INFLUENCE ANALYSIS FOR ONLINE SOCIAL NETWORKS

XU ENLIANG

NATIONAL UNIVERSITY OF SINGAPORE

2014

INFLUENCE ANALYSIS FOR ONLINE SOCIAL NETWORKS

XU ENLIANG

(B.Sc., Northeastern University, China, 2009)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE

NATIONAL UNIVERSITY OF SINGAPORE

2014

Declaration

I hereby declare that this 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.

XU ENLIANG

July 9, 2014

To my parents.

Acknowledgements

First and foremost, I would like to thank my supervisors, Prof. Wynne Hsu and Prof. Mong Li Lee. Without their excellent guidance, continuous support and encouragement, this thesis cannot be done. I have benefited greatly from their insights and knowledge through regular discussions. I have learnt a lot from them in many aspects of doing research. Their dedication and preciseness have deeply influenced me in my research and my entire life.

I would like to thank my thesis committee Prof. Stéphane Bressan and Prof. Tan Chew Lim to give me insightful comments and constructive suggestions to improve my work.

I would like to thank Dr. Dhaval Patel for his generous help and inspiring discussions on my research, and for being a great friend. Dhaval has helped me a lot during my Ph.D study. He is very friendly and always ready to help whenever I have questions, even after leaving NUS for IIT Roorkee as an Assistant Professor.

I would also like to thank the following lecturers in School of Computing, NUS for giving me the opportunity to be a part-time teaching assistant: Prof. Lubomir Bic, Prof. Joxan Jaffar, Dr. Ang Chuan Heng, and Aaron Tan. As a part-time TA, I have gained valuable teaching experience, enhanced my knowledge and improved my communication skills through teaching tutorials and conducting labs. I extend my thanks to Ms Loo Line Fong and other administrative staffs in School of Computing for their always kind help.

I am also grateful to my lab mates in iLab: Ding Feng, Cheng Yuan, Deng

Fanbo, Gilbert Lim, Jin Yiping and lab mates in DB2: Chen Wei, Zhao Gang, Song Chonggang, Li Furong and other friends, to name a few.

Last, but not least, I give my sincere thanks to my parents for their endless love, unconditional support and encouragement.

Contents

Summary

The prevalence of online social media such as Facebook, Twitter, LinkedIn and YouTube has attracted considerable research in social influence analysis with applications in viral marketing, online advertising, recommender systems, information diffusion, and experts finding. Social influence occurs when one's emotions, opinions, or behaviors are affected by others. Most of the works on social influence analysis have largely been focused on validating the existence of influence, studying the maximization of influence spread in the whole network, inferring the "hidden" network from a list of observations, modeling direct influence in homogeneous networks, mining topic-level influence on heterogeneous networks, and conformity influence.

In this thesis, we perform influence analysis for online social networks by addressing three important issues in the discovery of influential nodes and influence relationships, which have been given little attention by existing works: *influential path*, *topic-level influence* and *consistent influencer*. We outline our approaches as follows.

First, we focus on influential path discovery. We show that influential paths can capture the dynamics of information diffusion better compared to influential edges. We propose a generative influence propagation model based on the Independent Cascade Model and Linear Threshold Model, which mathematically models the spread of certain information through a network. We formalize the top-k maximal influential path inference problem and develop an efficient algorithm, called TIP, to infer the top-k maximal influential paths. TIP makes use of the properties of top-k maximal influential paths to dynam-

ically increase the support and prune the projected databases. As databases evolve over time, we also develop an incremental mining algorithm, named IncTIP, to maintain the set of top-k maximal influential paths efficiently. We evaluate the proposed algorithms on two real world datasets (MemeTracker and Twitter). The experimental results show that our algorithms are more scalable and more efficient than the base line algorithms. In addition, influential paths can improve the precision of predicting which node will be influenced next.

Next, we investigate topic-level influence. We show that in many applications the underlying networks are not explicitly modeled, and temporal factor plays an important role in determining social influence, which is ignored by existing works. We take into account the temporal factor in social influence to infer the influential strength between users at topic-level. Our approach does not require the underlying network structure to be known. We propose a guided hierarchical LDA approach to automatically identify topics without using any structural information. We then construct the topic-level social influence network incorporating the temporal factor to infer the influential strength among the users for each topic. Experimental results on two real world datasets (Twitter and MemeTracker) demonstrate the effectiveness of our methods. Further, we show that the proposed topic-level influence network can improve the precision of user behavior prediction and is useful for influence maximization.

Finally, we propose to identify k-consistent influencers. We show that finding influential users at single time point only cannot capture whether the users are consistently influential over a period of time. We devise an efficient algorithm that utilizes a grid index to scan the users in the 2D personal-preference consistency space, thereby obtaining the rank of these users at a given time point. Then we design the TCI algorithm to identify the k-consistent influencers for

a given time interval. We conduct extensive experiments on three real world datasets (Citation, Flixster and Twitter) to evaluate the proposed methods. The experimental results demonstrate the effectiveness and efficiency of our methods. We show that the proposed k-consistent influencers is useful for identifying information sources and finding experts.

List of Tables

List of Figures

List of Publications

1. E. Xu, W. Hsu, M. Lee, and D. Patel. Top-k Maximal Influential Paths in Network Data. In *International Conference on Database and Expert Systems Applications (DEXA)*, pages 369-383, 2012.

2. E. Xu, W. Hsu, M. Lee, and D. Patel. Incremental Mining of Top-k Maximal Influential Paths in Network Data. In *Transactions on Large-Scale Data and Knowledge-Centered Systems (TLDKS)*, pages 173-199, 2013. (Invited Paper)

3. E. Xu, W. Hsu, M. Lee, and D. Patel. Inferring Topic-level Influence from Network Data. In *International Conference on Database and Expert Systems Applications (DEXA)*, pages 132-147, 2014.

4. E. Xu, W. Hsu, and M. Lee. k-Consistent Influencers in Network Data. Submitted to *International Conference on Information and Knowledge Management (CIKM)*, 2014.

Chapter 1

Introduction

1.1 Background

The advent of Web 2.0 has seen increasing and extensive participation of people in online activities like content sharing (e.g., text, images), social networking (e.g., Facebook, Twitter), and social bookmarking (e.g., ratings, tagging). With the prevalence of online social media, such as Facebook, Twitter, Flickr and YouTube, a huge amount of valuable information has been generated and made available, which has led to different kinds of research from many different domains, e.g. statistics, computer science, and sociology. The field of social network analysis has recently attracted great research interests in the computer science community. A social network can be represented as a graph, in which nodes represent users, and links represent the connections between users. Social networks are extremely rich in data, which can be divided into two main categories: *linkage* data and *content* data. The linkage data refers to the graph structure of the social network; whereas the content data contains the text, images and other kinds of data in the social networks.

One aspect of social network analysis is influence analysis. When a user purchased a product that his friend has just recently bought, he may have been influenced by his friend. Such phenomenon is called social influence. Social influence occurs when one's

emotions, opinions, or behaviors are affected by others[1]. Social influence takes many forms and can be seen in conformity, socialization, peer pressure, obedience, leadership, persuasion, sales, and marketing. The study of social influence has a long history in social sciences. Early works focused on the adoption of medical [32] and agricultural innovations [107]. Later, marketing researchers investigated the "word of mouth" diffusion process for *viral marketing* [12, 43, 71, 54]. With the rapid proliferation of online social media and the availability of user generated contents, influence analysis on social networks has attracted great research interests.

A basic problem in influence analysis on social networks is that of influence maximization: given a social network, find k nodes to target in order to maximize the spread of influence. Domingos and Richardson [37, 86] are the first to study the influence maximization problem as an algorithmic problem. Subsequently, Kempe et al. [55] formulate the problem as a discrete optimization problem. Considerable works have also been done on different aspects of social network influence, such as validating the existence of influence [37, 3], modeling information diffusion [55, 25, 46], learning influence probabilities [92, 45], inferring hidden networks [44, 75], topic-level influence analysis [102, 69, 109] and conformity influence [103]. In [18], Bonchi presents a survey on social network influence from a data mining perspective.

Social network influence analysis has been exploited in applications like recommender systems [96, 98, 99], information diffusion in social media [10, 22, 76, 88, 113, 115], experts finding [38, 102], and link prediction [33, 9]. Recently, some startups have utilized social influence for social media marketing. For example, Klout[2] measures the social influence scores of users by integrating their Facebook and Twitter profiles with Klout. Klout generates a score on a scale of 1-100 for a social user to represent his/her ability to engage other people and inspire social actions.

[1] http://en.wikipedia.org/wiki/Social_influence
[2] http://www.klout.com/

1.2 Motivation

Existing social network influence analysis research has largely been focused on discovering influential nodes (users, entities) and influence relationships (who influences whom) among nodes in the network [55, 64, 28, 27, 44, 102]. In the context of influence relationship discovery, existing works have investigated both macro-level and micro-level influence. For macro-level influence, Gomez et al. [44] infer top-k influential edges from a list of observations, which can only capture the influence relationship between two nodes. However, in many applications, knowing the actual paths of how influence is being propagated in the social networks can lead to better decision making and policy formulation. For micro-level influence, Tang et al. [102] study the topic-level influence between two users assuming the influence relationship among users are explicitly modeled. While this is useful for some applications that are concerned with only explicitly modeled relationships, many applications need to go beyond the connected users. In the context of influential nodes discovery, existing works [55, 64, 28, 27] find influential users at single time point only and do not capture whether the users are consistently influential over a period of time. However, consistency is a key factor in determining influence. In this thesis, we address these three issues and show that exploiting these issues can further benefit social influence analysis.

1.2.1 Mining Top-k Maximal Influential Paths

Discovering influential edges has important applications in viral marketing and personalized recommendations. Existing works infer top-k influential edges from a list of observations of when and where an event occurs. However, an influential edge can only capture the influence relationship between two nodes. Often times, it is equally, if not more important to know how the influence is being propagated. Knowing the paths of propagation is useful. For example in the surveillance of computer virus propagation, knowing the influential paths allow us to identify critical nodes and stop the virus propagation by bringing down these nodes. Finding the top-k influential paths in large-scale

social networks is non-trivial. The problem is further complicated by the fact that users are active and regularly upload new information to the online social media. Such updates may introduce new patterns or invalidate some existing patterns and demand the need for an incremental solution.

1.2.2 Inferring Topic-level Social Influence

Besides identifying the influential paths, it is also important to infer the influence relationship among users at topic-level. Existing methods [102, 69, 109] that discover topic-level influence assume that influence can only occur among known social connections (e.g. friends in Facebook). However, there are many social networks where the influence may occur among users who are not explicitly connected. For example, in Twitter, one user can influence another even when they are not explicitly following one another. Inferring topic-level influence without explicit connections is a challenging task. First, we need to design an effective algorithm that can extract meaningful topics from short texts such as tweets. Second, without the benefit of an explicit modeling of users' connection with each other, we need to infer influence relationships among users through the observation of their activities on social networks.

1.2.3 Identifying k-Consistent Influencers

For influential nodes discovery, existing works [55, 64, 28, 27] find influential users at a given time point. They do not care whether the users are consistently influential over a period of time. However, from the psychological perspective, it is consistency that builds trusts and thereby resulting in the greatest influence. Here, we advocate the need to incorporate the notion of consistency in determining the top influencers. This involves dynamically computing the total influence of each user and ranking them at each time point.

1.3 Contributions

In this thesis, we investigate three important issues related to the discovery of influential nodes and influence relationships, i.e. *influential path*, *topic-level influence* and *consistent influencer*. The overall framework of this thesis is shown in Figure 1.1. We first address the problem of mining top-k maximal influential paths. Later, we infer topic-level social influence from network data. Last, we study the problem of identifying k-consistent influencers. The main contributions of this thesis can be summarized as follows.

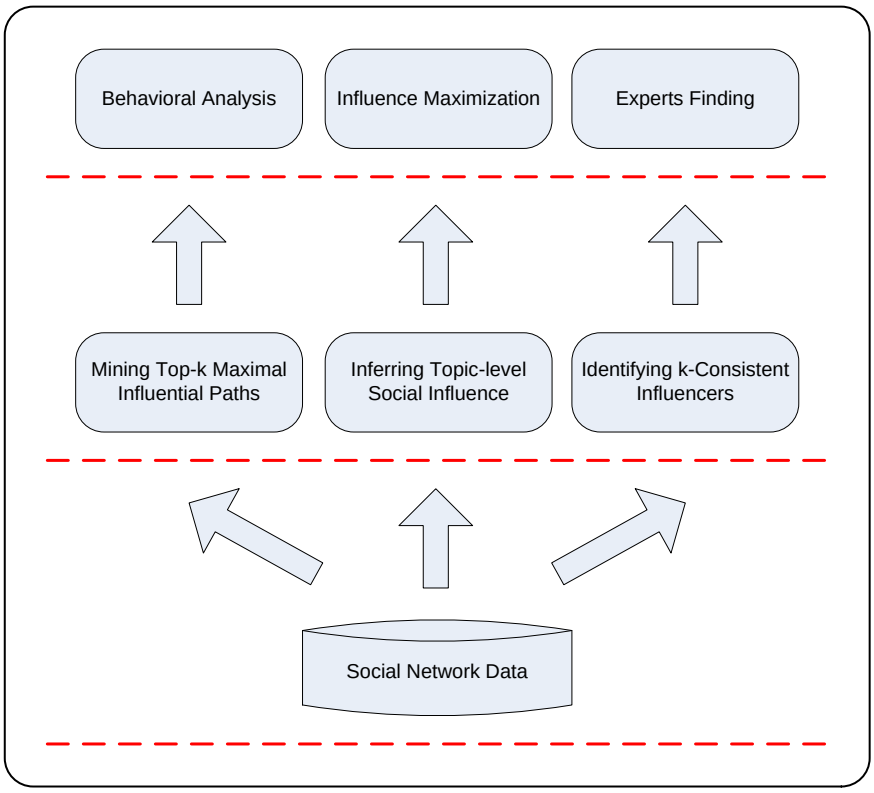


Figure 1.1: Thesis framework

1. We develop a method for inferring top-k maximal influential paths which can capture the dynamics of information diffusion better than influential edges. We propose a generative influence propagation model based on the Independent Cascade Model and Linear Threshold Model, which mathematically models the spread of certain information through a network. We formally define the top-k maximal influential path inference problem and develop an efficient algorithm, TIP, to infer top-k maximal influential paths. TIP makes use of the properties of top-k maximal influential paths to perform dynamic support-raising and projected database-pruning.

As databases evolve over time, we also develop an incremental mining algorithm, named IncTIP, to maintain the set of top-k maximal influential paths efficiently. Extensive experiments are conducted on two real world datasets (MemeTracker and Twitter). We show that our algorithms are more efficient than the base line algorithms and demonstrate the effectiveness of using influential paths for predicting which node will be influenced next.

2. We take into account the temporal factor in social influence to infer the influential strength between users at topic-level, without requiring the underlying network structure to be known. We propose a guided hierarchical LDA approach to automatically identify topics without using any structural information. We then construct the topic-level social influence network incorporating the temporal factor to infer the influential strength among the users for each topic. Experimental results on two real world datasets (Twitter and MemeTracker) demonstrate the effectiveness of our methods. Further, we show that the proposed topic-level social influence network can improve the precision of user behavior prediction and is useful for influence maximization.

3. We devise an efficient algorithm that utilizes a grid index to scan the users in the 2D personal-preference consistency space, thereby obtaining the rank of these users at a given time point. Then we design the TCI algorithm to identify the k-consistent influencers for a given time interval. We conduct extensive experiments on three real world datasets (Citation, Flixster and Twitter) to evaluate the proposed methods. The experimental results demonstrate the effectiveness and efficiency of our methods. We show that the proposed k-consistent influencers is useful for identifying information sources and finding experts.

1.4 Organization

The rest of this thesis is organized as follows. Chapter 2 discusses the related work. We review works that are most relevant to our research. These include works in information diffusion models, influence maximization, learning influence probabilities, inferring hidden networks, information cascades and blog networks, and topic-level influence analysis.

In Chapter 3, we develop a method for inferring top-k maximal influential paths which can truly capture the dynamics of information diffusion. As databases evolve over time, we also develop an incremental mining algorithm IncTIP to maintain top-k maximal influential paths efficiently.

In Chapter 4, we infer topic-level influence without requiring the underlying network structure to be known. We show that the proposed topic-level social influence network can improve the precision of user behavior prediction and is useful for influence maximization.

In Chapter 5, we design the TCI algorithm to identify k-consistent influencers for a given time interval. We show that the proposed k-consistent influencers is useful for identifying information sources and finding experts.

Finally, we conclude our studies and discuss some future work in Chapter 6.

Chapter 2

Related Work

In this section, we review works that have been done on different aspects of social influence. We also give a brief overview of some of the mathematical and computational techniques and models that have been developed in previous works.

2.1 Information Diffusion Models

Information diffusion refers to the spread of abstract ideas or technical information within a social system, where the spreading denotes flow or movement from a source to an adopter, typically via a communication link [87]. Such a communication can influence and alter an adopter's probability of adopting an innovation, where an adopter may be an individual, a group, or an organization. Examples include viral marketing, innovation of technologies, and infection propagation. There are two basic information diffusion models that capture the underlying dynamics of the diffusion process, namely, the Linear Threshold (LT) model and the Independent Cascade (IC) model.

The Linear Threshold model was first proposed by Grannovetter [47] and Schelling [94] in the context of the social sciences. It is often used in marketing research [37, 86, 47, 94]. The model gives each individual an influence threshold. An individual is activated when this threshold is exceeded. There is a cumulative effect of the linear threshold model, as it takes a critical number of influential neighbors to activate an individual.

Let $G = (V, E)$ be a graph where the set of vertices V represent individuals and the directed edges in E indicate the direction of influence. The Linear Threshold model works as follows. First, every vertex v randomly selects a value between [0,1] for its threshold λ_v. Next, influence cascades in discrete steps i = 0, 1, 2, ..., and let S_i denote the set of vertices activated at step i, with $S_0 = S$. S is the set of initially activated vertices. In each step $i \geq 1$, a vertex $v \in V \setminus \cup_{0 \leq j \leq i-1} S_j$ is activated if the weighted number of its activated in neighbors reaches its threshold, i.e. $\sum_{u \in \cup_{0 \leq j \leq i-1} S_j} w(u, v) \geq \lambda_v$. The process ends at a step t when $S_t = \emptyset$. Note that the linear threshold model is deterministic, as we know whether a node is active or not by just counting the sum of the weights of all active neighbors. It imposes the property that the sum of weights to a node is bounded by 1.

The Independent Cascade model was defined by Kempe et al. [55] and used in the context of marketing [43, 42]. Given a seed set $S \subseteq V$, the independent cascade model works as follows. Let $S_t \subseteq V$ be the set of nodes that are activated at step $t \geq 0$, with $S_0 = S$. At step t + 1, every node $u \in S_t$ may activate its out-neighbors $v \in V \setminus \cup_{0 \leq i \leq t} S_i$ with an independent probability of $p_{u,v}$. The process ends at a step t with $S_t = \emptyset$. The independent cascade model gives each individual the ability to influence their neighbors as soon as they are activated. This is opposed to the linear threshold model that relies on a cumulative effect. The independent cascade model has the property that a node has exactly one time step in which it is infected to infect other nodes. That is, each node is infectious for exactly one time step and then can no longer be infected, nor can it infect any other nodes. Along with the linear threshold model, this model is used for studying information diffusion on networks.

In [55], Kempe et al. also propose a broader framework, called General Threshold Model, which simultaneously generalizes the Linear Threshold (LT) and Independent Cascade (IC) models. In the General Threshold Model, each node v has a monotone threshold function f_v that maps the subsets of v's neighbor set to real numbers in $[0, 1]$, and a threshold θ_v chosen uniformly at random from the interval $[0, 1]$. A node v becomes active at time $t + 1$ if $f_v(S) \geq \theta_v$, where S is the set of neighbors of v that are active at

time t.

In our work on influential path discovery, we propose a generative influence propagation model based on the Independent Cascade Model and Linear Threshold Model, which can mathematically model the spread of certain information through a network.

2.2 Influence Maximization

A basic problem in social influence analysis is that of influence maximization: given a social network, find k nodes to target in order to maximize the spread of influence. Domingos and Richardson [37, 86] are the first to study the influence maximization problem as an algorithmic problem. They modeled social networks as Markov random fields where the probability of an individual adopting a technology (or buying a product) is a function of both the intrinsic value of the technology (or the product) to the individual and the influence of neighbors. The authors proposed three algorithms that approximately determine the influential users and showed that selecting the right set of users for a marketing campaign can make a substantial difference. [37, 86] built probabilistic models, and used these models to choose the best viral marketing plan, but there are many parameters to be trained in their scheme.

The algorithmic and computational aspects of the influence maximization problem are investigated in [55, 56, 59]. Kempe et al. [55] formulate the problem as a discrete optimization problem. A social network is modeled as a graph with vertices representing individuals and edges representing connections or relationship between two individuals. Influence is propagated in the network according to a stochastic cascade model. Three cascade models, namely the independent cascade model, the weight cascade model, and the linear threshold model, are considered in [55]. Given a social network graph, a specific influence cascade model, and a small number k, the influence maximization problem is to find k vertices in the graph (referred to as seeds) such that under the influence cascade model, the expected number of vertices influenced by the k seeds (referred to as influence spread) is the largest possible.

Kempe et al. prove that the optimization problem is NP-hard, and present a greedy approximation algorithm (Algorithm 1) which guarantees that the influence spread is within $(1 - 1/e)$ [80] of the optimal influence spread. The basic idea of the greedy algorithm is to calculate the influence set of each individual, and take turns to choose the node maximizing the marginal influence value until k nodes are selected. They also show through experiments that their greedy algorithm significantly outperforms the classic degree and centrality-based heuristics in influence spread.

Algorithm 1 Greedy(k, f)

1: initialize $S = \emptyset$;
2: **for** i =1 to k **do**
3: select $u = \arg\max_{w \in V \setminus S} (f(S \cup \{w\}) - f(S))$;
4: $S = S \cup \{u\}$;
5: **end for**
6: output S;

However, their algorithm has a serious drawback, which is its efficiency. A key element of their greedy algorithm is to compute the influence spread given a seed set, which turns out to be a difficult task (in fact, Chen et al. point out that the computation is ♯P-hard [27]). Instead of finding an exact algorithm, they run Monte-Carlo simulations of the influence cascade model for sufficiently many times to obtain an accurate estimate of the influence spread. As a result, even finding a small seed set in a moderately large network (e.g. 15000 vertices) could take days to complete on a modern server machine.

Recent studies aim to address this efficiency issue. In [64], Leskovec, Krause, and Guestrin address the influence maximization problem in two applications. The first application is to determine where sensors should be placed in a water distribution network such that contaminants can be quickly detected. The second application is to identify influential blogs. They present a Cost-Effective Lazy Forward (CELF) scheme to select new seeds. This scheme uses the sub-modularity property of the underlying objective to greatly reduce the number of evaluations on the influence spread of vertices. As reported in [64], CELF has the same influence spread as the original greedy algorithm of Kempe, Kleinberg, and Tardos [55], and achieves as much as 700 times speedup in their exper-

iments. There are two aspects to this speed up: (i) by speeding up function evaluations using the sparsity of the underlying problem, and (ii) by reducing the number of function evaluations using the submodularity of the influence functions. However, even though the "lazy-forward" optimization is significant, it still takes hours to find 50 most influential nodes in a network with a few tens of thousands of nodes, as shown in [28].

Kimura and Saito [57] propose a shortest-path based influence cascade model and provide efficient algorithms for finding the most influential nodes under these models. However, since the influence cascade models are different, they do not directly address the efficiency issue of the greedy algorithms for the cascade models studied in [55].

Even-Dar and Shapira [39] study the influence maximization problem in the context of probabilistic voter model. They present simple and efficient algorithms for solving this problem. Furthermore, in a special case, the popular heuristic which picks nodes in the network with the highest degree turns out to be an optimal solution.

