participants of a community only view contents while the rest will edit(9%) or create(1%). Hence, for users who frequently login to check news updates from friends, the online retrieval algorithm in Section 3.2 is not an appropriate choice. This is because the context may vary little in such a short period and it is a waste of CPU resources to repeatedly retrieve the same set of ads.

To tackle the issue, we propose a safe region algorithm that examines whether the top-$k$ ads have changed since the previous user read requests. This can be done efficiently by maintaining a safe region for each user. As long as the new context-aware query vector triggered by a user read operation is still located in the safe region, the top-$k$ ads can be directly presented to the user. Otherwise, we re-compute the new top-$k$ results and update the safe region.

3.3.1 Safe Region Construction

In Eqn. 3.3, the ranking function aggregates the relevance of static user profile and dynamic sliding window in the news feed. The window contains $w$ recent posts and can be represented in the form of a topic vector. When a new post is disseminated to a user, the oldest post in the window expires. The weight of the new window for each topic varies mildly and the top-$k$ relevant ads may remain the same. Thus, by constructing a rectangle in the high-dimensional topic space such that whenever the topic vector of the new window is still located in the rectangle, the top-$k$ ads for the user will not change. We call the high-dimensional rectangle a safe region, denoted by $S = (Q_u^{lb}, Q_u^{ub})$, 

37
Algorithm 4: GSR(User $u$)

1. $R \leftarrow$ Use TA to compute the relevant ads against $Q_u$
2. $Q_{lb}^u \leftarrow Q_u, Q_{ub}^u \leftarrow Q_u$
3. while True do
4. \hspace{1em} $w \leftarrow \text{DimensionSelect}(v)$
5. \hspace{1em} $\vec{\delta} \leftarrow \frac{1-\alpha}{m} \vec{e}_w$
6. \hspace{1em} for $a \in R$ do
7. \hspace{2em} $\phi(a) = \text{MinS}(a, Q_{lb}^u - \vec{\delta}, Q_{ub}^u + \vec{\delta})$
8. \hspace{1em} $S_u \leftarrow \min\{\phi(a) | a \in R\}$
9. \hspace{1em} for $a \in A \setminus R$ do
10. \hspace{2em} $\phi(a) = \text{MaxS}(a, Q_{lb}^u - \vec{\delta}, Q_{ub}^u + \vec{\delta})$
11. \hspace{1em} $S_l \leftarrow \max\{\phi(a) | a \in A \setminus R\}$
12. \hspace{1em} if $S_u \geq S_l$ then
13. \hspace{2em} $Q_{lb}^u \leftarrow Q_{lb}^u - \vec{\delta}$
14. \hspace{2em} $Q_{ub}^u \leftarrow Q_{ub}^u + \vec{\delta}$
15. \hspace{1em} else
16. \hspace{2em} return $(Q_{lb}^u, Q_{ub}^u)$

where $Q_{lb}^u$ stores the lower bound of coordinates in all the dimensions and $Q_{ub}^u$ stores the upper bound.

In [113], Zhang et al. proposed GIR to compute the maximal safe region such that the topic vector update within the region incurs no change for the current top-k results. However, it is prohibitively expensive to construct the optimal safe region, especially in high dimensional topic space. The method cannot meet the real time requirement in the social streaming environment. Thus, we propose a Greedy Safe Region (GSR) algorithm to incrementally build a safe region. As illustrated in Algorithm 4, we first store the top-k results for the current news feed window in $R$ and initialize the safe region to be the context aware query vector $Q_u$ (lines 1-2). In the following iterations, we pick the most promising topic/dimension to expand the current safe region (line 3). For each dimension, we first calculate the distance from $Q_u$ to the boundaries of the current safe
region \((Q^l_u, Q^u_u)\) in that dimension. Then, we select the dimension with the minimum distance for safe region expansion.

For the selected dimension \(w\), we examine whether it is safe to expand upwards and downwards by an expansion unit \(\vec{\delta} = 1 - \alpha_m\), since \(1 - \alpha_m\) is the maximum possible change in \(Q_u(w)\) for each new post. For the expanded safe region, if its minimum relevance to the current top-\(k\) ads, denoted by \(S_l\), is still larger than the maximum relevance to those not in \(R\), denoted by \(S_u\), then the expansion is safe. Otherwise, the algorithm terminates and returns the safe region expanded in partial dimensions.

**Theorem 1.** For a query vector \(Q_u\) with its bound vectors \(Q^l_u\) and \(Q^u_u\) returned by Algorithm 4, whenever \(Q^l_u(w) \geq x(w) \geq Q^u_u(w) \forall w \in T\), it corresponds to the same set of top-\(k\) ads as \(Q_u\).

The proof is trivial as the GSR algorithm maintains the invariant that, for any \(x\) s.t. \(Q^l_u(w) \geq x(w) \geq Q^u_u(w) \forall w \in T\), \(S_u \geq S_l\) where \(S_u \leftarrow \min\{\phi(a)|a \in R\}\) and \(S_l \leftarrow \max\{\phi(a)|a \in A \setminus R\}\). Note \(R\) is the current top-\(k\) ad set for \(Q_u\).

### 3.3.2 Computing MinS and MaxS

To obtain the values of \(S_u\) and \(S_l\), we are required to evaluate the minimum and maximum relevance score between an ad and a safe region, denoted by \(\text{MinS}\) and \(\text{MaxS}\) respectively. We can formulate such a problem as the following:

\[
\min / \max \sum_{w \in T} \text{rel}(a, w) \cdot \frac{x(w)}{\|x\|} \quad \text{s.t.} \quad Q^l_u(w) \leq x(w) \leq Q^u_u(w) \forall w \in T
\]

(3.6)

Note that in Eqn. 3.6 we divide the objective function by \(\|x\|\). A success safe region expansion requires, for any query \(x \in (Q^l_u, Q^u_u)\), \(x \cdot a_u \geq x \cdot a_l \forall a_u \in R\) and \(\forall a_l \in A \setminus R\).
This means the norm of \( x \) should not be taken into account of the relevance score between an ad and the safe region. If the normalization is not applied, MinS would choose \( Q_{ub}^{lb} \) and MaxS would choose \( Q_{ub}^{ub} \) as solutions of Eqn. 3.6. Since \( \| Q_{ub}^{lb} \| < \| Q_{ub}^{ub} \| \), it incurs an underestimation of MinS and an overestimation of MaxS, resulting in a much smaller safe region.

Eqn. 3.6 is essentially an optimization problem of finding two vectors in the rectangular area defined by the safe region bound vectors \( (Q_{ub}^{lb}, Q_{ub}^{ub}) \), which have the minimum and the maximum cosine similarities respectively against an ad vector \( a \). In other words, MinS and MaxS correspond to vectors \( x_{\text{max}} \) and \( x_{\text{min}} \) in the rectangular safe region which have the maximum and the minimum angles respectively to \( a \). However, it is inefficient to find the exact solution due to the nonlinear term in the objective function. To solve the issue, we propose to use a sphere that encloses the safe region constructed so far. Based on the sphere, we calculate MinS and MaxS to determine a termination of the GSR algorithm or further expansion of the current safe region.

Let \( x_c \) be the vector that passes the origin and the center of the rectangular safe region, i.e. \( x_c = \frac{1}{2} (Q_{ub}^{lb} + Q_{ub}^{ub}) \). We apply a minimum sphere to enclose the safe region and replace \( x_{\text{min}} \) and \( x_{\text{max}} \) to be the minimum and maximum angles to the bounding sphere. The new angles from an ad \( a \) to \( x_{\text{min}} \) and \( x_{\text{max}} \) for the sphere can be computed as:

\[
\theta(a, x_{\text{min}}) = \max\{\theta(a, x_c) - \arcsin\left(\frac{r}{\|x_c\|}\right), 0\} \tag{3.7}
\]

\[
\theta(a, x_{\text{max}}) = \theta(a, x_c) + \arcsin\left(\frac{r}{\|x_c\|}\right) \tag{3.8}
\]

where \( \theta(\ldots) \) denotes the angle between two vectors and \( r \) is the radius of the spherical safe region, i.e \( r = \| \frac{1}{2} (Q_{ub}^{ub} - Q_{ub}^{lb}) \| \).

Fig. 3.2 presents illustrative examples of \( x_{\text{min}} \) and \( x_{\text{max}} \) in a 3-dimensional space. In Fig. 3.2 the whole spherical safe region is a ball that lies in the positive quadrant. It
Figure 3.2: Computing MinS and MaxS when the safe region sphere lies in the positive quadrant

is visually intuitive about the min and max angles between $a$ and the sphere region. In this case, MinS and MaxS are directly computed via Eqn 3.7 and Eqn. 3.8 respectively. Compared to using rectangle to derive MinS and MaxS, the computation becomes much more efficient but the constructed safe region may be slightly smaller. This is because the sphere encloses the rectangle and it results in smaller $x_{\text{min}}$ and larger $x_{\text{max}}$, which consequently leads to larger MinS and smaller MaxS. In other words, the value of $S_l$ increases but the value of $S_u$ decreases, which makes the GSR algorithm more likely to terminate.

Fig. 3.3 illustrates a case worthy of our attention. The sphere region is now overlapped with x-y plane. In this case, we can still calculate $x_{\text{min}}$ using Eqn. 3.7 because the ad $a$ and query vector $Q_u$, which is the center of the safe region, are positive vectors and $x_{\text{min}}$ is guaranteed to lie on top of the x-y plane. To calculate $x_{\text{max}}$, as the safe region sphere overlaps with x-y plane, we need to determine if $x_{\text{max}}$ still lies on the surface of the sphere
Figure 3.3: Computing \textbf{MinS} and \textbf{MaxS} when the safe region sphere overlaps with x-y plane or the intersected area between the sphere and $x - y$ plane, which are shown in Fig. 3.3.

The following theorem shows how to determine the location of $x_{\text{max}}$ for an ad $a$.

\textbf{Theorem 2.} For an ad vector $a$, let $x^*_{\text{max}}$ be the vector obtained by directly applying \textbf{MinS} on the spherical safe region. Let $I$ be an index set such that $I = \{i | x^*_{\text{max}}(i) < 0\}$ and $S(i)$ be the region where the sphere intersects with the plane $x(i) = 0$ $\forall i \in I$. If $I$ is an empty set, $x_{\text{max}}$ can be calculated by Eqn 3.8. Otherwise, $x_{\text{max}}$ is obtained by:

$$x_{\text{max}} = \arg\max_q \{\theta(a, q)| q \in S(i) \ \forall i \in I\}$$ \hfill (3.9)

\footnote{For any point $q$, we also use $q$ to represent the vector which passes through the origin and $q$ as a point when there is no ambiguity, e.g. in $\theta(a, q)$, $q$ means a vector whereas in $q \in S(i)$, $q$ means a point.}
Proof. For the ease of presentation, we prove Theorem 2 in 3D space and the proof can be generalized to any finite dimensional space. Given the spherical safe region $B$ and the center of $B$, i.e. $x_c$, as shown in Figure 3.4, we draw a plane $P$ which is normal to $x_c$ and passes through $x_{\text{max}}^*$ (Here $x_{\text{max}}^*$ is used as the contact point between the sphere and the vector $x_{\text{max}}^*$). $a$ is the point where the ad vector intersects with $P$ and $R$ is the intersecting circle region between $B$ and $P$. It is easy to see that the line from $a$ to $x_{\text{max}}^*$ passes through the center of the intersecting circle plane between $B$ and $P$, i.e. $x_c'$. This means any point $q$ on the boundary of $R$ will have shorter distance to $a$ than that of $x_{\text{max}}^*$. 

Figure 3.4: The location of $x_{\text{max}}$ for an ad against a safe region.
Moreover, since $x_{max}^*$ has a negative coordinate, we can find a line segment $cd$, which is the intersection between $P$ and the boundary region $S_i$, that separates $a$ and $x_{max}^*$ on both sides of cd. For any point $q$ on the boundary of $R$, $\alpha$ is the angle between $a, x_c$ and $q, x_c$. Then the distance from $a$ to $q$ can be expressed as:

$$\xi(a, q) = \xi^2(a, x_c') + \xi^2(q, x_c') - 2\xi(a, x_c')\xi(q, x_c')\cos\alpha$$

where $\xi(.,.)$ denotes the distance between two points. Therefore $\xi(a, q)$ is a continuous unimodal function w.r.t $\alpha \in [0, 2\pi)$. This means $\max\{\xi(a, c), \xi(a, d)\}$ is larger than any point $q$ which is on the boundary of $R$ and lies on the same side with $a$ w.r.t line cd. This in term means $\theta(q, a) < \max\{\theta(a, c), \theta(a, d)\}$ because all points on $R$ have the same distance to the origin and $\theta(q, a)$ is proportional to $\xi(q, a)$.

With the above proof, we have shown that, $\max\{\theta(a, c), \theta(a, d)\}$ is the maximum possible angle within the region $R$. However we have not shown for all points on $B$ that such condition holds. Note that since $R$ contains the point $x_{max}^*$ which is a contact point between $B$ and $B$’s minimum bounding convex cone. Then all vectors from the origin to any points on $B$ will pass through $R$. Thus we can conclude that $\max\{\theta(a, c), \theta(a, d)\}$ is the maximum possible angle among all points on $B$ and prove Theorem 2.

Theorem 2 tells us that there are two cases when computing $x_{max}$. In the first case, if $x_{max}$ lies in the positive quadrant, we compute $x_{max}$ by Eqn. 3.8. In the second case, when $x_{max}$ goes beyond the positive quadrant, $x_{max}$ must lie in the intersected area between the sphere and boundaries of the positive quadrant. To obtain the exact location of $x_{max}$ for the second case, we project the ad $a$ onto the boundaries of the positive quadrant, where the boundaries contain a piece of the safe region sphere. It can be easily proven that $x_{max}$ is exactly the point which has the maximum angle away from $a$’s projection. Fig. 3.5 shows an example of identifying $x_{max}$. We plotted the intersection area $S$ between x-y
plane and the safe region sphere. \( a' \) is the projection of \( a \) onto \( x\)-\( y \) plane and we identify \( x_{\text{max}} \) from \( S \) as the point which has the maximum angle away from \( a' \) as shown in Fig. 3.5.

![Diagram](image)

Figure 3.5: Example of computing \( x_{\text{max}} \) when the safe region sphere overlaps with \( x\)-\( y \) plane.

### 3.3.3 Safe Region Based Query Processing

Having discussed how the safe region can be constructed, we are now ready to present how to process incoming queries triggered by user read operations. We have proved in Theorem 1 that for any query vector \( x \) within the safe region formed by \((Q_{lb}^u, Q_{ub}^u)\), the top-k results will be the same for all \( x \). A naive query processing technique for a query \( Q \) is to simply check if \( Q \geq Q_{lb}^u \land Q \leq Q_{ub}^u \). However such a checking rule is too strict and cannot handle the case where the safe region bound vectors and the query vector are not in the same scale. For example, there are two query vectors \( Q = (0.3, 0.5) \) and \( Q^* = (0.15, 0.25) \). Both vectors will have the same top-\( k \) ads since \( Q = 2 \cdot Q^* \) and the results are invariant under scalar multiplication of the query vector. Let a safe region be
defined by $Q_u^{lb} = (0.1, 0.2), Q_u^{ub} = (0.2, 0.4)$, $Q$ is not bounded by $(Q_u^{lb}, Q_u^{ub})$. However it is easy to see that $Q$ can indeed be processed by the safe region since $Q$ and $Q^*$ share the same result and $Q^*$ is bounded by $(Q_u^{lb}, Q_u^{ub})$. If the naive checking rule is adopted, a large number of re-computations for new safe regions are needed. Thus, we assign a more flexible checking rule for query processing with the safe region.

**Lemma 1.** For any query vector $Q_u$ and a safe region formed by $(Q_u^{lb}, Q_u^{ub})$, if $Q_u$ intersects the bounding sphere of the safe region, then $Q_u$ will also be in the safe region.

It is straightforward to prove Lemma 1 according to Theorem 1, as the scalar multiplication of the query vector does not change the query results. To efficiently check if a query vector $Q_u$ intersects with a sphere, we use Eqn. 3.7 where the ad vector $a$ is replaced with $Q_u$. Such checking has a worst time complexity of $O(|T|)$ since we need to compute the angle between two vectors with at most $|T|$ dimensions.

### 3.3.4 Optimizations

To further improve the performance of the safe region method, we propose two optimization techniques. The first seeks to efficiently evaluate $S_l$ and $S_u$ in each iteration of GSR algorithm. The second aims to avoid the online retrieval and safe region re-construction cost when a query vector $Q_u$ no longer exists in its original safe region.

In Algorithm 4, we evaluate $S_l$ by computing MinS for $k$ times and $S_u$ by computing MaxS $|A| - k$ times. This means evaluating $S_l$ requires almost a scan of all ads, which is too computationally expensive to evaluate $S_l$ for a large ad database. This motivates us to develop an efficient algorithm to compute MaxS only when necessary.

We design a TA-like approach to evaluate $S_l$. $S_l$ is the score of the top-1 ad in $|A \setminus R|$, which has the highest MaxS score against a safe region. As the ads have been sorted w.r.t each topic by the ads’ topic relevance scores, i.e $\text{rel}(a, w)$, we visit the ads in descending
order of \( \text{rel}(a, w) \) in topic \( w \)'s inverted list and perform random access to other topics' inverted lists for computing \( \text{MaxS} \). Meanwhile, we maintain an upper bound score \( b_w \) for the inverted list of each topic \( w \) and the maximum \( \text{MaxS} \) score of unvisited ads can be bounded by computing \( \text{MaxS} \) for \( b = (b_1, \ldots, b_{|T|}) \) against the safe region. If the top-1 ad, which has the highest \( \text{MaxS} \) score among all visited ads, has larger \( \text{MaxS} \) score than that of \( b \), we can terminate and return \( S_l \). Such optimization will greatly reduce the number of \( \text{MaxS} \) computations in Lines 6-8 of Algorithm 1.

![Figure 3.6: Example of computing if a SSR contains a query vector](image)

When the dynamic query vector \( Q_u \) deviates out of the safe region of user \( u \), we need to adopt the \textit{online retrieval} to obtain top-\( k \) ads on the fly and construct a new safe region in the meanwhile. Such computations are rather expensive. Thus for our second optimization, we propose a new idea to “salvage” the maintained safe regions as well as the associated top-\( k \) ads from other users. This is because all the query vectors in a safe region share the same top-\( k \) ads. When \( Q_u \) moves out of the safe region, we can search all

47
the safe regions of other users. If we can find a safe region from user $v$ that contains the new query vector $Q_u$ of user $u$, its top-$k$ ads are exactly the same as user $u$. Moreover, we can assign the safe region of $v$ directly to user $u$. In this way, the cost of online retrieval and safe region computation can be saved.

To quickly identify whether the new query vector $Q_u$ is contained in the safe regions of other users, we transform the problem into a standard range query in high dimensional space. As shown in Fig. 3.6, $B$ is a safe region centered at $x_c$ with radius $r$ and $Q$ is a query vector. Our original processing technique is to check if $Q$ intersects with $B$. By mapping $B$ to $B'$ with new center $x'_c$ and scaling $Q$ to $Q'$ where $x'_c$ and $Q'$ both lie on the boundary of the unit sphere, we transform the vector-sphere intersection problem into checking if the distance from $Q'$ to $x'_c$ is smaller than $r'$. This is because the minimum bounding convex cone is the same for both $B$ and $B'$. After the transformation, our goal is to find the MBRs whose distance to a query point is smaller than the radius. The new problem can be efficiently solved by high-dimensional indexes such as R-tree [54], k-d-tree [13] or iDistance [60]. R-tree is adopted in our actual implementation.

### 3.4 Hybrid Algorithm

In this section, we propose a hybrid model to combine the merits of online retrieval and safe region. Our strategy is to measure the dynamism of topic distributions in the streaming news feed of each user. If the topic distributions in a news feed vary dramatically as new posts flood in, we adopt the online retrieval method to avoid the cost of maintaining safe regions that update frequently. Otherwise, the topic distributions are relatively stable and the safe region method is suitable for the scenario.
3.4.1 Variance of Topic Distributions

To see if a user $u$ is suitable for deploying eager update approach, we need to analyze how topics change in the sliding window of $u$’s news feed. When the frequency of a topic to appear in the sliding window changes aggressively, it is almost impossible that the top-k result stays the same. Moreover, we are interested in the topic with the most aggressive dynamics. To measure the variance of topic distributions, we use i.i.d Poisson process $P_u$ of rate $\lambda_u$ to model the generation of new posts in a user’s news feed as it is frequently used to model the arrival of events. We assume that the number of topics in each post $d$ from user $u$ follows the discrete uniform distribution $F_u$ with range $\{1, 2, ..., f_u\}$. The topics in $d$ are then sampled via a multinomial distribution and each topic is selected with probability $p_{w,u}$. Let $D_{w,u}$ denote the total weightage of topic $w$ in a post $d$ from user $u$. The whole generative process to build a post for a user $u$ is summarized as follows:

- Draw a posting time from Poisson process $P_u \sim \text{Poisson}(\lambda_u)$

- Draw the number of topics for a post $F_u \sim \text{Uniform}(1, 2, ..., f_u)$

- Draw topics $D_{w,u}|F_u \sim \text{Multinomial}(p_{w,u})$

For $P_u$, we can estimate the posting rate $\lambda_u$ by $u$’s historical posting times. For the parameters of $F_u$ and $D_{w,u}$, we can estimate by analyzing all the topic vectors of $u$’s posts.