Chen, Wang, and Yang [28] present an efficient algorithm to find the top-k nodes in a social network and this algorithm improves upon the greedy algorithm of Kempe, Kleinberg, and Tardos [55] and also the algorithm of Leskovec et al. [64] in terms of its running time. Specially, they propose two faster greedy algorithms called NewGreedy and MixedGreedy, respectively. The main idea behind NewGreedy is to remove the edges that will not contribute to propagation from the original graph to get a smaller graph and do the influence diffusion on the smaller graph. The first round of MixedGreedy uses NewGreedy algorithm, and the rest rounds employ CELF algorithm. An earlier approach proposed by Kimura et al. [58] also removes edges that do not contribute to information diffusion, and does the propagation on the subnetwork. In addition, the authors also design a new degree discount heuristic algorithm, which they call DegreeDiscount, that achieves much better influence spread than classic degree and centrality based heuristics. They also note that the performance of this heuristic algorithm is comparable to that of the greedy algorithm while its running time is much less than that of the greedy algorithm. DegreeDiscount assumes that the influence spread increases with the degree of nodes.

Unlike the greedy algorithm, DegreeDiscount algorithm has no provable performance guarantee.

The work by Chen et al. [27] is the continuation of [28] in the pursuit of efficient and scalable influence maximization algorithms. In [28], Chen et al. explore two directions in improving the efficiency: one is to further improve the greedy algorithm of [55], and the other is to design new heuristic algorithms. The first direction shows improvement but is not significant enough, indicating that this direction could be difficult to continue. The second direction leads to new degree discount heuristics that are very efficient and generate reasonably good influence spread. The major issue is that the degree discount heuristics are derived from the uniform IC model where propagation probabilities on all edges are the same, which is rarely the case in reality. [27] is a major step in overcoming this limitation − their new heuristic algorithm, called *maximum influence arborescence* (MIA), works for the general IC model while still maintains good balance between efficiency and effectiveness. The main idea of the MIA heuristic is to use local arborescence structures of each node to approximate the influence propagation. The authors also conduct much more experiments than in [28] on more and larger scale graphs, and the results show that the MIA heuristic performs consistently better than the degree discount heuristic in all graphs. Actually, the degree discount heuristic can be viewed as a special case of the MIA heuristic restricted on the uniform IC model with all arborescences having depth one.

Since both [28] and [27] are designed using specific features of the IC model, they do not apply directly to the LT model. In term of design principle, Chen et al. [29] propose the LDAG algorithm to fill this gap in the research of scalable influence maximization algorithms in the LT model. LDAG is similar to the MIA algorithm [27]. Both uses local structures to make the influence computation tractable and reduce computation cost. However, the local structure and the influence computation are different: MIA uses local tree structures because that is the only structure making the influence computation tractable in the IC model, while LDAG uses local DAG structures, and thus could include

more influence paths in the local structure.

Narayanam and Narahari [101, 79] propose an efficient heuristic algorithm which is called the SPIN (ShaPley value based Influential Nodes) algorithm for the LT model. Their approach exploits the novel idea of modeling the information diffusion process as a cooperative game and using the Shapley values of the nodes to compute their network value or influence in the network. And they compare the performance of the proposed SPIN algorithms with well-known algorithms in the literature. Extensive experimentation on 4 synthetically generated random graphs and 6 real-world data sets show that the proposed SPIN approach is more powerful and computationally efficient. However, SPIN only relies on the evaluation of influence spreads of seed sets, and thus does not use specific features of the LT model. Moreover, SPIN is not scalable, with running time comparable (as shown in [55]) or slower than the optimized greedy algorithm [29].

Goyal et al. [46] propose a novel data-based approach for influence maximization. They introduce a new model called Credit Distribution (CD), which directly estimates influence spread by exploiting available propagation traces, without the need for learning influence probabilities or conducting Monte Carlo (MC) simulations. The credit distribution model learns the total influence credit accorded to a given set S by any node u and uses this to predict the influence spread of S. Their approach also learns the different levels of user influenceability, and takes the temporal nature of influence into account. Based on the CD model, Goyal et al. develop an approximation algorithm for influence maximization with high accuracy and scalability.

The aforementioned approaches attack the efficiency issue by either improving the greedy algorithm or using new heuristics. However, none of them take into consideration the community property of social networks. Wang et al. [112] propose a community-based method for mining top-k influential nodes, called Community-based Greedy Algorithm (CGA). The basic idea is to exploit the community structure property of social networks. Intuitively, a community is a densely connected subset of nodes that are only sparsely linked to the remaining network. Communities in a social network represent real

social groups, and thus individuals in a community will influence each other in the form of "word-of-mouth". The prohibitive cost of finding influential nodes over the whole network would be reduced greatly if we find influential nodes with regard to communities. The proposed CGA algorithm has two main components, an algorithm for detecting communities by taking into account information diffusion, and a dynamic programming algorithm for selecting communities to find influential nodes. The authors also provide provable approximation guarantees for CGA. Empirical studies on a large real-world mobile social network show that the CGA algorithm is more than an order of magnitudes faster than the state-of-the-art Greedy algorithm for finding top-k influential nodes and the error of CGA is small compared with Greedy algorithm.

However, these influence maximization methods ignore one important aspect of influence propagation in the real world. That is, not only positive opinions on products may propagate through the network, negative opinions are also propagating, and are often more contagious and stronger in affecting people's decisions. In [25], Chen et al. incorporate the emergence and propagation of negative opinions into the influence cascade model and study its impact together with positive influence in the influence maximization problem. They design an efficient algorithm to compute influence in tree structures, which is nontrivial due to the negativity bias in the model. And then they use this algorithm as the core to build a heuristic algorithm for influence maximization for general graphs.

Recently, a substantial amount of research has been done in the context of influence maximization. Although work has been done on improving the performance of greedy algorithms for influence maximization, scalability remains a significant challenge. In addition to the scale issues that are inherently there, these definitions of influential users ignore certain aspects of the real social networks such as the existence of multiple innovations (competing campaigns), and time factor.

Bharathi et al. [14] extend past work by focusing on the case when multiple innovations are competing within a social network such as when multiple companies market competing products using viral marketing. Specially, they augment the Independent

Cascade Model to capture the existence of competing campaigns in a network. Similar to Kempe et al. [55], they provide an approximation algorithm to computing the best response to an opponent's strategy in the "game of innovation". In the influence maximization game, players wish to maximize their individual influence given a randomized propagation scheme. It can be shown that mixed Nash Equilibria exist for this game. From here, Bharathi et al. show that best-response strategies exist for this game that are both monotone and submodular. This, coupled with discussion of "first mover" strategies, provides a framework for the behavioral basis of influence maximization in social networks. In this paper, the authors use diffusion models where the competing campaigns propagate exactly the same way, i.e. the probability of diffusion on a certain edge is the same for all campaigns and all campaigns start at the same time. However, this assumption is not true, as in the real world the competing campaigns may have different acceptance rates.

Liu et al. [70] study the categorical influence maximization (CIM) problem. Compare with identifying maximum influence vertices in a single category social network, CIM is much harder because it has to deal with large scale complex data. Specially, based on the observations from real mobile phone social network data, they propose a Probability Distribution based Search method (PDS) to tackle the CIM problem. The PDS method consists of three steps. It first solves the storage problems in mobile phone social networks. Second, it identifies influential vertices by the probability distribution. Third, it minimizes influential sets and maximizes the influence considering the vertex attributes. They also verify the PDS method by real data sets, a one-year mobile phone network data in a city in China.

Budak et al. [19] study the notion of competing campaigns in a social network. By modeling the spread of influence in the presence of competing campaigns, they provide necessary tools for applications such as emergency response where the goal is to limit the spread of misinformation. More specifically, they investigate efficient solutions to the eventual influence limitation (EIL) problem: Given a social network where a (bad) information campaign is spreading, who are the k "influential" people to start a counter-

campaign if our goal is to minimize the effect of the bad campaign? They introduce the *Multi-Campaign Independent Cascade Model* (MCICM), which models the diffusion of two cascades evolving simultaneously in a network. And they prove that the eventual influence limitation problem is NP-hard and show that a greedy method is guaranteed to provide a $1/(1-e)$ approximation.

In [26], Chen et al. extend the classical Independent Cascade model to study time-delayed influence diffusion and they consider the time-critical influence maximization problem under the proposed IC-M model. They prove the submodularity of IC-M, and propose fast heuristics MIA-M and MIA-C to find seed sets efficiently and effectively. MIA-M is based on a dynamic programming procedure that computes exact influence in tree structures, while MIA-C converts the problem to one in the original IC model and then applies existing fast heuristics to it.

Liu et al. [67] study the time constrained influence maximization problem, which is based on the Latency Aware Independent Cascade influence propagation model. They show that the problem is NP-hard, and prove the monotonicity and submodularity of the time constrained influence spread function. Based on this, they develop a greedy algorithm with performance guarantees. To improve the algorithm scalability, they propose to use Influence Spreading Paths (ISP) to quickly and effectively approximate the time constrained influence spread for a given seed set. Let $\sigma_T(S)$ be the expected number of nodes influenced by S within T time units. ISP calculates both $\sigma_T(S \cup \{v\})$ and $\sigma_T(S)$ by using Influence Spreading Paths. The Influence Spreading Paths starting from each seed set are calculated from scratch by Depth-First Search (DFS). Further, by employing faster marginal influence spread calculating methods, they propose Marginal Discount of Influence Spread Path (MISP) to improve the speed of ISP. MISP calculates influence spread $\sigma_T(u)$ for each single node u with Influence Spreading Paths starting from u, then select seed node with the largest discounted marginal influence spread one by one. Experimental results show that MISP is the fastest and multiple orders of magnitude faster than the simulation based greedy algorithm MC while achieving similar time constrained

influence spread.

Recently, Li et al. [66] study the problem of location-aware influence maximization. They devise two greedy algorithms with $1 - 1/e$ approximation ratio. The expansion-based algorithm estimates the upper bound of users' influences and adopts a best-first method to eliminate the insignificant users. The assembly-based algorithm assembles the precomputed information on small regions to answer a query. They also propose two efficient algorithms with $\epsilon \cdot (1 - 1/e)$ approximation ratio for any $\epsilon \in (0, 1]$. The first is a bound-based algorithm that uses the estimated upper bounds and lower bounds to select top-k seeds. The second is a hint-based algorithm that utilizes precomputed hints to identify top-k seeds.

All the above works study the influence maximization problem from different aspects, such as performance, community property, negative opinions, multiple innovations and location awareness. Our methods can also be applied to find k nodes such that the influence spread is maximized.

2.3 Learning Influence Probabilities

Saito et al. [92] focused on learning propagation probabilities under the IC model. They formalize this as a likelihood maximization problem and then apply the expectation maximization (EM) algorithm to solve it. While their formulation is elegant, there are two issues in their approach. First, since EM is an iterative algorithm, it may not be scalable to very large social networks. This is due to the fact that in each iteration, the EM algorithm must update the influence probability associated to each edge. Second, the propagation traces data that is used as input to learn probabilities is very sparse in particular, it follows a long tail distribution, that is, most of the users perform very few actions. As a result, the EM algorithm is vulnerable to overfitting and may result in poor quality seed sets.

Later, Saito et al. [90, 91] extended the IC and LT models to make them time-aware and proposed methods to learn influence propagation probabilities for these extended

models. They incorporate time delay in action propagations where the time delay is on a continuous time scale, for IC model in [90] and for LT model in [91]. They use EM based approaches to learn propagation probabilities as in their previous work [92]. In a recent paper, Saito et al. [93] recognize the issue of overfitting and propose to consider node attributes as well in learning probabilities.

Goyal et al. [45] also study the problem of learning influence probabilities from the history of user actions. They focus on the time varying nature of influence, and present the concept of user influential probability and action influential probability. The goal of this work is to find a model to best capture the user influence and action influence information in the network. They also show that their methods can be used to predict whether a user will perform an action and at what time, with higher accuracy for users with higher influenceability scores.

These works focus on learning influence probabilities under certain information diffusion models, which is different from our work.

2.4 Inferring Hidden Networks

Gomez et al. [44] study the diffusion of information among blogs and online news sources. They assume that connections between nodes cannot be observed and use the observed cascades to infer a sparse, "hidden" network of information diffusion. They propose an iterative algorithm called NetInf which is based on submodular function optimization. NetInf first reconstructs the most likely structure of each cascade. Then it selects the most likely edge of the network in each iteration. The algorithm assumes that the weights of all edges have the same values.

In [115], Yang et al. propose a Linear Influence Model to model the global influence of a node on the rate of diffusion through the (implicit) network. The main idea of this model is that each node has an influence function associated with it and the number of newly infected nodes is a function of influences of which other nodes got infected in the past. For each node they estimate an influence function that quantifies how many

subsequent infections can be attributed to the influence of that node over time. With a non-parametric formulation, the model can be efficiently estimated using a simple least squares procedure.

Mathioudakis et al. [75] investigate the problem of sparsifying influence networks. Given a social graph and a list of actions propagating through it, they design the SPINE algorithm to find the "backbone" of the network through the use of the independent-cascade model [55]. SPINE has two phases: the first phase selects a set of arcs that yields a finite log-likelihood, while the second phase greedily seeks a solution of maximum log-likelihood. The effectiveness of SPINE came from its ability to increase computation speed significantly.

The aforementioned works aim to infer top-k influential edges from a list of observations of when and where an event occurs. Influential edges can only capture the influence relationship between two nodes. In our work, we introduce the concept of "influential path" to capture the propagation of influence beyond two nodes.

2.5 Information Cascades and Blog Networks

Cascades have been studied for many years by sociologists concerned with the diffusion of innovation [87]. Cascades are used for studying viral marketing [62], and explaining trends in blogspace [58, 29]. Leskovec et al. [65] studied the properties and models of information cascades in blogs. Information diffusion models are also appropriately considered from the view of the blogosphere where a blogger may have a certain level of interest in a topic and is thus susceptible to talking about it. By discussing the topic, the blogger may influence other bloggers.

Gruhl et al. [48] present a study on information diffusion of various topics in the blogosphere along two dimensions, topical and individual, drawing on the theory of infectious diseases via a general cascade model. They formalize the idea of topics that run over long period of time and use theory of infectious diseases to analyze the flow of information. They further classify the long running topics as internal sustained discussion and

externally induced spikes and provide formal models for both of them. Furthermore, they propose an "expectation-maximization" algorithm which predicts the probability of an individual getting infected by a topic at a given epoch of time and validate the algorithm with both synthetic and real data.

In [1], Adar et al. have proposed the use of URL citations to infer the dynamics of information epidemics in the blogspace. They also show that the PageRank algorithm finds authoritative blogs. A variation, called iRank, is described to rank blogs based on their informativeness. In this scheme, each directed edge is assigned a weight W_{ij} = $w(\Delta d_{ij})$ where Δd refers to the time difference between the blogs citing a URL and $w(\Delta)$ is the weight function that gives importance to URL citations which are closer in time. The edge weights are then normalized and PageRank computation follows. This weighted graph is called the implicit information flow graph. iRank makes use of the temporal nature of blogs by differentially weighing each citation in the graph by the time difference between when the blog mentions a URL and how soon it is referenced by other blogs.

Weblogs link together in a complex structure through which information can flow. Such a structure is ideal for the study of the propagation of information. Adar et al. [2] study the pattern and dynamics of information spreading among blogs. Specifically, they are interested in determining the path information takes through the blog network by using the existing link structure of blogspace. This infection inference task is related to both link inference and link classification but makes use of non-traditional features unique to blog data. Their goal is to correctly label graph edges between blogs when one blog infects the other. The difficulty is that frequently blogs do not cite the source of their information and appear disconnected from all likely sources of that information (i.e. other infected blogs). Thus, they apply link inference techniques to infer the source of information spread in blogspace based on the timestamps of entries and the link structure of blogs. The authors describe a Support Vector Machine (SVM) and logistic regression based classifiers to find and label potential infection routes. However, their method relies

on the embedded explicit hyperlinks in blogs. And the interesting interaction that occurs in comments left by bloggers is not explored.

In [52], Java et al. study the performance of various algorithms such as PageRank and in-degree, on modeling influence of blogs. They present the results of applying the Linear Threshold Model and the Independent Cascade Model in the blogosphere and show how these techniques can automatically predict a set of influential blogs which are likely to be able to spread an idea most effectively. And they also show how splogs (spam blogs) affect some of the heuristics such as in-degree, while others such as greedy and PageRank perform well even in presence of splogs. Moreover, they suggest PageRank as an inexpensive approximation to the greedy heuristic in selecting the initial target set for activation.

2.6 Topic-level Influence Analysis

Anagnostopoulos et al. [6] and Singla et al. [97] propose methods to qualitatively measure the existence of influence. In [33], Crandall et al. study the correlation between social similarity and influence. However, no previous work has been conducted for quantitatively measuring the topic-level social influence on large-scale networks.

Tang et al. [102] introduce the problem of topic-based social influence analysis and present a method to quantify the influential strength in social networks. Given a social network and a topic distribution for each user, the problem is to find topic-specific sub-networks, and topic-specific influence weights between members of the sub-networks. They propose a Topical Affinity Propagation (TAP) model to model social influence in a network with respect to different topics, which are extracted by using topic modeling methods. Later, Wang et al. [109] extend the TAP model further by considering the dynamic social influence. They propose a pairwise factor graph (PFG) model to model the pairwise influence by mainly using the topological structures. In the factor graph model, the pairwise influence is modeled as a marginal probability of two hidden variables. As social influences are highly time-dependent, they further propose a dynamic factor graph

(DFG) model to incorporate the time information, which is described as a factor function across time windows. Experiments show that their approach can facilitate influence maximization.

In [69], Liu et al. introduce a probabilistic model for mining direct and indirect influence between the nodes of heterogeneous networks. They measure influence based on the clearly observable "following" behaviors and study how the influence varies with number of hops in the network.

Weng et al. [113] study the problem of identifying topic-sensitive influential users on Twitter by proposing an extension of the PageRank algorithm to measure the influence taking both the topical similarity between users and the link structure into account. Their method leverages LDA by creating a single document from all the tweets of a user and then discovering the topics by running LDA over this document.

In [4], Ahmed et al. propose a unified framework, the nested Chinese Restaurant Franchise (nCRF), to discover a unified hidden tree structure with unbounded width and depth while allowing users to have different distributions over this structure. They apply the framework to organize tweets into a hierarchical structure and show that this tree structure can be used to predict locations of unlabeled messages, resulting in significant improvements to state-of-the-art approaches, as well as revealing interesting hidden patterns.

These methods on topic-level influence depend on known social network structure, i.e. influence only occurs along social connections. The temporal factor, which plays an important role in determining the degree of influence, is also not considered. In our work on inferring topic-level influence, we take into account the temporal factor in social influence to infer the influential strength between users at topic-level, without requiring the network structure.

Chapter 3

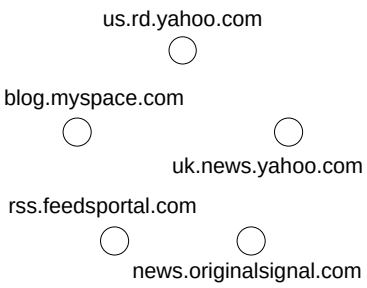
Mining Top-k Maximal Influential Paths

In this chapter, we develop a method for inferring top-k maximal influential paths. We propose a generative influence propagation model based on the Independent Cascade Model and Linear Threshold Model, which mathematically models the spread of certain information through a network. We formalize the top-k maximal influential path inference problem and develop an efficient algorithm, called TIP, to infer the top-k maximal influential paths. TIP makes use of the properties of top-k maximal influential paths to dynamically increase the support and prune the projected databases. As databases evolve over time, we also develop an incremental mining algorithm IncTIP to maintain top-k maximal influential paths. We evaluate the proposed algorithms on two real world datasets. The experimental results demonstrate the effectiveness and efficiency of both TIP and IncTIP.

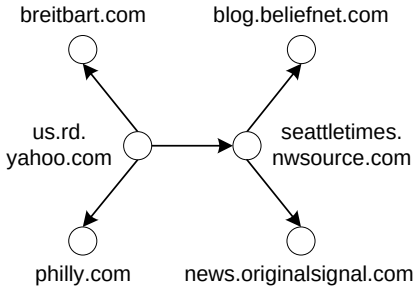
The remaining of this chapter is organized as follows: We start with the motivation of inferring top-k maximal influential paths in Section 3.1. In Section 3.2, we describe the preliminaries and problem statement. The TIP algorithm is explained in Section 3.3. In Section 3.4, we develop an incremental mining algorithm, named IncTIP, to maintain the set of top-k maximal influential paths efficiently. Experimental evaluation is reported in Section 3.5. Finally, we summarize our work in Section 3.6.

3.1 Motivation

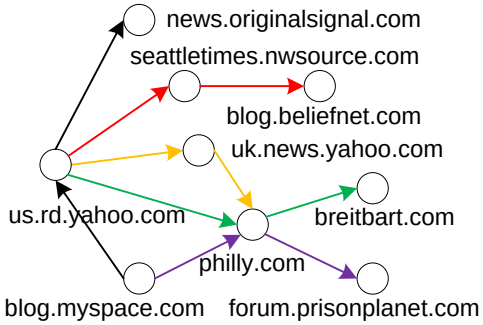
Early attempts to find the top-k influential users/nodes in a social network assume the existence of a social graph with edges labeled with probabilities of influence between users [55, 57, 64, 79, 28, 27]. However, this assumption is not realistic as such edges are often implicit or even unknown in the networks. Recent works aim to infer the "hidden" network from a list of observations of when and where an event occurs [44, 75]. The work in [44] infers top-k influential edges in the context of information propagation among blogs and online news sources where bloggers write about newly discovered information without explicitly citing the source. In other words, we can only observe the time when a blog gets influenced but not where it got the influence from.



(a) Top-5 influential nodes



(b) Top-5 influential edges



(c) Top-5 influential paths

Figure 3.1: MemeTracker dataset

Figures 3.1(a) and 3.1(b) show the top-5 influential nodes and top-5 influential edges obtained from MemeTracker dataset [63]. Each node in the network is a news website and a directed edge from node a to node b indicates that information has propagated from a to b. Based on the influential edges, we can only know that when the website *seattle-*

times.nwsource.com has new information, it gets propagated to either *blog.beliefnet.com* or *news.originalsignal.com* or both. However, if we have the top-5 influential paths as shown in Figure 3.1(c), then we see that a new piece of information gets propagated from *us.rd.yahoo.com* to *seattletimes.nwsource.com* to *blog.beliefnet.com*. Further, we observe that many of the influential paths pass through philly.com, making it a critical node. Critical nodes should have mirror sites as any disruption to these critical nodes may lead to news blackout.

Identifying critical nodes have many useful applications. In social network sites such as Twitter, identifying critical nodes in the rumor paths enables effective strategies to be formulated that target these critical nodes to counter the spread of rumors. Another important application of top-k influential paths and critical nodes is in the surveillance of computer virus propagation. Inferring the top-k influential paths from the list of sites infected by computer virus allows one to better understand how the virus spreads over time and stop the virus propagation by bringing down the critical nodes.

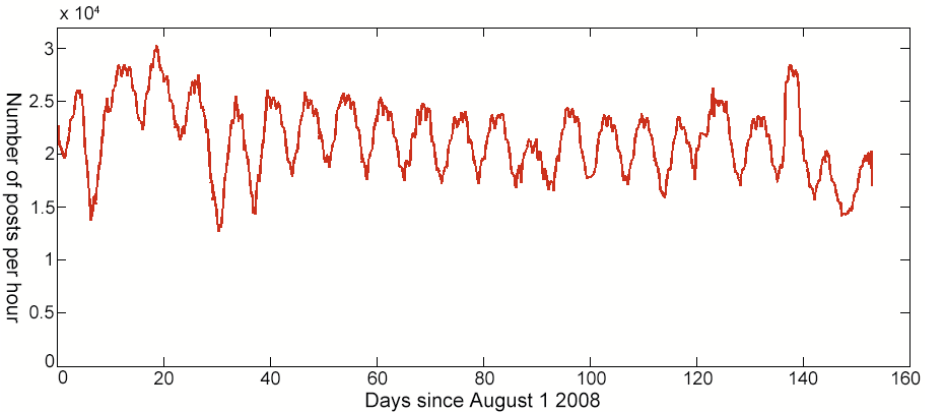


Figure 3.2: Number of news articles produced in MemeTracker dataset

One complication in the identification of influential paths is due to the fact that users of the online social media are active and regularly upload new information to the social media. For example, news websites regularly publish important information in MemeTracker dataset. On average, 20,000 news articles are produced per hour from August 2008 to January 2009 in the MemeTracker dataset (see Figure 3.2). Such updates may introduce new patterns or invalidate some existing patterns. Recomputing top-k maximal

influential paths for each update is very inefficient. Clearly, an incremental algorithm is needed to maintain the set of top-k influential paths efficiently.