Let $X_{w,v}$ be the random variable describing the weightage of topic $w$ appears in a user $v$’s sliding window. Given the aforementioned generative process, $X_{w,v}$ can be written as:

$$X_{w,v} = \sum_{n \in N(v)} \sum_{1 \leq i \leq M_{v,n}} D_{w,n}(F_n)$$ (3.10)
Where $N(v)$ is all $v$' neighbours and $M_{v,n}$ is a random variable describing how many posts are selected from a neighbour $n$ to form the news feed window of $m$ posts for user $v$. Then, based on Eqn. 3.10, the variance of a topic $w$ in a user $v$'s news feed can be defined as:

$$\text{Var}[X_{w,v}] = \text{Var}\left[ \sum_{n \in N(v)} \sum_{1 \leq i \leq M_{v,n}} D_{w,n}(F_n) \right]$$ (3.11)

which can be further expanded to

$$\text{Var}[X_{w,v}] = \sum_{n \in N(v)} \frac{(f_n + 1)^2 p_{w,n}^2}{4} m \lambda_{v,n}(1 - \lambda_{v,n}) - \sum \sum_{a,b \in N(v)} m \lambda_{v,a} \lambda_{v,b} \frac{(f_a + 1)(f_b + 1) p_{w,a} p_{w,b}}{4}$$ (3.12)

**Derivation of Eqn. 3.11** 

Var$[X_{w,v}]$ is defined in Eqn. 3.11 as:

$$\text{Var}[X_{w,v}] = \text{Var}\left[ \sum_{n \in N(v)} \sum_{1 \leq i \leq M_{v,n}} D_{w,n}(F_n) \right]$$

By the definition of variance, we obtain the following expansion of Var$[X_{w,v}]$:

$$\text{Var}[X_{w,v}] = \sum_{n \in N(v)} \text{Var}\left[ \sum_{1 \leq i \leq M_{v,n}} D_{w,n}(F_n) \right]$$ (3.13)

$$+ \sum \sum_{a,b \in N(v)} \text{Cov}\left[ \sum_{1 \leq i \leq M_{v,a}} D_{w,a}(F_a), \sum_{1 \leq i \leq M_{v,b}} D_{w,b}(F_b) \right]$$ (3.14)
We compute Eqn. 3.13 and 3.14 separately. For a user \( v \) and a topic \( w \), \( \text{Var}\left[\sum_{1 \leq i \leq M_{v,n}} D_{w,n}(F_n)\right] \) can be expressed as:

\[
\text{Var}\left[\sum_{1 \leq i \leq M_{v,n}} D_{w,n}(F_n)\right]
= \mathbb{E}\left[\left( \sum_{1 \leq i \leq M_{v,n}} D_{w,n}(F_n) \right)^2 \right] - \mathbb{E}\left[ \sum_{1 \leq i \leq M_{v,n}} D_{w,n}(F_n) \right]^2
= \mathbb{E}\left[\mathbb{E}\left[\left( \sum_{1 \leq i \leq M_{v,n}} D_{w,n}(F_n) \right)^2 | M_{v,n} \right] - \mathbb{E}\left[ \sum_{1 \leq i \leq M_{v,n}} D_{w,n}(F_n) | M_{v,n} \right]^2 \right]
\]

Since we know that \( M_{v,n} \) is independent of \( D_{w,n}(F_n) \) and each \( D_{w,n}(F_n) \) is independent of each other:

\[
\mathbb{E}\left[\mathbb{E}\left[\left( \sum_{1 \leq i \leq M_{v,n}} D_{w,n}(F_n) \right)^2 | M_{v,n} \right] - \mathbb{E}\left[ \sum_{1 \leq i \leq M_{v,n}} D_{w,n}(F_n) | M_{v,n} \right]^2 \right]
= \mathbb{E}\left[M_{v,n}^2 \mathbb{E}[D_{w,n}(F_n)]^2 | M_{v,n} \right]
= \mathbb{E}[D_{w,n}(F_n)]^2 \mathbb{E}[M_{v,n}^2 | M_{v,n}] = \mathbb{E}[D_{w,n}(F_n)]^2 \mathbb{E}[M_{v,n}^2]
\]

and in a similar way:

\[
\mathbb{E}\left[\sum_{1 \leq i \leq M_{v,n}} D_{w,n}(F_n) | M_{v,n} \right]^2 = \mathbb{E}[D_{w,n}(F_n)]^2 \mathbb{E}[M_{v,n}^2] \]

(3.15)

now we have:

\[
\text{Var}\left[\sum_{1 \leq i \leq M_{v,n}} D_{w,n}(F_n)\right] = \mathbb{E}[D_{w,n}(F_n)]^2 \text{Var}[M_{v,n}]
\]

Since \( \text{Var}[M_{v,n}] = m\lambda_{w,n}(1 - \lambda_{w,n}) \), we only need to derive the unknown term \( \mathbb{E}[D_{w,n}(F_n)] \).

\[
\mathbb{E}[D_{w,n}(F_n)] = \mathbb{E}[\mathbb{E}[D_{w,n}(F_n)|F_n]] = \sum_{f=1}^{f_n} \frac{f \cdot p_{w,n}}{f_n} = \frac{(f_n + 1)p_{w,n}}{2}
\]

51
Then, for any $n$ and $w$, we are able to evaluate:

$$
\text{Var}\left[ \sum_{1 \leq i \leq M_{v,n}} D_{w,n}(F_n) \right] = \frac{(f_n + 1)^2 p_{w,n}^2 m \lambda_{v,n} (1 - \lambda_{v,n})}{4} \tag{3.16}
$$

Next we derive $\text{Cov}\left[ \sum_{1 \leq i \leq M_{v,a}} D_{w,a}(F_a), \sum_{1 \leq i \leq M_{v,b}} D_{w,b}(F_b) \right]$ for any $a, b \in N(v)$ and $a \neq b$. Let $A = \sum_{1 \leq i \leq M_{v,a}} D_{w,a}(F_a)$ and $B = \sum_{1 \leq i \leq M_{v,b}} D_{w,b}(F_b)$. It follows that:

$$
\text{Cov}\left[ \sum_{1 \leq i \leq M_{v,a}} D_{w,a}(F_a), \sum_{1 \leq i \leq M_{v,b}} D_{w,b}(F_b) \right] = \mathbb{E}[AB] - \mathbb{E}[A] \mathbb{E}[B]
$$

From Eqn. 3.15 we can get $\mathbb{E}[A] = \mathbb{E}[D_{w,a}(F_a)] \mathbb{E}[M_{v,a}]$ and $\mathbb{E}[B] = \mathbb{E}[D_{w,b}(F_b)] \mathbb{E}[M_{v,b}]$.

The only left part is $\mathbb{E}[AB]$ which can be derived as the follows:

$$
\mathbb{E}[AB] = \mathbb{E}[AB|M_{v,a}, M_{v,b}]
$$

$$
= \mathbb{E}\left[ \sum_{1 \leq i \leq M_{v,a}} D_{w,a}(F_a) \cdot \sum_{1 \leq i \leq M_{v,b}} D_{w,b}(F_b) | M_{v,a}, M_{v,b} \right]
$$

$$
= \mathbb{E}[D_{w,a}(F_a)] \mathbb{E}[D_{w,b}(F_b)] \mathbb{E}[M_{v,a} M_{v,b}]
$$

Then it is natural to have:

$$
\text{Cov}[A, B] = \mathbb{E}[D_{w,a}(F_a)] \mathbb{E}[D_{w,b}(F_b)] \text{Cov}[M_{v,a}, M_{v,b}]
$$

$$
= - \frac{(f_a + 1)(f_b + 1)p_{w,a}p_{w,b} m \lambda_{v,a} \lambda_{v,b}}{4} \tag{3.17}
$$

By combining the above results from Eqn. 3.16 and 3.17 we can derive $\text{Var}[X_{w,v}]$.

Here, $\lambda_{v,n}$ measures the probability of selecting a post from a neighbor $n$ for user $v$. Since all the neighbours of $v$ compose posts that follow i.i.d Possion processes and it
has been shown in [39] that the sum of i.i.d Poisson distribution follows a multinomial
distribution, we have $\lambda_{v,n} = \lambda_n \sum_{n' \in N(v)} \lambda_{n'}$.

To calculate $\text{Var}[X_{w,v}]$ according to Eqn. 3.12 we need to traverse all the pairs of
neighbours $(a, b)$ for each user $v$. Suppose the average node degree in a social network is $z$, the computation complexity is $O(z^2 |V|)$, which is very high for dense social graphs. To reduce the computational cost, we rewrite the term in Eqn. 3.12 as:

$$
\frac{1}{4} \sum_{a,b \in N(v)} \sum_{a \neq b} m \lambda_{v,a} \lambda_{v,b} (f_a + 1)(f_b + 1)p_{w,a}p_{w,b}
$$

$$
= \frac{1}{4} \sum_{a \in N(v)} m \lambda_{v,a} (f_a + 1)p_{w,a} \sum_{b \in N(v)} \lambda_{v,b} (f_b + 1)p_{w,b}
$$

$$
= \frac{1}{4} \sum_{a \in N(v)} m \lambda_{v,a} (f_a + 1)p_{w,a} \left( \sum_{b \in N(v)} \lambda_{v,b} (f_b + 1)p_{w,b} \right)
$$

$$
- \frac{1}{4} \sum_{a \in N(v)} m \lambda_{v,a}^2 (f_a + 1)^2 p_{w,a}^2
$$

In this way, we can pre-compute $\sum_{b \in N(v)} \lambda_{v,b} (f_b + 1)p_{w,b}$ for each user $v$ and the complexity is reduced to $O(z|V|)$.

### 3.4.2 Hybrid Retrieval Strategy

$\text{Var}[X_{w,v}]$ only captures the variance of topic distributions in the news feed. We need to further combine it with the static personal interests to measure its impact in choosing an appropriate retrieval strategy. By applying coefficient of variation on the linear combination of static interests and dynamic topic distributions in the news feed, we have

$$
\rho(v) = \max_{w \in T} \frac{1 - \alpha}{m} \sqrt{\text{Var}[X_{w,v}]} + \frac{1 - \alpha}{m} \cdot \text{rel}(u, w) \cdot \text{E}[X_{w,v}] 
$$

(3.18)
In addition, the ratio of the read frequency of user $v$ to the write frequency of $v$’s neighbors will also affect the retrieval strategy selection and is ignored in the above model, which only considers the variance of topic distributions for a sequence of write operations. To bridge the gap, we extend $\rho(v)$ by taking the read frequency $\eta_v$ of $v$ into account.

$$\rho^*(v) = \sum_{n \in N(v)} \frac{\lambda_n}{\eta_v} \cdot \rho(v)$$ (3.19)

Finally, we can use $\rho^*(v)$ to determine the retrieval strategy for user $v$. If $\rho^*(v)$ is smaller than a pre-defined threshold $\rho_{\text{max}}$, we adopt the safe region strategy for user $v$. Otherwise online retrieval is used when $v$ logsins/refreshes its personal social page.

### 3.5 Experimental Study

In this section, we study the performance of the three proposed methods on real social network datasets with billions of edges. In the following experiments, we focus on evaluating the efficiency of the proposed methods. The effectiveness evaluation is beyond the scope of the chapter because we simply adopt the previous topical mining and relevance measurement techniques which have already been shown to be effective [8, 73]. In addition, the idea of considering newsfeed as a dynamic context was also supported by a recent work from Twitter [73].

We use Online to denote the online retrieval method presented in Section 3.2 and use GSR and Hybrid to denote the methods proposed in Sections 3.3 and 3.4 respectively. All the methods are implemented with C++ and run in memory on a CentOS server (Intel i7-3820 3.6GHz CPU with 8 cores and 60GB RAM).

**Advertisement Datasets.** We use Amazon products [70] and AOL keyword queries\(^2\) as two representative ad repositories. The Amazon dataset consists of 548,552 products.

\(^2\)http://www.gregsadetsky.com/aol-data/
associated with their metadata and review information; whereas in the AOL dataset, there are over 7 million keyword queries. We then apply existing topic modeling method \cite{15} upon the products and keyword queries, resulting in ads in the form of probabilistic topic distributions with fixed number of dimensions.

**Social Network Datasets.** We use two real datasets, *Twitter* and *News*, from SNAP\footnote{http://snap.stanford.edu/} to evaluate the performance in different structures of social networks. The *Twitter* dataset contains 41.6 million users and 476 million tweets; the *News* dataset contains 1.42 million websites extracted from a collection of 96 million online articles. For the News dataset, the vertex in the graph denotes a website whereas the edge means that there is a link from one website to another. To test the scalability with increasing graph sizes, we sampled 10M, 20M, 30M, 40M nodes for the *Twitter* dataset and 0.2M, 0.6M, 1M, 1.4M nodes for the News dataset.

To simulate a social network with high-speed news feeding, we extract 1 million articles or tweets to generate a sequence of write operations and the order is determined by the timestamp associated with each article or tweet. Each write operation consists of an article or tweet and the associated website or user in the network. The remaining items are used to model static personal interests. We simply apply LDA to the remaining articles or tweets of each user to construct the topic distributions. The input stream to the system consists a sequence of read and write operations. The user who triggers the read operation can be any node in the network. When a write operation arrives, we simply analyze the post and store it in the database. When a read operation arrives, we call the proposed methods to return top-\textit{k} ads. Since our target is to guarantee the real-time delivery of relevant ads, we are interested to measure the average elapsed time in retrieving the top-\textit{k} ads for each read operation in the following experiments.
Table 3.2: All parameter settings used in the experiments of Chapter 3

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Twitter dataset</th>
<th>News dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Users</td>
<td>10, 20, 30, \textbf{40} (M)</td>
<td>0.2, 0.6, 1, \textbf{1.4} (M)</td>
</tr>
<tr>
<td>#Edges</td>
<td>0.7, 1.1, 1.2, \textbf{1.3} (B)</td>
<td>1.0, 1.9, 2.6, \textbf{3.1} (M)</td>
</tr>
<tr>
<td>AvgDegree</td>
<td>76.4, 56.8, 46.1, \textbf{38.9}</td>
<td>5.2, 3.1, 2.6, \textbf{2.2}</td>
</tr>
<tr>
<td>(\alpha)</td>
<td>0.1, 0.3, \textbf{0.5}, 0.7, 0.9</td>
<td></td>
</tr>
<tr>
<td>(k)</td>
<td>1, 2, \textbf{3}, 4, 5</td>
<td></td>
</tr>
<tr>
<td>R/W</td>
<td>1.0, 2.0, \textbf{4.0}, 8.0, 16.0</td>
<td></td>
</tr>
<tr>
<td>#Topics</td>
<td>5, 10, \textbf{15}, 20, 25</td>
<td></td>
</tr>
</tbody>
</table>

Parameters & Settings. As shown in Table 3.2, we evaluate the scalability w.r.t. increasing \(\alpha\) (the weight of static interests in the ranking function), \(k\) (the number of ads to be embedded in the news feed) and \(|V|\) (the number of users in a social network). In addition, we simulate the user activities in social networks with different read/write ratio \(R/W\). The read operation refers to a user login or refreshing the news feed. The write operation means a user composes, likes or shares a post. We will see that the proposed methods have different biases on this parameter. We also evaluate the number of topics from 5 to 25. For the number of posts in one’s news feed window, we use the default value \((m = 20)\) in Twitter and when a user logins, 20 latest articles/tweets are returned.

3.5.1 Tuning \(\rho\)

We first investigate the effect of threshold parameter \(\rho\) (Equation 3.18) that indicates how dynamic a user’s news feed is in the Hybrid model. If \(\rho\) is large, the news feed of most users are considered as non-dynamic and they will adopt the GSR method for ad recommendation. There may be frequent update of their safe regions, incurring high CPU cost. If \(\rho\) is small, most users will adopt the Online method to retrieve the top-\(k\) ads when a read operation is triggered. CPU resources may be wasted if the read frequency is high but the top-\(k\) ads update infrequently. Therefore there exists a sweet spot for \(\rho\). As shown in Fig. 3.7 we select 0.06, 0.08, 0.03 and 0.02 for News-AOL, News-Amazon,
Twitter-AOL and Twitter-Amazon datasets respectively. Therefore, we will use these optimal setups for the Hybrid method in the remaining experiments of this section.

![Graphs showing time vs. ρ for different datasets](image)

Figure 3.7: Vary ρ

### 3.5.2 Varying α

In our ranking function for top-\(k\) ads retrieval, we consider the relevance to the static user interests and dynamic news feed that are combined linearly by the parameter \(α\). In the first evaluation of the three proposed methods, we examine the performance w.r.t. varying \(α\). As shown in Fig. 3.8, the performance of the Online method is invariant under different \(α\) for all datasets. This is because the Online method always recomputes the top-\(k\) ads whenever there is a read operation. The computation cost remains the same when \(α\) varies.
The GSR method shows superior performance over Online for large $\alpha$. When $\alpha$ increases, the relevance score between an ad and a user is more likely to be dominated by the static user interests and less affected by the dynamic update in the news feed. Hence, the constructed safe region can last longer before its next re-construction. However, when $\alpha$ is very small, the GSR method becomes very sensitive to the news feed update and its performance can degrade to a point that it becomes inferior to the Online method. The arrival of new posts in the news feed incurs frequent re-construction of safe region which is more expensive than retrieving top-k ads in the Online method.

The Hybrid method combines the advantages of the Online and the GSR methods and shows superior performance. It can outperform GSR by up to 30x speedups and outperform Online by up to 11x speedups in our experiments. This is because the hybrid
model can automatically select a retrieval strategy for each user based on our proposed cost model to optimize the performance. It can avoid repetitive retrieval of the same set of ads as in the Online method. It can also avoid frequent safe region re-construction as in the GSR method when the news feed updates at a high speed. Hence, we can see that its performance is not as sensitive to $\alpha$ as the GSR method. For different values of $\alpha$, it can select a suitable retrieval strategy for each user. The experimental results verified the effectiveness of our proposed hybrid model.

We can also see that the ad database derived from the Amazon dataset results in slower performance than that from the AOL dataset. This is because the textual information in the Amazon dataset is more abundant. It contains product descriptions of books, music and movies while the ads in the AOL dataset are simply keyword search queries. The vectors of topic distributions in the AOL dataset is much more sparse with the values of many columns being or close to 0. It leads to an early termination of the TA algorithm to retrieve top-$k$ ads, which is a component in all the three proposed methods.

Finally, as shown in the figure, when $\alpha$ varies, most users can adapt themselves by selecting a proper retrieval strategy. Another interesting observation is that the performance of Hybrid is more stable in the Twitter dataset than in the News dataset. This is because the posts in the News dataset have longer text and cover more topics. After aggregating the topic distributions in the window, the variation in the news feed would be more dramatical.

### 3.5.3 Increasing $k$

When we increase the number of recommended ads, i.e. $k$, it takes longer to perform recommendation for all three methods as shown in Fig. 3.9. First, all the methods need to retrieve top-$k$ relevant ads using the TA algorithm. It is obvious that when $k$ increases,
the $k$-th score becomes larger and it needs to scan more items in the sorted lists until the $k$-th score is smaller than the upper bound of the unvisited items. Second, the effectiveness of a safe region is affected by $k$. Based on our observations on the experiments, when $k$ is large, the $k$-th and $(k+1)$-th item become less distinguishable which makes it more difficult to construct an effective safe region. Nevertheless, **Hybrid** still significantly outperforms the other two methods and its speedup is 10x in the Twitter-Amazon dataset.

### 3.5.4 Vary Read/Write Ratio

In this experiment, we examine the performance and robustness in a dynamic streaming environment with varying read/write ratio. In this experimental setup, we divide the 1 million sampled write operations into 10 blocks, each with 100K write operations. For
each block, we manually control the read/write ratio inside. In particular, we first increase the ratio of each block from 1, 2, 4, 8 to 16 and then decrease afterwards. It means in the first block, we have 100K read operations and 100K write operations. When the ratio is 16, there are 1600K read operations with 100K write operations in one block.

We report the total running time of handling the read operations in Fig. 3.10. The running time for the write operations is the same for all the three methods and thus ignored. Since Online always retrieve the top-k results on the fly for a read operation, its running time drastically increases when there are more read operations. It is interesting to observe that GSR significantly outperforms Online when the read/write ratio is very high, say 8 or 16 in the figure. This is because the context for the next read operation varies little given such a high read/write ratio. The constructed safe region can support
more read operations before its next re-construction. However, the total processing time of GSR still grows with the read frequency. When more read operations are triggered by randomly picked users, it becomes more likely to detect a user whose safe region requires re-construction. The throughput of Hybrid is higher than both Online and GSR and demonstrates higher adaptivity to the dynamic workloads. It periodically updates $\rho^*(v)$ for each user $v$ to track the dynamism of $v$’s news feed and apply a suitable retrieval strategy. Thus, we have seen that the performance of Hybrid is robust against streaming blocks with different read/write ratios.

![Graphs showing varying number of topics](image)

Figure 3.11: Vary Number of Topics
3.5.5 Scalability

In the final set of experiments, we evaluate the scalability of the three proposed methods w.r.t increasing number of topics and graph size.

We increase the number of topics from 5 to 25 and the performance of all methods w.r.t increasing number of topics is presented in Fig. 3.11. It shows that all methods run slower when there are more topics. Higher dimensions in the topic distribution vectors will result in more computation cost in the TA algorithm as well as less effectiveness in the constructed safe regions. Nevertheless, Hybrid remains superior over the other two methods in all experiments and it shows that Hybrid is more capable to handle larger number of topics.

Figure 3.12: Vary Graph Sizes
Lastly, we show the experimental results when varying graph sizes shown in Figure 3.12. Not surprisingly, Online is not affected by larger graphs since the retrieval of ads is independent of graph size. Surprisingly, we found that GSR and Hybrid show better performance when the graph size becomes larger. We interpret the reason as the following: although the social graph is larger, the average degree in a larger graph is actually smaller in our experiments as the graph statistics suggest in Table 3.2. Smaller average degrees mean the news feeds are less dynamic since only the posts written by a neighbour on the social graph will appear in one’s news feed. As safe region based methods are highly dependent on the dynamism of news feeds, it is intuitive to understand that they are more efficient with larger number of nodes in the social graph.