3.2 Preliminaries

An influence network aims to capture the propagation of influence among a set of entities based on a list of observations. We model the network using a directed graph $G = (V, E)$ where V and E are the sets of nodes and edges respectively.

A node u in V denotes an entity and can be $active$ or $inactive$. It is considered $active$ if it has been influenced. Nodes can switch from being inactive to active, but not vice versa. When a node u gets influenced, it in turns may influence each of its currently inactive neighbors v with some small probability. Node u can only influence its neighbor v if their time difference is within some time threshold τ.

Each directed edge $(u, v) \in E$ has a weight $weight(u, v) \in [0, 1]$ denoting the likelihood of node v being influenced by node u. Suppose t_u and t_v are the times at which nodes u and v get influenced respectively. Then $weight(u, v)$ = 0 if $t_v \leq t_u$, i.e., nodes cannot be influenced by nodes from the future time points. Otherwise, $weight(u, v)$ = $e^{-\frac{t_v - t_u}{\alpha}}$, where α is radius of influence.

We associate each node u with an influence measure which is computed from the weights of the edges connecting u to its active neighboring nodes as follows:

$$influence(u, S) = 1 - \prod_{w \in S} (1 - weight(w, u)) \tag{3.1}$$

where S is the set of active neighbors of u.

One immediate concern is the cost of updating $influence(u, S)$ when the status of nodes change. Since the node status changes frequently, this update cost can be computationally expensive. We derive an expression that allows $influence(u, S)$ to be updated incrementally.

Suppose a new neighboring node w of u becomes active. Then

$$\begin{aligned} influence(u, S \cup \{w\}) &= 1 - (1 - weight(w,u)) * \prod_{u' \in S} (1 - weight(u', u)) \\ &= 1 - (1 - weight(w,u)) * (1 - influence(u, S)) \\ &= influence(u, S) + (1 - influence(u, S)) * weight(w, u) \end{aligned} \tag{3.2}$$

We observe that the influence measure $influence(u, S)$ is both monotonic and sub-modular.

A function $f(.)$ is $monotonic$ if $f(S) \leq f(S')$, for $S \subseteq S'$. From Equation 3.2, we have

$$influence(u, S \cup \{w\}) - influence(u, S) = (1 - influence(u, S)) * weight(w, u) \geq 0$$

A function $f(.)$ is $submodular$ if $f(S \cup \{w\}) - f(S) \geq f(S' \cup \{w\}) - f(S')$, for $S \subseteq S'$. This means that adding a node w to S increases the score more than adding w to S' when $S \subseteq S'$. We show that $influence(u, S)$ is sub-modular as follows:

$$\begin{aligned} &influence(u, S \cup \{w\}) - influence(u, S) - (influence(u, S' \cup \{w\}) - influence(u, S')) \\ &= (1 - influence(u, S)) * weight(w, u) - (1 - influence(u, S')) * weight(w, u) \\ &= (influence(u, S') - influence(u, S)) * weight(w, u) \end{aligned} \tag{3.3}$$

By monotonicity, $influence(u, S') \geq influence(u, S)$. Hence,

$$(influence(u, S') - influence(u, S)) * weight(w, u) \geq 0$$

Definition 1. Observation. *An* observation $o = < (u_1, t_1), (u_2, t_2), \cdots, (u_n, t_n) >$ *is a sequence of tuples* (u_i, t_i) *where* t_i *is the time when node* u_i *becomes active, and* $\forall\ i < j$, $t_i < t_j$. *Further,* $u_i \neq u_j\ \forall\ i \neq j$. *The* $length$ *of observation* o, *denoted as* $|o|$, *is the*

number of (u_i, t_i) *tuples in* o.

Definition 2. Influential Path. *An* influential path *is a sequence of nodes, denoted as* p = $< v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_n >$, *such that weight*$(v_i, v_{i+1})$ *is larger than some user defined threshold for all* i, $1 \leq i \leq n-1$. *The* $length$ *of* p *is given by* $|p|$ = $n-1$.

Definition 3. Support. *An observation* o supports *an influential path* p *if*

- $\forall v \in p$, $v \in \{ u_i \mid (u_i, t_i) \in o \}$, *and*
- *if* u_i *and* u_j *are nodes in* o *that correspond to* $v_{i'}$ *and* $v_{i'+1}$, *then* $0 < t_j - t_i < \tau$, $1 \leq i' \leq n-1$.

Let D be an observation database, which consists of a set of observations. The $support$ of an influential path p, denoted as $support(p)$, is the number of observations in D that support p.

The $score$ of a path $p = < v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_n >$ w.r.t. an observation o is defined as

$$score(p, o) = \log(influence(v_1, S) \prod_{1 \leq i \leq n-1} weight(v_i, v_{i+1})) - \log \epsilon, \tag{3.4}$$

where $\epsilon \in [0, 1]$ is some small value and S is the set of active neighbors of v_1 w.r.t. o.

Let S_p be the set of observations in D that support influential path p. The *total score* of p, denoted as $total\_score(p)$, is defined by

$$total\_score(p) = \sum_{o \in S_p} score(p, o). \tag{3.5}$$

An influential path $p = < v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_m >$ is a $sub\text{-}path$ of another influential path $p' = < v'_1 \rightarrow v'_2 \rightarrow \cdots \rightarrow v'_n >$, denoted as $p \sqsubseteq p'$, if and only if $\exists\ i_1, i_2, \cdots, i_m$, such that $1 \leq i_1 < i_2 < \cdots < i_m \leq n$, and $v_1 = v'_{i_1}$, $v_2 = v'_{i_2}$, $\cdots$, and $v_m = v'_{i_m}$. We also call p' a $super\text{-}path$ of p.

An influential path p is **maximal** if there exists no influential path p' such that $p \sqsubseteq p'$ and support(p) = support(p').

Definition 4. Top-k Maximal Influential Path. *An influential path p is a* top-k maximal influential path *if p is maximal and there exist no more than $(k-1)$ maximal influential paths whose total score is greater than that of p.*

The following theorem states the relation between the support and total score of two maximal influential paths. This theorem is utilized by our proposed algorithm in Section 3.3 to effectively prune off the search space.

Theorem 1. *For any two maximal influential paths p and p', if $support(p) > support(p')$ and $\epsilon < e^{-|D|(|o|+1)\tau}$, then $total\_score(p) > total\_score(p')$ where o is an observation with maximum length in database D.*

Proof. Let p be a maximal influential path with support s and length $|p|$. We can calculate the total score of path p as

$$\begin{aligned}
total_score(p) &= \sum_{o \in S_p} score(p, o) \\
&> (\log e^{-\tau} + |p| * \log e^{-\tau} - \log \epsilon) * s \\
&= -s\tau - s|p|\tau - s * \log \epsilon \\
&= -s\tau - s|p|\tau - s * \log \epsilon + \log \epsilon - \log \epsilon \\
&= (-\log \epsilon) * (s-1) + (-s\tau - s|p|\tau - \log \epsilon) \\
&> (-\log \epsilon) * (s-1) + (-s\tau - s|p|\tau - \log e^{-|D|(|o|+1)\tau}) \\
&= (-\log \epsilon) * (s-1) + (|D|(|o|+1) - s(|p|+1))\tau \\
&> (-\log \epsilon) * (s-1)
\end{aligned}$$

Since $(|D|(|o|+1) - s(|p|+1)) \geq 0$, we have $(\log e^{-\tau} + |p| * \log e^{-\tau} - \log \epsilon) * s > (-\log \epsilon) * (s-1)$. Note that $(\log e^{-\tau} + |p| * \log e^{-\tau} - \log \epsilon) * s$ is the lower bound for the $total\_score$ of any maximal influential path with support s, and $(-\log \epsilon) * (s-1)$ is the upper bound for the $total\_score$ of any maximal influential path with support $(s-1)$. Further, the value of $total\_score$ decreases with the length of a path. Hence, $(\log e^{-\tau} + |p| * \log e^{-\tau} - \log \epsilon) * s > (-\log \epsilon) * (s-1)$ implies that the $total\_score$ of any

maximal influential path with support s is greater than all the maximal influential paths whose support is less than s. □

3.3 The TIP Algorithm

In this section, we first briefly review works in sequential pattern mining that are related to our TIP method and then give the details of the TIP algorithm. Sequential pattern mining [100], which discovers frequent subsequences as patterns in a sequence database, is an important data mining problem. Recent studies have developed two major classes of sequential pattern mining method: Apriori-based approaches such as GSP [100], SPADE [117], SPAM [8] and Pattern-Growth-based approaches such as FreeSpan [49], PrefixSpan [83]. The key advantage of PrefixSpan is that it does not generate any candidate sequences. Its general idea is to examine only the frequent prefix subsequences and project only their corresponding postfix subsequences into projected databases because any frequent subsequence can always be found by growing a frequent prefix.

A major challenge in mining frequent sequential patterns from a large data set is the fact that such mining often generates a huge number of patterns satisfying the min_sup threshold, especially when min_sup is set low. This is because if a sequential pattern is frequent, each of its sub-patterns is frequent as well. A large sequential pattern will contain an exponential number of smaller, frequent sub-patterns. To overcome this problem, closed frequent sequential pattern mining methods have been proposed (e.g. CloSpan [114], BIDE [111]). CloSpan [114] adopts a two-phase strategy for mining closed sequential patterns. In the first phase, it finds a superset of the set of final closed patterns. In the second phase, it eliminates non-closed patterns using a hash index.

Mining closed patterns may significantly reduce the number of patterns generated and is information lossless because it can be used to derive the complete set of sequential patterns. However, setting min support is a subtle task: A too small value may lead to the generation of thousands of patterns, whereas a too big one may lead to no answer found. To come up with an appropriate min support, one needs prior knowledge about the mining

query and the task-specific data, and be able to estimate beforehand how many patterns will be generated with a particular threshold. TFP [50] and TSP [106] algorithms have been proposed to discover top-k frequent closed patterns without a predefined min_sup threshold. TFP [50] is an FP-tree based frequent pattern mining algorithm. It starts the mining process with min_sup threshold equal to 1, and raises the support threshold during both the FP-tree construction and the mining of the FP-tree. Giannotti et al. [41] introduce a novel form of sequential pattern, called Temporally-Annotated Sequence (TAS), representing typical transition times between the events in a frequent sequence.

Our TIP algorithm is different from the above sequential pattern mining methods. TIP is a prefix-based influential path mining method in the context of information diffusion. We also consider the temporal information which is ignored by existing works.

Next, we describe our method, TIP, for mining top-k maximal influential paths without the need to specify a minimum support threshold. Initially, the minimum support threshold min_sup is set to 1. TIP is a prefix-based influential path mining method. It extends the classical projection-based pattern growth method [83] with time constraint. Instead of projecting observation databases by considering all possible occurrences of prefixes, TIP examines the frequent prefix sub-paths and projects only the corresponding valid observations which satisfy the time constraint into the projected databases. The influential paths are then extended by exploring the valid frequent nodes in the projected databases.

Given an influential path $p = < v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_n >$ and a node α, we can extend p by α if the last node of p, i.e. v_n, can $influence$ α, that is, the time difference between t_{v_n} and t_α is within the time threshold τ. We denote the extension as $p \rightarrow \alpha = < v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_n \rightarrow \alpha >$.

Let $p' = p \rightarrow \alpha$ be an extension of p. we say p is a $prefix$ of p' and α is a $suffix$ of p'. For example, in our sample observation database D as shown in Table 3.1, $< a \rightarrow d \rightarrow g >$ is a prefix of path $< a \rightarrow d \rightarrow g \rightarrow i >$ and $< i >$ is its suffix.

Let S_p be the set of observations that support influential path p. Suppose each $o \in S_p$

is of the form $< (u_1, t_1), (u_2, t_2), \cdots, (u_a, t_a), (u_{a+1}, t_{a+1}), \cdots, (u_b, t_b) >$. Then we define the $p\text{-}projected$ database as $D_p = \{ < (u_{a+1}, t_{a+1}), \cdots, (u_b, t_b) > \}$ if the last node $v_n \in p$ corresponds to $u_a \in o$ and the time difference $t_{a+1} - t_a$ is less than τ.

Table 3.1: A sample observation database D

ID	Observation
o_1	<(a,1) (d,5) (g,10) (i,16)>
o_2	<(c,8) (e,15) (f,20)>
o_3	<(c,4) (d,10) (g,16) (i,20)>
o_4	<(c,3) (e,12) (i,36)>
o_5	<(c,5) (e,9) (h,20) (i,24)>

Consider the sample observation database in Table 3.1. Let time threshold τ = 20. The projected database for path $< c \rightarrow e >$ is $D_{<c \rightarrow e>}$ = $\{< (f, 20) >, < (h, 20), (i, 24) >\}$. Note that for observation o_4, the time stamp of e is 12, while the next time stamp is 36. Since the time difference is 24 which is more than τ, node e cannot influence node i, and hence $< (i, 36) >$ is not included in the projected database.

Algorithm 2 TIPMiner(D, k, τ)

Require: global variable $PathSet$
Require: observation database D, an integer k and time threshold τ
Ensure: Top-k maximal influential path set $PathSet$
1: $V \leftarrow$ nodes in D
2: Initialize min_sup = 1
3: Initialize $PathSet$ = $\emptyset$
4: Let $root$ be the root node
5: **for each** node $v \in V$ **do**
6: Create child node v of $root$ and record support count and IDs of the supporting observations of v
7: Update $PathSet$ by calling TIP($< v >$, $D_{<v>}$, k, min_sup, τ, $PathSet$)
8: **end for**
9: return $PathSet$

Having defined the concept of path-projected databases, we next describe the framework TIPMiner for mining the top-k maximal influential paths from a given observation database D. Algorithm 2 gives the details. It first finds all the nodes in D and sorts them in decreasing order of their support values. A global variable $PathSet$ is used to keep

track of the set of top-k maximal influential paths. This global variable is updated by calling Algorithm TIP (see Algorithm 3) for each node.

Algorithm TIP finds the top-k maximal influential paths by constructing projected databases. Inputs to TIP algorithm are an influential path p, the p-projected database D_p, the number of maximal influential paths k, minimum support threshold min_sup, time threshold τ, and $PathSet$. The outputs are the set of top-k maximal influential paths $PathSet$.

Given an influential path p, TIP algorithm attempts to extend p by first obtaining the p-projected database D_p. Initially, the path consists of only one node. Given a path p, we first check if this path is promising (Lines 1-3). A path is promising if its support is no less than the minimum support threshold. We calculate the $total\_score$ of path p (Lines 4-5). Line 6 checks whether there exists an influential path $p' \in PathSet$ such that p is a sub-path or super-path of p'. If p' exists, we perform maximal influential path verification (Lines 8-15). If p' is a sub-path of p, then we replace p' by p in the $PathSet$ since p is now the maximal influential path (Lines 12-14). However, if p' is a super-path of p, then p is not a maximal influential path and can be discarded (Lines 9-11).

If p' does not exist and $PathSet$ contains less than k maximal influential paths, then we add p to the $PathSet$ (Lines 17-18). Otherwise, if $PathSet$ already contains k maximal influential paths, we check the $total\_score$ of p. If the $total\_score$ of p is larger than any of the k maximal influential paths in $PathSet$, we replace the path with the smallest $total\_score$ by p (Lines 19-24). By Theorem 1, we raise min_sup to the support of the path whose $total\_score$ is the minimum in $PathSet$ (Lines 26-29). This allows us to prune off unpromising paths.

Next, the algorithm attempts to extend p by finding all the frequent nodes $\alpha \in D_p$ such that we can extend p to $p \rightarrow \alpha$ (Lines 30-41). We scan the p-projected database D_p to find every frequent node α, such that path p can be extended to $p \rightarrow \alpha$, and insert α into a priority queue Q (Lines 31-36). We recursively call TIP algorithm to extend another path using the next frequent node in Q (Lines 37-41). The algorithm terminates

Algorithm 3 TIP(p, D_p, k, min_sup, τ, $PathSet$)

Require: global variable $PathSet$, min_sup
Require: a path p, D_p, an integer k and time threshold τ
Ensure: Top-k maximal influential path set $PathSet$
if $support(p) < min\_sup$ **then**
  return
end if
let S_p be the set of observations that support p
calculate $total\_score(p) = \sum_{o \in S_p} score(p, o)$
check whether a discovered influential path $p' \in PathSet$ exists, s.t. either $p \sqsubseteq p'$ or $p' \sqsubseteq p$, and $support(p) = support(p')$
if such super-path or sub-path exists **then**
  for each $p' \in PathSet$ such that $support(p') = support(p)$ **do**
    if $p \sqsubseteq p'$ **then**
      return
    end if
    if $p' \sqsubseteq p$ **then**
      replace p' with p
    end if
  end for
else
  if $|PathSet| < k$ **then**
    $PathSet = PathSet \cup \{p\}$
  else
    let path $q \in PathSet$ such that $\nexists\, q' \in PathSet, total\_score(q') < total\_score(q)$
    if $total\_score(p) > total\_score(q)$ **then**
      replace q with p
    end if
  end if
end if
if $|PathSet| = k$ **then**
  let path $q \in PathSet$ such that $\nexists\, q' \in PathSet, total\_score(q') < total\_score(q)$
  $min\_sup = support(q)$
end if
$Q \leftarrow$ empty priority queue
compute the frequency of each node in D_p
for each frequent node α **do**
  if p can be extended to $p \rightarrow \alpha$ **then**
    Q.insert(α)
  end if
end for
while !Q.isEmpty() **do**
  $\alpha = Q$.pop()
  create child node α of the last node of p and record support count and IDs of the supporting observations of α
  Call TIP($p \rightarrow \alpha$, $D_{p \rightarrow \alpha}$, k, min_sup, τ, $PathSet$)
end while
return

when Q is empty.

Let us now use the example in Table 3.1 to illustrate the TIP algorithm. The entity with the highest support value is c (see Table 3.2). We obtain the projected database $D_{<c>}$ as shown in Table 3.3. The frequent nodes with their support values are shown in Table 3.4. We insert these nodes into the priority queue Q and recursively call TIP to extend $< c >$. Since node e has support 3 in Q, we extend $< c >$ to $< c \rightarrow e >$.

Table 3.2: Frequent nodes in D

Node	Count
c	4
i	4
e	3
d	2
g	2
a	1
f	1
h	1

Table 3.3: $< c >$-projected database $D_{<c>}$

ID	Observation
o_2	<(e,15) (f,20)>
o_3	<(d,10) (g,16) (i,20)>
o_4	<(e,12) (i,36)>
o_5	<(e,9) (h,20) (i,24)>

Table 3.4: Frequent nodes in $D_{<c>}$

Node	Count
e	3
i	2
d	1
f	1
g	1
h	1

Conceptually, the TIP algorithm is constructing a prefix search tree where node in the tree corresponds to an influential path starting from the root to the node and its support

is shown next to the node as shown in Figure 3.3. The number along each edge denotes the $total\_score$ of the path from the root to the end node of the edge. We assume that the time threshold $\tau = 20$ and $\epsilon = e^{-64}$. We observe that $< c \rightarrow e >$ are supported by three observations o_2, o_4 and o_5 in Table 3.1. The scores with respect to these observations are as follows:

$$\begin{aligned} score(p, o_2) &= \log(influence(c, S) * weight(c, e)) - \log \epsilon \\ &= \log e^{-\frac{15-8}{1.0}} - \log e^{-64} \\ &= 57 \end{aligned}$$

Similarly, we have $score(p, o_4)$ = 55 and $score(p, o_5)$ = 60. Thus the $total\_score$ of the influential path $p = < c \rightarrow e >$ is $total\_score(p)$ = 57 + 55+ 60 = 172. In the same manner, we build $< c \rightarrow e >$-projected database and extend $< c \rightarrow e >$ to $< c \rightarrow e \rightarrow f >$.

Suppose we wish to find the top-2 maximal influential paths. After obtaining the paths $< c \rightarrow e >$ and $< c \rightarrow i >$, the min_sup is raised to 2. This implies that all the branches rooted at node a are pruned as their support values are less than 2. Similarly, branches rooted at node e are also pruned as they have already been traversed previously from node c. The bold lines in Figure 3.3 show the explored paths.

To further improve the efficiency of TIP algorithm, we propose two optimization strategies.

Early Termination by Equivalence. Early termination by equivalence is a search space reduction technique developed in CloSpan [114]. Let $N(D)$ represent the total number of nodes in D. The property of early termination by equivalence shows that if two influential paths $p \sqsubseteq p'$ and $N(D_p) = N(D_{p'})$, then $\forall\ \gamma$, $support(p \rightarrow \gamma) = support(p' \rightarrow \gamma)$. It means the descendants of p in the prefix search tree cannot be maximal. Furthermore, the descendants of p and p' are exactly the same. We can utilize this property to quickly prune the search space of p.

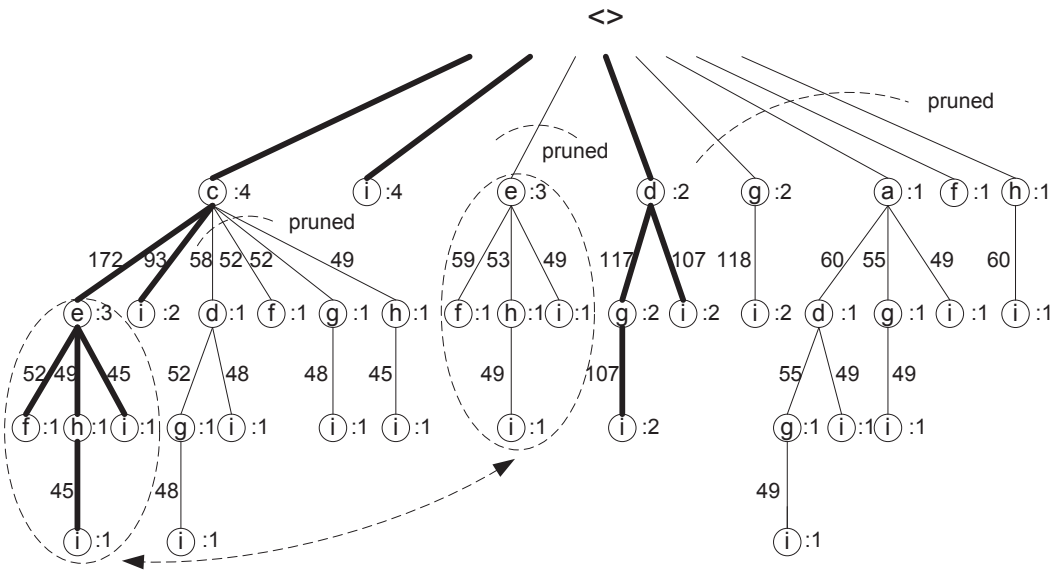


Figure 3.3: Prefix search tree for sample database

Pseudo Projection. As with traditional projection-based mining method, the major cost of TIP is the construction of projected databases. To reduce the cost of projection, we apply the pseudo-projection technique [83]. Instead of constructing a *physical projection* by collecting all the postfixes, we use pointers referring to the observations in the database as a *pseudo projection*. Every projection consists of two pieces of information: $pointer$ to the observation in database and $offset$ of the postfix in the observation. This allows us to avoid physically copying postfixes: only pointers to the projected point are saved for each observation. Thus, it is efficient in terms of both running time and space.

3.4 Incremental Mining

One challenge in finding the top-k maximal influential paths in social networks is that most users are active and updates tend to be frequent and voluminous. In general, there are three kinds of updates: (1) new observation arrives. This corresponds to an INSERT operation. (2) new follow-up action is observed later. This corresponds to an APPEND operation. (3) an existing observation is no longer valid and should be removed. This corresponds to a DELETE operation. We can consider APPEND as deleting an existing

observation and inserting a new one. For example, if we wish to append the tuple $< (g, 26) >$ to observation o_2 in Table 3.1, we first delete o_2 and insert the observation o_2': $< (c, 8)(e, 15)(f, 20)(g, 26) >$ into D.

Invoking TIP for each update is infeasible. In this section, we describe an incremental mining algorithm to mine top-k maximal influential paths. We first briefly review incremental sequential pattern mining algorithms that are most relevant to our incremental mining method. Sequential pattern mining [100] is to find frequent subsequences from a sequence database. In many applications, databases are updated incrementally, which leads to the study of incremental mining of sequential patterns. Incremental sequential pattern mining methods can be classified into two categories, Apriori-based methods (e.g. ISM [82], ISE [74], and GSP+ [119]) and projection-based methods (e.g. IncSpan [31], IncSpan+ [81], PBIncSpan [30], and ISPBS [68]). Apriori-based incremental mining methods would generate huge set of candidate sequences, while projection-based incremental mining methods can avoid this by using pattern growth approach to mine sequential patterns.

Cheng et al. [31] propose an incremental mining algorithm, called IncSpan, by taking advantage of PrefixSpan [83]. IncSpan buffers a set of semi-frequent sequences for incremental mining. Later, Nguyen et al. [81] clarify that IncSpan cannot find the complete set of sequential patterns in the updated database and propose a new algorithm called IncSpan+. In [30], Chen et al. argue that in general IncSpan+ cannot find complete set of sequential patterns, and propose a new incremental mining algorithm based on prefix tree, called PBIncSpan. PBIncSpan constructs a prefix tree to represent the sequential patterns and maintains the tree structure using width pruning and depth pruning when database updates.

Our incremental mining method IncTIP is quite different from existing works on incremental pattern mining. We extend the pattern growth method with time constraint, and introduce a score function to measure different patterns.

Next, we describe our IncTIP algorithm to allow for incremental mining of top-k

maximal influential paths. The main idea in incremental mining is to leverage on the computations done previously. In order to do this, we need to store additional information for each node, namely the support count for each of its extended child and the IDs of the supporting observations. Figure 3.4 shows the additional information we keep for $root$ and node c, e in the explored paths of the prefix tree.

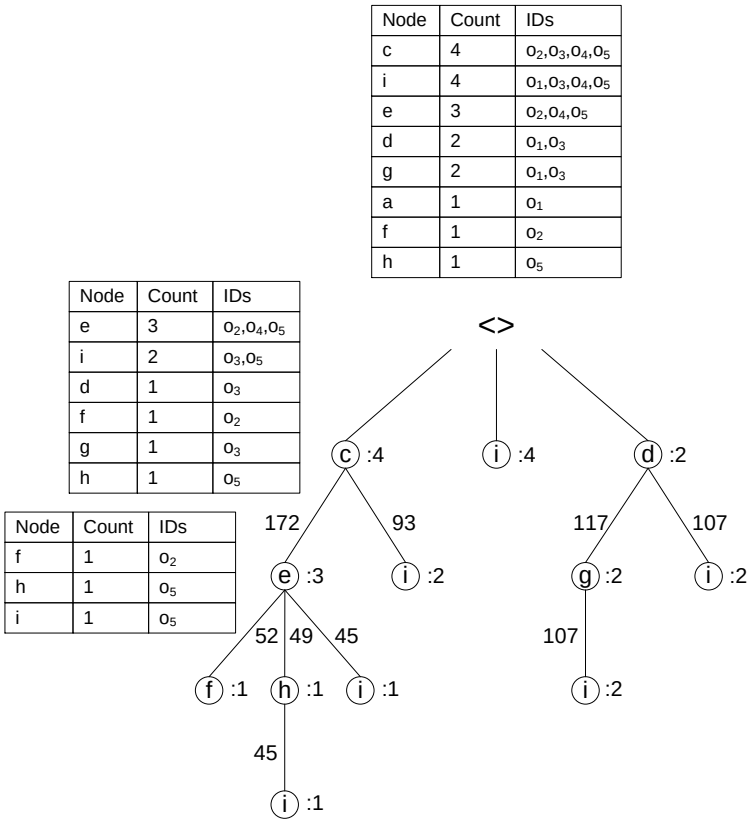


Figure 3.4: Prefix tree with additional information for $root$ and node c, e

The IncTIP algorithm for incremental mining of top-k maximal influential paths is given in Algorithm 4. The inputs are database D, set of updates U, an integer k, top-k maximal influential path set $PathSet$ and the corresponding final min_sup, and time threshold τ. The output is the set of top-k maximal influential paths $PathSet$. For each update, we first check whether it is INSERT or DELETE (Line 2). If the update is INSERT of an observation o, then for each node v in o, we scan additional information table of $root$ and check whether v is frequent or not (Line 5). If it is frequent, we update the $PathSet$ by calling the INSERT algorithm (Lines 6-7). Otherwise, we call the TIP algorithm (Lines 8-12). If the update is DELETE of an observation o, we update the

$PathSet$ by calling the DELETE algorithm (Lines 15-17). The global variable $PathSet$, which keeps track of the set of top-k maximal influential paths, is updated by calling the appropriate algorithms. Algorithm 5 and 6 gives the details of INSERT and DELETE respectively. We will illustrate them in detail in the following subsections with our running example.

Algorithm 4 IncTIP(D, U, k, min_sup, τ, $PathSet$)

Require: global variable $PathSet$, min_sup
Require: database D, set of updates U, an integer k and time threshold τ
Ensure: Top-k maximal influential path set $PathSet$

```
1: let root be the root node
2: for each update in U do
3:    if INSERT o then
4:       for each node v ∈ o do
5:          scan additional information table of root, check whether v is frequent or not
6:          if v is frequent then
7:             Update PathSet by calling INSERT(D, v, o, k, min_sup, τ, PathSet)
8:          else
9:             let I be the set of observations in D ∪ {o} that support v
10:            let I_<v> be v-projected database
11:            Update PathSet by calling TIP(v, I_<v>, k, min_sup, τ, PathSet)
12:         end if
13:      end for
14:   else
15:      if DELETE o then
16:         Update PathSet by calling DELETE(D, root, o, k, min_sup, τ, PathSet)
17:      end if
18:   end if
19: end for
20: return PathSet
```

3.4.1 Insert Observation

Suppose we insert a new observation o_6: $< (a, 2)(d, 7)(i, 13) >$ into the sample observation database D in Table 3.1. The new observation database D' after insertion is shown in Table 3.5.

Table 3.5: New database D' after insertion

ID	Observation
o_1	<(a,1) (d,5) (g,10) (i,16)>
o_2	<(c,8) (e,15) (f,20)>
o_3	<(c,4) (d,10) (g,16) (i,20)>
o_4	<(c,3) (e,12) (i,36)>
o_5	<(c,5) (e,9) (h,20) (i,24)>
o_6	<(a,2) (d,7) (i,13)>

Recall that in our previous running example for the TIP algorithm, we find top-2 maximal influential paths and the min_sup is finally raised to 2. So all the branches rooted at node a are pruned as their support values are less than 2 (see Figure 3.3). However, after inserting observation o_6, the support of node a becomes 2, implying that we should mine influential paths starting at node a. Based on the additional information for the root node as shown in Table 3.6, we know that in the original database observation o_1 supports node a. So observations that support node a are observation o_1 and the inserted observation o_6.

Table 3.6: Additional information for root node

Node	Count	IDs
c	4	o_2,o_3,o_4,o_5
i	4	o_1,o_3,o_4,o_5
e	3	o_2,o_4,o_5
d	2	o_1,o_3
g	2	o_1,o_3
a	1	o_1
f	1	o_2
h	1	o_5

For node a, we call the TIP algorithm (Algorithm 3). We obtain $< a >$-projected database $I_{<a>}$ as shown in Table 3.7. The frequent nodes with their support values are shown in Table 3.8. We insert these nodes into the priority queue Q and recursively call TIP to extend $< a >$. Since node d has support 2 in Q, we extend $< a >$ to $< a \rightarrow d >$. By recursively calling the TIP algorithm, we obtain the path $< a \rightarrow d \rightarrow i >$ and the other paths are pruned.

Table 3.7: $< a >$-projected database $I_{<a>}$

ID	Observation
o_1	<(d,5) (g,10) (i,16)>
o_6	<(d,7) (i,13)>

Table 3.8: Frequent nodes in $I_{<a>}$

Node	Count
d	2
i	2
g	1

For node d, we update $PathSet$ by calling the INSERT algorithm, as we observe from the prefix search tree in Figure 3.3 that node d has already been traversed in the previous mining result.

Algorithm 5 gives the details of INSERT algorithm. The inputs are database D, node v, observation o, an integer k, min_sup, time threshold τ, and top-k maximal influential path set $PathSet$. The output is the set of top-k maximal influential paths $PathSet$. We first check whether observation o supports v (Line 1). If o supports v, we update the $support$ of v and $total\_score$ of path $< root \rightarrow \cdots \rightarrow v >$ and meanwhile update IDs of the supporting observations of node v (Lines 2-5). For each child α of node v, if α is frequent, we recursively call the INSERT algorithm (Lines 7-8). Otherwise, we call the TIP algorithm to explore branches that are pruned previously for possible top-k maximal influential paths (Lines 9-15). Finally, we update the top-k maximal influential path set $PathSet$ (Lines 18-21) and min_sup (Lines 22-23). With the insertion of observation o_6, we update the $support$ of d to 3. Based on the additional information for node d as shown in Table 3.9, we know that the child node i is supported by o_6, so we update the $support$ of i and meanwhile update the $total\_score$ of path $< d \rightarrow i >$.

For node i in observation o_6, as it is already traversed, we call the INSERT algorithm and update the $support$ of i to 5. Figure 3.5 shows the prefix search tree constructed after inserting observation o_6. The bold lines represent the explored paths.

Algorithm 5 INSERT(D, v, o, k, min_sup, τ, $PathSet$)

Require: global variable $PathSet$, min_sup
Require: database D, node v, observation o, an integer k and time threshold τ
Ensure: Top-k maximal influential path set $PathSet$

if o supports v **then**
  $support(v)$ = $support(v)$ + 1
  let path $p = < root \rightarrow \cdots \rightarrow v >$
  $total\_score(p)$ = $total\_score(p)$ + $score(p, o)$
  add ID of o to IDs of the supporting observations of node v
  for each child α of v **do**
    if α is frequent **then**
      Call INSERT(D, α, o, k, min_sup, τ, $PathSet$)
    else
      if o supports α **then**
        let I be the set of observations in $D \cup \{o\}$ that support $< p \rightarrow \alpha >$
        let $I_{<p \rightarrow \alpha>}$ be $< p \rightarrow \alpha >$-projected database
        Call TIP($< p \rightarrow \alpha >$, $I_{<p \rightarrow \alpha>}$, k, min_sup, τ, $PathSet$)
      end if
    end if
  end for
end if
let path $q \in PathSet$ such that $\nexists\, q' \in PathSet, total\_score(q') < total\_score(q)$
if $\exists\, p' \in T \setminus PathSet$ such that $total\_score(p') > total\_score(q)$ **then**
  replace q with p'
end if
let path $q \in PathSet$ such that $\nexists\, q' \in PathSet, total\_score(q') < total\_score(q)$
min_sup = $support(q)$
return

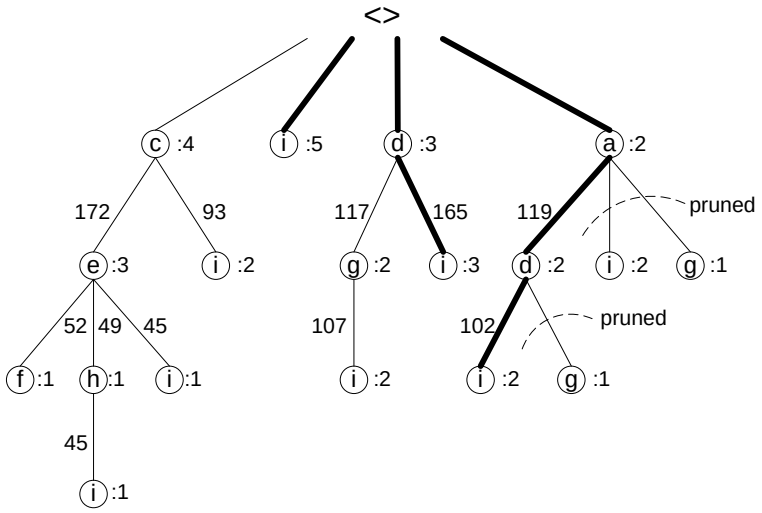


Figure 3.5: Prefix search tree for new database after inserting observation o_6

Table 3.9: Additional information for node d

Node	Count	IDs
g	2	o_1,o_3
i	2	o_1,o_3

3.4.2 Delete Observation

Suppose we delete observation o_4: $< (c,3)(e,12)(i,36) >$ from the sample observation database D in Table 3.1. We update $PathSet$ by calling the DELETE algorithm. Algorithm 6 gives the details of DELETE algorithm. The inputs are database D, node v, observation o, an integer k, min_sup, time threshold τ, and top-k maximal influential path set $PathSet$. The output is the set of top-k maximal influential paths $PathSet$. We first scan the additional information table of node v to find every node α such that o supports α (Line 1). For each node α, we update the $support$ of α and $total\_score$ of path $< root \rightarrow \cdots \rightarrow \alpha >$ and meanwhile update IDs of the supporting observations of node α (Lines 3-6). We recursively call the DELETE algorithm on node α (Line 7). After deleting observation o, we update the top-k maximal influential path set $PathSet$ (Lines 9-12) and min_sup (Lines 13-14). Finally, we call the TIP algorithm to explore branches that are pruned previously for possible top-k maximal influential paths (Lines 15-20).

As observation o_4 is deleted from the sample database D, the support of node c, e and i will decrease. Note that we utilize the additional information for each node in the prefix tree as shown in Figure 3.4. Starting from the root node, based on the additional information for root node (Table 3.6), we know node c and i are supported by observation o_4, so we decrease their support by 1. As for node c, based on its additional information (Table 3.10), the child e is also supported by o_4, so we update the $support$ of node e and meanwhile update the $total\_score$ of path $< c \rightarrow e >$. Figure 3.6 shows the prefix search tree after deleting observation o_4. The bold lines represent the explored paths.

Algorithm 6 DELETE(D, v, o, k, min_sup, τ, $PathSet$)

Require: global variable $PathSet$, min_sup
Require: database D, node v, observation o, an integer k and time threshold τ
Ensure: Top-k maximal influential path set $PathSet$
1: scan additional information table of node v, find every node α such that o supports α
2: **for each** node α **do**
3: $support(\alpha) = support(\alpha) - 1$
4: let path $p = < root \rightarrow \cdots \rightarrow \alpha >$
5: $total\_score(p) = total\_score(p) - score(p, o)$
6: remove ID of o from IDs of the supporting observations of node α
7: **Call** DELETE(D, α, o, k, min_sup, τ, $PathSet$)
8: **end for**
9: let path $q \in PathSet$ such that $\nexists\, q' \in PathSet, total\_score(q') < total\_score(q)$
10: **if** $\exists\, p' \in T \setminus PathSet$ such that $total\_score(p') > total\_score(q)$ **then**
11: replace q with p'
12: **end if**
13: let path $q \in PathSet$ such that $\nexists\, q' \in PathSet, total\_score(q') < total\_score(q)$
14: $min\_sup = support(q)$
15: scan additional information table of $root$, find every node v' that is not frequent
16: **for each** node v' **do**
17: let I be the set of observations in $D \setminus \{o\}$ that support v'
18: let $I_{v'}$ be v'-projected database
19: **Call** TIP(v', $I_{v'}$, k, min_sup, τ, $PathSet$)
20: **end for**
21: **return**

Table 3.10: Additional information for node c

Node	Count	IDs
e	3	o_2,o_4,o_5
i	2	o_3,o_5
d	1	o_3
f	1	o_2
g	1	o_3
h	1	o_5

3.4.3 Complexity Analysis

In this section, we provide a brief analysis of the time and space complexity of TIP and IncTIP algorithms. The major cost of the TIP algorithm is the construction of projected databases. In the worst case, when no pruning takes place, TIP constructs a projected database for every observation in the database. Thus, both the worst-case time and space

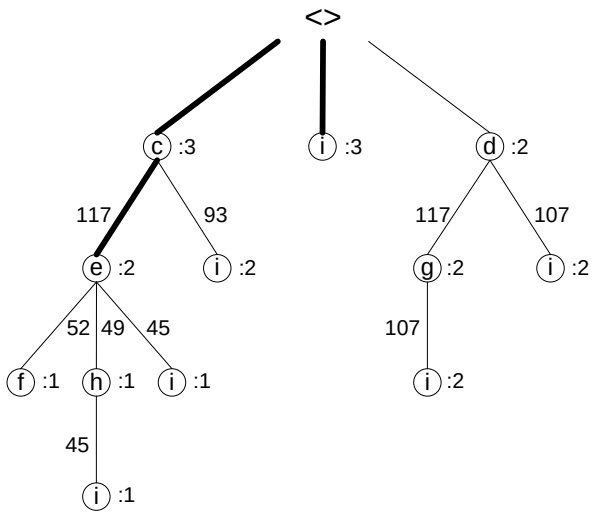


Figure 3.6: Prefix search tree for new database after deleting observation o_4

complexities are $O(N^L)$ where N is the number of tuples in the database and L is the maximum length of all observations. In addition, since we use pseudo-projection in our implementation, the space complexity can be reduced to the order of the size of the database.

Similar to the TIP algorithm, the worst-case time complexity of IncTIP is $O(N^L)$ where N is the number of tuples in the database and L is the maximum length of all observations. For the IncTIP algorithm, we keep child node information for each node in the prefix tree to facilitate incremental mining. So the worst-case space complexity of IncTIP is $O((N + C)^L)$ where C is the number of child nodes for each node in the tree.

3.5 Experimental Evaluation

In this section, we conduct experiments to evaluate the effectiveness and efficiency of our proposed TIP and IncTIP algorithms. In the first set of experiments, we compare the TIP algorithm with the Naïve algorithm that finds the top-k influential paths without any optimization techniques. We also analyze the effectiveness of the two optimization strategies by implementing two versions of TIP, TIP^{early} and TIP^{pp}, where TIP^{early} utilizes only the early termination strategy without pseudo projection whereas TIP^{pp} utilizes only the

pseudo projection technique without early termination. In the second set of experiments, we compare efficiency of TIP and IncTIP algorithms for incremental mining.

All algorithms are implemented in Java language. The experiments are performed using an Intel Core 2 Quad CPU 2.83 GHz system with 3GB of main memory and running Windows XP operating system.

We use two real world datasets for performance evaluation. The first real world dataset is the MemeTracker data [63]. This MemeTracker dataset contains the quotes, phrases, and hyperlinks of the articles/blogposts that appear over prominent online news sites from August 2008 to April 2009. Each post contains a URL, time stamp, and all of the URLs of the posts it cites. Nodes are mostly news portals or news blogs and the time stamps in the data capture the time that a quote/phase was used in a post. Finally, there are directed hyperlinks among the posts. We use these hyperlinks to trace the flow of information. A site publishes a piece of information and uses hyperlinks to refer to the same or closely related pieces of information published by other sites. An observation is thus a collection of time-stamped hyperlinks among different sites that refer to the same or closely related pieces of information. We record one observation per piece – or closely related pieces – of information. We extract the most active media sites and blogs with the largest number of posts, and generate 46,352 observations.

Another real world dataset is the Twitter dataset [116, 60]. This Twitter dataset consists of 17,214,780 tweets published by 1,746,259 users over a 7 month period from June 1 2009 to December 31 2009. For each tweet the following information is available: user, time and content. We preprocess the tweets by removing tweets that are not in English or have no hashtags. We use hashtags to identify the topic of each tweet and generate 129,043 observations for our experiments.

Table 3.11 shows the characteristics of the two real world datasets used in the experiments including the number of input observations (Cardinality), average observation length (Avg Len), maximum observation length (Max Len) and minimum observation length (Min Len).

Table 3.11: Datasets characteristics

Datasets	Cardinality	Avg Len	Max Len	Min Len
MemeTracker	46,352	13.72	42	3
Twitter	129,043	8.56	38	3

3.5.1 Efficiency Experiments

Efficiency of TIP. In this set of experiments, we evaluate the efficiency of TIP algorithm. For the MemeTracker dataset, we generate the top-10 (i.e. k = 10) maximal influential paths by setting time threshold τ to 1000 minutes and radius of influence α to 1.0. We set time threshold τ to 1000 minutes, as we observe that the time lapse in the MemeTracker dataset tends to be long. We randomly sample the dataset to vary the database size from 10k to 40k. As can be seen from Figure 3.7, TIP algorithm outperforms the Naïve algorithm with early termination playing a greater role in reducing the runtime of TIP.

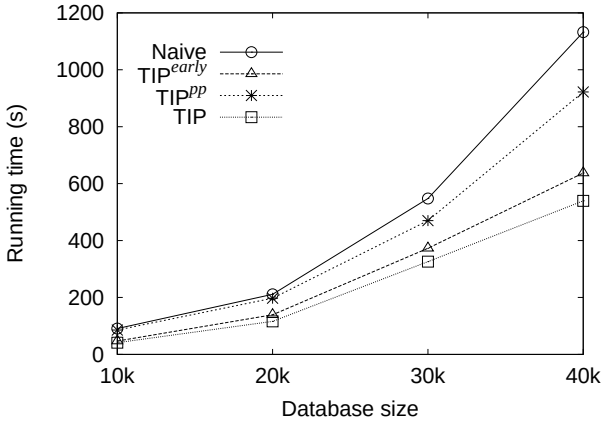


Figure 3.7: Performance of varying database size on MemeTracker dataset

For the Twitter dataset, we set k = 10, τ = 10 minutes and α = 1.0. As the time lapse in the Twitter dataset tends to be short, we set time threshold τ to 10 minutes. We generate the top-10 maximal influential paths by varying database size from 10k to 129k. Figure 3.8 shows the result. We observe that TIP algorithm remains efficient as the database size increases. In particular, the early termination optimization strategy is more effective in reducing the runtime compared to the pseudo projection.

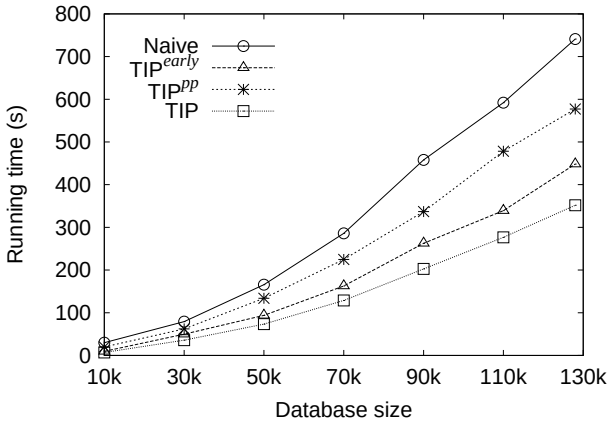


Figure 3.8: Performance of varying database size on Twitter dataset

Note that the runtime on Twitter dataset is less than the runtime on MemeTracker dataset. This is because the average observation length of the Twitter dataset is smaller than that of the MemeTracker dataset. Furthermore, for the Twitter dataset time threshold τ is set to 10 minutes, whereas for the MemeTracker dataset τ is set to 1000 minutes.

Efficiency of IncTIP. We also evaluate the efficiency of IncTIP algorithm. For the MemeTracker dataset, we set the original database size to 25k and vary the size of update database from 5k to 20k. We set the number of maximal influential paths k = 10, time threshold τ = 1000 minutes, and radius of influence α = 1.0. Figure 3.9 shows the result. We observe that as the size of update database increases, the running time for both algorithms increases. However, IncTIP is more efficient than TIP. The reason is that each time when the database updates, TIP has to mine from scratch, but IncTIP only deals with the update part.

For the Twitter dataset, we generate the top-10 maximal influential paths by setting time threshold τ to 10 minutes and radius of influence α to 1.0. We set the original database size to 30k and vary the update database size from 10k to 50k. As shown in Figure 3.10, IncTIP algorithm outperforms TIP algorithm and the performance gap widens as the size of update database increases. This is because IncTIP only deals with the update part, whereas TIP has to mine from scratch for each database update.