3.6 Summary

In this chapter, we studied the context-aware advertisement recommendation problem for high speed social news feeding. We first formulated a general ranking function of ads against each user in the social network by combing his/her interests and dynamic contents in the news feed. The Online method was first proposed to retrieve a user’s news feed and re-compute the recommended ads based on TA algorithm when there is a read operation triggered. Then the GSR method is developed, which maintains a safe region and only re-computes the recommended ads whenever the safe region is found invalid against updated news feed. Subsequently, we developed the Hybrid method to analyze users in terms of the dynamism of their news feed and determine a suitable retrieval strategy so as to speedup the recommendation process. Extensive experiments on real-world social networks and ad datasets have verified the efficiency and robustness of the hybrid model.
Chapter 4

Targeted Influence Maximization for CSA

In Chapter 3, we have studied how DTA channel can be used for targeted dissemination of ads. In this chapter, we examine another SMM channel: celebrity social ads (CSA) channel. CSA explores the power of viral marketing by the word-of-mouth propagation effects between friends on SNPs. Influence Maximization (IM) is a key algorithmic problem in this channel. 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. [65] 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 [17, 101]. 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 [11, 58, 30, 74, 28] 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 [28] 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 [104] to find the top-$k$ seed users. IRR further 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:
• We propose a KB-TIM query to support scalable social IM in online advertising platforms.

• 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.

• 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.

• 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 4.1 and the problem definition of KB-TIM query in Section 4.2. We extend the RIS method and propose WRIS in Section 4.2.2. Then two disk-based methods, RR and IRR, are presented in Sections 4.3 and 4.4 respectively. We report the experimental results in Section 4.5. Finally we summarize in Section 4.6.

4.1 Preliminary

In this section, we review the classic influence maximization problem as well as the state-of-the-art solutions to the problem.
4.1.1 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 [46], linear threshold (LT) model [50] and general triggering model [65] have been proposed. In this chapter, we adopt IC model because it has been widely used in previous work [101, 31, 32, 75]. Note that the methods proposed in this chapter can support LT and general triggering model as well.

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 $v$. Note that our proposed methods are also independent of how $p(e)$ is set. 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 $P(\text{head}) = p(e)$ 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:
Definition 2 (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.

![Social Network Diagram](image)

Figure 4.1: The social network adopted in Chapter 4

Example 4. In Figure 4.1, 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 \rightarrow v) has been proven to be \#P hard in [31], the calculation of the probability is feasible in this small example graph. Take p({e, g} \rightarrow b) as an example: since p(e \rightarrow b) = 0.5 and p(g \rightarrow 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} \rightarrow b) = 1 - [1 - p(e \rightarrow b)] \cdot [1 - p(g \rightarrow 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 E[I(S*)] = \sum_{v \in V} p(S* \rightarrow v) = 1 + 0.75 + 0.6875 + 0.375 + 1 + 0 + 1 = 4.8125.

4.1.2 Reverse Influence Set (RIS)

The state-of-the-art solutions to the IM problem [17, 101] 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 [17, 101]:

**Definition 3.** 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 E[I(S*)]. Based on this idea, the query processing framework of RIS works as follows:

1. Generate \(\theta\) random RR sets from G.
2. Use the standard greedy algorithm on the maximum coverage problem \[104\] 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 \[101\], it was proved that when \( \theta \) is sufficiently large, RIS returns near-optimal results with high probability:

**Theorem 3 (101).** If \( \theta \geq (8 + 2\varepsilon) \cdot |V| \cdot \frac{\ln |V| + \ln \left( \frac{|V|}{k} \right) + \ln 2}{OPT_k \cdot \varepsilon^2} \), RIS returns a \((1 - 1/e - \varepsilon)\)-approximate solution with at least \( 1 - \frac{1}{|V|} \) probability.

The proof sketch in \[101\] 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 \( \theta \geq (8 + 2\varepsilon) |V| \cdot \frac{\ln |V| + \ln \left( \frac{|V|}{k} \right) + \ln 2}{OPT_k \cdot \varepsilon^2} \), we have \( |\frac{F_\theta(S) \cdot |V|}{\theta} - \mathbb{E}[I(S)]| < \frac{\varepsilon}{2} \cdot OPT_k \) holds with probability \( 1 - \frac{1}{|V|} \) 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 \( 1 - \frac{1}{|V|} \) probability.

**Example 5.** 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 4.1 (\( d \) 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.
4.2 Targeted Influence Maximization

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

4.2.1 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 [57] 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 [115] on the topic space, the impact of an advertisement with a keyword set \( Q.T \subseteq T \) on a user \( v \) is defined as

\[
\phi(v, Q) = \sum_{w \in Q.T} tf_{w,v} \cdot idf_w
\]

(4.1)

72
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
\[
\mathbb{E}[I^Q(S)] = \mathbb{E}\left[ \sum_{v \in I(S)} \phi(v, Q) \right] = \sum_{v \in V} p(S \mapsto v) \cdot \phi(v, Q)
\]

(4.2)

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

**Definition 4.** (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 6.** In Figure 4.1 each node is associated with a user profile depicting the user’s preferences on different topics. Suppose a KB-TIM query is \( Q = (\{\text{music}\}, 2) \), the optimal seed set \( S^* \) selected by \( Q \) should be \( \{b, e\} \) and the expected impact is 
\[
\mathbb{E}[I^Q({\text{music}})(S^*)] = \sum_{v \in V} p(S^* \mapsto v) \cdot \phi(v, \{\text{music}\}) = 1.05+1.03+0.75+0.6+0.1875\cdot0.5+0.0+0.0+0.0 = 1.5.
\]
The result is different from the general IM problem in Example 5 as we are now considering targeted influence maximization.

### 4.2.2 Weighted RIS Sampling

To facilitate our presentation, all notations frequently used are listed in Table 4.1. In the traditional IM problem, all the users are treated as candidates to be influenced. However,
### Table 4.1: Notations frequently used across Chapter 4

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G, V, E, T$</td>
<td>the social network, the vertex set, the edge set and the topic space respectively.</td>
</tr>
<tr>
<td>$Q(Q.T, Q.k)$</td>
<td>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.</td>
</tr>
<tr>
<td>$OPT_{Q,k}^{Q.T}$</td>
<td>the maximum expected spread among all seed set with size $Q.k$ for query keywords $Q.T$.</td>
</tr>
<tr>
<td>$F_0(S)$</td>
<td>the number of RR sets, which is covered by seed set $S$.</td>
</tr>
<tr>
<td>$tf_{w,v}$</td>
<td>the preference weight between a user $v$ and a topic $w$.</td>
</tr>
<tr>
<td>$idf_w$</td>
<td>the inverse document frequency for a topic $w$.</td>
</tr>
<tr>
<td>$\phi_w$</td>
<td>sum of the relevance scores among all users for a given topic $w$, i.e., $\sum_{v \in V} tf_{w,v} \cdot idf_w$.</td>
</tr>
<tr>
<td>$\phi(v, Q)$</td>
<td>the relevance score of a KB-TIM query $Q$ to a user $v$, i.e., $\sum_{w \in Q.T} tf_{w,v} \cdot idf_w$.</td>
</tr>
<tr>
<td>$\phi_Q$</td>
<td>sum of the relevance scores among all users to $Q$, i.e., $\sum_{v \in V} \phi(v, Q)$.</td>
</tr>
<tr>
<td>$\theta$</td>
<td>the number of RR sets needed to process $Q$ using WRIS.</td>
</tr>
<tr>
<td>$\bar{\phi}_w$</td>
<td>the proportion of RR sets w.r.t a topic $w$ among all RR sets needed to process $Q$, i.e. $\phi_w/\phi_Q$.</td>
</tr>
<tr>
<td>$p_s(v, w)$</td>
<td>the probability to sample a user $v$ w.r.t a topic $w$ in discriminative WRIS sampling.</td>
</tr>
<tr>
<td>$I(S)$</td>
<td>an instance of the influence spread of a seed set $S$.</td>
</tr>
<tr>
<td>$I^Q(S)$</td>
<td>influence score of $I(S)$ w.r.t query keywords $Q.T$, $I^Q(S) = \sum_{v \in I(S)} \phi(v, Q)$.</td>
</tr>
<tr>
<td>$\theta^Q$</td>
<td>the number of RR sets needed process query $Q$ using RR or IRR index.</td>
</tr>
<tr>
<td>$\theta^Q_w$</td>
<td>the number of RR sets needed, w.r.t topic $w$, to process $Q$ using RR or IRR index.</td>
</tr>
</tbody>
</table>

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 3 does not hold, i.e., $E[F_0(S)/|V|] \neq E[I^Q(S)]$. Therefore, we need to find a new unbiased estimator for $E[I^Q(S)]$. In this section, we propose a weighted RIS (WRIS) sampling technique for the KB-TIM query processing.
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:

\[
    p_s(v, Q) = \frac{\phi(v, Q)}{\sum_{v \in V} \phi(v, Q)} = \frac{\phi(v, Q)}{\phi_Q}
\]

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 2.** \( \mathbb{E} \left[ \frac{F_\theta(S)}{\theta} \right] \cdot \phi_Q = \mathbb{E} [I^Q(S)] \)

**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:

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

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

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

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 \( \theta \) number of vertices from \( G \) with a probability of \( p_s(v, Q) \) (in Eqn. 4.3) 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 $\theta$ value as the minimum number of random RR sets to guarantee that $WRIS$ can also achieve a theoretical bound like Theorem 3 for RIS. Theorem 4 gives a sufficient lower bound for $\theta$ such that our WRIS algorithm is a $(1 - 1/e - \varepsilon)$-approximate solution with high probability.

**Theorem 4.** If $\theta$ satisfies:

$$\theta \geq (8 + 2\varepsilon)\phi_Q \cdot \frac{\ln |V| + \ln (|V|^{|Q,k|}) + \ln 2}{OPT_{Q,k} \cdot \varepsilon^2}$$

(4.6)

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

**Proof.** The proof is similar to that in Theorem 3 (S1-S4).

S1: We have proved in Lemma 2: $\mathbb{E}[\frac{F_\theta(S)}{\theta}] \cdot \phi_Q = \mathbb{E}[I^Q(S)]$.

S2: Given a KB-TIM query $Q$, let $\rho$ be the probability that $S$ intersects with a random RR set, then $F_\theta(S)$ can be regarded as the sum of $\theta$ i.i.d Bernoulli variables with mean $\rho$. We can then derive:

$$Pr[|\phi_Q \cdot \frac{F_\theta(S)}{\theta} - \mathbb{E}[I^Q(S)]| \geq \frac{\varepsilon}{2} \cdot OPT_{Q,k}^Q]$$

$$= Pr[|F_\theta(S) - \rho \theta| \geq \frac{\varepsilon \theta}{2\phi_Q} \cdot OPT_{Q,k}^Q]$$

$$= Pr[|F_\theta(S) - \rho \theta| \geq \frac{\varepsilon \cdot OPT_{Q,k}^Q}{2\phi_Q \cdot \rho} \cdot \rho \theta]$$

(4.7)

Let $\delta = \varepsilon \cdot OPT_{Q,k}^Q/(2\phi_Q \cdot \rho)$. By the Chernoff bounds, Eqn. 4.6 and the fact that $\rho = \mathbb{E}[I^Q(S)]/\phi_Q \leq OPT_{Q,k}^Q/\phi_Q$, we have:
\[ \text{r.h.s of Eqn. 4.7} < 2 \exp \left( \frac{-\delta^2}{2 + \delta} \cdot \rho \theta \right) \]
\[ = 2 \exp \left( \frac{-\varepsilon^2}{8\phi_Q^2 \rho + 2\varepsilon \phi_Q \cdot \text{OPT}_{Q,k}^{{Q,T}}} \cdot \theta \right) \]
\[ \leq 2 \exp \left( \frac{-\varepsilon^2}{8\phi_Q \cdot \text{OPT}_{Q,k}^{{Q,T}} + 2\varepsilon \phi_Q \cdot \text{OPT}_{Q,k}^{{Q,T}}} \cdot \theta \right) \]
\[ = 2 \exp \left( \frac{-\varepsilon^2 \cdot \text{OPT}_{Q,k}^{{Q,T}}}{(8 + 2\varepsilon) \cdot \phi_Q \cdot \theta} \right) \] (4.8)
\[ \leq \frac{1}{|V| \cdot (|V|_{Q,k})} \] (4.9)

This means \(|\phi_Q \cdot \frac{F_{\theta}(S)}{\theta} - \mathbb{E}[I^Q(S)]| \leq \frac{\varepsilon}{2} \cdot \text{OPT}_{Q,k}^{{Q,T}}\) holds with probability \(1 - |V|^{-1}\) simultaneously for all \(S\) s.t \(|S| = k\) by union bound property.

**S3-S4:** Let \(S_k\) be the vertex set returned by WRIS, and \(S_k^+\) be the vertex set that maximizes \(F_{\theta}(S_k^+)\) (i.e., \(S_k^+\) intersects with the largest number of RR sets generated by WRIS). As \(S_k\) is obtained by using a \((1 - 1/e)\)-approximate algorithm for the maximum coverage problem on the RR sets generated by WRIS, we have \(F_{\theta}(S_k) \geq (1 - 1/e) \cdot F_{\theta}(S_k^+)\).

Let \(S_k^0\) be the optimal solution for the KB-TIM query \(Q\) on \(G\), i.e., \(\mathbb{E}[I^Q(S_k^0)] = \text{OPT}_{Q,k}^{{Q,T}}\).

We have \(F_{\theta}(S_k^+) \geq F_{\theta}(S_k^0)\), which leads to \(F_{\theta}(S_k) \geq (1 - 1/e) \cdot F_{\theta}(S_k^0)\). Finally we conclude the proof of Theorem 4 as the follows:

\[
\mathbb{E}[I^Q(S_k)] > \phi_Q \cdot \frac{F_{\theta}(S)}{\theta} - \frac{\varepsilon}{2} \cdot \text{OPT}_{Q,k}^{{Q,T}}, \text{By S2} \\
\geq (1 - 1/e) \cdot \phi_Q \cdot \frac{F_{\theta}(S_k^0)}{\theta} - \frac{\varepsilon}{2} \cdot \text{OPT}_{Q,k}^{{Q,T}} \\
\geq (1 - 1/e) \cdot \phi_Q \cdot \frac{F_{\theta}(S_k^0)}{\theta} - \frac{\varepsilon}{2} \cdot \text{OPT}_{Q,k}^{{Q,T}} \\
\geq (1 - 1/e) \cdot (1 - \varepsilon/2) \cdot \text{OPT}_{Q,k}^{{Q,T}} - \frac{\varepsilon}{2} \cdot \text{OPT}_{Q,k}^{{Q,T}} \\
> (1 - 1/e - \varepsilon) \cdot \text{OPT}_{Q,k}^{{Q,T}}
\]
Thus, Theorem 4 is proved.

With the revised sampling technique in Lemma 2, the proof is similar to that in Theorem 3 (S1-S4).

4.3 Disk-based RR Index for KB-TIM Query Processing

Existing methods [17, 101] 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.

4.3.1 Discriminative WRIS Sampling

We note that the sampling probability \( p_s(v, Q) \) appeared in Eqn. 4.3 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

$$p_s(v, Q) = \frac{\sum_{w \in Q,T} t_{f,v,w} \cdot idf_w}{\phi_Q}$$

$$= \sum_{w \in Q,T} t_{f,v,w} \cdot idf_w \cdot \frac{\sum_{v \in V} t_{f,v,w} \cdot idf_w}{\phi_Q}$$

$$= \sum_{w \in Q,T} t_{f,v,w} \cdot \frac{\sum_{v \in V} t_{f,v,w} \cdot idf_w}{\phi_Q}$$

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

$$p_s(v, Q) = \sum_{w \in Q,T} p_s(v, w) \cdot p_w$$  \hspace{1cm} (4.10)

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 the same theoretical bound as in Theorem 4 based on the following lemma:

**Lemma 3.** Given a query $Q$, let $\theta^Q_w$ 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 $\theta^Q$ is the total number of RR sets and $\theta^Q = \sum_{w \in Q,T} \theta^Q_w$. If $\theta^Q \geq \theta$ and $\frac{\theta^Q}{\theta} = p_w$, we can achieve the same theoretical bound as in Theorem 4 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. 4.10. 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 \( \theta^Q \) RR sets. Therefore the expected fraction of RR sets w.r.t \( v \) among the \( \theta^Q \) RR samples is:

\[ \frac{\sum_{w \in Q.T}[(\theta^Q \cdot p_w) \cdot p_s(v, w)]}{\theta^Q} = p_s(v, Q) \] which is the same as the WRIS method. Besides, we take more RR samples than the desired threshold specified in Eqn. 4.6 for WRIS. Thus we conclude that the discriminative sampling achieves at the same theoretically accuracy as the WRIS method as in Theorem 4.

**4.3.2 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 \( \theta^Q_w = \theta \cdot p_w \) in the offline sampling. Hence, we need to find a \( \theta_w \) 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 4.** For any keyword \( w \), if we choose

\[ \hat{\theta}_w = (8 + 2\varepsilon)(\sum_{v \in V} tf_{w,v}) \cdot \frac{\ln |V| + \ln \binom{|V|}{K} + \ln 2}{OPT_1^{(w)}} \cdot \varepsilon^2 \] (4.11)

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

**Proof.** According to Eqn. 4.6 and the definition of \( p_w \), we have:

\[ \theta \cdot p_w = (8 + 2\varepsilon)(\sum_{v \in V} tf_{w,v}) \cdot \frac{\ln |V| + \ln \binom{|V|}{Q.k} + \ln 2}{OPT_{Q.k}^{Q,T} / \text{idf}_w \cdot \varepsilon^2} \] (4.12)
Algorithm 5: BuildRR(TopicSet \( T \))

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

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: \( E[I^{Q,T}(S^w)] \leq OPT_{Q,k}^{Q,T} \) according to the definition of \( OPT_{Q,k}^{Q,T} \). Then:

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

Since the influence spread is monotonic w.r.t the size of the seed set \([65]\), \( OPT_{1}^{\{w\}} \leq OPT_{Q,k}^{\{w\}} \leq OPT_{Q,k}^{Q,T} / idf_w \). In addition, as \( Q.k \leq K, \ln \left( \frac{|V|}{Q,k} \right) \leq \ln \left( \frac{|V|}{K} \right) \) when \( K \leq |V|/2 \). Thus we can conclude that \( \hat{\theta}_w \geq \theta \cdot p_w \).

The index construction procedure based on the offline sampling is shown in Algorithm 5. 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 \( v \).
Figure 4.2: Example of basic RR index structures for keyword “music” and “book”

Note that in Line 3 of Algorithm 5, $OPT^1_{\{w\}}$ is unknown and needs to be estimated in advance. We adopt the weighted iterative estimation method in [101] 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 7.** Figure 4.2 shows the RR index built for keywords “music” and “book” for the running example in Figure 4.1. In this example, we estimate $\theta_{\text{music}} = 9$ and $\theta_{\text{book}} = 6$. Then, we sample 9 random RR sets for keyword “music” and 6 for “book”. Based on the RR sets, we construct $L_{\text{music}}$ and $L_{\text{book}}$ to store the inverse mapping between vertex and RR set.

### 4.3.3 Improved Estimation of $\theta_w$

Based on Lemma 4, 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 $\theta_w$ based on the observation that $\frac{\ln |V| + \ln \binom{|V|}{K}}{\ln |V| + \ln \binom{|V|}{Q,k}} + \frac{\ln 2}{\ln |V| + \ln \binom{|V|}{Q,k} + \ln 2}$ can be approximated to $\frac{K}{Q,k}$ when $|V|$ is significantly larger than $K$ and $Q.k$.

**Lemma 5.** For any keyword $w$, if we choose

$$\theta_w = (8 + 2\varepsilon)\left(\sum_{v \in V} tf_{w,v}\right) \cdot \frac{\ln |V| + \ln \binom{|V|}{K} + \ln 2}{OPT_K^{Q,w}} \cdot \varepsilon^2$$

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

**Proof.** As $OPT_{Q,k}^{Q,w} \leq OPT_{Q,k}^{Q,T} / idf_w$ according to Lemma 4, we have the inequality: $\theta \cdot p_w \leq (8 + 2\varepsilon)\left(\sum_{v \in V} tf_{w,v}\right) \cdot \frac{\ln |V| + \ln \binom{|V|}{K} + \ln 2}{OPT_{Q,k}^{Q,w}} \cdot \varepsilon^2$. To prove $\theta_w \geq \theta \cdot p_w$, we need to justify:

$$\frac{\ln |V| + \ln \binom{|V|}{K} + \ln 2}{OPT_K^{Q,w}} \cdot \varepsilon^2 \geq \frac{\ln |V| + \ln \binom{|V|}{Q,k} + \ln 2}{OPT_{Q,k}^{Q,w}} \cdot \frac{OPT_{Q,k}^{Q,w}}{OPT_K^{Q,w}}.$$  

Let $r = \frac{\ln |V| + \ln \binom{|V|}{Q,k} + \ln 2}{OPT_K^{Q,w}} \cdot \frac{OPT_K^{Q,w}}{OPT_{Q,k}^{Q,w}}$. Since $\ln |V| + \ln \binom{|V|}{Q,k} + \ln 2$ can be approximated to $\frac{K}{Q,k}$, $r = \frac{K}{Q,k}$.

Let $S^K$ be the seed set to achieve $OPT_K^{Q,w}$ with $|S^K| = K$, according to the submodular property of the influence spread function \[65\], there exists a set $S \subset S^K$ with $|S| = Q,k$ s.t. $\frac{\mathbb{E}[I^{(w)}(S)]}{Q,k} \geq \frac{OPT_K^{Q,w}}{K}$. According to the definition of $OPT_K^{Q,w}$, $\frac{OPT_K^{Q,w}}{Q,k} \geq \frac{\mathbb{E}[I^{(w)}(S)]}{Q,k} \geq \frac{OPT_K^{Q,w}}{K}$. This means $r \geq 1$. Lastly, we can conclude that $\theta_w \geq \theta \cdot p_w$ whenever $\theta_w$ satisfies Eqn. 4.13.

To utilize the improved $\theta_w$ for constructing the index, we simply replace $\hat{\theta}_w$ with $\theta_w$ in Line 3 of Algorithm 5. $OPT_K^{Q,w}$ will be estimated similarly as how we estimated $OPT_1^{Q,w}$.

### 4.3.4 KB-TIM Query Processing

The algorithm for KB-TIM query processing based on RR index is shown in Algorithm 6. To process a query, we need to retrieve $\theta^Q$ number of RR sets from the RR index from all the query keywords. To determine $\theta^Q$, we need to ensure the conditions in Lemma 3.
Algorithm 6: QueryRR(KB-TIM $Q$)

1. $\theta^Q \leftarrow \min \left\{ \frac{\theta_w}{p_w} | w \in Q.T \right\}$
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. $L^Q_w \leftarrow L_w$
4. for $i = 1$ to $Q.k$ do
   5. $v_i \leftarrow$ the user that covers the most RR sets in $R^Q$.
   6. $S^Q \leftarrow S^Q \cup \{v_i\}$
5. for $w \in Q.T$ do
   6. for $rr \leftarrow L^Q_w[v_i]$ do
      7. if $rr$ is not covered then
         8. mark $rr$ as covered
      9. if $rr \in R^Q_w$ then
         10. remove $rr$ from $R^Q_w$
6. return $S^Q$

so that the algorithm has the result guaranteed as in Theorem 4. Note that we cannot set $\theta^Q$ to $\theta$ via Eqn. 4.13 because the optimal influence spread $OPT^Q_{Q, k}$ is unknown. To determine a proper $\theta^Q$, we set

\[
\theta^Q = \min \left\{ \frac{\theta_w}{p_w} | w \in Q.T \right\}
\]

(4.14)

When $\theta^Q$ 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 3 (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 [104] on $(R^Q, L^Q)$ (lines 6 - 14).

**Example 8.** Suppose $Q.T = \{\text{“music”}, \text{“book”}\}$ and $k = 2$. If the ratio of RR sets for keyword “music” to “book” is 9 : 4, we can determine $\theta^Q$ as 13 because

\[
\min \left\{ \frac{13 \cdot \theta_{\text{music}}}{9}, \frac{13 \cdot \theta_{\text{book}}}{4} \right\} = \min \left\{ \frac{13 \cdot 9}{9}, \frac{13 \cdot 6}{4} \right\} = 13.
\]

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 [10] 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 \(1 - |V|^{-1}\).

Proof. We only need to ensure the two conditions fulfilled for Lemma 3. It is obvious that we have \(\frac{\theta^Q}{\theta^Q} = \theta_w\) according to Algorithm 6. Then we left to prove \(\theta^Q \geq \theta\) with the \(\theta\) in Equation 4.6. For a KB-TIM query \(Q\), let \(w^* = \arg\min_w \{\theta_w \mid w \in Q.T\}\), then we have \(\theta^Q \cdot p_{w^*} = \theta_w^*\). According to Lemma 5, \(\theta_w^* \geq \theta \cdot p_{w^*}\). Finally we can conclude with \(\theta^Q \geq \theta\).

4.4 Incremental RR Index (IRR)

Although the RR index significantly improves the KB-TIM query processing compared to the WRIS method, it has to load \(\theta^Q_w\) 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 design an index that incrementally loads relevant RR sets into memory for query processing.
4.4.1 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 $IL^1_w, IL^2_w, \ldots, IL^m_w$. $IR_w$ also contains $m$ partitions $IR^1_w, IR^2_w, \ldots, IR^m_w$ generated from the partitions in $IL_w$.

$$IR^i_w = \begin{cases} 
\{rr | rr \in R_w \land rr \cap IL^i_w \neq \emptyset\} & \text{if } i = 1 \\
\{rr | rr \in R_w \land rr \cap IL^i_w \neq \emptyset \land rr \notin \bigcup_{1 \leq j < i} IR^j_w\} & \text{if } i > 1
\end{cases}$$

In other words, $IR^i_w$ picks from the remaining RR sets in $R_w$ that contain users in partition $IL^i_w$. $IP_w$ 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 $\theta^Q_w$ 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 $\theta^Q_w$ samples.

Example 9. Figure 4.3 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 4.2. In this example, we set the partition size in $IL^i_w$ 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
Figure 4.3: Example of incremental RR index structures for keyword “music” and “book”

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 4.2. Hence, it has an entry in $IP_{music}$ mapping to $rr_2$.

Algorithm 7: BuildIRR(TopicSet $T$, BlockSize $\delta$)

1. $IRRIndex \leftarrow (\emptyset, \emptyset, \emptyset)$
2. for $w \in T$ do
   3. Compute $\theta_w$ according to Lemma 5
   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. $IL^p_w \leftarrow \delta$ rows $L_w$
       12. $IR^p_w \leftarrow \{rr|rr \cap IL^p_w \neq \emptyset \wedge \exists 1 \leq j < p IR^j_w\}$
       13. $p \leftarrow p + 1$
       14. Remove the first $\delta$ rows from $L_w$
15. Store $(IR, IL, IP)$
The construction of IRR index is presented in Algorithm 7. 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 \( \delta \) users, to form \( IL^p_w \) where \( p \) is the partition ID. The matching RR sets partition \( IR^p_w \) is built against \( IL^p_w \). The construction terminates when all the users are visited.

4.4.2 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 [42]. 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 [104], 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 8. Given a query, we first calculate $\theta^Q$ 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
Figure 4.4: Example of processing a KB-TIM query $Q = (\{\text{“music”, “book”}\}, 2)$ (lines 23-30). The algorithm terminates when $Q.k$ seed users are found. An example is shown as the following.

**Example 10.** We use the same KB-TIM query $Q$ as in Example 8 where $Q = (\{\text{“music”, “book”}\}, 2)$. $\theta^Q$ is 13 as shown in Example 8. Therefore the corresponding $\theta^Q_{\text{music}}$ and $\theta^Q_{\text{book}}$ are 9 and 4 respectively. We demonstrate the steps of the incremental evaluation in Figure 4.4. The priority queue $pq$ and keyword upper bound $kb$ are shown. Besides, we list out the status and upper bound scores for all users with all query keywords. For example, user $c$ have status of $F$ and $T$ for “music” and “book” respectively. This means the score of $c$ is w.r.t “music” is not set whereas the score w.r.t “book” is set. The values that are updated from the previous steps are highlighted for the ease of presentation.
All the variables are initialized in step 0. Note that the scores of some users are set during initialization as we explained Algorithm 8. In step 1.a, we load the first partition from the IRR indices shown in Figure 4.3 and the keyword upper bound $kb$ is updated to 4 and 2 for ‘music” and “book” respectively. User b and c update their corresponding status with upper bound scores and both of them are pushed to $pq$. In step 1.b, we pop b from $pq$. Since b is COMPLETE and the score of b is larger than the sum of scores in $kb$, i.e. $7 > 4 + 2$, b is selected as the first seed user. Then the RR sets covered by b will be removed and the corresponding upper bound scores will be updated. At this stage, we are not able to confirm that c is the next seed user though c is COMPLETE because the score of c is smaller than the sum of scores in $kb$, i.e. $4 < 4 + 2$. Thus step 3 is required to load a new partition from the IRR index. e,d are loaded for ‘music” and d,f are loaded for “book”. After updating all the status and scores of the new loaded users, they are pushed to $pq$. Then c is popped from $pq$. Since we found c can be updated to a smaller value, i.e 2, (2, c) will be re-inserted into $pq$ after popping (4, c) as indicted in Figure 4.4. Lastly we pop d and d is selected to the seed set as d is COMPLETE and the score of b is larger than the current bound in $kb$.

Theorem 5. The impact scores of top-k seed users returned by Algorithm 8 and Algorithm 6 are the same.

Proof. Suppose Algorithm 8 iteratively returns k seed users \{u_1, u_2, \ldots, u_k\}, Algorithm 6 returns \{v_1, v_2, \ldots, v_k\}. If the top-k impact scores are not the same, since Algorithm 6 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 8, $u_i$ 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 $\sum_{w \in Q.T} kb[w] \leq \text{score}(u_i)$. Both cases contradict with $\text{score}(u_i) < \text{score}(v_i)$.

The $\text{case 2)}$: $v_i = u_i$. 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 8 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 6. Thus, $\text{score}(v_i) = \text{score}(u_i)$, which contradicts with $\text{score}(u_i) < \text{score}(v_i)$.

4.5 Experimental Study

In this section, we study the performance of KB-TIM query processing on two real datasets. We use WRIS (in Section 4.2.2) as a baseline solution because it uses online sampling and can be considered as a variant of the state-of-the-art RIS methods [101, 17]. We compare the methods based on offline sampling (RR and IRR index in Sections 4.3 and 4.4 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 chapter due to the incapability to scale for large graph sizes and number of topics. 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).

4.5.1 Experimental Setup

Datasets. We use two real-world datasets, Twitter and News, from SNAP [1] 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 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 4.2. We also plot the frequency distribution of incoming degrees in Figure 4.5, 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. 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. 4.6, \( \hat{\theta}_w \) in Eqn. 4.11 and \( \theta_w \) in Eqn. 4.13. ε is set to 0.1 for all experiments as it is the most accurate setting adopted in [101] (no experiment is done

---

Table 4.2: All parameter settings used in the experiments of Chapter 4

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Twitter dataset</th>
<th>News dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Users</td>
<td>10M, 20M, 30M, 40M</td>
<td>0.2M, 0.6M, 1M, 1.4M</td>
</tr>
<tr>
<td>#Edges</td>
<td>0.7B, 1.1B, 1.2B, 1.3B</td>
<td>1.0, 1.9M, 2.6M, 3.1M</td>
</tr>
<tr>
<td>AveDegree</td>
<td>76.4, 56.8, 46.1, 38.9</td>
<td>5.2, 3.1, 2.6, 2.2</td>
</tr>
<tr>
<td>#QWords</td>
<td>1, 2, ..., 5, 6</td>
<td></td>
</tr>
<tr>
<td>k</td>
<td>10, 15, ..., 30, ..., 50</td>
<td></td>
</tr>
</tbody>
</table>
Figure 4.5: In-degrees distributions for both datasets

Table 4.3: Disk space and running time of using \( \hat{\theta}_w \) and \( \theta_w \) for constructing indices for news datasets

<table>
<thead>
<tr>
<th>Data</th>
<th>Disk Size (GB)</th>
<th>Time (Secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RR</td>
<td>IRR</td>
</tr>
<tr>
<td></td>
<td>( \hat{\theta}_w )</td>
<td>( \theta_w )</td>
</tr>
<tr>
<td>n0.2M</td>
<td>37</td>
<td>4.1</td>
</tr>
<tr>
<td>n0.6M</td>
<td>47</td>
<td>5.7</td>
</tr>
<tr>
<td>n1.0M</td>
<td>54</td>
<td>6.7</td>
</tr>
<tr>
<td>n1.4M</td>
<td>62</td>
<td>7.6</td>
</tr>
</tbody>
</table>

K is set to 100 since the largest \( Q.k \) in our experiment is 50. For IRR, the partition size \( \delta \) 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 4.2.

### 4.5.2 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. 4.11) and \( \theta_w \) (Eqn. 4.13) for index construction. The disk space and running time of using \( \hat{\theta}_w \) and \( \theta_w \) for index construction of news datasets
are presented in Table 4.3. 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 $\theta_w$ 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.

Table 4.4: Disk space and running time for constructing indices for various datasets with $\theta_w$

<table>
<thead>
<tr>
<th>Data</th>
<th>Disk Size (GB)</th>
<th>Time (Secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>uncomp</td>
<td>compress</td>
</tr>
<tr>
<td>RR</td>
<td>4.1</td>
<td>4.1</td>
</tr>
<tr>
<td>IRR</td>
<td>5.7</td>
<td>5.8</td>
</tr>
<tr>
<td>RR</td>
<td>6.7</td>
<td>6.9</td>
</tr>
<tr>
<td>IRR</td>
<td>7.6</td>
<td>7.8</td>
</tr>
<tr>
<td>RR</td>
<td>88</td>
<td>94</td>
</tr>
<tr>
<td>IRR</td>
<td>84</td>
<td>93</td>
</tr>
<tr>
<td>RR</td>
<td>77</td>
<td>87</td>
</tr>
<tr>
<td>IRR</td>
<td>55</td>
<td>63</td>
</tr>
</tbody>
</table>

Table 4.5: Sum of $\theta_w$ and mean rr set size for increasing graph size

<table>
<thead>
<tr>
<th>(News)</th>
<th>Sum of</th>
<th>Mean of</th>
<th>Twitter</th>
<th>Sum of</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>[V]</td>
<td>$\theta_w$</td>
<td>rr size</td>
<td>[V]</td>
<td>$\theta_w$</td>
<td>rr size</td>
</tr>
<tr>
<td>0.2M</td>
<td>385M</td>
<td>2.7</td>
<td>10M</td>
<td>218M</td>
<td>94.9</td>
</tr>
<tr>
<td>0.6M</td>
<td>640M</td>
<td>2.3</td>
<td>20M</td>
<td>247M</td>
<td>71.9</td>
</tr>
<tr>
<td>1M</td>
<td>798M</td>
<td>2.1</td>
<td>30M</td>
<td>276M</td>
<td>55.0</td>
</tr>
<tr>
<td>1.4M</td>
<td>926M</td>
<td>2.0</td>
<td>40M</td>
<td>374M</td>
<td>26.7</td>
</tr>
</tbody>
</table>

Since RR and IRR use inverted lists, we can apply FastPFOR\(^3\) 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 4.4. For 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

---

\(^3\)https://github.com/lemire/FastPFOR
size depends on two factors: the number of RR sets, i.e. $\theta_w$ in Lemma 5, and the average size of a RR set. On one hand, $\theta_w$ increases for large graphs since $\theta_w$ depends on $|V|$. We report the sum of $\theta_w$ among all keywords and average rr set size for both datasets in Table 4.5. 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 4.2, 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, $\theta_w$ and average RR set size are two conflicting factors as $|V|$ varies. In the news dataset, $\theta_w$ 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.

97
4.5.3 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 4.6. It is obvious that the methods based on offline sampling are significantly faster than 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 $\theta_w$ 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 4.6. 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. $\theta$ in Theorem 4, 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.

<table>
<thead>
<tr>
<th>$Q.k$</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>25</th>
<th>30</th>
<th>35</th>
<th>40</th>
<th>45</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>News</td>
<td>6.10</td>
<td>7.34</td>
<td>8.75</td>
<td>10.2</td>
<td>14.0</td>
<td>19.3</td>
<td>33.6</td>
<td>78.1</td>
<td>170</td>
</tr>
<tr>
<td>Twitter</td>
<td>8.00</td>
<td>14.5</td>
<td>23.0</td>
<td>34.0</td>
<td>40.0</td>
<td>51.0</td>
<td>58.5</td>
<td>69.0</td>
<td>81.0</td>
</tr>
</tbody>
</table>

Table 4.7: Influence spread when varying $Q.k$.

<table>
<thead>
<tr>
<th>$Q.k$</th>
<th>WRIS</th>
<th>RR($\theta_w$)</th>
<th>RR</th>
<th>IRR</th>
<th>WRIS</th>
<th>RR</th>
<th>IRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>289.6</td>
<td>289.4</td>
<td>289.4</td>
<td>289.4</td>
<td>8673.9</td>
<td>8672.6</td>
<td>8672.6</td>
</tr>
<tr>
<td>15</td>
<td>356.5</td>
<td>356.3</td>
<td>356.4</td>
<td>356.4</td>
<td>10666</td>
<td>10666</td>
<td>10666</td>
</tr>
<tr>
<td>20</td>
<td>408.9</td>
<td>409.0</td>
<td>409.0</td>
<td>409.0</td>
<td>12088</td>
<td>12092</td>
<td>12092</td>
</tr>
<tr>
<td>25</td>
<td>448.7</td>
<td>448.6</td>
<td>448.6</td>
<td>448.6</td>
<td>13100</td>
<td>13099</td>
<td>13099</td>
</tr>
<tr>
<td>30</td>
<td>480.3</td>
<td>480.3</td>
<td>480.2</td>
<td>480.2</td>
<td>13929</td>
<td>13925</td>
<td>13925</td>
</tr>
<tr>
<td>35</td>
<td>506.7</td>
<td>506.8</td>
<td>506.7</td>
<td>506.7</td>
<td>14618</td>
<td>14616</td>
<td>14616</td>
</tr>
<tr>
<td>40</td>
<td>528.6</td>
<td>528.8</td>
<td>528.8</td>
<td>528.8</td>
<td>15209</td>
<td>15209</td>
<td>15209</td>
</tr>
<tr>
<td>45</td>
<td>548.6</td>
<td>548.0</td>
<td>548.1</td>
<td>548.1</td>
<td>15731</td>
<td>15735</td>
<td>15735</td>
</tr>
<tr>
<td>50</td>
<td>564.9</td>
<td>564.9</td>
<td>564.9</td>
<td>564.9</td>
<td>16211</td>
<td>16214</td>
<td>16214</td>
</tr>
</tbody>
</table>

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 4.6, 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 4.7. To justify our improved estimation of \( \theta_w \) in Sec. 4.3.3, we also report the news dataset result for running RR method on indices constructed by using \( \hat{\theta}_w \). As shown in Table 4.7, 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.

### 4.5.4 Vary the Number of Query Keywords

When we increase the number of query keywords \( |Q.T| \) from 1 to 6, as shown in Figure 4.7, 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.
4.5.5 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 4.8. 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.
4.5.6 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 \([101, 31, 32, 75]\) and linear threshold (LT) model \([101, 75, 65]\). For IC model, the propagation probability set to \(p(e) = \frac{1}{N_v}\) which is widely adopted in \([101, 31, 32, 75]\). For LT model, following existing work \([101, 65, 33]\), 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 4.8, we illustrate the top-8 most influential users for 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 chapter 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 work. In this chapter, we focus on the efficiency issue and our method can be easily applied to other propagation models like the general triggering model \[65, 75\]. This is because our proposed \textbf{WRIS} is an extension of \textbf{RIS} and \textbf{RIS} has been shown to incorporate all propagation models in \[101\]. The only difference is that \textbf{RIS} uniformly samples vertices to create RR sets whereas \textbf{WRIS} conducts a weighted sampling approach. Since influence propagation models is independent of the vertex sampling methods, \textbf{WRIS} can directly adopt other propagation models supported by \textbf{RIS}.

\section{Summary}

In this chapter, we studied the Keyword-Based Targeted Influence Maximization (KB-TIM) query on social networks. We first proposed an online sampling \textbf{RIS} method \textbf{WRIS} that returns a solution with a \((1 - 1/e - \varepsilon)\) approximation ratio. Then a disk based index \textbf{RR} is developed so that the query processing can be done in real time. Subsequently, an incremental index and query processing technique, i.e. \textbf{IRR}, is presented to further boost
<table>
<thead>
<tr>
<th>Method</th>
<th>Keyword</th>
<th>News dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>WRIS(IC)</td>
<td>“software”</td>
<td>kb.vmware.com, davidstef9.wordpress.com, suse.ehelp.pl, softwarelivre.net, en.wikipedia.org, codeplex.com, virtualgeek.typepad.com, linux.pl</td>
</tr>
<tr>
<td>WRIS(IC)</td>
<td>“journal”</td>
<td><a href="http://www.biblegateway.com">www.biblegateway.com</a>, hugh.journalspace.com, journal.peishan.org, signefavor17.spaces.live.com, bookology.wordpress.com, journals.aol.com, earticle.wordpress.com, jazzitalia.net</td>
</tr>
<tr>
<td>WRIS(LT)</td>
<td>“journal”</td>
<td>journals.aol.com, bizjournals.com, journal.peishan.org, bookalytics.com, bookology.wordpress.com, <a href="http://www.biblegateway.com">www.biblegateway.com</a>, jornaldugeek.com, jazzitalia.net</td>
</tr>
<tr>
<td>RIS</td>
<td>N.A</td>
<td>en.wikipedia.org, blogger.com, myweb.yahoo.com, wrzuta.pl, 2.bp.blogspot.com</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Keyword</th>
<th>Twitter dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>WRIS(IC)</td>
<td>“software”</td>
<td>BarackObama, cnmbrok, THE_REAL_SHAQ, jimmyfallon, britneyspears, aplusk, kevinrose, twitter</td>
</tr>
<tr>
<td>WRIS(LT)</td>
<td>“software”</td>
<td>biz, kevinrose, BarackObama, jimmyfallon, cnmbrok, twitter, mashable, britneyspears</td>
</tr>
<tr>
<td>WRIS(IC)</td>
<td>“journal”</td>
<td>kevinrose, Barack Obama, RyanSeacrest, THE_REAL_SHAQ, twitter, TheEllenShow, britneyspears, taylorswift13</td>
</tr>
<tr>
<td>WRIS(LT)</td>
<td>“journal”</td>
<td>ev, twitter, TheOnion, jimmyfallon, cnmbrok, BarackObama, NotTinaFey, britneyspears</td>
</tr>
<tr>
<td>RIS</td>
<td>N.A</td>
<td>ev, kevinrose, mashable, TheEllenShow, cnmbrok, BarackObama, jimmyfallon, britneyspears</td>
</tr>
</tbody>
</table>
the performance of RR method. Extensive experiments on real-world social network data have verified the theoretical findings and efficiency of our solutions.
Chapter 5

Personalized Influential Tags
Exploration for SIA

The last SMM channel studied in this thesis is the self influential ads (SIA) channel. Through this channel, an advertiser directly makes an ad post on his personal social homepage/channel and hope the ad would be effectively propagated to wider audiences by his friends or followers. Thus, exploring influential content features for the advertiser becomes the key to successful SIA deployments.

As previously discussed in Section 1.2.3, political campaigns have an urgent demand to take advantage of SNPs as important channels for opinion poll to analyze their “selling points”. It is highly desirable for any candidate, say Hilary Clinton, to evaluate the “effect” of her standpoints and judge which ones can influence more people, e.g. infrastructure rebuild and US-China relation in Fig. 1.1. To win more voters, her publicity manager should then spend more time on these issues for Hilary’s subsequent public speeches or tweets. Another useful application is social media marketing, where businesses also want to position their marketing strategies by identifying the influential product features (e.g., high-tech, energy-saving) which are propagated to more people.
in SNs. Similarly, researchers want to explore their most influential research contributions from academia SNs. Last but not the least, with an emerging trend of “we-media” (aka “self-media”), long-tail users are eager to know which topics to be published would receive more attentions from their potential SN audiences.