We then compare the performance of IncTIP algorithm with an existing incremen-

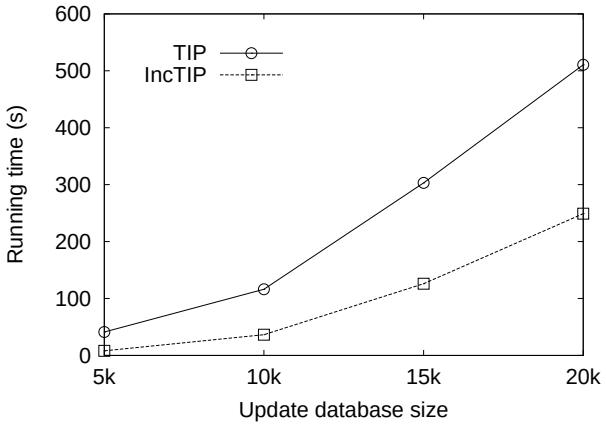


Figure 3.9: Performance of varying update database size on MemeTracker dataset

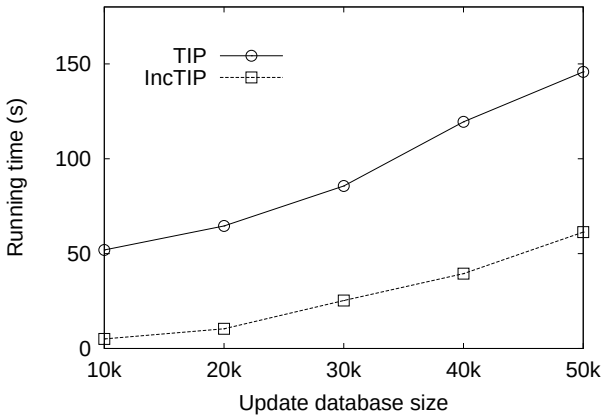


Figure 3.10: Performance of varying update database size on Twitter dataset

tal mining algorithm IncSpan [31]. We evaluate IncTIP and IncSpan by varying update database size on the real world datasets. For both algorithms, we set the parameters such that they will generate the same number of patterns.

Figure 3.11 shows the result on MemeTracker dataset by varying update database size from 5k to 20k. We can see that IncTIP outperforms IncSpan and the performance gap gets larger and larger as the update database size increases. This is because IncTIP utilizes time information to prune off the search space during mining process. Similar trend is observed for the Twitter dataset as shown in Figure 3.12.

Memory Usage. Note that in order to facilitate incremental mining, we keep additional information for each node in the prefix tree. Thus, IncTIP algorithm will incur additional memory cost. In this set of experiments, we compare the memory usage of TIP and

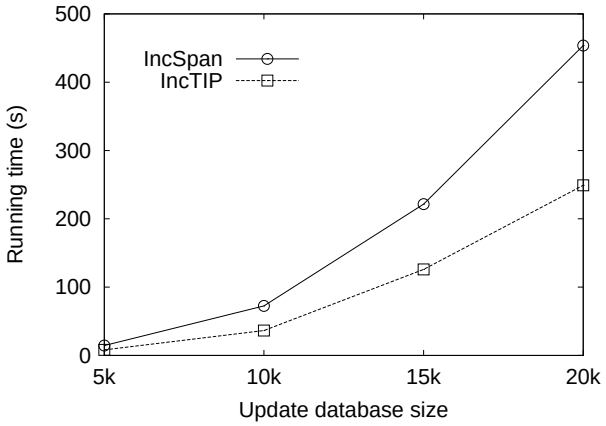


Figure 3.11: Performance of varying update database size on MemeTracker dataset

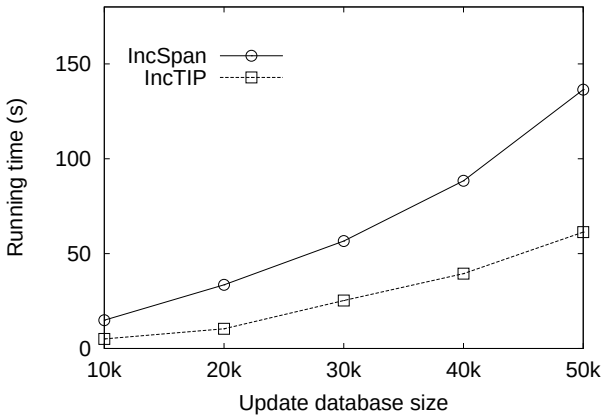


Figure 3.12: Performance of varying update database size on Twitter dataset

IncTIP. Figure 3.13 shows the memory usage of TIP and IncTIP on the MemeTracker dataset. The original database size is 25k and the size of update database varies from 5k to 20k. We set the number of maximal influential paths k = 10, time threshold τ = 1000 minutes, and radius of influence α = 1.0. We can see that as the update database size increases, the memory usage of both algorithms increases. However, IncTIP algorithm incurs more memory usage than TIP, as IncTIP keeps additional information to facilitate incremental mining.

For the Twitter dataset, we set the original database size to 30k and vary the size of update database from 10k to 50k. We set k to 10, τ to 10 minutes and α to 1.0. As can be seen from Figure 3.14, IncTIP incurs more memory usage than TIP for different update database sizes. The reason is that IncTIP has to keep additional information to facilitate

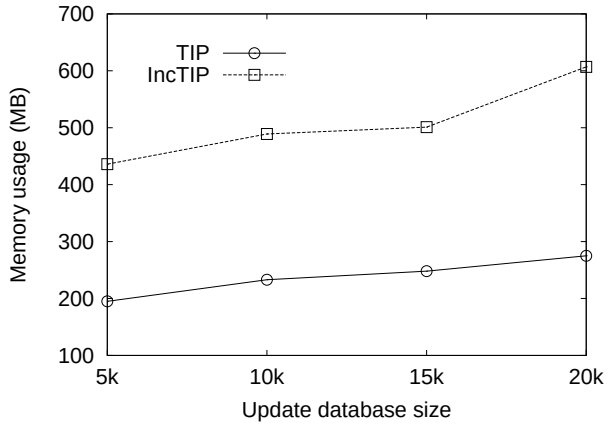


Figure 3.13: Memory usage by varying update database size on MemeTracker dataset

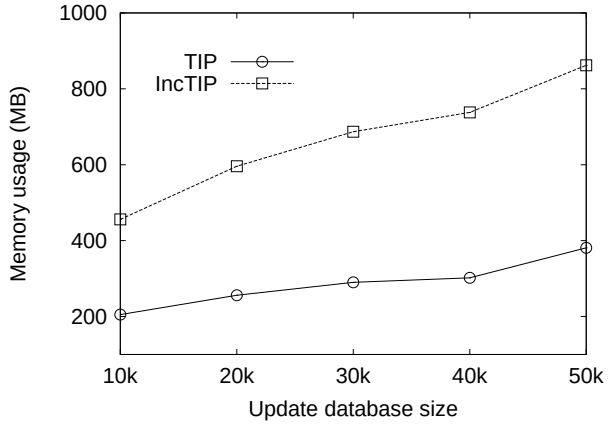


Figure 3.14: Memory usage by varying update database size on Twitter dataset

incremental mining.

3.5.2 Sensitivity Experiments

Effect of k. Next, we investigate the effect of the number of maximal influential paths, k, on the performance of TIP algorithm. For the MemeTracker dataset, we set the database size to 20k, time threshold τ to 1000 minutes and vary k from 5 to 25. Figure 3.15 shows the result. As can be seen, the runtime for both TIP and Naïve algorithm increases as k increases. However, TIP algorithm outperforms the Naïve algorithm and the gap in runtime widens as k increases.

For the Twitter dataset, we set the database size to 50k, time threshold τ to 10 minutes and vary k from 5 to 25. As shown in Figure 3.16, the runtime of TIP algorithm is half that

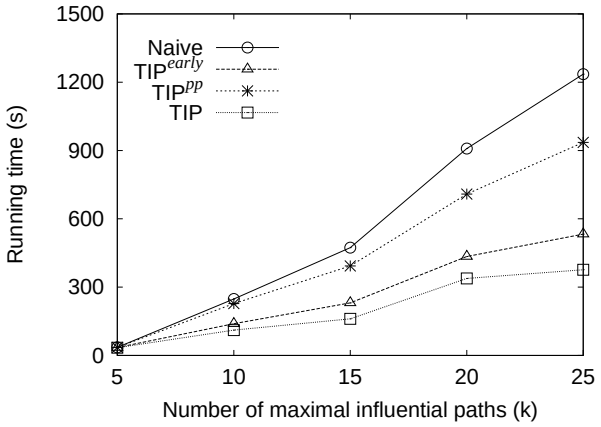


Figure 3.15: Performance of TIP by varying k on MemeTracker dataset

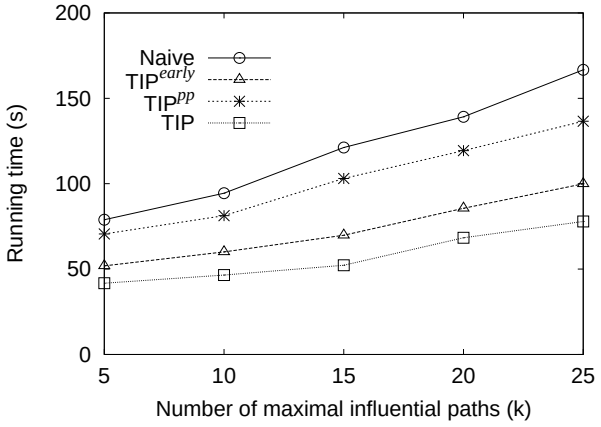


Figure 3.16: Performance of TIP by varying k on Twitter dataset

of the Naïve algorithm demonstrating that TIP remains efficient even when k increases.

We also investigate the effect of the number of maximal influential paths, k, on the performance of IncTIP algorithm. For the MemeTracker dataset, we set the original database size to 25k, update database size to 20k and time threshold τ to 1000 minutes. Figure 3.17 shows the runtime of IncTIP and TIP by varying k from 5 to 25. We can see that the runtime of both algorithms increases as k increases. However, IncTIP algorithm outperforms TIP algorithm by a large margin and the performance gap gets larger and larger as k increases.

For the Twitter dataset, we vary k from 5 to 25 by setting the original database size to 30k, update database size to 50k and time threshold τ to 10 minutes. As shown in Figure 3.18, the runtime of both IncTIP and TIP increases as k increases. However, IncTIP

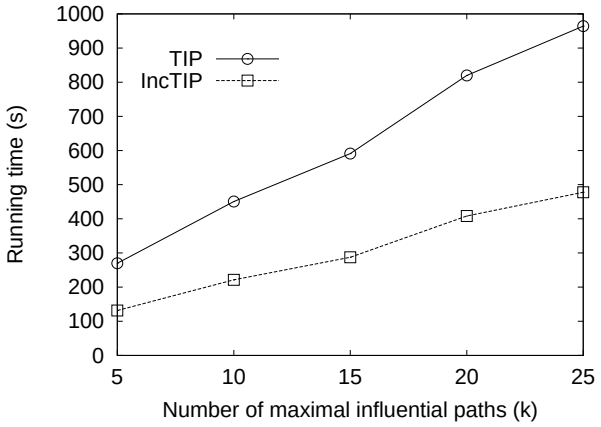


Figure 3.17: Performance of IncTIP by varying k on MemeTracker dataset

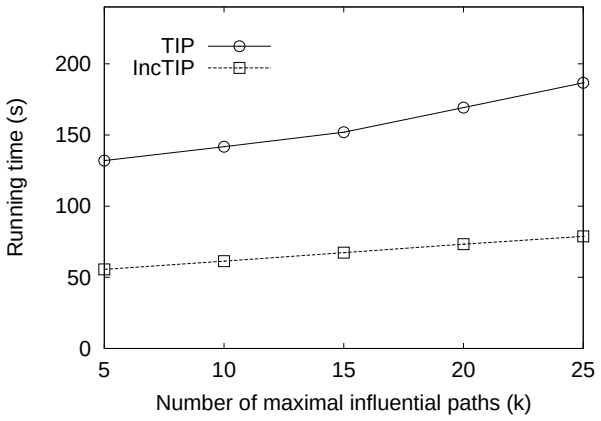


Figure 3.18: Performance of IncTIP by varying k on Twitter dataset

algorithm outperforms TIP algorithm for different values of k.

Effect of τ. Here, we examine the effect of varying the time threshold τ on the performance of TIP algorithm. Note that increasing τ is equivalent to increasing the search space, i.e. the number of potential influential paths. For the MemeTracker dataset, we set the database size to 20k, number of maximal influential paths k to 10 and vary time threshold τ from 1000 to 3000 minutes. Figure 3.19 shows that the runtime for all algorithms increases as τ increases. Similar trend is observed here with the TIP algorithm showing a significant reduction in runtime as compared to the Naïve algorithm.

For the Twitter dataset, we set the database size to 50k, number of maximal influential paths k to 10 and vary time threshold τ from 10 to 50 minutes. As shown in Figure 3.20, the runtime for both TIP and Naïve algorithm increases as τ increases. However, TIP

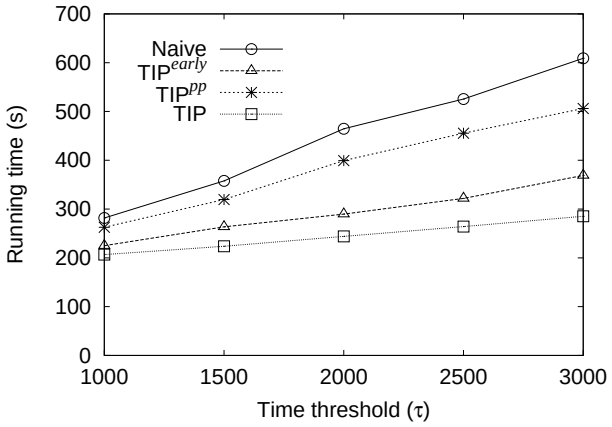


Figure 3.19: Performance of TIP by varying τ on MemeTracker dataset

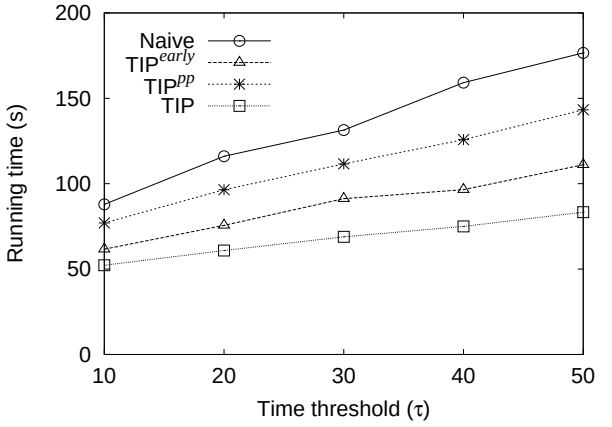


Figure 3.20: Performance of TIP by varying τ on Twitter dataset

algorithm outperforms the Naïve algorithm and the gap in runtime widens as τ increases.

We also examine the effect of varying the time threshold τ on the performance of IncTIP algorithm. For the MemeTracker dataset, we set the original database size to 25k, update database size to 20k and k to 10. Figure 3.21 shows the runtime of IncTIP and TIP by varying τ from 1000 to 3000 minutes. We can see that the runtime of both algorithms increases as τ increases. However, IncTIP algorithm outperforms TIP algorithm and the performance gap widens as τ increases.

For the Twitter dataset, we vary τ from 10 to 50 minutes by setting the original database size to 30k, update database size to 50k and k to 10. As can be seen from Figure 3.22, the runtime of both IncTIP and TIP increases as τ increases. However, IncTIP algorithm is more efficient than TIP algorithm for different values of τ.

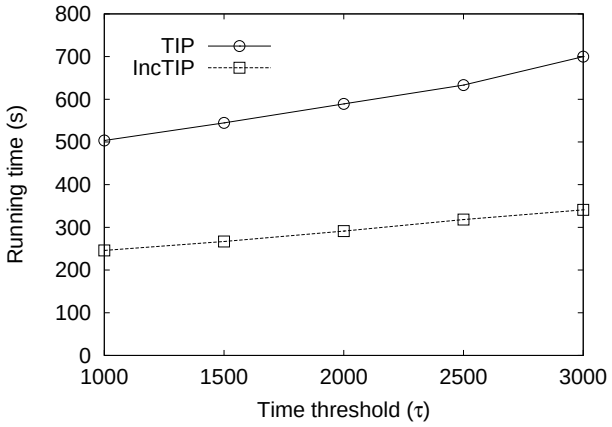


Figure 3.21: Performance of IncTIP by varying τ on MemeTracker dataset

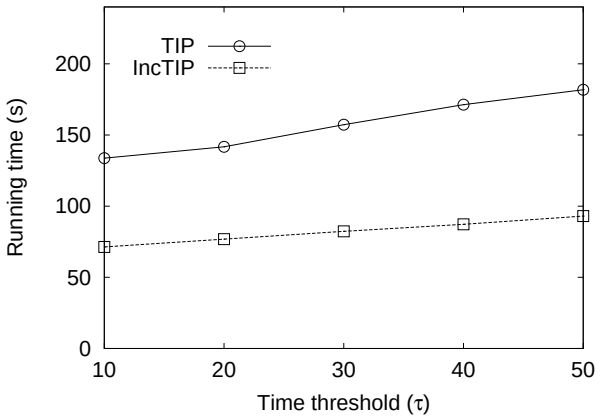


Figure 3.22: Performance of IncTIP by varying τ on Twitter dataset

3.5.3 Effectiveness Experiments

Effectiveness of TIP. In the final set of experiments, we demonstrate the effectiveness of using maximal influential paths for prediction. To do cross validation, we partition the MemeTracker dataset into 4 folds (25% each). We use 75% of the total observations for training and the remaining 25% for testing. We run the TIP algorithm on the training data to generate the top-k maximal influential paths. For each influential path $p = < v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_{n-1} \rightarrow v_n >$ generated, we obtain the corresponding rule

$$r = \{< v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_{n-1} > \Rightarrow < v_n >\}$$

with

$$confidence(r) = \frac{support(< v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_{n-1} \rightarrow v_n >)}{support(< v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_{n-1} >)}.$$

For each rule $< v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_{n-1} > \Rightarrow < v_n >$, we determine the number of observations in the testing data that support $p' = < v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_{n-1} >$. If there is at least one support observation in the testing data, we assign the probability of node v_n being influenced to the confidence of the rule, i.e. $\frac{support(p)}{support(p')}$. If we have more than one rule predicting that node v_n will be influenced, we assign the maximum confidence of the rules as the probability of node v_n being influenced.

The set of predicted nodes are sorted in decreasing order of the probability of getting influenced. We consider a node to be the next influenced node if it is among the top-n nodes. Here top-n nodes are the first n non-duplicate nodes with highest probability of being influenced.

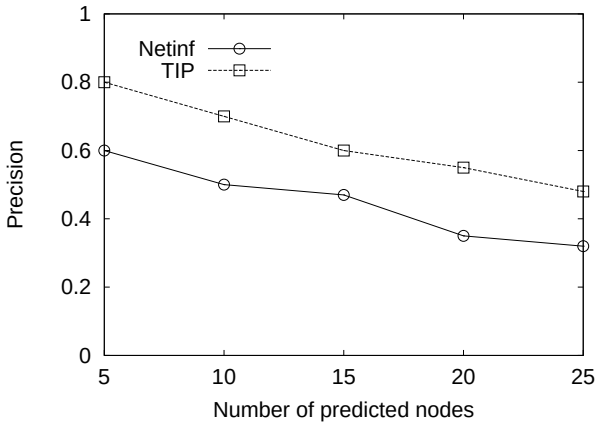
Let X be the set of nodes influenced in test data, and Y be the set of nodes predicted to be influenced in test data, then precision and recall are defined by the following equations:

$$precision = \frac{|X \cap Y|}{|Y|} \tag{3.6}$$

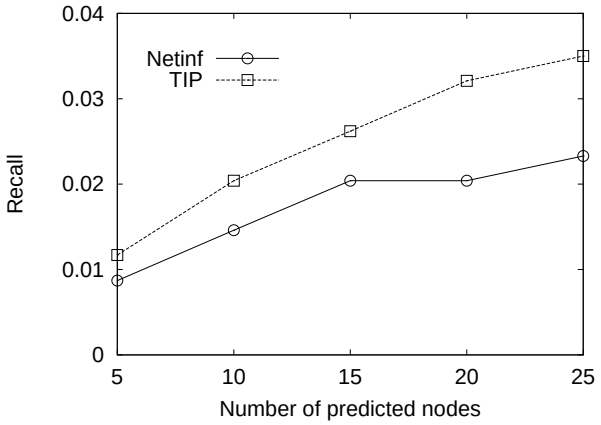
$$recall = \frac{|X \cap Y|}{|X|} \tag{3.7}$$

We compare the prediction accuracy of TIP algorithm with NetInf algorithm [44], which can only infer influential edge between two nodes. Similarly, we run NetInf algorithm on the training data to generate a set of influential edges, say $< i \rightarrow j >$. We assign the probability of node j being influenced as $\frac{support(<i \rightarrow j>)}{support(<i>)}$.

We perform 4-fold cross validation for evaluating the prediction performance of both algorithms. Figure 3.23 shows the precision and recall results by varying the number of predicted nodes, n, from 5 to 25. We observe that TIP algorithm significantly outperforms NetInf algorithm for different values of n. This is because influential paths are more informative than influential edges and hence in predicting which node will be influenced



(a) Precision



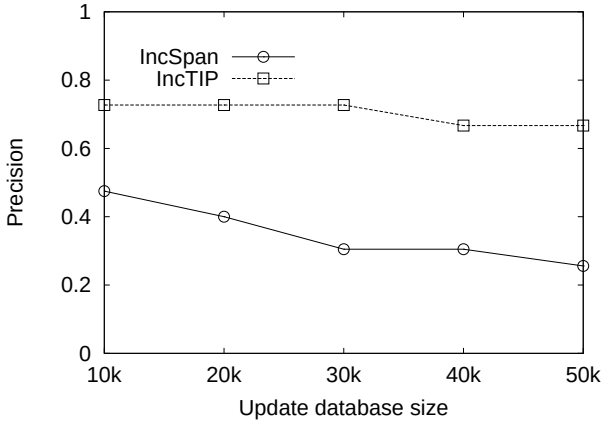
(b) Recall

Figure 3.23: Precision and recall on MemeTracker dataset

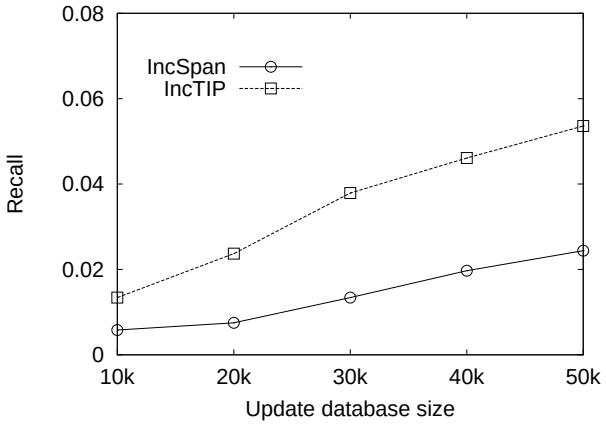
next, the TIP algorithm tends to be more accurate than NetInf algorithm.

Effectiveness of IncTIP. We evaluate the effectiveness of IncTIP algorithm on the Twitter dataset. We partition the dataset into training data and testing data. The size of the training data varies from 10k to 50k. We set time threshold τ to 10 minutes, and radius of influence α to 1.0. We run IncTIP on the training data to generate a set of rules and use the top-10 nodes for prediction. Similarly, we run IncSpan on the training data to generate a set of rules and select the top-10 predicted nodes.

We compare the prediction accuracy of IncTIP with IncSpan [31]. Figure 3.24 shows the precision and recall results by varying the size of update database (training data) from 10k to 50k. We observe that IncTIP outperforms IncSpan in both precision and recall measures. Further, the gap in both precision and recall between IncTIP and IncSpan widens as update database size increases. This demonstrates the effectiveness of IncTIP algorithm.