In this chapter, we propose a novel social influence query known as personalized social influential tag exploration (PITEX): Given a target user, it extracts a size-$k$ tag set that maximizes the user’s social influence from a set of tags which characterize the content propagated in an SNP. We study the research challenges that naturally arise in supporting efficient PITEX processing. The first challenge is to generate a total set of tags from the contents propagated in an SNP and analyze the correlation between the tags and influence spread. Thanks to the existing topic-based social influence models [11, 82, 58, 84, 28], we can generate both the tag set and its correlation to influence spread by analyzing the interaction history of SNP users. The second challenge is to analyze the complexity of answering PITEX query. We prove that it is not only a NP-hard problem but also NP-hard to be approximated within any constant ratio. We introduce a sampling-based framework and employ Monte Carlo (MC) sampling [65, 71] and Reverse Random set (RR) sampling [17, 101] to estimate the influence spread and find the best size-$k$ tag set. The third challenge is to achieve high performance, as millions of SNP users may issue PITEX queries to explore their personalized influence tags. Due to PITEX’s hardness, the framework with the existing sampling techniques is both theoretically and experimentally inefficient according to our studies. Thus, we focus on devising efficient algorithms to achieve real-time performance with theoretical guarantee.

PITEX can satisfy users’ requirements in various real applications. The first one is to promote social marketing campaigns, such as viral marketing [16]. For example, suppose that a restaurant has many friends in Facebook caring about healthy-food. The restaurant can leverage PITEX to select featured tags, such as “low-calories” and
“organic”, to arouse interests of its friends for widely spreading its products. Another application is to improve SN user experiences by enabling more social exposure (as studied in [9], SN users have strong incentives to make their contents accessed by more people). Consider a user having many Twitter followers who are basketball fans. PITEX can recommend NBA related hashtags to enhance social interactions of this user, e.g., replies and likes, and thus in turn, improve the user experience. Due to its immense practical potentials, it calls for supporting online PITEX computation, as there would be a huge number of users in SNs posing PITEX queries and each query wants to be answered efficiently.

To achieve instant performance to answer PITEX queries, we first propose approaches for optimizing online sampling. We devise a lazy propagation sampling algorithm to probe as few edges as possible when estimating the influence spread, which significantly outperforms MC and RR sampling techniques. We also develop a best-effort exploration strategy that effectively estimates bounds of influence spread and thus avoids computing the actual influence spread of tag sets with small influence. Furthermore, to enable real-time influence computation, we devise an effective index structure that materializes the “influencer” of uniformly sampled users, and develop efficient pruning and materialization techniques to support fast influence computation with moderate index size. We prove that all the approaches proposed in this chapter can achieve a $\frac{1-\varepsilon}{1+\varepsilon}$ approximate solution.

To summarize, we make the following contributions.

(1) To the best of our knowledge, we are the first to study personalized social influential tags exploration. We formalize the problem and prove its hardness (Section 5.1).

(2) We introduce a sampling-based framework and employ MC and RR sampling to answer PITEX. We theoretically analyze these two strategies and pinpoint the drawbacks of applying both strategies in processing PITEX. (Section 5.2).
(3) We devise lazy propagation sampling and best-effort exploration techniques for optimizing online sampling (Section 5.3). To enable real-time computation, we develop an index structure together with effective pruning techniques. The proposed approaches can significantly reduce the running time while preserving the theoretical bound (Section 5.4).

(4) We evaluate the performance on four large-scale real datasets. Results of our experimental study have shown the effectiveness and efficiency of our methods (Section 5.5).

5.1 The PITEX Problem

This section introduces the problem of personalized social influential tags exploration. We first formally define the problem in Section 5.1.1 and then provide theoretical analysis on the problem hardness in Section 5.1.2.

5.1.1 Problem Definition

We model a social network as a directed graph $G(V,E)$, where $V$ is a set of users and each edge $e = (u,v)$ in $E$ captures the friendship or follow relationship from $u$ to $v$. Like previous literature in SNPs [94, 96, 44], we introduce a set $\Omega$ of tags that characterizes activities in the SN, such as updates, replies and forwards. Examples of such tag set include hashtags in Twitter and topical keywords extracted from the text of the activities [11, 82].

Information Propagation Model. To capture how information is propagated in an SN, we adopt the topic-aware propagation model [11, 82, 58, 84, 28], which is extensively studied recently. The idea of the model is to consider that the influence of user $u$ on another user $v$ depends on not only their social connection $e = (u,v)$, but also the content (i.e., tags) that $u$ publishes. For example, $u$ may influence $v$ only given activities related to sports tags. To formalize this, the model introduces an activation probability $p(e|W)$ for each edge $e = (u,v)$, where $W$ is a subset of $\Omega$. To compute $p(e|W)$, it introduces a
set of hidden topics \( Z = \{z_1, z_2, \ldots, z_{|Z|}\} \), and considers the probability depends on the following two factors. (1) **Topic-based influence probability** \( p(e|z) \) is the likelihood that \( u \) influences \( v \) through edge \( e \) under topic \( z \in Z \), i.e., if \( u \) is interested in the content of topic \( z \), how likely will \( v \) is activated by \( u \) to be interested in that content. (2) **Tag distribution in topic** \( p(w|z) \) represents the likelihood of sampling a tag \( w \) from topic \( z \). The above two types of probabilities can be jointly learnt from “a log of past propagation” \([11]\).

Then, the activation probability \( p(e|W) \) is computed by combining these probabilities as

\[
p(e|W) = \sum_{z \in Z} p(e|z) \cdot p(z|W) \\
= \sum_{z \in Z} p(e|z) \cdot \frac{\prod_{w \in W} p(w|z)p(z)}{\sum_{z' \in Z} \prod_{w \in W} p(w|z')p(z')}, \tag{5.1}
\]

where \( p(z) \) denotes the prior probability.

Based on \( p(e|W) \) in each edge, we are now ready to present the influence in the entire SN by considering the process of information propagation. In this chapter, we use the independent cascade (IC) model \([31, 65]\) as it has been widely adopted\(^1\). In the IC model, each user in \( V \) is either in an active state or inactive state. Initially every user is inactive. Given a tag set \( W \subseteq \Omega \), a target user, say \( u \), is selected to become active at time step 0. Then, each active user at time step \( i \) attempts to activate the inactive neighbors with the aforementioned probability \( p(e|W) \), i.e., the likelihood of \( v \) being activated by \( u \) is \( p(e|W) \). Note that each active user has only one chance to activate his neighbors; after that the user stays active and stops the activation. At time step \( i + 1 \), the newly activated users will also activate their neighbors. This process terminates when no more users are

\(^1\)The approaches proposed in this chapter can also support other propagation models, such as linear threshold model \([50]\) and the more general triggering model \([65, 101]\).
activated. To evaluate the effect of this process, the influence spread $I(u|W)$ is introduced to denote the number of active users after the process terminates.

**Personalized Social Influential Tags Exploration.** Given the influence propagation model, our problem is formally defined as follows:

**Definition 5.** *(Personalized Social Influential Tags Exploration (PITEX))* Given a social network $G$, a PITEX query consists of a target user $u$ who is initially activated and a number $k$, and it aims to find a $k$-size tag set $W^*$ that maximizes $u$’s expected influence spread, *i.e.*, $W^* = \arg\max_{W \subseteq \Omega, |W| = k} \mathbb{E}[I(u|W)]$.

Note that PITEX can be extended by considering other factors, e.g., tag popularity (i.e., the new PITEX is a combination of both influence and popularity). Nevertheless, this chapter focuses on maximizing social influence as the problem is already computationally complex (see Section 5.1.2). To facilitate our presentation, all the frequently used notations are listed in Table 5.1.

**Example 11.** Figure 5.1 shows an example, consisting of a social graph with topic-based influence probabilities on each edge, and a tag topic probability $p(w|z)$ table. Under a uniform prior $p(z_j) = \frac{1}{|Z|}$, influence probability of edge $e = (u_1, u_2)$ w.r.t. tag set $\{w_1, w_2\}$ is computed as $p(e|\{w_1, w_2\}) = 0.4 \cdot 0.5 + 0.5 \cdot 0.5 + 0 \cdot 0 = 0.2$. The influence spread of $u_1$ across the entire social graph w.r.t $\{w_1, w_2\}$ is $\mathbb{E}[I(u_1|\{w_1, w_2\})] = 1.5125$. When $u_1$...
issues a PITEX query for two tags \( k = 2 \), \( W^* = \{w_3, w_4\} \) will be selected as the result since it generates the maximum expected influence spread \( \mathbb{E}[I(u_1|W)] \) from \( u_1 \) among all the size-2 tag sets.

### 5.1.2 Problem Hardness

This section analyzes the complexity of PITEX. It turns out that not only this problem is NP-hard, but also there exists no constant approximation algorithm unless NP=P. To prove this claim, we first show the following lemma.

**Lemma 6.** (\( k \)-label s-t reachability) Consider a directed multi-graph \( G = (V, E, \mathcal{L}) \) where \( V \) is a set of vertices, \( E \) is a set of edges, and \( \mathcal{L} \) is a set of labels. Each edge \( e \in E \) is associated with a label \( l \in \mathcal{L} \). Given a pair of vertices \( s, t \) in \( G \) and a number \( k \), the problem, which checks if there exists a label set \( L \subseteq \mathcal{L} \) and \( |L| = k \) s.t. \( s \) reaches \( t \) in the subgraph of \( G \) induced by \( L \) is NP-hard.
Proof. To prove this lemma, we first note that the \( k \)-label \( s-t \) reachability problem is equivalent to the problem of finding the minimum number of labels required so that \( s \) reaches \( t \) by a simple polynomial time reduction. Thus, if one of them is NP-hard, so is the other.

Next, we prove the problem of finding the minimum label set for \( s \) to reach \( t \) is NP-hard, by a reduction from the set cover problem. Given an instance of the set cover problem with a universe set \( U = \{u_1, ..., u_n\} \) and a collection of subsets \( S = \{S_1, S_2, ..., S_m\} \), \( S_i \subseteq U \), we construct a directed multi-graph \( G' \) with \( n + 1 \) vertices \( \{u'_1, ..., u'_{n+1}\} \) and \( m \) labels \( \{l_1, ..., l_m\} \). Then if \( u_i \in S_j \), we add an edge \((u'_i, u'_{i+1})\) with a label \( w_j \) into \( G' \). It is not difficult to see that the problem of finding the minimum labels that \( u'_1 \) reaches \( u'_{n+1} \) on \( G' \) can be solved, only if the set cover problem is solved. As the reduction is in polynomial time and the set cover problem is NP-hard, we finish the proof.

With Lemma 6, we now show the hardness of PITEX.

**Theorem 6.** Given an instance \( \langle G, u, k \rangle \) of PITEX, it is NP-hard to obtain a solution that achieves an expected influence spread of at least \( \frac{1}{\sqrt{n}} \\text{OPT} \) where \( n \) is the number of vertices in \( G \) \((n = |V|)\) and \( \text{OPT} \) is the maximum influence spread of \( u \) under any \( k \)-size tag set.

**Proof.** We prove by a reduction from the \( k \)-label \( s-t \) reachability problem. Given an instance \( \langle G = (V, E, \mathcal{L}), k, s, t \rangle \) of the \( k \)-label \( s-t \) reachability problem, we construct an influence graph \( G' \) as follows. First, we construct the vertex set of \( G' \) as \( V = \{u_1, u_2, ..., u_n\} \) and \( V' = \{u'_1, u'_2, ..., u'_{n^2-n}\} \). Second, we construct a tag set \( \Omega = \{w\} \) and a topic set \( Z = \{z\} \) in \( G' \) such that \( |\Omega| = |Z| = |\mathcal{L}| \), \( p(w_i|z_i) = 1 \) for \( i = 1, ..., |\Omega| \), and \( p(w_i|z_j) = 0 \) for any \( i \neq j \). Third, for each edge \( e = (u_i, u_j) \in E \) associated with a label \( l_t \) in \( G \), we add an edge to the corresponding vertices \( u_i \) and \( u_j \) in \( G' \) and set \( p(e|z_t) = 1 \). In addition, we also create another set \( E' \) of edges for
Given the constructed graph $G'$ with $n^2$ vertices, let us assume that there exists a polynomial time algorithm $A$ that can solve PITEX and achieves an influence spread of at least $\OPT' = \OPT/\sqrt{n^2}$. Then, we can create another algorithm to solve the $k$-label $s$-$t$ reachability problem $\langle G = (V, E, L), k, s, t \rangle$ by simply considering the following cases.

**Case 1:** If the produced influence spread $\OPT' \leq n - 1$, we must have $\OPT \leq n - 1$, which can be proved by contradiction: If $\OPT > n - 1$, it means vertex $s$ must reach vertex $t$. According to the construction of $E'$, $s$ must also influence all the vertices in $V'$, and thus we have $\OPT \geq n^2 - n + 2$ (i.e., $n^2 - n$ vertices in $V'$ and $s, t$). As $\OPT/\sqrt{n^2} \leq \OPT' \leq n - 1$, we have $n^2 - n + 2 \leq \OPT \leq n \cdot (n - 1)$, which induces an incorrect result $2 \leq 0$. Now that we know $\OPT \leq n - 1$, it is trivial to see that $s$ cannot reach $t$ in graph $G$.

**Case 2:** If the influence spread $\OPT' > n - 1$, we also have $\OPT \geq \OPT' > n - 1$. Based on the above analysis, we must have $\OPT \geq n^2 - n + 2$. As $n^2 - n + 2 > n - 1$ always holds, we can see $s$ can reach $t$ in $G$.

However, according to Lemma 6, the $k$-label $s$-$t$ reachability problem is NP-hard. Thus, we have a contradiction: if there exists an algorithm $A$ that solves the PITEX problem and achieves expected influence spread of at least $\OPT/\sqrt{|V|}$, then P=NP. We thus prove the theorem.

### 5.2 Sampling-Based Framework

One straightforward solution to PITEX is to enumerate all possible size-$k$ tag sets and select the one with the maximum influence spread. However, the computation of influence spread $\mathbb{E}[I(u|W)]$ of user $u$ for any tag set $W$ is expensive due to its #P-hardness [31]. To
address this issue, we introduce a sampling-based framework on top of the enumeration-based approach to achieve a tight approximation ratio. Before presenting the details, we introduce the chernoff bounds \cite{34} which are frequently used in our analysis.

**Lemma 7.** Let $X$ be the sum of $\theta$ independent and identical r.v.s sampled from a distribution on $[0, 1]$ with a mean $p$. For any $\delta > 0$, the followings hold \cite{34},

\[
\Pr[X − \theta p ≥ \delta \cdot \theta p] \leq \exp\left(-\frac{\delta^2}{2 + \delta}\right),
\]

\[
\Pr[X − \theta p ≤ −\delta \cdot \theta p] \leq \exp\left(-\frac{\delta^2}{2}\right)
\]

Given a social graph $G$, a user $u$ and a number $k$, the framework enumerates every possible $k$-size tag set $W \subseteq \Omega$, and computes the influence spread $\mathbb{E}[I(u|W)]$ of $u$ under tag set $W$. Specifically, it employs sampling based strategies to implement `EstimateInfluence` (see Algo. 9). It first computes the size of samples $\theta_W$ (SAMPLESIZE) and then estimates influence spread $\hat{\mathbb{E}}[I(u|W)]$ (SAMPLEESTIMATE) based on the samples. Finally, the framework returns the tag set $W^*$ with the maximum estimated influence.

To support the framework, we can adopt existing sampling strategies. Next, we present two state-of-the-art strategies and then discuss how to determine the sample size $\theta_W$.

**Monte Carlo Sampling.** Inspired by \cite{65}, we can use a Monte Carlo (MC) method for computing $\hat{\mathbb{E}}[I(u|W)]$. It generates an instance of sample $g$ from $G$ as follows. It starts from $u$ and traverses $G$ by removing each edge with a probability $1 − p(e|W)$ until no vertex can be reached. Let $I_g(u|W)$ denote the number of nodes reachable from $u$ in
Suppose we generate multiple sample instances, i.e., \( \{g_1, g_2, \ldots, g_{\theta_W}\} \). The expected influence spread is then estimated by:

\[
\hat{E}_{MC}[\mathcal{I}(u|W)] = \sum_{i=1}^{\theta_W} \mathcal{I}_{g_i}(u|W)/\theta_W.
\]

**Reverse Reachable Set Sampling.** We can also apply the recently proposed reverse reachable set (RR) sampling [17] to estimate \( \hat{E}[\mathcal{I}(u|W)] \). It first uniformly samples a vertex \( v_i \) from the set \( \mathcal{R}_W(u) \) of vertices that can be reached by \( u \) on \( G \) by removing each edge if \( p(e|W) = 0 \), and then generates a subgraph \( g_i \) from \( G \) by removing each edge \( e \) with a probability \( 1 - p(e|W) \). If \( u \) reaches \( v_i \) on \( g_i \), we set \( 1[u \leadsto v_i] = 1 \); otherwise, \( 1[u \leadsto v_i] = 0 \). By sampling multiple vertices \( v_1, \ldots, v_{\theta_W} \) from \( \mathcal{R}_W(u) \), the influence spread is estimated by:

\[
\hat{E}_{RR}[\mathcal{I}(u|W)] = \sum_{i=1}^{\theta_W} 1[u \leadsto v_i] \cdot |\mathcal{R}_W(u)|/\theta_W.
\]

A critical issue in the sampling strategies is to determine sufficient number \( \theta_W \) of sample instances, which works for any tag set \( W \) to ensure estimation accuracy. Existing works on RR sampling [101] have provided the following bound.

**Lemma 8.** [101] Whenever \( \theta_W \) satisfies

\[
\theta_W = \frac{2 + \varepsilon}{\varepsilon^2} \cdot |\mathcal{R}_W(u)| \cdot \frac{\log(\delta) + \log \left( \frac{|\mathcal{V}|}{k} \right) + \log 2}{E[\mathcal{I}(u|W)]}, \tag{5.2}
\]

then \( |\hat{E}_{RR}[\mathcal{I}(u|W)] - E[\mathcal{I}(u|W)]| < \varepsilon \cdot E[\mathcal{I}(u|W)] \) holds with a probability of at least \( 1 - \delta^{-1} \left( \frac{|\mathcal{V}|}{k} \right)^{-1} \) for the RR method.

On the other hand, existing works using MC sampling [65, 71] do not provide any theoretical results on \( \theta_W \) and simply use a large number, say 10000. According to our studies, the above error bound can also be applied to MC sampling when solving PITEX, i.e.,

**Lemma 9.** Whenever \( \theta_W \) satisfies Eqn. (5.2), the inequality: \( |\hat{E}_{MC}[\mathcal{I}(u|W)] - E[\mathcal{I}(u|W)]| < \varepsilon \cdot E[\mathcal{I}(u|W)] \) holds with a probability of at least \( 1 - \delta^{-1} \left( \frac{|\mathcal{V}|}{k} \right)^{-1} \) for the MC method.

117
Proof. We state the following probability bound:

\[
Pr \left[ \left| \hat{E}_{MC}[\mathcal{I}(u|W)] - \mathbb{E}[\mathcal{I}(u|W)] \right| \geq \varepsilon \cdot \mathbb{E}[\mathcal{I}(u|W)] \right) = Pr \left[ \left| \sum_{i=1}^{\theta_W} g_i(u|W) \right| - \theta_W \cdot \mathbb{E}[\mathcal{I}(u|W)] \right| \geq \varepsilon \cdot \theta_W \mathbb{E}[\mathcal{I}(u|W)] \right) \tag{5.3}
\]

Note that \( \sum_{i=1}^{\theta_W} g_i(u|W) \) is a sum of \( \theta_W \) i.i.d. r.v.s and each variable \( \frac{g_i(u|W)}{|R_W(u)|} \) lies in \([0, 1]\). Moreover, we have \( \mathbb{E} \left[ \sum_{i=1}^{\theta_W} g_i(u|W) \right] = \mathbb{E}[\mathcal{I}(u|W)] \). Thus, the following holds according to the chernoff bound [34]:

\[
\text{Eqn. (5.3)} < 2 \cdot \exp \left( -\frac{\varepsilon^2}{2 + \varepsilon} \cdot \theta_W \cdot \frac{\mathbb{E}[\mathcal{I}(u|W)]}{|R_W(u)|} \right) = \frac{1}{\delta \cdot \left( \frac{|\Omega|}{k} \right)} \tag{5.4}
\]

Therefore, Lemma 9 is proved. \( \square \)

In summary, Lemmas 8 and 9 show that both MC and RR sampling methods may use the same sample size \( \theta_W \) for the solutions to achieve the same error bound. We show the approximation ratio of the proposed enumeration-based approach, as formally stated in the following.

**Theorem 7.** The enumeration-based method always has a \( \frac{1 - \varepsilon}{1 + \varepsilon} \) approximation ratio to the optimal solution to the PITEX problem with a probability of \( 1 - \delta^{-1} \).

Proof. Let \( W^* \) and \( \hat{W} \) respectively denote the optimal tag set and the tag set returned by the enumeration-based method. Recall that \( \hat{\mathbb{E}}[\mathcal{I}(u|W)] \) is the influence spread estimated by either MC or RR. Based on Lemmas 8 and 9, the estimation has an error bound for any \( k \)-size tag set \( W \) with a probability of at least \( 1 - \delta^{-1} \left( \frac{|\Omega|}{k} \right)^{-1} \) when \( \theta_W \) is large. By union bound [104], the estimation has the error bound with a probability of at least \( 1 - \delta^{-1} \) simultaneously for all \( k \)-size tag sets. Thus, the following holds with a probability
of $1 - \delta^{-1}$:

$$E[I(u|W)] > \frac{\hat{E}[I(u|\hat{W})]}{1 + \varepsilon} \geq \frac{\hat{E}[I(u|W^*)]}{1 + \varepsilon} \geq 1 - \varepsilon \cdot \frac{\hat{E}[I(u|W^*)]}{1 + \varepsilon}$$

which proves the theorem.

Theoretical Analysis. Interestingly, we find out that the framework using either MC or RR sampling to solve PITEX could lead to inefficient solutions due to the unexpected large complexities to generate one sample instance. As we have shown that both methods use the same $\theta_W$ to achieve the same error bound, to prove the above claim, clearly we only need to analyze the expected time complexity of computation for one sample instance of MC and RR.

The complexity can be naturally measured by the expected number of edges visited by MC (RR) sampling for one sample instance, denoted by $\text{ENE}^{MC}$ ($\text{ENE}^{RR}$). Existing work [101] has shown the following lemma for $\text{ENE}^{RR}$.

Lemma 10. [101] Given the set $(R_W(u))$ of vertices reachable by $u$ by removing each edge with $p(e|W) = 0$ from $G$ and the set $(E_W(u))$ of edges connecting vertices in $R_W(u)$, we have

$$\text{ENE}^{RR} = O(|E_W(u)| \cdot E[I(v^{in} \leadsto v^*|W)])$$

(5.5)

where $E[I(v^{in} \leadsto v^*|W)])$ is the influence probability from a randomly selected vertex $v^{in}$, with the probability of selecting $v^{in}$ proportional to its in-degree, to activate another uniformly selected vertex $v^*$.

As no prior work has shown any result for $\text{ENE}^{MC}$ of MC sampling, we first present the following lemma.

Lemma 11. Let $E[I(u \leadsto v^{ot}|W)]$ denote the influence probability from the query user $u$ to activate a randomly selected vertex $v^{ot}$ with the probability of selecting $v^{ot}$ proportional
to its out-degree.

\[
\text{ENE}^{MC} = O\left( |E_W(u)| \cdot \mathbb{E}[\mathcal{I}(u \sim v^*|W)] \right)
\] (5.6)

Proof. Consider an instance of sample \( g \) in MC sampling. Let \( \mathcal{R}_W^g(u) \) denote the set of vertices in \( g \) reachable from \( u \), and \( p_g \) denote the probability of randomly selecting an edge from \( E_W(u) \) that starts from a vertex in \( \mathcal{R}_W^g(u) \). Then, we have \( \text{ENE}^{MC} = \mathbb{E}[|E_W(u)| \cdot p_g] \) where the expectation is taken over the randomness of \( \mathcal{R}_W^g(u) \).

Let us choose a random vertex \( v \in V \) with the probability proportional to its out-degree, denoted by \( \text{deg}(v) \), and let \( 1[v \in \mathcal{R}_W^g(u)] \) be an indicator function such that \( 1[v \in \mathcal{R}_W^g(u)] \) equals 1 if \( v \in \mathcal{R}_W^g(u) \) and equals 0 otherwise. We have:

\[
\frac{\text{ENE}^{MC}}{|E_W(u)|} = \mathbb{E}[p_g] = \sum_g (\Pr[g] \cdot p_g)
\]

\[
= \sum_g \left( \Pr[g] \cdot \frac{\sum_v \text{deg}(v) \cdot 1[v \in \mathcal{R}_W^g(u)]}{|E_W(u)|} \right)
\]

\[
= \sum_v \left( \frac{\text{deg}(v)}{|E_W(u)|} \cdot \sum_g (\Pr[g] \cdot 1[v \in \mathcal{R}_W^g(u)]) \right)
\]

\[
= \sum_v \left( \frac{\text{deg}(v)}{|E_W(u)|} \cdot \Pr[u \sim v] \right) = \mathbb{E}[\mathcal{I}(u \sim v^*|W)]
\]

which completes the proof. \( \square \)

Now, we are ready to analyze the time complexity of the sampling strategies by multiplying \( \text{ENE}^{MC} \) (\( \text{ENE}^{RR} \)) with \( \theta_W \) analyzed in lemmas 8-11. Let \( \Lambda = \frac{\varepsilon + \delta}{\varepsilon + \delta + \log 2} \) and \( \mathbb{E}[\mathcal{I}(u \sim v^*|W)] \) be the influence probability from \( u \) to activate an uniformly selected vertex \( v^* \in \mathcal{R}_W(u) \). We can see that the complexity of the enumeration method with MC is \( O \left( \Lambda \cdot |E_W(u)| \cdot \frac{\mathbb{E}[\mathcal{I}(u \sim v^*|W)]}{\mathbb{E}[\mathcal{I}(u \sim v^*|W)]} \right) \). On the other hand, the complexity of RR is \( O \left( \Lambda \cdot |E_W(u)| \cdot \frac{\mathbb{E}[\mathcal{I}(v^*\sim v^*|W)]}{\mathbb{E}[\mathcal{I}(u \sim v^*|W)]} \right) \).
Figure 5.2: Examples where MC and RR have high computation complexities

Note that in the scenario where $\frac{E[I(u \sim v^*|W)]}{E[I(u \sim v^t|W)]}$ is a constant w.r.t. $G$, MC produces a linear time algorithm. The similar conclusion holds for RR when $\frac{E[I(v^t_{n-1} \sim v^t|W)]}{E[I(u \sim v^t|W)]}$ is also a constant. However, the following two realistic counterexamples pinpoint scenarios where both methods may have quadratic complexities for even a sparse graph, which are prohibitive for processing real-world large SNs.

**Example 12.** In Fig. 5.2 (a), an input graph consists of a root vertex $u$ which has an influence edge to each of the remaining $n$ vertices with a probability of $\frac{1}{n}$. This example pictures the situation where a user who has a lot of followers but has a low impact in the social network.

When $u$ is the query vertex, we have $E[I(u \sim v^t|W)] = 1 \cdot 1 + \sum_{i \in 1..n} 0 \cdot \frac{1}{n} = 1$ and $E[I(u \sim v^*|W)] = \frac{1}{n+1} \cdot 1 + \sum_{i \in 1..n} \frac{1}{n+1} \cdot \frac{1}{n} = \frac{2}{n+1}$. Under such scenario, $\frac{E[I(u \sim v^t|W)]}{E[I(u \sim v^t|W)]} = O(n)$, which brings quadratic complexity to MC.

**Example 13.** In Fig. 5.2 (b), an input graph consists of a central vertex $v$ which has an influence edge to $n$ vertices with a probability 1. There is another set of $n$ vertices which has an influence probability of $\frac{1}{n}$ to $v$. The example presents the situation where a celebrity who has a huge number of followers and also follows a lot of users in the social
network. However, the influence from a celebrity to a normal individual is trivially much higher than the other way around.

When any $u_j$ is the query vertex, we have $\mathbb{E}[I(v^m \leadsto v^*|W)] = \frac{n}{2n^2} \cdot (\frac{1}{2n+1} \cdot 1 + \sum_{i=1..n} \frac{1}{2n+1} \cdot 1) + \sum_{i=1..n} \frac{1}{2n} \cdot 1 = \frac{3n+2}{4n+2}$ and $\mathbb{E}[I(u_j \leadsto v^*|W)] = \frac{1}{2n} \cdot 1 + \frac{1}{2n} \cdot \frac{1}{n} + \sum_{i=1..n} \frac{1}{2n} \cdot \frac{1}{n} = \frac{1}{n} + \frac{1}{2n^2}$ for any $u_j$. Under such scenario, $\frac{\mathbb{E}[I(v^m \leadsto v^*|W)]}{\mathbb{E}[I(u_j \leadsto v^*|W)]} = O(n)$, which brings quadratic complexity to RR.

Given the above counter examples, MC and RR could both take $O(\Lambda \cdot |E_W(u)| \cdot |R_W(u)|)$ to evaluate the influence score which is obviously not scalable as $|R_W(u)| = O(|V|)$ and $|E_W(u)| = O(|E|)$ in highly connected social graphs. In the following section, we develop a series of optimized sampling methods to speed up the influence estimation process.

### 5.3 Optimizing Online Sampling

To reduce the sampling complexity in the above framework, this section develops two techniques, lazy propagation sampling and best effort exploration. The former reduces the sampling cost when evaluating influence spread $\mathbb{E}[I(u|W)]$ for any tag set $W$, while the latter improves the enumeration-based approach by pruning a large number of tag sets without evaluating their influence spread.

#### 5.3.1 Lazy Propagation Sampling

According to our previous analysis, the main reason why MC and RR fail to deliver efficient influence estimation is the unexpected high complexity to generate one sample instance which has to probe a large number of edges. For example, in Fig. 5.2 (a), when MC is applied to estimate $u$'s influence, the generation of each individual sample instance
Algorithm 10: LazySample \((u, G, \theta_W, \delta, \epsilon)\)

1. \(s \leftarrow 0;\)
2. for \(i = 1, \ldots, \theta_W\) do
   3. \(h \leftarrow \{u\}; // The traversal frontier.\)
   4. while \(h \neq \emptyset\) do
      5. \(v \leftarrow h\text{.pop}();\)
      6. if \(v\) is not initialized then
         7. \(c_v \leftarrow 0, h_v \leftarrow \emptyset\)
         8. for \(nbr \in v\text{’s neighbour sets}\) do
            9. \(X(nbr) \leftarrow \text{geometric r.v. instance}\)
            10. \(h_v \leftarrow h_v \cup \{\langle v, nbr, X(nbr)\rangle\}\)
      11. while \(h_v\text{.top}() == c_v\) do
          12. \(\langle v, X(nbr)\rangle \leftarrow h_v\text{.pop}();\)
          13. \(h \leftarrow h \cup \{nbr\}\) if nbr is not visited.
          14. \(X'(nbr) \leftarrow \text{geometric r.v. instance}\)
          15. \(h_v \leftarrow h_v \cup \{\langle nbr, X'(nbr) + c_v\rangle\}\)
      16. \(c_v \leftarrow c_v + 1; s \leftarrow s + 1;\)
      17. if \(\frac{s}{|R_{W(u)}|} \geq 1 + (1 + \epsilon)\frac{\sqrt{2}}{\epsilon^2} \cdot \log\left(\frac{2}{\delta(\Omega_k)}\right)\) then
         18. break
      19. Reset all visited vertex to not visited
   20. return \(\frac{s}{\theta_W}\)

has to probe every \(u\text{’s out-going} edge e\) to test if \(e\) is activated (with a probability \(p(e|W)\)).

Obviously, this process will lead to many “unnecessary” probings for the unactivated edges, as existing works on learning propagation probabilities in real-world SNs often deliver sparse influence graphs [11, 48, 24], i.e., the propagation probability is low for a large number of edges. Similarly, in Fig. 5.2(b), each sample instance generated by RR has to probe all \(v\text{’s in-coming} edges\), even though the activation probabilities are low. Thus the key to speed up the influence estimation is to avoid probing as many such “unnecessary” edges as possible. Towards this goal, we develop a lazy propagation sampling strategy that only probes the edges when they are activated.

The key challenge is to predict the case in which an edge \(e\) will be activated. According to the sampling process, the events of any edge \(e\) being activated across \(\theta_W\) sample
instances can be considered as independent uniform random variables (r.v.s), each of which is equivalent to a coin toss with a head probability of \( p(e|W) \). This means the edge will become activated (and thus needs to be probed) once a head appears after a number of tosses, which is literally a geometric distribution with parameter \( p(e|W) \). Based on this observation, to decide when to probe edge \( e \), we only need to sample a geometric r.v. that determines the next sampling instance in which \( e \) is activated. For example, suppose that \( \theta_W \) of the MC sample instances are generated by starting from \( u \) in Fig. 5.2 (a). Instead of probing every edge \( e \) from \( u \) to \( v_i \) for \( \theta_W \) instances, our proposed strategy first samples a geometric r.v., say 3, for edge \( e \), which means \( e \) will not be activated until the third instance. Thus, it can safely skip probing \( e \) in the first two instances. Then, after the third instance, we sample a geometric r.v. again to determine the next sample instance that probes \( e \). In this way, our strategy only probes the edge \( \frac{\theta_W}{n} \) instead of \( \theta_W \) times in expectation. The method is also applicable for RR and one can picture that the reverse probings from \( v \) to each of its in-coming neighbors in Fig. 5.2 (b) is reduced by \( n \) times.

To show the correctness of the above strategy, we establish the equivalence between the number of heads observed by \( \theta_W \) tosses (the number of times an edge is activated in our problem) and the number of success indicated by a sequence of geometric r.v.s with their sum smaller or equal to \( \theta_W \).

**Lemma 12.** The following two r.v.s are statistically identical: 1) the number of heads observed by \( \theta \) Bernoulli trials with success probability of \( p \); 2) a random number \( Y \) s.t. \( \sum_{i=1}^{Y} X_i \leq \theta \) and \( \sum_{i=1}^{Y+1} X_i > \theta \) where \( X_i \)s are i.i.d geometric r.v.s with parameter \( p \).
Proof. Let $Y$ denote event 1), then $\Pr[Y = y] = \binom{\theta}{y} p^y (1 - p)^{\theta - y}$, $\forall y \in 1..\theta$. Let $Y'$ denote event 2) and $X_i$ are i.i.d geometric r.v.s with probability $p$, we derive the followings:

$$
\Pr[Y' = y] = \sum_{s=y..\theta} \Pr[\sum_{i=1..y} X_i = s] \cdot \Pr[X_{y+1} > \theta - s]
$$

$$
= \sum_{s=y..\theta} \binom{s-1}{y-1} p^y (1 - p)^{s-y} (1 - p)^{\theta-s}
$$

$$
= \left\{ \binom{\theta}{y} + \sum_{s=y+1..\theta} \binom{s-1}{y-1} \right\} \cdot p^y (1 - p)^{\theta-y}
$$

$$
= \binom{\theta}{y} p^y (1 - p)^{\theta-y} = \Pr[Y = y]
$$

where the second equality is based on the fact the sum of $y$ geometric random variable with success probability of $p$ is equal to a negative binomial distribution with parameter $y$ and $p$. Then it is safe to say two events are statistically identical since they have the same probability for all instances. \qed

The pseudo-code of the optimized MC with lazy propagation sampling is presented in Algo. 10. Above all sampling instances, for any $v \in \mathcal{R}_W(u)$, it initializes a heap $h_v$ and pushes every $v$’s out-going neighbor $nbr$ with a geometric r.v., i.e., $(nbr, X(nbr))$ into $h_v$ (lines 6-10). It also maintains a counter $c_v$ to keep track of the number of time $v$ has been visited. Once an r.v. in $h_v$ is equal to $c_v$, the corresponding neighbor will be probed, and then a new r.v. is generated to decide the next time to probe the neighbor (See Lines 11-15). Lastly, we impose a stopping condition to early terminate the sampling process (See Line 17). The intuition is that, when the sum of observed random values are sufficiently large, the estimation can be stopped early since the true value has a large error tolerance range. The proof that such stopping condition ensures the same approximation guarantee as Theorem 7 can be easily established by existing works on martingale simulations [36, 100].
Example 14. Fig. 5.3 shows a simplified example for running the lazy propagation on an influence graph. Let us assume the sampling process always starts from $u$. The shaded nodes are the ones visited during a particular sampling iteration and the updated values are marked in red. In iteration $i = 1$, $u$ is initialized with $h_u$ containing its neighbor $v$ and a geometric instance, i.e. 3. This means the next visit to $v$ is in the third visit, and thus this iteration can safely terminate. It is only in iteration $i = 3$ where $c_u$ is updated to 3 as $u$ is visited in each of the last three sampling processes, that $u$ visits $v$ and re-assigns $v$ with another geometric r.v. Meanwhile, $v$ is visited for the first time ($c_v$ is set to 1) so it will be initialized with $h_v$ containing its neighbors $a$ and $b$. As the earliest visit to $v$’s neighbor happens in the second visit of $v$, the sampling iteration terminates at this point. Finally, $a$ will be visited in the iteration $i = 5$. 

Figure 5.3: Example of lazy propagation algorithm
Lemma 13. Let $\text{ENE}^{\text{LP}}$ denote the expected number of edges visited by a lazy propagation instance under the IC model, the following holds:

$$\text{ENE}^{\text{LP}} = O(|R_W(u)| \cdot \mathbb{E}[\mathcal{I}(u \sim v^*|W)])$$

Proof. Consider an instance of sample $g$ in lazy propagation sampling. Let $R^g_W(u)$ denote the set of vertices in $g$ reachable from $u$, only edges which has both end vertices in $R^g_W(u)$ are traversed. According to the IC model, the influence probability through any edge $(x \rightarrow y)$ is inverse proportional to the in-degree of $y$. If $c(x)$ denote the number of vertices which are active and incident to $x$ \[65, 61, 101\], $\text{ENE}^{\text{LP}}$ is evaluated as the following:

$$\text{ENE}^{\text{LP}} = \sum_g \Pr[g] \cdot \sum_{x \in R^g_W(u)} c(x)$$

$$= \sum_g \Pr[g] \cdot \sum_{x \in R^g_W(u)} \frac{|\{(x, y) \in E | y \in R^g_W(u)\}|}{\text{deg}^{\text{in}}(x)}$$

$$\leq \sum_g \Pr[g] \cdot |R^g_W(u)|$$

$$\leq |R_W(u)| \sum_g \Pr[g] \cdot \frac{|R^g_W(u)|}{|R_W(u)|}$$

$$\leq |R_W(u)| \cdot \mathbb{E}[\mathcal{I}(u \sim v^*|W)]$$

where $v^*$ is uniformly selected from $R_W(u)$. \hfill \Box

With Lemma 13 one can easily see the complexity for the lazy propagation method is $O(\Lambda \cdot |R_W(u)|)$ where $\Lambda = \frac{2+\varepsilon}{\varepsilon^2} (\log(\delta) + \log(\frac{|R|}{|u|}) + \log 2)$, which shows that lazy propagation sampling is indeed more efficient than MC and RR as it visits much smaller number of edges.

127
5.3.2 Best-Effort Exploration

Although enumeration-based sampling solves PITEX with high accuracy, it has to evaluate all size-$k$ tag sets, which is exponential against $k$. Moreover, the complexity of sampling strategies for each tag set $W$ is also expensive. This motivates us to develop effective pruning methods in order to avoid evaluating all $k$-size tag sets.

To this end, we introduce a best-effort exploration strategy. The basic idea of this strategy is to progressively select a tag $w$ into a partial solution tag set, i.e., $W$ with $|W| < k$. Then, it estimates the upper bound of the influence spread for any $k$-size tag set containing the partial solution $W$. Obviously, if the involvement of the newly selected tag makes the partial solution set impossible to be the optimal one (i.e., the upper bound is already smaller than a known solution), we can prune all size-$k$ tag sets containing this partial solution. The essential challenge for the best effort exploration is to estimate the upper bound for the influence w.r.t. any partial tag set. In this chapter, we introduce a bound estimation method by considering two scenarios for the tag-topic distribution ($p(w|z)$): 1) $p(w|z)$ is sparse [88, 106], meaning a large number of tags have no probability to appear in many topics; 2) $p(w|z)$ is dense [72, 110], meaning the topics are often expressed in most of the tags. Under the aforementioned scenarios, we have the following lemma of the upper bound of the influence spread for a partial tag set $W$.

**Lemma 14.** Given a partial tag set $W$ with $|W| < k$, an upper bound of the influence probability for each edge $e \in E$, denoted by $p^+(e|W)$, can be estimated as:

$$
p^+(e|W) = \min \left( \max_{z \in Z} p(e|z), \prod_{w \in W \cup W^*} p(w|z) p(z) \prod_{z' \in Z} p(w|z') p(z') \right)
$$

(5.7)
\[ W.L.O.G, \ p^+(e|\emptyset) = \max_{z \in Z} p(e|z). \]

**Proof.** It is straightforward to see that: \( p^+(e|W) \leq \max_{p(e|W') > 0 \land z \in Z} p(e|Z) \). Thus we are only left to prove: \( p^+(e|W) \leq \text{Eqn. 5.8} \). To achieve this, let \( p^+_z(e|W) = \frac{\prod_{w \in W} p(w|z)p(z)}{\sum_{z' \in Z} \prod_{w \in W} p(w|z')p(z')} \), we have the following inequalities:

\[
\log(p^+_z(e|W)) = \log\left(\frac{\prod_{w \in W} p(w|z)p(z)}{\sum_{z' \in Z} \prod_{w \in W} p(w|z')p(z')}\right) = \sum_{w \in W} \log(p(w|z)p(z)) - \log\left(\sum_{z' \in Z} \prod_{w \in W} p(w|z')p(z')\right) \quad (5.9)
\]

Note when \( p(w|z) = 0 \) for any \( w \in W \), \( p^+_z(e|W) = 0 \) and no estimation is needed. Subsequently, we apply the jensen inequality to Eqn. 5.9.

\[
\log(p^+_z(e|W)) \leq \sum_{w \in W} \log(p(w|z)p(z)) - \sum_{z' \in Z} p(z') \sum_{w \in W} \log(p(w|z')) \\
\leq \log\left(\prod_{w \in W} p(w|z)p(z)\right) - \log\left(\prod_{w \in W} \prod_{z' \in Z} p(w|z')p(z')\right) \\
\leq \log\left(\prod_{w \in W} p(w|z)p(z)\prod_{z' \in Z} p(w|z')p(z')\right) \quad (5.10)
\]

As natural log is a monotone increasing function, \( p^+_z(e|W) \leq \prod_{w \in W} \frac{p(w|z)p(z)}{\prod_{z' \in Z} p(w|z')p(z')} \), \( \forall W \subset \Omega \). The rest is simply to plug in Eqn. 5.1 to complete the proof. \( \square \)

We note that given a tag set \( W \), \( p^+(e|W) \geq p(e|W') \) for any \( W' \supset W \) s.t. \( |W'| = k \). Based on the bounds \( p^+(e|W) \) of edge probability, we can devise our best exploration strategy to enable early termination for PITEX.