(a) Precision



(b) Recall

Figure 3.24: Precision and recall on Twitter dataset

3.6 Summary

In this chapter, we have focused on influential path discovery. We develop a method for inferring top-k maximal influential paths, which can truly capture the dynamics of in-

formation diffusion. We propose a generative influence propagation model based on the Independent Cascade Model and Linear Threshold Model, which mathematically models the spread of certain information through a network. We formalize the top-k maximal influential path inference problem and develop an efficient algorithm, called TIP, to infer the top-k maximal influential paths. TIP makes use of the properties of top-k maximal influential paths to dynamically increase the support and prune the projected databases. As databases evolve over time, we extend TIP to allow for incremental mining. The extended algorithm, named IncTIP, leverages on the computation performed in previous stages to maintain the set of top-k maximal influential paths efficiently. We evaluate the proposed algorithms on two real world datasets (MemeTracker and Twitter). The experimental results show that our algorithms are more scalable and more efficient than the base line algorithms. In addition, influential paths can improve the precision of predicting which node will be influenced next.

Chapter 4

Inferring Topic-level Social Influence

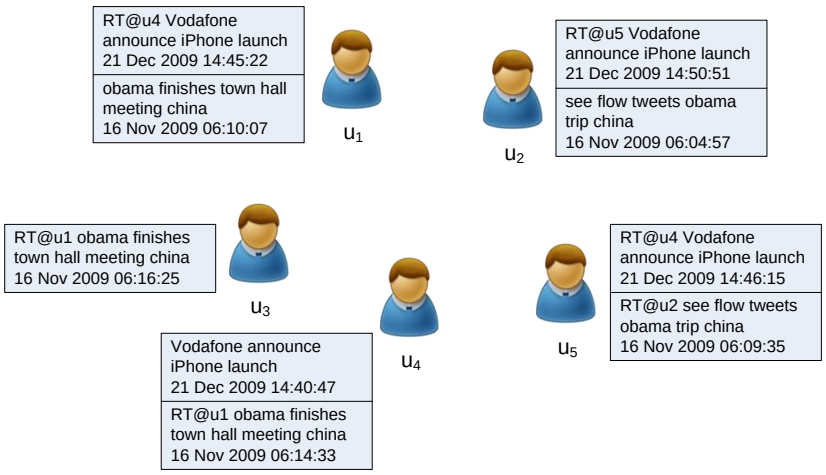
In this chapter, we take into account the temporal factor in social influence to infer the influential strength between users at topic-level. We propose a guided hierarchical LDA approach to automatically identify topics without using any structural information. We then construct the topic-level social influence network incorporating the temporal factor to infer the influential strength among the users for each topic. Experimental results on two real world datasets demonstrate the effectiveness of our methods. Further, we show that the proposed topic-level influence network can improve the precision of user behavior prediction and is useful for influence maximization.

The remaining of this chapter is organized as follows: We start with the motivation of inferring topic-level social influence in Section 4.1. In Section 4.2, we define some terminologies and give an overview of our two-step approach. Guided hierarchical LDA is described in Section 4.3. In Section 4.4, we infer topic-level influence network. We conduct experiments in Section 4.5. Finally, we summarize our work in Section 4.6.

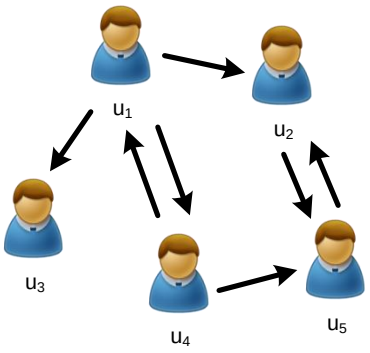
4.1 Motivation

Research on social influence has focused on discovering influential nodes (users, entities) and influence relationships (who influences whom) between nodes in the network [55, 64, 28, 27, 44]. Knowing the influential users and their influence relationships al-

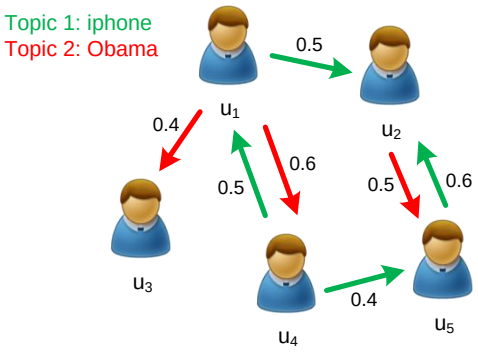
lows a company to target only a small number of influential users, thus leading to more effective online advertising and marketing campaigns. However, most often than not, influential users typically tweet on many topics and their followers generally follow them for different reasons. As a result, they may not be the ideal targets for targeted marketing.



(a) A set of users and the tweets they publish



(b) Influence based on re-tweet information



(c) Topic-level influence network

Figure 4.1: Example topic-level influence analysis

Figure 4.1(a) shows 5 users and the tweets they make at different times. Based on their re-tweet information, we can construct the influence among the users as shown in Figure 4.1(b). We note that user u_1 is the most influential person as his/her tweets are re-tweeted by 3 other users. Yet, when we analyze the contents of the tweets, we discover that user u_1 only influences u_2 on the topic "iphone", whereas for the same topic "iphone", user u_4 influences users u_1 and u_5. Hence, if we wish to conduct a marketing campaign on "iphone", the most influential person, i.e. u_1, may not be the ideal target. Instead, we

should target u_4.

Further, temporal factor also plays an important role in differentiating the degree of influence among different users. For example, users u_1, u_3 and u_4 are connected to each other in Figure 4.1(b) and they have tweeted about "Obama" at time stamps 06:10, 06:16 and 06:14 respectively. Without utilizing the time information, the degree of influence from u_1 to u_3 and to u_4 is the same. However, in real life, we observe that the influence is the greatest when the time lapse is the shortest [44]. In other words, the influence from u_1 to u_4 should be greater than that from u_1 to u_3.

To address this, the works in [102, 69, 109] have looked into capturing the micro-level mechanisms of influence, e.g. the influence relationship between two users on a specific topic. However, they require the connection among users to be explicitly modeled. In other words, suppose we wish to analyze the influence relationships among users on Twitter, these works can only report the topic-specific influence relationships among the followers where the follow relationships are explicitly modeled in Twitter. While this is useful for applications that concern only the explicitly modeled relationships, many applications need to go beyond the connected users.

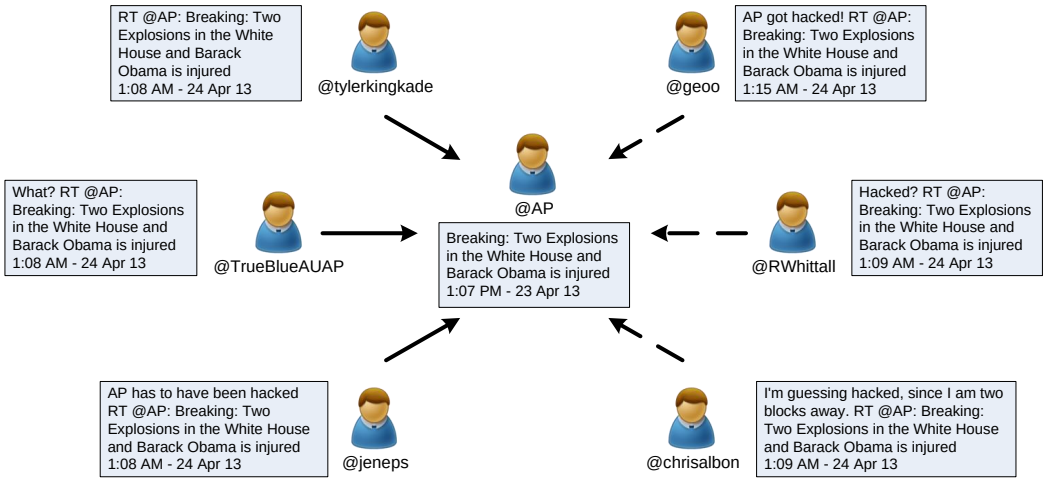


Figure 4.2: "Two Explosions in the White House and Barack Obama is injured" rumor

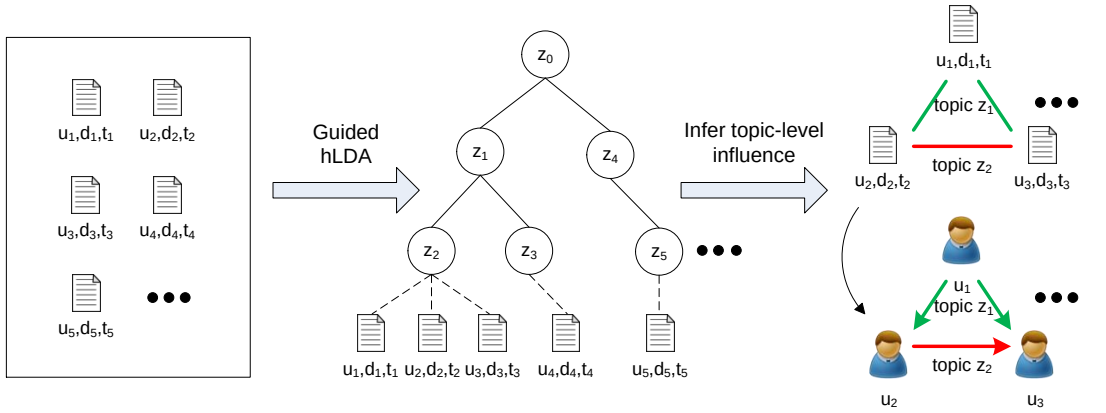
Figure 4.2 shows the rumor "Two Explosions in the White House and Barack Obama is injured". The Twitter account of the Associated Press (@AP) was hacked and a tweet that reported a fake White House explosion caused the Dow Jones Index to drop more than

140 points within minutes. This tweet was retweeted by almost 1,500 Twitter users within a short span of a few minutes and many of these users are not explicitly connected via the follow relationship in Twitter. Clearly, there is a need to capture topic-level influence among users that are not explicitly connected.

There are two challenges that we need to address. First, we need to design an effective algorithm that can extract meaningful topics from short texts such as tweets. Second, without the benefit of an explicit modeling of users' connection with each other, we need to infer influence relationships among users through the observation of their activities on social networks.

4.2 Preliminaries

A *topic-level influence network* is denoted as $G = (U, E)$, where U is the set of users and E is the set of labeled directed edges between users. An edge $e \in E$ from node u to node v with label (z, w) denotes that user u influences user v on topic z with an influential strength of w, $w \geq 0$. Figure 4.3 gives an overview of our two-step approach to discover topic-level user influence network.



(a) Set of time-stamped documents (b) Guided hLDA tree (c) Topic-level influence network

Figure 4.3: Overview of proposed solution

Given a collection of time-stamped documents D where a tuple $< u, d, t > \in D$ indicates that user u has published document d at time t, and a set of users who published these documents, the first step is to cluster the set of documents into various groups based

on their topics. Unfortunately, in most cases, the topics of the documents are not known. A popular approach is to apply an unsupervised probabilistic generative model, LDA, proposed in [17], to generate the set of topics for the documents. In this model, a document is defined as a bag of words from a fixed vocabulary $V = \{w_j\}$, $1 \leq j \leq L$. A latent *topic* z is characterized by a multinomial distribution over the words $w \in V$ such that $\sum_{w \in V} p(w|z) = 1$. A document is then represented as a random mixture ψ over latent topics where $\sum_z \psi_z = 1$.

As a standard method in topic modeling, LDA has been extended in a variety of ways [89, 85, 84, 23]. The works in [85, 84] introduce labeled LDA that uses the hashtags in microblogs as labels to guide the generative process of LDA so that the learned latent topics can be more meaningful. While labeled LDA tends to give more interpretable topics than LDA, it is dependent on the availability of hashtags. However, as shown in [7], only about 10% of tweets in Twitter contain hashtags. Moreover, on Twitter hashtags may belong to more than one topics and thus be misleading in guiding topic models. In addition, both LDA and labeled LDA require the number of topics to be pre-determined. This may not be practical for social networks where the number of topics discussed varies greatly. Hierarchical LDA model (hLDA) [16, 15] generates topic hierarchies from an infinite number of topics. Unlike LDA, it does not restrict the given number of topics and allows arbitrary breadth and depth of topic hierarchies. However, both LDA and hLDA are unsupervised latent topic models. They should work for documents that are long in length and dense in word distribution. When applying to short texts such as tweets and microblogs, the results are poor and lack meaningful interpretations [108].

Here, we propose a guided topic modeling approach based on the hierarchical LDA model [16, 15] to overcome the two limitations. The key idea is to utilize additional knowledge in the form of known popular topics to bias the path selection in the hierarchical LDA topic generation such that documents that belong to the same path are more similar than documents of another path. Since hierarchical LDA based topic generation allows infinite number of topics, we effectively remove the need to pre-determine the

number of topics.

Once the documents are clustered, the second step is to compute the KL-divergence based similarity between pairs of documents in each cluster. Then we utilize user and temporal information of each document to obtain the influential strength among the users for each topic and construct the topic-level user influence network.

4.3 Guided Hierarchical LDA

In this section, we briefly review the original hierarchical LDA model [16, 15] and then describe our proposed guided hierarchical LDA topic model. In the original hierarchical LDA model, a document is generated by choosing a path from the root to a leaf, and as it moves along the path, it repeatedly samples topics along that path, and then samples the words from the selected topics. The path selection is based on the nested Chinese Restaurant Process (nCRP) which is a stochastic process that assigns probability distribution to an infinitely branched tree. In nCRP, the first customer sits at the first table, and the nth subsequent customer sits at a table drawn from the following distribution:

$$\begin{aligned} p(occupied\ table|previous\ customers) &= \frac{n_i}{\gamma + n - 1} \\ p(next\ unoccupied\ table|previous\ customers) &= \frac{\gamma}{\gamma + n - 1} \end{aligned} \tag{4.1}$$

where n_i is the number of customers currently at table i, and γ is a real-valued parameter which controls the probability of choosing new tables.

Careful observation of this distribution shows that the probability of choosing a table depends on the number of customers already assigned to the table at that level. Thus, tables with more customers will have a higher probability to be selected. However, this does not consider the similarity of the customers at the table. For short documents such as tweets and microblogs, the length of each path is short and hence it is vital to ensure the similar documents are assigned to the same table as early on the path as possible.

Fortunately, in real life social networks, we often have some rough ideas what are the

hot topics being discussed and the commonly used words associated with these topics. For example, one hot topic in the recent months is "gun control" and the commonly associated words may include "victims", "killed". Another hot topic is "bird flu" with associated words such as "H5N1" and "H7N9". We assume that for each hot topic the associated representative words do not change much over a period of time. Taking advantage of such knowledge, we propose to guide the topic generation of hierarchical LDA model by biasing the path selection at the beginning of each path by favoring the table (the preferred table) whose customers are most similar to the incoming customer. We compute the cosine similarity between the incoming customer (tweet) and the hot topics to decide the preferred table. This is achieved by changing the probability distribution of path selection at level 2 as follows:

$$\begin{aligned} p(preferred\ table|previous\ customers) &= \frac{n_i + \delta}{\gamma + n + \delta} \\ p(next\ occupied\ table|previous\ customers) &= \frac{n_i}{\gamma + n + \delta} \\ p(next\ unoccupied\ table|previous\ customers) &= \frac{\gamma}{\gamma + n + \delta} \end{aligned} \tag{4.2}$$

where δ adds an increment to the table where the most similar customers are seated.

More specifically, in our guided hierarchical LDA model, a document is drawn by first choosing an L-level path and then drawing the words from the L topics which are associated with the nodes along that path. The generative process is as follows:

(1) For each table k in the infinite tree,

  (a) Draw a topic $\beta_k \sim$ Dirichlet(η).

(2) For each document, $d \in \{1, 2, \ldots, D\}$,

  (a) Let c_1 be the root node.

  (b) Let hot be the most similar hot topics to d.

  (c) Mark the table corresponding to the hot as "preferred table".

    i. Draw a table from c_1 using Equation (4.2).

ii. Set c_2 to be the restaurant referred to by that table.

(d) For each level $l \in \{3, \ldots, L\}$,

i. Draw a table from c_{l-1} using Equation (4.1).

ii. Set c_l to be the restaurant referred to by that table.

(e) Draw a distribution over levels in the tree, $\psi_d \mid \{m, \pi\} \sim \text{GEM}(m, \pi)$.

(f) For each word,

i. Choose level $z_{d,n} \mid \psi_d \sim \text{Discrete}(\psi_d)$.

ii. Choose word $w_{d,n} \mid \{z_{d,n}, c_d, \beta\} \sim \text{Discrete}(\beta_{c_d}[z_{d,n}])$, which is parameterized by the topic in position $z_{d,n}$ on the path c_d.

where $z_{d,n}$ denotes the topic assignments of the nth word in the dth document over L topics, $w_{d,n}$ denotes the nth word in the dth document, and m, π, γ and η are the same hyperparameters used in hierarchical LDA [16, 15].

Figure 4.4 shows the graphical model representation of guided hLDA. The node labeled T refers to a collection of an infinite number of L-level paths drawn from the modified nCRP. Given an observed T, c_d represents the path for document d in the infinite path collection. The node labeled Λ represents the set of hot topics. The dependency of T on both Λ and δ is indicated by the directed edges from Λ and δ to T.

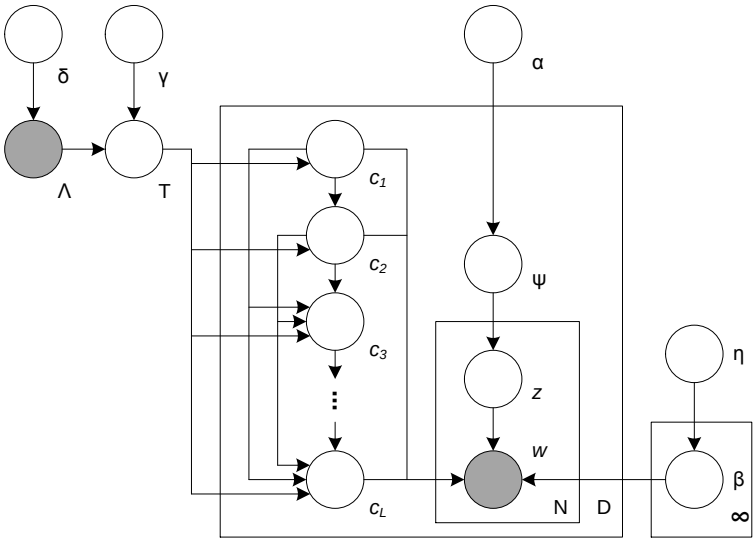


Figure 4.4: Graphical model of guided hLDA

Having defined guided hierarchical LDA model, the next step is to learn the model from data. We adopt the Gibbs sampling approach and iteratively sample each variable conditioned on the rest. First, we sample a path c_d for each document conditioned on the path assignment of the rest documents in the corpus and the observed words:

$$\begin{aligned} & p(c_d|w, c_{-d}, z, \eta, \gamma, \delta) \\ \propto\ & p(c_d|c_{-d}, \gamma, \delta)p(w_d|c, w_{-d}, z, \eta) \end{aligned} \tag{4.3}$$

where c_{-d} and w_{-d} denote the vectors of path allocation and observed words leaving out c_d and w_d respectively. $p(w_d|c, w_{-d}, z, \eta)$ is the probability of the data given a particular choice of path and $p(c_d|c_{-d}, \gamma, \delta)$ is the prior on paths implied by the modified nested Chinese Restaurant Process.

Given the path assignment, we sample the level allocation variable $z_{d,n}$ for word n in document d conditioned on all the other variables:

$$\begin{aligned} & p(z_{d,n}|z_{-(d,n)}, c, w, m, \pi, \eta) \\ \propto\ & p(z_{d,n}|z_{d,-n}, m, \pi)p(w_{d,n}|z, c, w_{-(d,n)}, \eta) \end{aligned} \tag{4.4}$$

where $z_{-(d,n)}$ and $w_{-(d,n)}$ denote vectors of level allocation and observed words leaving out $z_{d,n}$ and $w_{d,n}$ respectively. The first term in Equation 4.4 is a distribution over levels and the second term is the probability of a given word based on the topic assignment.

Figure 4.5 shows the 3-level guided hLDA tree obtained for 6 sample tweets. We observe that for the guided hLDA tree, the documents sharing the same path are more often than not on the same topic compared to documents on other paths. With this, we can identify the topics for each document. This leads us to the next step in the discovering of topic-level user influence network.

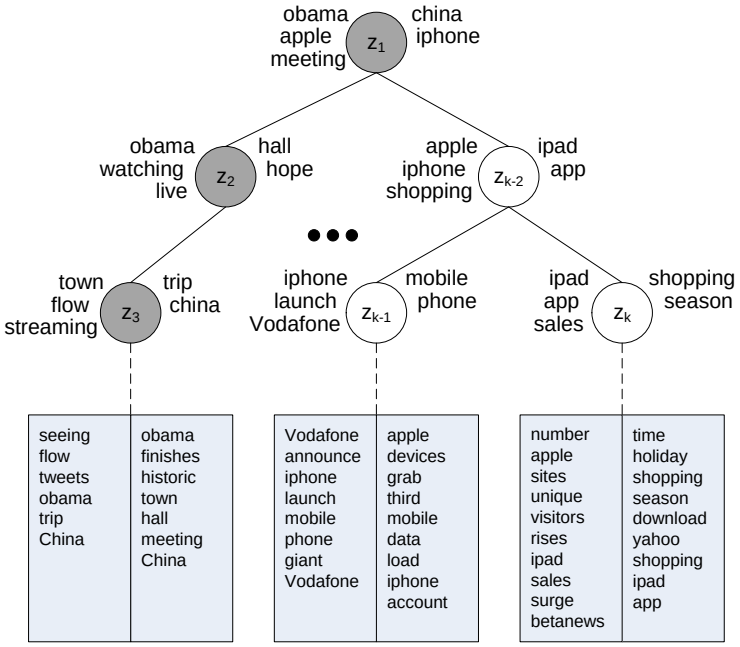


Figure 4.5: Example 3-level guided hLDA tree. Each tweet is assigned a path starting from the root of the tree. Each node is a topic which is a distribution over words and words with highest probability at each topic are shown.

4.4 Topic-level Influence Network

Having organized the documents into topic-specific groups, our next task is to determine the influential strength among the users on each topic. From the proposed guided hLDA model, we find the topic-specific documents by following each path in the model. Let d_u be the document published by user u and d_v be the document published by user v. Suppose that d_u and d_v share the same path that corresponds to topic z.

We say that user u influences user v on topic z if the time associated with d_v is greater than d_u. Furthermore, we realize that the degree of influence is greater when the time lapse between documents is less. We model this effect using a time decay function $g(d_u, d_v)$. Let t_u and t_v be the times at which users u and v post documents d_u and d_v respectively. Then, we have

$$g(d_u, d_v) = \begin{cases} e^{-\frac{\Delta}{\alpha}}, & \text{if } t_u < t_v \\ 0, & \text{otherwise} \end{cases} \tag{4.5}$$

where $\Delta = t_v - t_u$ and $\alpha > 0$.

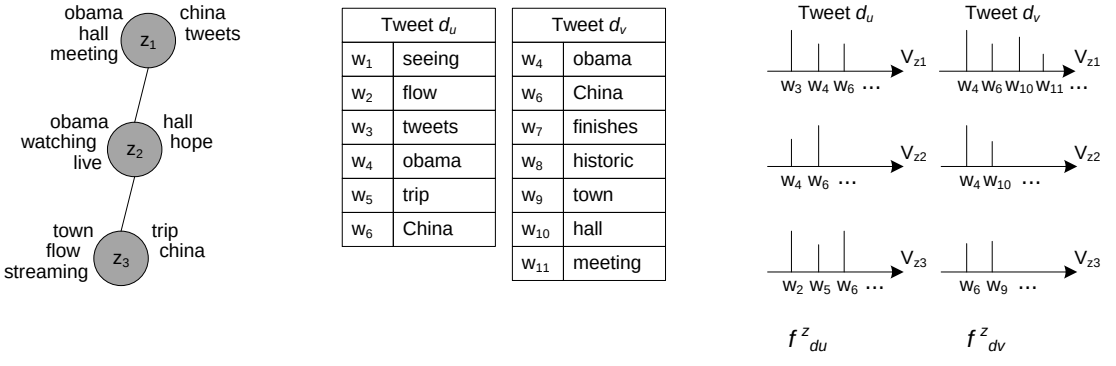
The parameter α controls the time window to compute $g(d_u, d_v)$. Note that for a fixed α, $e^{-\frac{\Delta}{\alpha}} \to 1$ when $\Delta \to 0^+$ and $e^{-\frac{\Delta}{\alpha}} \to 0$ when $\Delta \to +\infty$. This implies that if user v posts a document just after u then u may have an influence on v. On the other hand, if v posts a document after a long elapse time, then u has little influence on v.

Another factor determining the strength of influence between user u and user v on topic z is the degree of similarity among the documents published by u and v on topic z. Let D_u and D_v be the sets of documents published on topic z by users u and v respectively.

For each pair of documents (d_u, d_v) where $d_u \in D_u$ and $d_v \in D_v$, we obtain the normalized topic-word distributions of d_u and d_v on topic z from guided hLDA model, denoted as $f^z_{d_u}$ and $f^z_{d_v}$ respectively (see Figure 4.6). The similarity of these two documents on topic z is evaluated based on the commonly used measure $S(f^z_{d_u}, f^z_{d_v})$ [73]:

$$\begin{aligned} sim(d_u, d_v) &= 10^{-S(f^z_{d_u}, f^z_{d_v})} \\ &= 10^{-[KL(f^z_{d_u}||\frac{f^z_{d_u}+f^z_{d_v}}{2})+KL(f^z_{d_v}||\frac{f^z_{d_u}+f^z_{d_v}}{2})]} \end{aligned}$$

where $KL(P||Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$ defines the divergence from distribution Q to P. We use $S(f^z_{d_u}, f^z_{d_v})$ instead of commonly used KL divergence to measure the similarity between probability distributions. Because $S(f^z_{d_u}, f^z_{d_v})$ is symmetric and there is no problem with infinite values since $\frac{f^z_{d_u}+f^z_{d_v}}{2} \neq 0$, if either $f^z_{d_u} \neq 0$ or $f^z_{d_v} \neq 0$.



(a) Topic hierarchy (b) Words in tweets (c) Topic-word distribution

Figure 4.6: (a) Topic hierarchy for tweet d_u and d_v. (b) Words in tweet d_u and d_v. (c) Topic-word distribution for tweet d_u and d_v at each level. Distribution of words in tweet d_u and d_v at each topic w.r.t all the words assigned to that topic.

With this, we define the influential strength between u and v on topic z as follows.

$$strength(u, v) = max_{d_u \in D_u, d_v \in D_v}[g(d_u, d_v) * sim(d_u, d_v)] \tag{4.6}$$

Using max function reflects the scenario whereby a user may publish many documents on a topic. As long as one of his published document has large overlapped with another user, we may conclude that this user has influenced the other user.

Algorithm 7 TIND(T, τ, σ)

Require: guided hLDA tree T, time threshold τ, and similarity threshold σ
Ensure: topic-level influence network G

```
Initialize G = ∅
for each path p in T do
    let D be the set of documents associated with path p
    for each pair of documents in D do
        let d_u be the document published by user u at time t_u
        let d_v be the document published by user v at time t_v
        if |t_u − t_v| ≤ τ then
            compute sim(d_u, d_v) for each topic z along p
            if sim(d_u,d_v) ≥ σ then
                compute strength(u,v)
                if edge between u and v does not exist then
                    if t_u < t_v then
                        G = G ∪ (u,v) with label (z,strength(u,v))
                    else
                        G = G ∪ (v,u) with label (z,strength(u,v))
                    end if
                end if
                if strength(u,v) > max_strength_uv then
                    max_strength_uv = strength(u,v)
                    update the label for edge (u,v) to (z,max_strength_uv)
                end if
            end if
        end if
    end for
end for
return G
```

Algorithm 7 shows the details of our TIND algorithm. The input is a guided hLDA tree T, time threshold τ, and similarity threshold σ. The output is topic-level influence network G. For each path in the tree T, we obtain the set of documents D associated with

the path (Line 3). For each pair of documents in D, we check if their time difference is within the threshold τ (Line 7). If yes, we calculate their similarity for each topic along the path (Line 8). If the similarity for a topic exceeds the threshold σ, we add an edge (u, v) or (v, u) to G with weight w denoting the maximum influential strength between u and v on topic z (Lines 9-22). Finally, in Line 26, we return the constructed topic-level influence network.

4.5 Experimental Evaluation

In this section, we present the results of experiments conducted to evaluate our proposed method. We implemented the proposed algorithm in C#. The experiments are carried out on an Intel Core 2 Quad CPU 2.83 GHz system with 3GB RAM running Windows.

We use two real world datasets in our experiments. The first is the Twitter dataset [116, 60], which covers a 7 month period from June 1 2009 to December 31 2009. To make our experiments manageable, we use a subset of this Twitter dataset, which consists of 64,451 tweets published by 880 users. Each tweet has the following information: user, time and content. We preprocess the tweets by stemming and removing stopwords. The tweets are then manually categorized into 6 hot topics and each topic is described by top-5 representative words as shown in Table 4.1.

Table 4.1: Characteristics of Twitter data

Topic	Top-5 representative words	# tweets	# ground truth
freeiran	iran, khamenei, tehran, regime, islamic	10,469	992
litchat	litchat, good, think, literature, books	13,511	940
lovestories	karma, forgive, love, lovestories, get	12,502	406
ObamaCN	obama, china, watch, town, hall	1,706	154
supernatural	supernatural, de, dean, que, assistir	13,504	550
Yahoo	yahoo, search, content, site, fav	12,759	326

We generate the ground truth as follows. A user u is said to be influenced by v on topic z if there is a "follow" relationship from u to v and both u and v have published tweets on topic z with the tweets published by v on z being earlier than that by u. The

last column of Table 4.1 gives the number of influence relationships among the users for each topic.

For our second dataset, we extract from the MemeTracker dataset [63] the quotes, phrases, and hyperlinks of articles/blogposts that appear in prominent online news sites from August 2008 to April 2009. Each post contains a URL, time stamp, and all of the URLs of the posts it cites. Nodes are mostly news portals or news blogs and the time stamps in the data capture the time that a quote/phase was used in a post. There are also directed hyperlinks among the posts. A site publishes a piece of information and uses hyperlinks to refer to the same or closely related pieces of information published by other sites.

We use the hyperlink information to obtain the ground truth for this dataset. A site u is influenced by another site v on topic z if there exists a hyperlink from u to v on topic z. Table 4.2 shows the characteristics of this MemeTracker dataset. The default values for the time threshold τ and similarity threshold σ are 20 hours and 0.5 respectively.

Table 4.2: Characteristics of MemeTracker data

Top-5 topics	Top-5 representative words	# documents	# ground truth
election	obama, mccain, campaign, vote, political	14,846	2,228
social media	blog, social, media, twitter, post	32,962	5,453
Iraq war	government, military, iraq, security, troop	15,379	1,080
finance	financial, market, credit, money, banks	10,293	2,033
apple	apple, iphone, store, macbook, ipod	11,668	2,059

4.5.1 Effectiveness Experiments

We carried out two sets of experiments to evaluate the effectiveness of our approach. In the first set of experiments, we evaluate the effectiveness of the guided hierarchical LDA model for grouping the documents into topic-specific clusters. The second set of experiments compare our TIND algorithm with the TAP method [102] which requires the network structure to be known for inferring topic-level influence relationships.

Guided hLDA vs. Clustering. We first evaluate the effectiveness of guided hierarchical LDA model for grouping documents into topic-specific clusters. We compare the guided

hierarchical LDA model with the original hierarchical LDA model and a clustering based method. The clustering based method compares each tweet with the 6 known hot topics using cosine similarity and groups the tweet under the most similar topic.

For each topic cluster, we determine the influence relationships among the users whose tweets are in the cluster. Let E_{truth} be the set of influence relationships in the ground truth for a topic, and E_θ be the set of influence relationships obtained at various cut-off thresholds, θ. Then the precision and recall of the models are defined as follows:

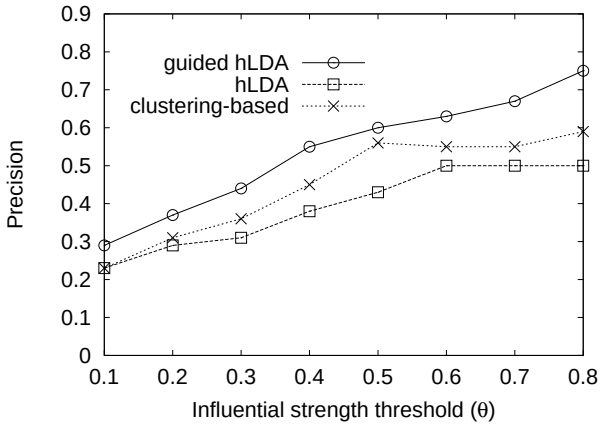
$$precision = \frac{|E_{truth} \cap E_\theta|}{|E_\theta|} \tag{4.7}$$

$$recall = \frac{|E_{truth} \cap E_\theta|}{|E_{truth}|} \tag{4.8}$$

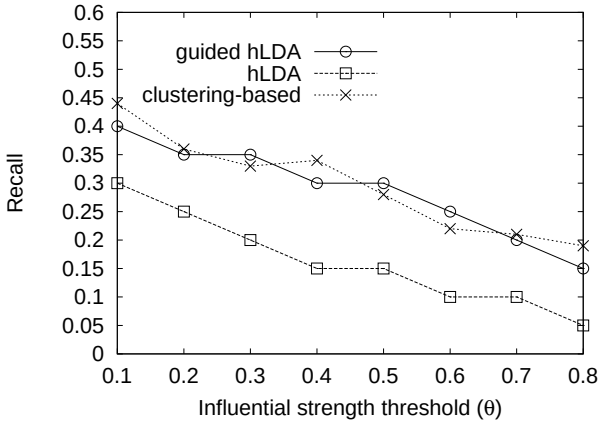
Figure 4.7 shows the average precision and recall on Twitter data for all the 6 topics in the Twitter dataset as we vary θ from 0.1 to 0.8. We observe that the precision of guided hLDA outperforms that of the original hLDA and the clustering based method. Further, the gaps in precision widen as θ increases. The recall for all three models decreases as θ increases. This is because all the models predict only the influence relationships with influential strength greater than θ. As a result, the number of influence relationships decreases, leading to lower recall. Guided hLDA and clustering based method outperform hLDA in both precision and recall measures, because both methods utilize the hot topics to do clustering, while hLDA does not utilize any additional information.

Figure 4.8 shows the average precision and recall on MemeTracker dataset as we vary θ from 0.1 to 0.8. Once again, we observe that guided hLDA outperforms both clustering based method and hLDA especially when θ is large.

Figure 4.9 shows the hierarchical topic tree generated by guided hLDA and hLDA as well as example tweets assigned to each path. We observe that hLDA may assign tweets with different topics into the same branch, while guided hLDA can correctly assign tweets into the appropriate branch based on their topics.



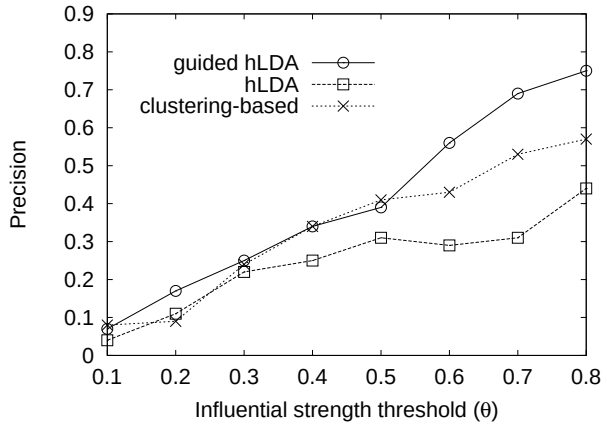
(a) Precision



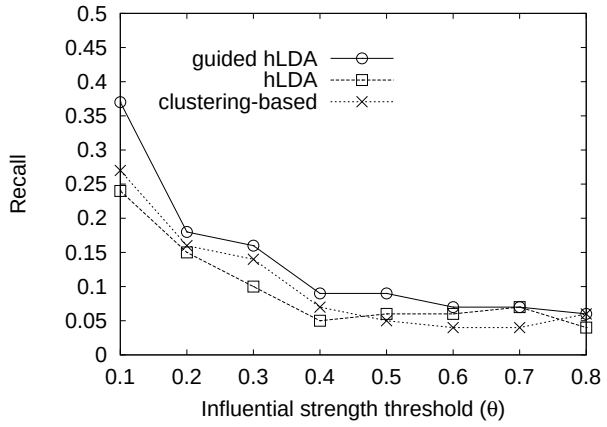
(b) Recall

Figure 4.7: Guided hLDA vs. clustering for varying θ on Twitter data

We also examine the effect of varying the time threshold τ on the precision and recall. Figure 4.10 shows the average precision and recall of all 6 topics in the Twitter dataset when we vary τ from 10 to 50 minutes. We observe that as τ increases, the average precision for guided hLDA, clustering based method and hLDA do not change much. However, the recall for all models increase. This is because as τ increases, the number of influence relationships obtained from all models also increase, leading to better recall. Guided hLDA performs better than clustering based method and hLDA in both precision and recall measures for different values of τ. Similar trend is observed for the MemeTracker dataset as shown in Figure 4.11.



(a) Precision

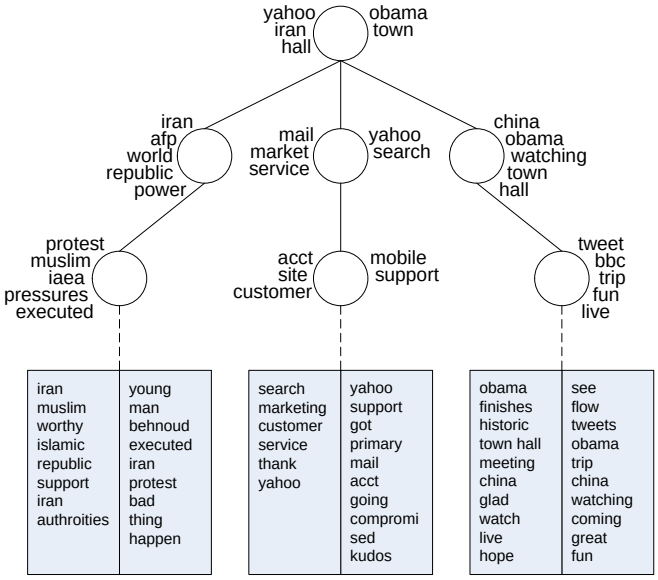


(b) Recall

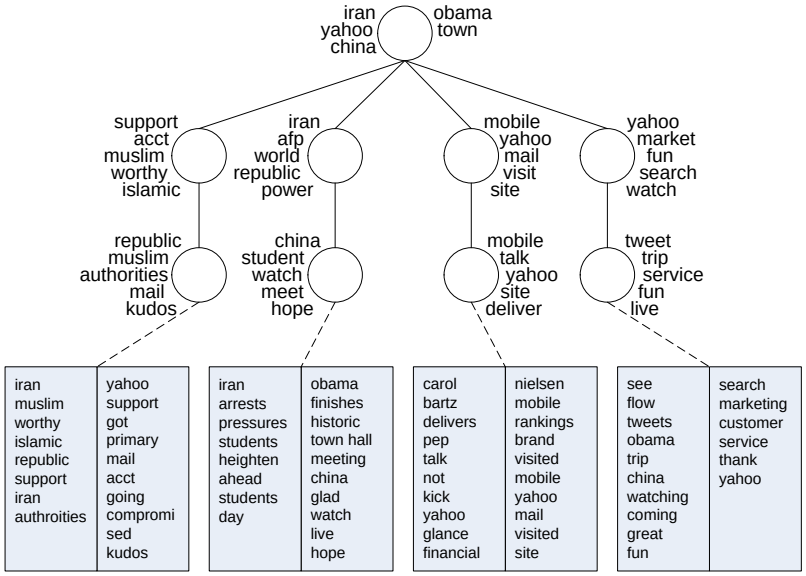
Figure 4.8: Guided hLDA vs. clustering for varying θ on MemeTracker data

TIND vs. TAP. Next, we compare the performance of our TIND algorithm with the existing topic-level influence method TAP [102]. TAP assumes the documents are already grouped into topics. Based on the groupings, it then utilizes the explicit modeled connections among users to derive the influence relationships for the topic.

We first apply the guided hierarchical LDA to obtain the topic-specific clusters. For each topic cluster, we generate topic-level influence relationships using both TIND and TAP. Figures 4.12 and 4.13 show the precision and recall of both methods on the 6 topics in the Twitter dataset as we vary θ from 0.1 to 0.8. We observe that in all the topics, TIND has higher or comparable precision than TAP. Overall, the recall for TIND is also higher than TAP. For 3 of the topics "litchat", "lovestories" and "Obama", the gap between the

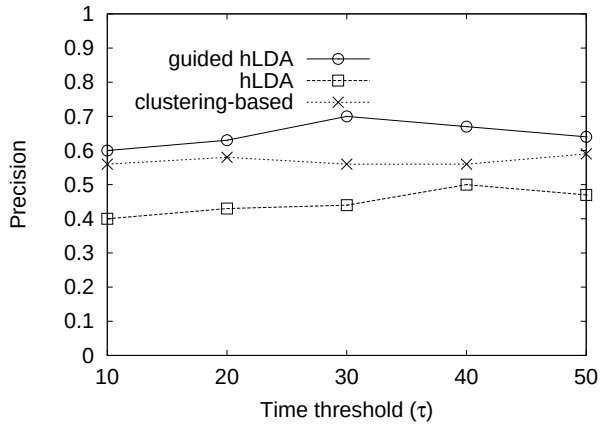


(a) Guided hierarchical LDA

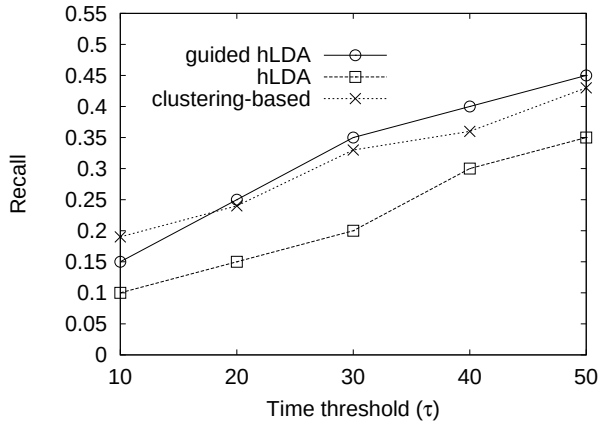


(b) Hierarchical LDA

Figure 4.9: Guided hierarchical LDA vs. hierarchical LDA. (a) Topic hierarchical tree generated by guided hierarchical LDA as well as example tweets assigned to each path. (b) Topic hierarchical tree generated by hierarchical LDA as well as example tweets assigned to each path. Each node is a topic which is a distribution over words. And the top-5 most probable words at each topic are shown.



(a) Precision

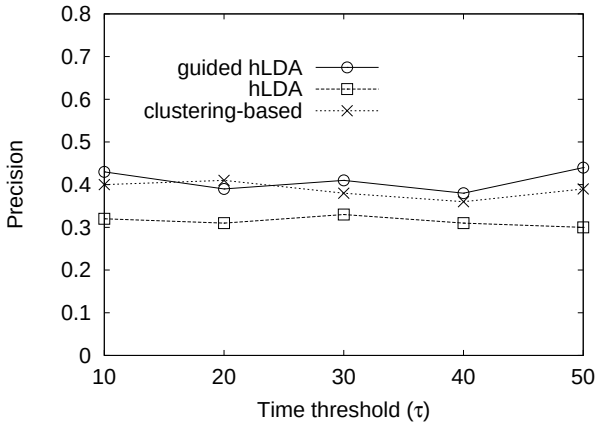


(b) Recall

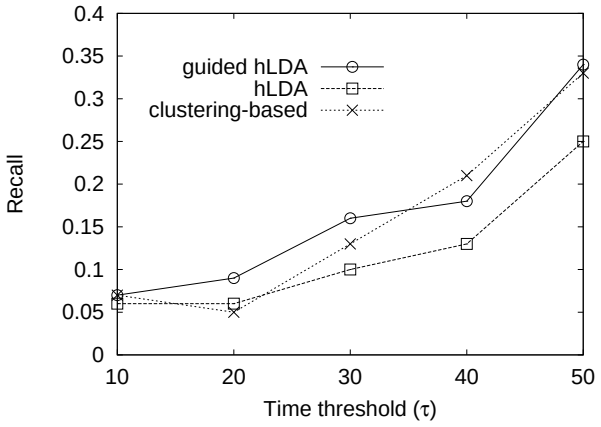
Figure 4.10: Guided hLDA vs. clustering for varying τ on Twitter data

recall of TIND and TAP narrows when θ is more than 0.4. This is because TIND computes influential strength by taking into account the time factor, hence it is able to infer more accurately the influence relationships at a given influential strength threshold.

Figures 4.14 and 4.15 show the precision and recall of both methods on 6 topics in the MemeTracker dataset as we vary θ from 0.1 to 0.8. We observe that in all the topics, TIND has higher recall than TAP. For the topic "election", the gap between the recall of TIND and TAP narrows when θ is more than 0.4. Overall, the precision for TIND is also higher than TAP. For 3 of the topics "social media", "finance" and "technology", the precision for TIND is higher than TAP when θ is more than 0.2.



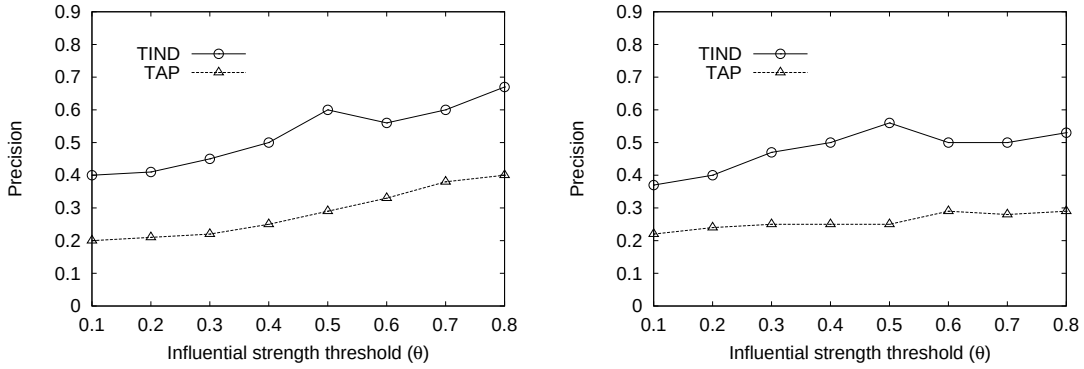
(a) Precision



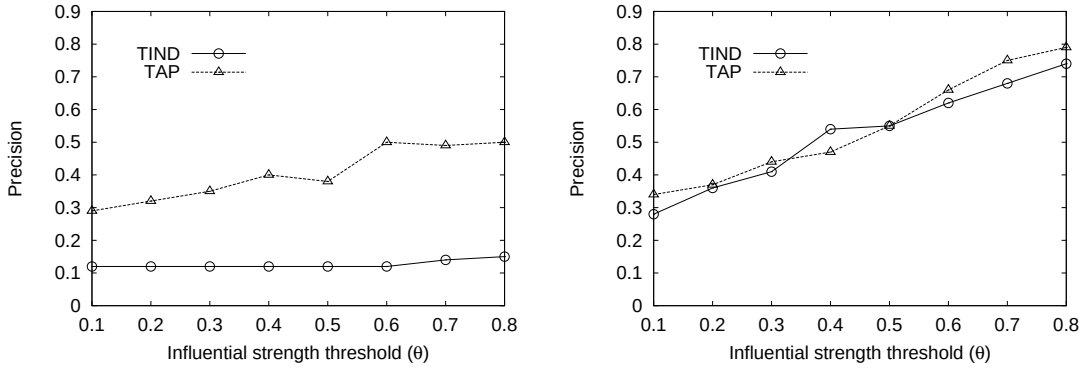
(b) Recall

Figure 4.11: Guided hLDA vs. clustering for varying τ on MemeTracker data

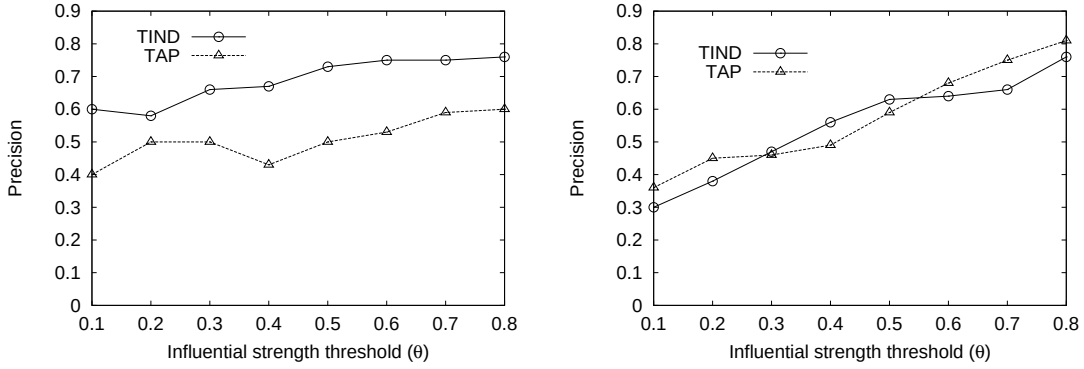
There are two reasons why TIND is better than TAP. One is the temporal factor. TAP does not consider the temporal factor whereas TIND takes into account the temporal factor. For the influential strength, TAP computes the influential strength based on document-topic distribution, which is at user level. On the other hand, TIND computes the influential strength based on topic-word distribution, which is at document level. So the influential strength obtained by TIND tends to be higher and more accurate than TAP.