Algorithm 11 provides the pseudo-code of the best effort framework. Let \( \Omega \) to be an ordered set where the order can be arbitrary, and we denote \( w_i < w_j \) if \( w_i, w_j \in \Omega \) and \( i < j \). It employs a max-heap \( \mathcal{H} \) to preferentially access (partial) solutions with larger upper bound of influence spread. Initially, it inserts an empty tag set \( \langle \emptyset, 0 \rangle \) into
Algorithm 11: BestEffortSample \((u, G, k, \varepsilon, \delta)\)

1. Initialize a max-heap \(H\);
2. \(W^* \leftarrow \emptyset, I^* \leftarrow 0\);
3. Insert \((\emptyset, 0)\) into \(H\);
4. While \(H \neq \emptyset\) do
   5. \(\langle W, I \rangle \leftarrow H_.\text{pop}()\);
   6. If \(W = k\) then
      7. \(I \leftarrow \text{EstimateInfluence} (u, G, W, \varepsilon, \delta)\);
      8. If \(I > I^*\) then \(W^* \leftarrow W, I^* \leftarrow I\);
   9. Else
      10. \(I \leftarrow \text{EstimateUpperBound} (u, G, W, \varepsilon, \delta)\);
      11. If \(I \leq I^*\) then continue;
      12. For \(w < w' \forall w' \in W \land w \in \Omega\) do
         13. \(W' \leftarrow W \cup \{w\}\);
         14. Insert \((W', I)\) into \(H\);
   15. return \((W^*, I^*)\);

\(H\) and selects tag sets iteratively. In each iteration, it pops a partial solution \(W\) with its upper bound \(I\) from \(H\). If \(W\) already has \(k\) tags, it simply computes the estimated influence spread by applying the lazy propagation sampling to update the current best solution (Lines 6-9). Otherwise where \(W\) has less than \(k\) tags, it first estimates the upper bound for \(W\) using \(\text{EstimateUpperBound}\) (Line 11) and prunes \(W\) if its upper bound is smaller or equal to the current best size-\(k\) solution. Lastly it adds one more tag \(w\) into \(W\) for forming \(W'\) and inserts \(W'\) back into \(H\). The algorithm terminates when \(H\) is empty and returns the best solution \((W^*, I^*)\).

**Theoretical analysis:** Note that the number of tag sets examined in the enumeration-based sampling is \(\binom{|\Omega|}{k}\) whereas the number is \(\sum_{i=1, \ldots, k} \binom{|\Omega|}{k}\) for best-effort exploration in the worst case. To retain the same theoretical guarantee as stated in Theorem 7 the sample size \(\theta_W\) required for best effort sampling w.r.t. any partial tag set \(W\) should
satisfy:
\[
\theta_W \geq \frac{2 + \varepsilon}{\varepsilon^2} \cdot |\mathcal{R}_W(u)| \cdot \frac{\log(\delta) + \log \left( \sum_{i=1}^{k} \binom{|\Omega|}{i} \right) + \log 2}{\mathbb{E}[I(u|W)]} \tag{5.11}
\]

In the “worst” case, the best-effort strategy is required to estimate the influence for \(\phi_k\) tag sets, where \(\phi_k = \sum_{i \in [1,k]} \left( \binom{|\Omega|}{i} \right) \). However, we can derive that: \(\phi_k \leq \left( \binom{|\Omega|}{k} \right) \frac{|\Omega| - k + 1}{|\Omega| - 2k + 1} \).

Since in practice \(k \ll |\Omega|\) which leads \(O(\phi_k) = O\left( \binom{|\Omega|}{k} \right) \). Thus the “worst” complexity of best-effort sampling is the same as that of enumeration-based sampling.

### 5.4 Real-time Influence Estimation

Although the online optimization strategies in the previous section can significantly improve the performance, they still cannot enable online exploration, as influence estimation incurs high computational complexities, and, even worse, the estimation may be invoked many times. To address this issue, this section presents more efficient influence estimation. We first introduce an index-based estimation method in Sec. 5.4.1 and then discuss pruning strategies and an efficient materialization technique for reducing the index size in Sections 5.4.2 and 5.4.3 respectively.

#### 5.4.1 RR-Graph Index Structure

The limitation of our previous influence estimation is that it has to on-the-fly regenerate samples for each user and tag set, which is very expensive. This motivates us to develop a more efficient index-based method. Different from the previous one, this method performs sampling in an offline manner and constructs an index structure over the obtained samples. For online estimation, given any user and any tag set, it estimates the influence spread based on the constructed index, instead of regenerating the samples.
The design of the index structure is inspired by the RR sampling. Recall that RR sampling uniformly samples a vertex \( v_i \) and checks if \( u \) influences \( v \) w.r.t tag set \( W \). As many vertices sampled may be far away from \( u \), redundant samples are generated, which leads to an inefficient solution. However, such a disadvantage could become an advantage for estimating influence for any query user (instead of a specific user) as the uniformly sampled vertices are generated independently of the query users. Based on this idea, we design the reverse reachable sample graph (or RR-Graph for simplicity) structure, which is formally defined as follows.

**Definition 6.** (Reverse Reachable Sample Graph) Given a social network \( G = (V, E) \) and a vertex \( v \), reverse reachable sample graph of \( v \), denoted by \( G^\text{RR}_v \), is sampled as follows: 1) Generate a random number \( c(e) \in [0, 1] \) for all \( e \in E \); 2) \( V(v) \subseteq V \) contains vertices that reach \( v \) in \( G \) after removing all edges s.t. \( p(e) < c(e) \), where \( p(e) = \max_{i=1}^Z p(e|z_i) \); 3) \( E(v) \subseteq V \) contains edges with both ends in \( V(v) \) and associated with their \( c(e) \).

We can see that \( G^\text{RR}_v \) can be used as an instance of sample for any user and tag set in RR sampling due to the following reasons. First, as \( p(e) = \max_{i=1}^Z p(e|z_i) \geq p(e|W) \) for any \( W \), it will not miss any vertex that would influence \( v \). Second, given any \( W \), \( c(e) \) can be used to examine if removing edge \( e \) by simply checking if \( c(e) > p(e|W) \). In such a way, a RR-GRAPH generated can guarantee no vertex is missed from the corresponding RS iteration, and thus assure influence spread is not underestimated for any user and any tag set. The reason is that the edge probability, i.e. \( p(e) \), on which we construct the RR-GRAPHS, is larger than \( p(e|W) \) for all the edges and all the tag combinations.

Next, we introduce a key operation on RR-GRAPH, namely tag-aware reachability, which is defined as follows.
Definition 7. (Tag-Aware Reachability in RR-Graph) Given a tag set $W$ and a RR-Graph $G_{v}^{RR}$, a vertex $u$ is said to reach $v$, denoted by indicator function $1[u \rightsquigarrow v_i | G_{v_i}^{RR}, W] = 1$, if there exists a path in $G_{v}^{RR}$ such that all edges in the path satisfy $p(e|W) \geq c(e)$.

Example 15. Consider our running example with social network $G$ shown in Fig. 5.1. Fig. 5.4 shows four RR-Graphs, i.e. $G_{u_2}^{RR}$, $G_{u_6}^{RR}$, $G_{u_3}^{RR}$ and $G_{u_7}^{RR}$, with the random numbers associated with their edges. Moreover, given tag set $W = \{w_3, w_4\}$, we can see $u_1$ cannot reach $u_2$ in $G_{u_2}^{RR}$ since $p(u_1 \rightarrow u_2 | W) = 0.13 < c(u_1 \rightarrow u_2) = 0.3$. In contrast, $u_1$ can reach $u_6$ in $G_{u_6}^{RR}$ given $W$ via $u_1 \rightarrow u_3 \rightarrow u_4 \rightarrow u_6$ as $p(e|W) \geq c(e)$ for all edges in the path.

Now, we are ready to present the index-based influence estimation, which is described in Algorithm 12. For offline sampling, it generates a sufficient amount $\theta$ (determination of $\theta$ will be discussed later) of RR-Graphs for randomly picked vertices. For online estimation, it simply checks tag-aware reachability from $u$ to $v$ in each RR-Graph $G_{v}^{RR}$, and estimates influence as $\frac{\sum_{i=1}^{\theta} 1[u \rightsquigarrow v_i | G_{v_i}^{RR}, W]}{\theta} \cdot |V|$. This estimation is much more efficient than previous methods, as the costly sampling process is avoided and the RR-Graphs are
Algorithm 12: EstimateInfluence+ \((u,G,W,\varepsilon,\delta)\)

```
// Offline -- RR-Graph sampling
1 Compute the offline sample size \(\theta\);
2 for \(i = 1, \ldots, \theta\) do
3 Sample a random vertex \(v_i\) from \(V\);
4 \(G_{vi}^{RR} \leftarrow \text{GenerateRRGraph}(G,v_i)\);
// Online -- RR-Graph matching
5 for \(\forall G_{vi}^{RR} \text{ s.t. } u \in G_{vi}^{RR}\) do
6 \(1[u \leadsto v_i \mid G_{vi}^{RR},W] \leftarrow \text{IsReachable}(u,v_i,G_{vi}^{RR},W)\);
7 \(\hat{\mathbb{E}}[\mathcal{I}(u\mid W)] \leftarrow \frac{\sum_{i=1}^{\theta} 1[u \leadsto v_i \mid G_{vi}^{RR},W]}{\theta} \cdot |V|\);
8 return \(\hat{\mathbb{E}}[\mathcal{I}(u\mid W)]\)
```

usually much smaller than the original graph \(G\). Moreover, as the samples are generated offline, we only process RR-Graphs containing \(u\) since \(u\) cannot reach the others.

**Example 16.** To estimate the influence of \(u_3\) for the tag set \(W = \{w_3, w_4\}\), the index-based method first examines RR-Graphs containing \(u_3\), i.e., \(\{G_{u_6}^{RR}, G_{u_4}^{RR}, G_{u_7}^{RR}\}\). As we only have \(1[u_3 \leadsto u_6 \mid G_{u_6}^{RR},W]\) and \(1[u_3 \leadsto u_7 \mid G_{u_7}^{RR},W]\) being 1, the influence can be estimated by \(\frac{2}{3} \cdot 7 = 3.5\).

**Determination of Offline Sample Size.** To guarantee the theoretical bounds mentioned previously, we need to make sure a sufficient amount \(\theta\) of RR-Graphs are sampled offline. To this end, we have that whenever \(\theta\) satisfies

\[
\theta = \frac{2 + \varepsilon}{\varepsilon^2} \cdot |V| \cdot (\log(\delta) + \log\phi_K + \log 2),
\]

where \(K\) is a parameter such that \(K\) is an upper bound of any possible \(k^2\), the sampling-based framework with the index-based influence estimation has a \(\frac{1 - \varepsilon}{1 + \varepsilon}\) approximation ratio to the optimal solution with a \(1 - \delta^{-1}\) probability (which can be proved similarly as Theorem 7).

\(^2\)In PITEX, \(k\) will not be too large as any content of a social user containing too many tags will not be practical. In our experiments, we set \(K = 10\).
Complexity Analysis. To study the time complexity of EstimateInfluence+, we have the following lemma.

Lemma 15. For any \( u \) and \( W \), the expected time to estimate \( \mathbb{E}[\mathcal{I}(u|W)] \) using Algo. 12 is \( O(\mathbb{E}^2[I(u*)])^{\frac{\log(\delta) + \log K + \log \log 2}{\epsilon^2}} \) where \( \mathbb{E}[\mathcal{I}(u*)] \) is the influence spread of \( u \) on a social graph with probability of each edge being \( p(e) = \max_{i=1}^Z p(e|z_i) \).

Proof. As the influence of \( u \) estimated directly by RR-Graphs generated offline is computed as \( \mathbb{E}[\mathcal{I}(u*)] = \frac{r}{\theta} |V| \) where \( r \) is the number of RR-Graphs containing \( u \). Thus \( r = \mathbb{E}[\mathcal{I}(u*)] |V|^{\theta} \). This means if the first \( \theta' = \frac{\theta}{\mathbb{E}[\mathcal{I}(u|W)]} \) RR-Graphs are used to estimate \( \mathbb{E}[\mathcal{I}(u|W)] \), the number of RR-Graphs containing \( u \) is \( r' = \frac{r}{\mathbb{E}[\mathcal{I}(u|W)]} \). Moreover, according to Lemma 9, using \( \theta' \) RR-Graphs for estimation can deliver an approximation of error ratio between \((1 - \varepsilon, 1 + \varepsilon)\) with probability of at least \( 1 - \delta - \frac{1}{\Omega_K} \). It follows that the expected estimation complexity is computed by \( r' \cdot \text{ENE} \) where \( \text{ENE} \) is the expected number of visited edges to compute one RR-Graph.

Given a tag set \( W \), for each RR-Graph \( G^\text{RR}_v \) containing \( u \), we can perform a breadth first search (BFS) from \( u \) to check if \( u \) reaches \( v \) in \( G^\text{RR}_v \). Consider an instance of sample \( g \). Let \( \mathcal{R}^g(u) \) denote the set of vertices in \( g \) reachable from \( u \) and \( \mathcal{R}^g_W(u) \) denote the set of vertices in \( g \) reachable from \( u \) after removing any edge \( e \) if \( p(e|W) < c(e) \), we have \( \text{ENE} = \sum_g \text{Pr}[g] \cdot \sum_{x \in \mathcal{R}^g_W(u)} |\{(x, y) \in E | y \in \mathcal{R}^g(u)\}|. \) This is because we only visit the vertices in \( \mathcal{R}^g_W(u) \) but probe edges which points to a vertex in \( \mathcal{R}^g(u) \). Since \( \sum_{x \in \mathcal{R}^g_W(u)} |\{(x, y) \in E | y \in \mathcal{R}^g(u)\}| \leq c \cdot \mathbb{E}[\mathcal{I}(u|W)] \cdot |\mathcal{R}^g(u)| \) for some \( c \), we have

\[
\text{ENE} \leq O\left(\mathbb{E}[\mathcal{I}(u|W)] \cdot \sum_g \text{Pr}[g] \cdot |\mathcal{R}^g(u)|\right) \leq O\left(\mathbb{E}[\mathcal{I}(u*)] \mathbb{E}[\mathcal{I}(u|W)]\right)
\]

Given the above derivation, the expected estimation complexity is bounded by \( r' \cdot \text{ENE} \leq \frac{r}{\mathbb{E}[\mathcal{I}(u|W)]} \cdot \mathbb{E}^2[I(u*)] \). Finally, Lemma 15 is obtained by simplify the expression. \( \square \)
The term $E^2[I(u\mid \ast)]$ is often small due to the power law principle of social networks when $u$ is a randomly selected user. Thus, Algo. 12 can be much more efficient than previous influence estimation methods using online sampling. However, $E^2[I(u\mid \ast)]$ of an influential user could still be significantly large to prevent efficient query processing.

5.4.2 Effective Pruning Techniques

In this section, we present effective pruning techniques to further speed up the checking of tag-aware reachability in RR-GRAphs, i.e., IsReachable in Algo. 12. Note that existing graph reachability techniques [61, 109] cannot be applied as they assume static graph structure or only allow incremental insertion/deletion of vertices/edges. However, each edge in an RR-GRAph exists only when its influence probability $p(e\mid W) \geq c(e)$ for a given $W$. Thus the graph structures may be significantly vary for different tag sets.

To address this issue, we develop a filter-and-verification approach. Given a user $u$ and a tag set $W$, the filter step prunes the unpromising RR-GRAphs and generates a candidate set. Then, the verification step checks reachability for every candidate RR-GRAph $G^{\text{RR}}_v$ by removing the edges of $G^{\text{RR}}_v$ s.t. $p(e\mid W) < c(e)$ and performing breath-first search from $u$ in the resulted graph. This section focuses on presenting effective pruning techniques in the filter step.

The idea of our pruning technique is inspired by the edge cut in graph connectivity. Given a user $u$ and an RR-GRAph $G^{\text{RR}}_v$, we select a set of edges such that $u$ reaches $v$ in $G^{\text{RR}}_v$ given tag set $W$ only if there exists at least one edge in the selected set satisfying $p(e\mid W) \geq c(e)$. Thus, instead of traversing $G^{\text{RR}}_v$, we can prune $G^{\text{RR}}_v$ if all selected edges meet $p(e\mid W) < c(e)$. Obviously, the quality of the selected edge cut determines the pruning effectiveness. Intuitively, we want to select a set of edges which are close to inactivated status, i.e. $p(e)$ is slightly larger than $c(e)$ for an edge $e$ in a cut. Since $p(e\mid W) \leq p(e)$,
the cut could easily prune corresponding RR-GRAPH if \( p(e|W) < c(e) \). The edge cut construction is essentially an s-t minimum cut problem. Although the problem has been extensively studied \([43, 12, 56]\), existing methods take at least a quadratic time (w.r.t. input graph size) to obtain merely an approximate solution on a graph with non-integral weights. To efficiently construct the cut, we apply a simple yet effective approach. By comparing two cuts: \( E_{\text{cut}}(u|v)' \) and \( E_{\text{cut}}(u|v)'' \), where \( E_{\text{cut}}(u|v)' \) consists of u’s out-going edges since u’s neighbor can reach v and \( E_{\text{cut}}(u|v)'' \) consists of edges from v’s incoming neighbors, who can be reached by u, to v, we take the cut with a higher chance to prune as the filter.

**Example 17.** Consider RR-GRAPH \( G_{u_7}^{RR} \) in Fig. 5.4. We compare two cuts: \( E_{\text{cut}}(u_3|u_7)' = \{e_1 = (u_3, u_4), e_2 = (u_3, u_6)\} \) and \( E_{\text{cut}}(u_3|u_7)'' = \{e_3 = (u_4, u_7), e_4 = (u_6, u_7)\} \). By assuming values of \( p(e|W) \) are independent uniform r.v.s in \([0, p(e)]\), we can deduce the probabilities of the two cuts to prune \( G_{u_7}^{RR} \): 0.2 and 0.1 respectively. Thus we take \( E_{\text{cut}}(u_3|u_7)' \) as the edge cut for \( G_{u_7}^{RR} \).

To enable fast pruning on top of the cut constructed, we employ an inverted index structure. Given a user u in a PITEX query, we first select the subset \( G_u^{RR} \) of the RR-GRAPHs that contain u, and compute edge cut \( E_{\text{cut}}(u|v) \) for each \( G_v^{RR} \in G_u^{RR} \). Then, we construct inverted lists that map edges to RR-GRAPHs as follows. An inverted list of edge e contains a sorted set of RR-GRAPHs satisfying \( e \in E_{\text{cut}}(u|v) \) where the RR-GRAPHs are sorted by the \( c(e) \) of e in ascending order. These inverted lists can be used to facilitate pruning for various tag sets. Specifically, given a tag set W, we examine each edge in the inverted lists. For each edge, we simply compute \( p(e|W) \), scan the inverted list of e until \( p(e|W) < c(e) \), and include all the visited RR-GRAPHs into the candidate set of the filter step. It is not difficult to show that all the unvisited RR-GRAPHs can be safely pruned without any additional computation.
Algorithm 13: RetainRRGraphs \((u, G)\)

1. \(G' = (V', E') \leftarrow \) a lazy sample from \(u\) on \(G\) (Algo. 17)
2. \(v \leftarrow\) a uniform sample from \(V'\)
3. \(V(v) \leftarrow \{v' \mid v' \in V' \land (v' \sim v)\} \) on \(G'\)
4. \(E(v) \leftarrow \{(v'', v') \mid (v'', v') \in E' \land v'' \in V(v)\}\)
5. \(c(e) \leftarrow\) a uniform sample in \([0, p(e)) \forall e \in \text{eset}(v)\)
6. return \(G^\text{RR}_v = (V(v), E(v))\)

Example 18. Consider the RR-Graphs in Fig. 5.4 and a query user \(u_3\). To answer a PITEX query of \(u_3\), we first select the RR-Graphs containing \(u_3\), i.e., \(\{G^\text{RR}_{u_6}, G^\text{RR}_{u_4}, G^\text{RR}_{u_7}\}\). Then, we compute edge cuts and construct the inverted lists. Consider two edges, \(e_1 = (u_3, u_4)\) and \(e_2 = (u_3, u_6)\). The inverted lists are \(e_1 \rightarrow \{\langle G^\text{RR}_{u_7}, 0.2 \rangle, \langle G^\text{RR}_{u_6}, 0.4 \rangle, \langle G^\text{RR}_{u_4}, 0.6 \rangle\}\) and \(e_2 \rightarrow \{\langle G^\text{RR}_{u_6}, 0.2 \rangle, \langle G^\text{RR}_{u_7}, 0.4 \rangle\}\). Given a tag set \(W = \{w_1, w_2\}\), we compute \(p(e_1|W) = 0\) and \(p(e_2|W) = 0.25\). Then, we can skip the inverted list of \(e_1\) as all \(c(e)\) in the list is larger than \(p(e_1|W)\). Similarly, for the inverted list of \(e_2\), we only need to visit \(G^\text{RR}_{u_6}\). Finally, we obtain a RR-Graph candidate set \(\{G^\text{RR}_{u_6}\}\). We can see that only one out of five RR-Graphs needs to be accessed, while the remaining are safely pruned.

5.4.3 RR-Graphs Delay Materialization

To ensure the theoretical guarantee of the RR-Graphs based scheme, a large number of RR-Graphs instances are pre-computed and materialized. However, the size of all RR-Graph instances might be too large to fit into the memory for query processing on large graphs. To resolve this issue, we develop a delay materialization approach. The idea is to avoid storing the RR-Graphs during the index phase and only record how many RR-Graphs contain a user \(u\), i.e. \(\theta(u)\), for all \(u \in V\). Whenever a user \(u\) issues a query, we “recover” \(\theta(u)\) RR-Graphs and the rest of the processing remains the same as previously described in this section.
Example 19. The delay materialization scheme maintains seven entries for the RR-Graphs shown in Fig. 5.4. \( \theta(u_1) = 2, \theta(u_2) = 1, \theta(u_3) = 3, \theta(u_4) = 3, \theta(u_5) = 0, \theta(u_6) = 2, \theta(u_7) = 1 \). If \( u_3 \) issues a query, after generating three RR-Graphs, e.g. \( \{G_{u_6}^{RR}, G_{u_4}^{RR}, G_{u_7}^{RR} \} \), the processing continues as illustrated in Example 18.

Since the delay materialization scheme does not store any RR-Graph, to maintain the theoretical guarantee, it is crucial to ensure the samples generated during the query phase retain the same random distributions of the RR-Graphs initially constructed offline. Note that we cannot simply generate RR-Graphs by its definition since the naive scheme would likely generate RR-Graphs which exclude the query user \( u \). This contradicts to the fact that all RR-Graphs which needs to be recovered in the query phase must contain \( u \). There are two major issues for recovering the RR-Graphs: 1) the probability to sample any RR-Graph structure in the query phase must be equal to the one sampling the same RR-Graph structure which contains the query user in the index phase; 2) the random value \( c(e) \) generated on each edge \( e \) of the recovered RR-Graphs must have the same distribution with that on the edges of the RR-Graphs originally generated offline.

For issue 1), a subgraph \( G' = (V', E') \) is first extracted by a lazy sample \( g \) (Sec. 5.3.1) from \( u \) on \( G \), where \( V' \) are the activated vertices in \( g \) and \( E' \) are the live edges in \( g \). Then, we uniformly sample a vertex \( v' \in V' \) and recover a RR-Graph as \( G_{v'}^{RR} = (V(v'), E(v')) \), where all vertices in \( V' \) reach \( v \) on \( G' \) and all edges have their both ends in \( V' \). For issue 2), since the lazy sample (Algo. 10) does not assign a Bernoulli r.v. to each edge to decide if the edge is live or not, we assign a random value \( c(e) \in [0, p(e)) \) to each edge of the recovered RR-Graph for further influence estimations. The detailed procedure is presented in Algo. 13. The following theorem ensures the correctness for the recovering scheme.
Theorem 8. Using RR-Graphs recovered by Algo. [13] to estimate the influence for any user \( u \) and tag set \( W \) has the same approximation guarantee as that estimated by Algo. [12].

Proof. Let \( v^* \) to be a uniformly selected vertex from \( V \) and \( X \) to be the event that a particular RR-Graph of \( v^* \), i.e. \( G_{v^*}^{\text{RR}} \) is sampled given the condition that \( G_{v^*}^{\text{RR}} \) contains the query user \( u \). Moreover, let \( g \) denote a random subgraph of \( G \) by removing each edge with a probability of \( 1 - p(e) \) and \( R_g(u) \) denote the set of vertices reached by \( u \) on \( g \) with \( R(u) \) as the random version of \( R_g(u) \) by considering the randomness in \( g \). We then derive the following equation:

\[
\Pr[X] = \Pr[G_{v^*}^{\text{RR}}|u \in G_{v^*}^{\text{RR}}] = \sum_g \Pr[g] \cdot \sum_{v^* \in V} \Pr[v^*] \cdot b_g(u \leadsto v^*)
\]

\[
= \Pr[g] \cdot \sum_{v^* \in R_g(u)} \Pr[v'] \cdot b_g(u \leadsto v')
\]

\[
= \Pr[G_{v'}^{\text{RR}}|v' \in R(u)]
\]

where \( b_g(u \leadsto v^*) = 1 \) if \( u \) reaches \( v^* \) on \( g \) and 0 otherwise. Note that the resulting event is exactly the same as the procedure in Algo. [13].

At this point, we establish the statistical equivalence between pure index and delay sampling approaches in terms of generating graph structures. We has yet to prove the random values generated by pure index and delay sampling approaches share the same distribution. Let \( c(e) \) denote the random values generated by the pure index approach, the edge being considered in a RR-Graph must be live, i.e. \( c(e) < p(e) \). Given any tag set \( W \) and \( p(e|W) < p(e) \), the following equality is easy to obtain: \( \Pr[c(e) < p(e|W)|c(e) < p(e)] = \frac{\Pr[c(e) < p(e|W)]}{\Pr[c(e) < p(e)]]} = \frac{p(e|W)}{p(e)} \). This is exactly the probability to generate a random value \( c'(e) \in [0, p(e)) \) which is live w.r.t. \( W \), as presented in Algo. [13].
5.5 Experiments

This section evaluates the performance of our approaches for solving PITEX. First, we examine the empirical convergence of the sampling-based framework. Then, we compare the approaches in various parameter settings. Finally, we evaluate the scalability and conduct a case study.

5.5.1 Experimental Setup

Datasets. We conduct experiments on four real datasets, namely lastfm, diggs, dblp and twitter. 1) lastfm is a social music sharing dataset from an online site\(^3\) lastfm contains a social network and an action log which records users’ activities of voting items (i.e., musics). 2) diggs is an open social news dataset\(^4\). Like lastfm, it also contains a social network and an action log. 3) dblp is a DBLP co-author graph which is downloaded from an online academic search service\(^5\). 4) twitter is a social network built from the re-tweet and reply actions of users in Twitter. The dataset is downloaded from SNAP\(^6\).

We adopt the TIC model [11] to compute topic-aware influence probability \(p(e|z)\) of each edge for lastfm and diggs based on their action logs. Following previous settings [28, 11], we set the number of topics of these two datasets as 20. We also select top-50 frequent items (i.e., music/news) in these datasets to form the tag set \(\Omega\). Since dblp has no action log, we follow the settings in [28] to use research fields as topics and compute \(p(e|z)\) of two authors by categorizing their related conferences using the topics. We also use the keywords in the names of the conferences to formulate tag set \(\Omega\). For twitter dataset, we first collect top-250 hashtags from re-tweets and replies in the edges as the tag set \(\Omega\), and consider all hashtags of an individual user as a document. Then, we apply

\(^3\)http://www.last.fm/
\(^4\)http://www.isi.edu/lerman/downloads/digg2009.html
\(^5\)http://dblp.uni-trier.de/xml/
\(^6\)http://snap.stanford.edu/
LDA \cite{15} on all the documents and obtain the topic distribution of each user. Given an edge $e = (u, v)$, we compute $p(e|z)$ based on the topic distributions of $u$ and $v$. The statistics of these four datasets are listed in Table 5.2.

**Query set and parameters.** To evaluate the performance, for each dataset, we first filter the users with no outgoing edge and divide the rest of the users into three groups based on their out-degrees: high (top 1%), mid (top 1-10%) and low (the rest) with “mid” as the default group. For each user group, we generate 100 PITEX queries with randomly selected users within the group and average the results of the queries. We also consider various parameter settings, including $\varepsilon$, $\delta$ and $k$, and the effect of the parameters will be discussed in Sec. 5.5.3.

**Approaches.** We use best-effort exploration (Sec. 5.3.2) to compare the online influence estimation (Sec. 5.2 and 5.3): MC, RR and lazy sampling (LAZY), as well as the offline methods (Sec. 5.4.1): index-based influence estimation (INDEXEst), index-based estimation with pruning (INDEXEst+) and delay index materialization (DELAYMat). Result of the simple enumeration-based method is not reported because it is not scalable and always worse than best-effort exploration.
Index Sizes and Time. We report the index sizes and construction time for all offline methods in Table 5.3. The size of RR-GRAPHs index is much larger than the original data size, since millions of RR-GRAPHs are stored in order to answer queries with accuracy guarantee. DelayMat (Sec. 5.4.3) drastically reduces the index sizes as it only keeps one entry for each user to record how many RR-GRAPHs contain the user. Moreover, DelayMat has a faster construction time than RR-GRAPHs index since it does not need to physically store the RR-GRAPH instances.

Experiment Settings. All the methods are implemented with C++ and ran on a CentOS server (Intel i7-3820 3.6GHz CPU with 8 cores and 60GB RAM).

Figure 5.5: Evaluating empirical convergence of sampling-based framework
5.5.2 Evaluation of Sampling Convergence

We first evaluate the empirical convergence of the proposed sampling-based framework. For each dataset, we consider the user with the largest out-degrees and its most influential tag as the tag set $W$. Then, we vary $\theta_W$ and use MC/RR/Lazy sampling strategy to estimate the influence.

Results in Fig. 5.5 show an interesting pattern. Recall that we have proved that both MC/RR/Lazy sampling strategies need the same sample size $\theta_W$ to guarantee the same error bound (see Lemmas 8 and 9). However, as observed from Fig. 5.5, MC/Lazy sampling converges faster than RR sampling: the former requires a smaller sample size than the latter for influence estimation. This can be explained by their estimation mechanisms. MC/Lazy uses $\theta_W$ i.i.d. $[0, 1]$ r.v.s while RR uses $\theta_W$ Bernoulli r.v.s for influence estimation. Ensuring theoretical bound of both methods requires the chernoff-hoeffding inequality, and Bernoulli r.v.s is the worst case for the chernoff-hoeffding inequality to hold [34]. Thus, RR converges slower than MC/Lazy.

5.5.3 Comparison of Approaches

In this section, we compare the approaches proposed in this chapter by varying parameters.

Varying user group. We first vary user groups within which queries are generated from. Other parameters, i.e., $\varepsilon$, $\delta$, $k$, are fixed to the default values: 0.7, 1000, 3 respectively, and the defaults are used throughout this section unless otherwise specified. We examine the efficiency of computing the most influential tag set as well as its influence spread.

Efficiency of the compared methods is presented in Fig. 5.6. Among the online sampling methods, Lazy shows superior performance over MC/RR on all the datasets, which validates our claims in Sec. 5.2. Moreover, we can see that index-based approach INDEX-
Figure 5.6: Efficiency comparison of methods when varying query user group.

Est significantly outperforms the online sampling methods. Take the dblp dataset as an example: INDEXEst achieves 1031x, 504.7x, and 1562x speedups for the high, mid, and low user group respectively. This is because the costly sampling process is avoided and influence can be estimated using the pre-computed RR-Graphs in INDEXEst. On top of that, INDEXEst+ further improves the efficiency and outperforms INDEXEst by 4.9x, 6.4x, 4.7x for different user groups respectively, which shows the effectiveness of the edge-cut based pruning techniques. DelayMat runs slightly slower than INDEXEst+ (but still performs better than INDEXEst), as it has to “recover” RR-Graphs on-the-fly. However, we use it to trade significant reduction on index sizes as indicated in Table 5.3.

In addition, All the methods run faster for user group with lower degrees. The reason
is that users with lower degrees tend to have less candidate tag sets with high influence scores, which makes the best-effort strategy easier to perform pruning.

The influence spreads achieved by different methods are shown in Fig. 5.7. We can see that the approaches have comparable influence spread in most of the cases, because all returned results are within the $1 - \frac{\varepsilon}{1 + \varepsilon}$ approximate ratio as stated in Theorem 7. We also note that the difference in the results generated from the various approaches is slightly larger on dblp and twitter datasets than that on the lastfm and diggs datasets. The reason is that the variance of random samples generated from larger social networks is higher, which leads to more volatile results.
As Lazy always shows the best performance among all online sampling methods and the influence scores returned are also within the error bound, we only compare Lazy with other offline solutions in the remaining part of this section.

Varying parameter $\varepsilon$. We vary $\varepsilon$ from 0.3 to 0.9. As shown in Fig. 5.8, the running time of all methods drops with a smaller $\varepsilon$ as a larger $\varepsilon$ leads to fewer samples being generated (by Lemma 8 and 9). Similar to the results when varying user groups, INDEXEst shows its dominating performance over the online Lazy sampling method, e.g. respectively achieving speedups up to 712x, 2417x, 849x, and 90x on the four datasets. The edge-cut pruning techniques of INDEXEst+ further boost the performance from INDEXEst by 5x, 4x, 4x and 2x on the datasets respectively. These results again demonstrate the
Figure 5.9: Influence spread comparison of methods when varying $\varepsilon$

effectiveness of the RR-GRAPH-based method and edge-cut based pruning techniques. In addition, DELAYMAT is as efficient as INDEXEST+ but only requires to use much smaller index spaces.

Fig. 5.9 shows the influence spread achieved by different methods. We can see that the influence spreads of these methods are quite close when $\varepsilon$ is small (e.g., $\varepsilon = 0.3$), and when $\varepsilon$ is larger, the differences also become larger. Since a larger $\epsilon$ leads to fewer samples being generated, which in turn makes the influence estimation less accurate. Note that the fluctuations in the influence result reported across all datasets are similar when varying other parameters, e.g., vary user group (Fig. 5.7), and thus we omit the results on influence spread in the remaining of this section. We also omit the results for varying
δ is because setting a smaller δ does not make the estimation differs significantly from that with a larger δ according to our experiments.

**Varying parameter k.** We examine the performance of all the methods by varying the tag number k.

We report the results in Fig. 5.10. First, when comparing the approaches, we have similar observations: INDEXEst is at least two orders of magnitude better than Lazy, and INDEXEst+ continues to be superior against INDEXEst. Second, we also observe that the improvement achieved by INDEXEst+ and DELAYMAT against INDEXEst increases for larger k. This is because, when k is larger, INDEXEst needs to check more tag sets and the pruning technique based on inverted index can enable more filtering power.
Third, although the running time increases when increasing $k$ in most cases, the running time of all the approaches does not explode exponentially w.r.t. increasing $k$ despite an exponential growth in the number of k-size tag sets (Fig. 5.10). This is because the tag-topic probability densities\footnote{The tag-topic probability density is the ratio between the number of non-zero $p(w|z)$ entries and $|\Omega| \cdot |Z|$.} in most datasets are low, e.g., 0.16, 0.08, 0.32 and 0.17 for lastfm, diggs, dblp and twitter respectively. Note that when the density value is low, more tag sets tend to have zero probability against many topics and they are efficiently pruned by our best effort strategy (note that all the reported approaches adopt best effort exploration as mentioned previously).

### 5.5.4 Evaluation on Scalability

This section evaluates the scalability. we vary the number of tags $|\Omega|$ and the number of topics $|Z|$ on the largest twitter dataset. As shown in Fig. 5.11, on the one hand, with the increase of $|W|$, although the running time increases for all the methods as the number of feasible size-$k$ tag sets becomes larger, INDEXEst dominates performance against other methods and shows much better scalability against number of tags. On the other hand, with the increase of $|Z|$, it is interesting to see that the running time
Table 5.4: An example case study of PITEX query on \texttt{dblp} dataset

<table>
<thead>
<tr>
<th>Researcher</th>
<th>Inferential Tags</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Michael Jordan</td>
<td>systems, theory, speech learning, applications</td>
<td>0.80</td>
</tr>
<tr>
<td>Yann LeCun</td>
<td>image, recognition, theory neural, representation</td>
<td>0.87</td>
</tr>
<tr>
<td>Jiawei Han</td>
<td>mining, structures, complexity optimization, programming</td>
<td>0.67</td>
</tr>
<tr>
<td>Jure Leskovec</td>
<td>society, communications, internet global, analysis</td>
<td>0.67</td>
</tr>
<tr>
<td>Micheal Stonebraker</td>
<td>systems, distributed, dependable data, management</td>
<td>0.73</td>
</tr>
<tr>
<td>Jim Gray</td>
<td>parallel, distributed, theory principles, storage</td>
<td>0.73</td>
</tr>
<tr>
<td>Richard Karp</td>
<td>mathematical, automata, combinatorial complexity, algorithms</td>
<td>0.93</td>
</tr>
<tr>
<td>Lasie Lamport</td>
<td>models, cryptography, theory, foundation, assessment</td>
<td>0.87</td>
</tr>
</tbody>
</table>

decreases. The reason is as follows. When there are more topics, the tag-topic probability density becomes lower as each tag is often heavily distributed in one or few topics. As explained previously, lower tag-topic probability densities lead to better pruning abilities triggered by the best effort strategy.

5.5.5 An Example Case Study

To illustrate the effectiveness of PITEX, this section provides a case study on the co-authorship social graph \texttt{dblp}. We select eight well-known computer scientists in different areas, machine learning (Jordan and LeCun), data mining (Han and Leskovec), databases (Stonebraker and Gray), and theory (Karp and Lamport). We generate PITEX queries with $k = 5$ for them. We also conduct a real user survey to evaluate the effectiveness. Specifically, we ask a group of PhD students with backgrounds in database management as annotators to manually judge the effectiveness of the system-recommended tags: given
a target scientist, say Han, and a selected tag, say mining, the student will label 1 if she thinks the tag reflects the scientist’s influential work, and 0 otherwise. For each query, we measure its accuracy by the ratio of 1’s in the returned $k$ tags, and average the accuracy from all the students. Table 5.4 provides the selected tags as well as the survey results. We can see that the tags successfully summarize the “selling-points”, such as “distributed”, “data” and “management” of Stonebraker, and the average accuracy of all queries is 0.78. This example illustrates the potential usage of personalized social influential tags exploration in SNs.

5.6 Summary

In this chapter, we introduced a new social influence problem, personalized social influential tags exploration (PITEX). We formalized this problem and proved the problem is NP hard to approximate within any constant ratio against the optimal solution. We developed a sampling-based framework and analyzed the drawbacks of two state-of-the-art sampling strategies for solving PITEX. To reduce the sampling complexities, we first proposed a lazy propagation sampling method to probe as fewer edges as possible and then employed a best-effort strategy to prune tag sets with insignificant influence spread. We further devised an index structure together with effective and materialization techniques to enable instant PITEX processing. We conducted extensive experiments on real large-scale social networks and the experimental results showed the effectiveness and efficiency of our methods.
Chapter 6

Conclusion

SMM is a rapid growing industry due to the booming of ubiquitous SNPs. By leveraging two distinct advantages of SMM: targeted reach and easy sharing, over traditional marketing approaches, this thesis aims to provide effective and efficient solutions for SMM. More specifically, we combine social information from both user profiles and user sharing behavior and propose novel solutions for each of the mainstream SMM channels: the Directed Targeted Ads (DTA) channel through which ads are posted to directed match user interests; the Celebrity Social Ads (CSA) channel through which ads are propagated from celebrities to generate a wide spread on SNPs; and the Self Influential Ads (SIA) channel through which ads are carefully designed and got propagated from the advertisers or users themselves. In the remaining of this chapter, we summarize the contributions of this thesis in Section 6.1 and highlight directions for further study in Section 6.2.

6.1 Summary

For DTA, we formulated the context-aware advertisement recommendation problem for high speed social news feeding. We first formulated a general ranking function of ads
against each user in the social network by combing the his/her interests and dynamic contents in the news feed. The Online method was first proposed to retrieve a user’s news feed and re-compute the recommended ads based on TA algorithm when there is a read operation triggered. Then the GSR method is developed, which maintains a safe region and only re-computes the recommended ads whenever the safe region is found invalid against updated news feed. Subsequently, we developed the Hybrid method to analyze users in terms of the dynamism of their news feed and determine a suitable retrieval strategy so as to speedup the recommendation process. Results of our extensive experimental study on real-world social networks and ad datasets have verified the efficiency and robustness of the hybrid model.

For CSA, we proposed the Keyword-Based Targeted Influence Maximization (KB-TIM) query to leverage user profile for viral marketing purposes. 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.

For SIA, we introduced a new type of social recommendation, personalized social influential tags recommendation (PITEX). We formalized this problem and proved the problem is NP-hard to approximate within a \(1/|V|\) ratio against the optimal solution. We developed a sampling-based framework with Monte Carlo and Reverse Random Set sampling strategies and analyzed the theoretical bound of the framework. We devised a best-effort sampling approach and an effective index for real-time influence estimation to further improve the performance while still keeping the theoretical guarantee. We
conducted extensive experiments on real large-scale social networks and the experimental results show our methods achieved high efficiency.

6.2 Future Work

In this thesis, we have developed efficient and effective advertisement solutions for SMM. However, there remain many opportunities for further exploration. The context-aware advertisement recommendation proposed in Chapter 3 assumes a static advertisement database in order to build an effective index for processing advertisement requests issued by dynamic news feed refreshes. However, given the massive number of advertisers on SNPs, it is more practical to consider dynamic updates of both news feed and advertisements. The current methods need to be adjusted or re-designed for dynamic advertisements since the safe region is no longer valid once there are new advertisements inserted or existing advertisements updated.

In Chapter 4, the keyword-based targeted IM maximizes the expected influence over the sum of influenced users’ interests to the given advertisements. However, this model is designed to simplify the existing topic aware influence models which have different influence probabilities for different topics. Although the solutions proposed in Chapter 4 efficiently solves the keyword-based targeted IM, we can still examine the more complex topic aware influence models since no efficient solutions with theoretical guarantees has been proposed. Moreover, dynamic graph update is always a challenging problem. In Chapter 4, we build sampling index on top of static graph thus the index needs to be rebuilt once there are updates happened in the social influence graph. Thus, a novel strategy to handle dynamic graph update would be an interesting future direction.

In Chapter 5, the solutions proposed for the personalized influential tags recommendation problem still have issues in time and space complexities. Although the best-effort
pruning strategies introduced reduce the number of influence score computation, the worst
time complexity is still exponential against the number of recommend tags. In addition,
the language model adopted in Chapter 5 is a bag-of-word naive Bayesian model. This
means the keywords appeared in the context is probabilistic independent given the hidden
topic distributions. The model setting can be further improved by more comprehensive
models, e.g. considering co-occurring patterns of keywords. Moreover, it would be inter-
esting to see how to extend the current solution for supporting the improved language
model.

Other than the aforementioned possible extensions of this thesis, there are a lot of ar-
reas to explore for effective and efficient SMM strategies. For example, one could consider
utilizing location information for promoting advertisements. As location data indicates
the geographical information of users, effective advertisements could be delivered to po-
tential users if they are close to the advertising business. In addition to the location-aware
SMM, time is another important criterion to be considered. As users often have certain
products to purchase at certain periods, one could mine such information and deliver the
advertisements at the right time. Moreover, the problem of how to combine the addi-
tional information, e.g. location and time, with social networks for SMM strategies is
yet another challenging direction.
Bibliography


[29] Wei Chen, Alex Collins, Rachel Cummings, Te Ke, Zhenming Liu, David Rincon, Xiaorui Sun, Yajun Wang, Wei Wei, and Yifei Yuan. Influence maximization in social networks when negative opinions may emerge and propagate. 2011.


[35] Michael Curtiss, Iain Becker, Tudor Bosman, Sergey Doroshenko, Lucian Grijincu, Tom Jackson, Sandhya Kunnatur, Soren Lassen, Philip Pronin, Sriram Sankar,


[82] Cindy Xide Lin, Qiaozhu Mei, Jiawei Han, Yunliang Jiang, and Marina Danilevsky. The joint inference of topic diffusion and evolution in social communities. In *ICDM*, pages 378–387, 2011.


[84] Lu Liu, Jie Tang, Jiawei Han, Meng Jiang, and Shiqiang Yang. Mining topic-level influence in heterogeneous networks. In *CIKM*, 2010.