(a) "freeiran" (b) "litchat"

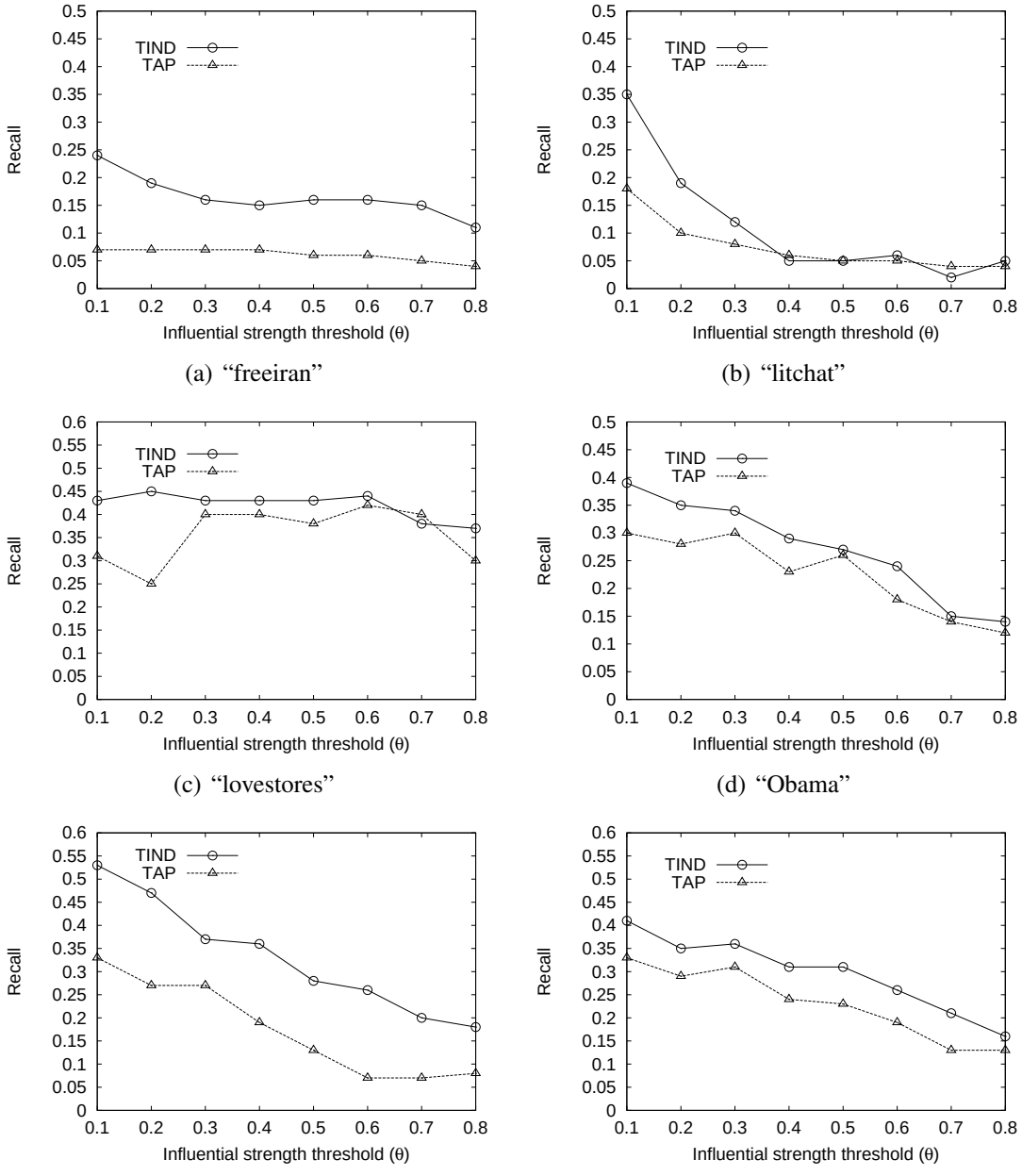


(c) "lovestores" (d) "Obama"



(e) "supernatural" (f) "Yahoo"

Figure 4.12: Precision of TIND vs. TAP for varying θ on Twitter data

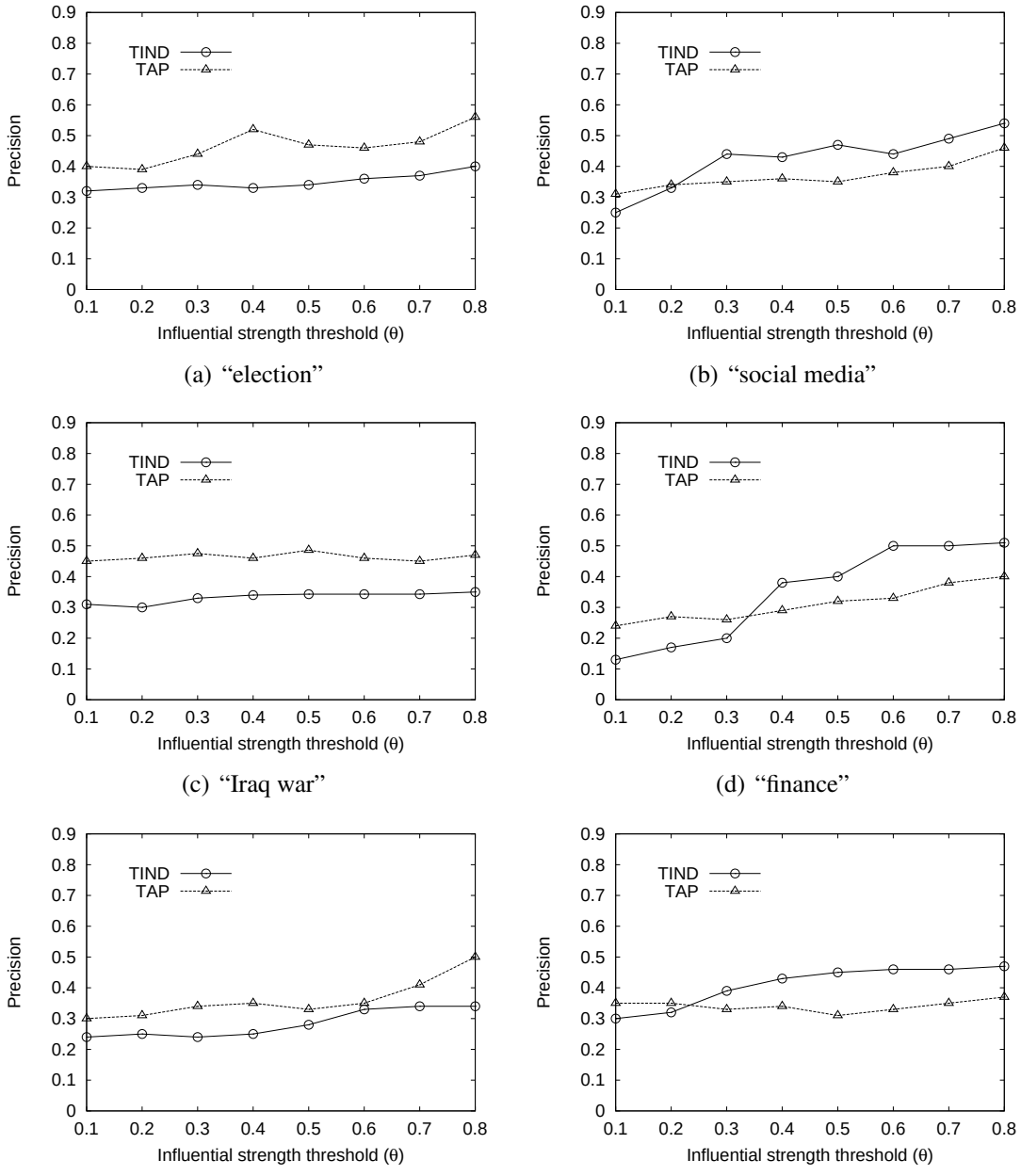


(a) "freeiran" (b) "litchat"

(c) "lovestores" (d) "Obama"

(e) "supernatural" (f) "Yahoo"

Figure 4.13: Recall of TIND vs. TAP for varying θ on Twitter data

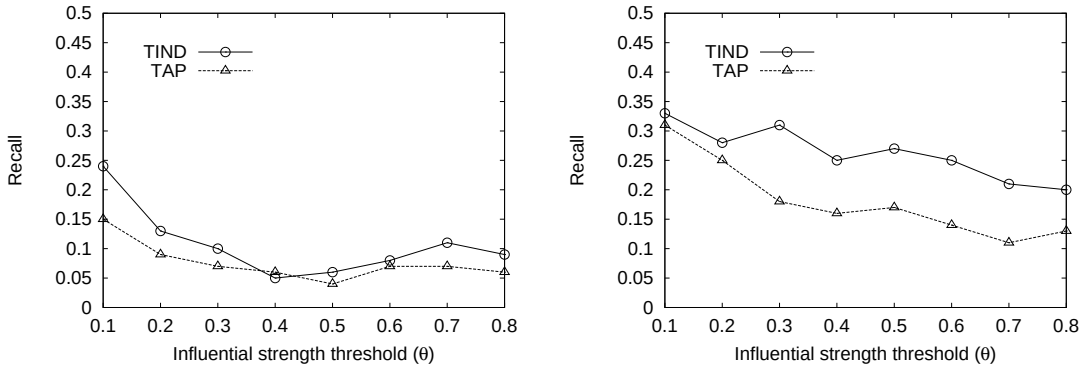


(a) "election" (b) "social media"

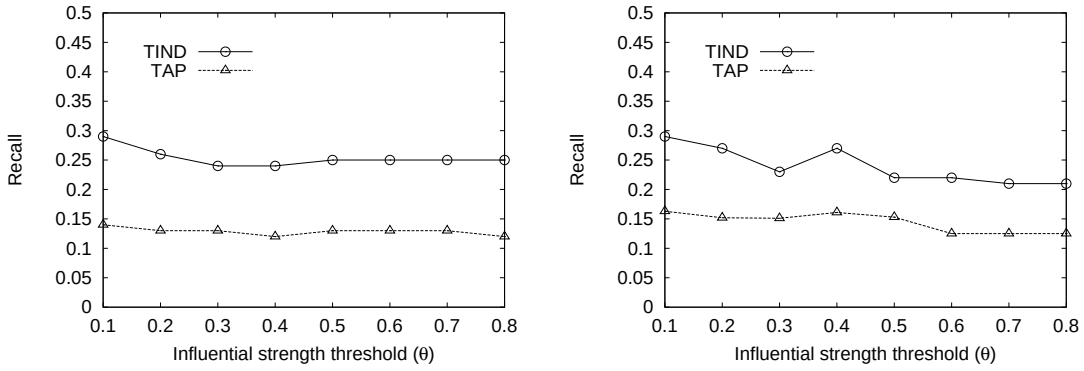
(c) "Iraq war" (d) "finance"

(e) "apple" (f) "technology"

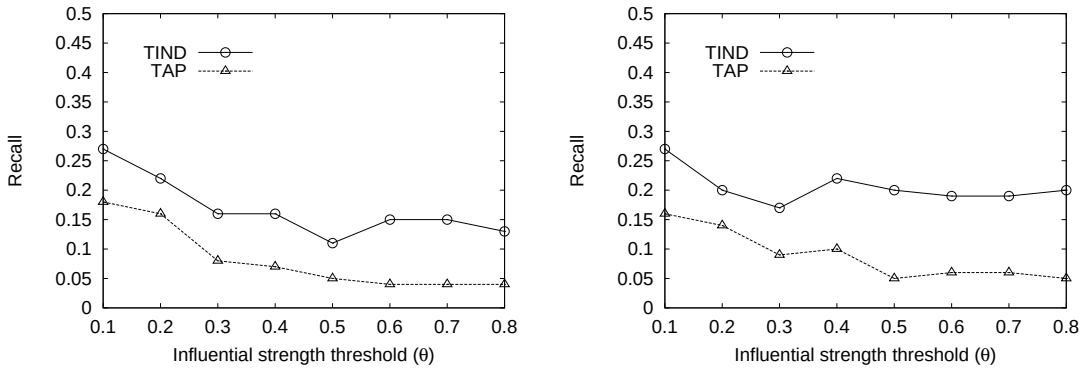
Figure 4.14: Precision of TIND vs. TAP for varying θ on MemeTracker data



(a) "election" (b) "social media"



(c) "Iraq war" (d) "finance"

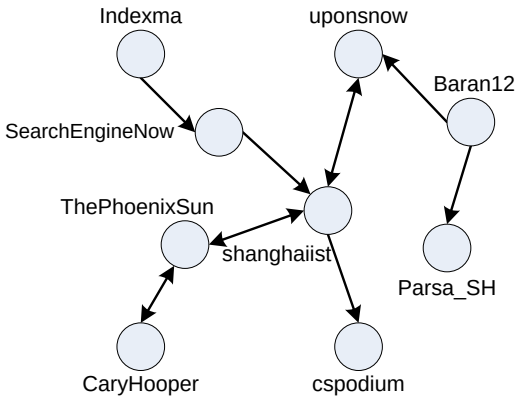


(e) "apple" (f) "technology"

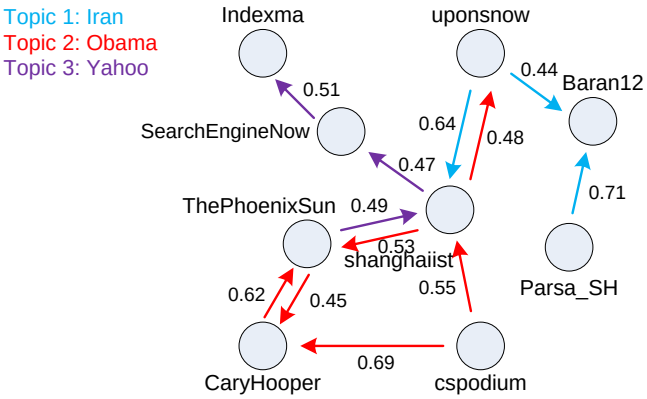
Figure 4.15: Recall of TIND vs. TAP for varying θ on MemeTracker data

4.5.2 Case Study

Figure 4.16(a) shows a sample of the follow relationships of users in the Twitter dataset, while Figure 4.16(b) shows the topic-level influence network obtained by our method. We see that when there is a following relationship from users u to v in Figure 4.16(a), our method will correctly infer that v influences u on the associated topic. For example, user $Indexma$ is following user $SearchEngineNow$, and our network shows that user $Indexma$ is influenced by user $SearchEngineNow$ on the topic "Yahoo".



(a) Following relationships



(b) Topic-level influence relationships

Figure 4.16: Topic-level influence network case study on Twitter data. (a) Following relationships of users in Twitter data. Each node is a user in Twitter. The directed edge from user u to v indicates that user u is a follower of v. (b) Topic-level influence relationships inferred by our method. Each node represents a user. Directed edge from user v to u indicates that user v influences u on a specific topic. Edge weights indicate the influential strength on that topic.

In addition, our method can also infer influence relationship between two users although they are not following each other. For example, there is no edge between user $CaryHooper$ and $cspodium$ in Figure 4.16(a), indicating that $CaryHooper$ is not following $cspodium$. However, our topic-level influence network discovers that $cspodium$ influences $CaryHooper$ on topic "Obama". When examining the tweets of $CaryHooper$, we realize that his tweets are very similar to $cspodium$'s and have been posted soon after $cspodium$'s tweets, indicating that $cspodium$ could have some influence on $CaryHooper$.

4.5.3 Applications

Topic-level influence networks have many applications. Here, we demonstrate how it is useful for user behavior prediction [69] and influence maximization [55, 64, 28, 27].

User Behavior Prediction. User behavior prediction is defined as whether a user will post a tweet on the same topic after another user has posted a tweet. An accurate prediction can lead to more effective target marketing and user recommendation.

Existing methods to perform user behavior prediction fall into either similarity-based methods or follower-based methods. The follower-based methods use the "following" relationship of the users in Twitter data to establish the edges among users w.r.t. specific topics; whereas in the similarity-based methods, the edges among users are determined based on the degree of similarity between two users. We say two users are similar based on the contents of the tweets they post. Let D_u and D_v be the set of tweets posted by users u and v respectively. The similarity between user u and v is defined as:

$$user\_sim(u, v) = max_{d_u \in D_u, d_v \in D_v}[\frac{d_u \cdot d_v}{||d_u||||d_v||}] \tag{4.9}$$

To demonstrate the effectiveness of topic-level influence network in user behavior prediction, we compare the precision obtained using the follower-based, similarity-based, TAP and TIND method. We sort the 64,451 tweets according to their time stamps and partition the data into two sets: the first half is used for training and the latter half is used for testing. Note that if the time difference between the two tweets posted by u and v is

larger than a given time threshold, we consider the two tweets to be unrelated and there will be no edge between u and v. In this experiment, we set the time threshold τ to 30 minutes for all the 4 methods.

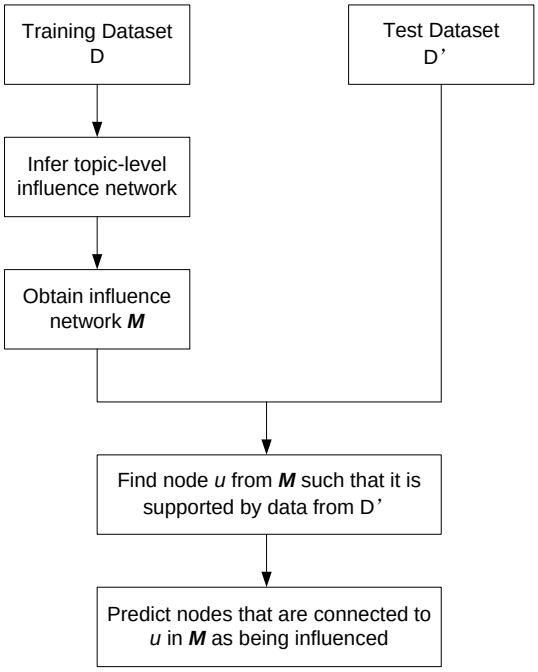
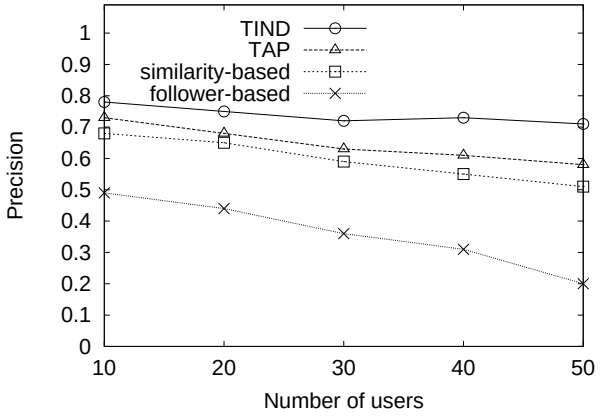


Figure 4.17: Prediction strategy

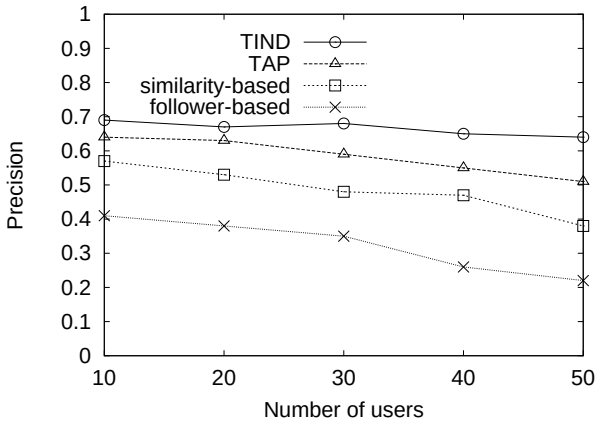
Figure 4.17 outlines our prediction strategy. We use the training dataset to construct four networks using the four methods. Based on the constructed networks, we perform user behavior prediction as follows: Let M be the constructed network. For each user u in the training set, let Y_u^M be the set of users that are connected to u in M. Let X_u be the set of users who have posted a tweet on the same topic within the time threshold after u's tweet in the test dataset. Then the precision of model M is given as:

$$precision(M) = \frac{\sum_u |X_u \cap Y_u^M|}{\sum_u |Y_u^M|} \tag{4.10}$$

We use the influential strength threshold of 0.6 and similarity threshold of 0.6 as determined in an empirical study. We plot the precision of the four models as we vary the number of users involved in the training and testing datasets. We repeat the experiment for two topics, i.e. "Obama" and "Yahoo". Figure 4.18 shows the results.



(a) User behavior prediction on topic "Obama"



(b) User behavior prediction on topic "Yahoo"

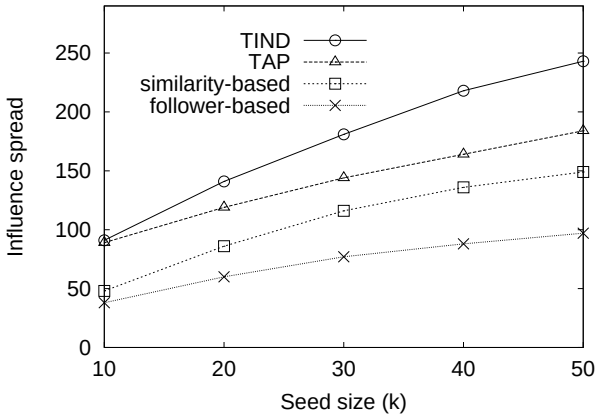
Figure 4.18: User behavior prediction

We observe that as the number of users increases, the precision for similarity-based and follower-based models decreases whereas the topic-level influence network is more stable. This is because similarity-based and follower-based models simply predict the most popular users without taking into consideration the topic. On the other hand, topic-level influence network predicts accurately users that are interested in the specific topic. Note that both TIND and TAP consider topic information, however, TIND outperforms TAP as TAP relies on the network structure. This demonstrates that topic-level influence can indeed improve the performance of user behavior prediction.

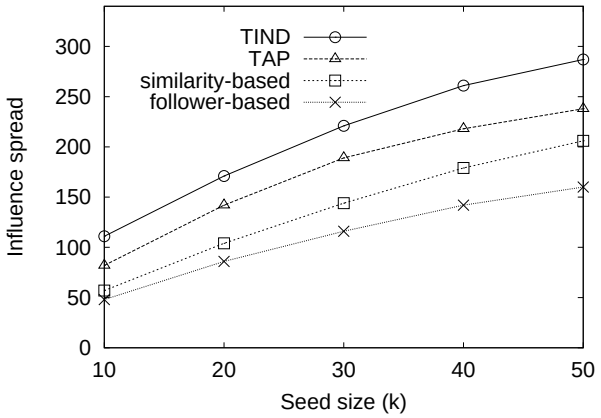
Influence Maximization. The problem of influence maximization in a social network is to find k nodes in the network such that the expected number of nodes influenced by these

k nodes is maximized. The work in [55] proposed a greedy algorithm to identify the k nodes. At each iteration, it selects a node that leads to the largest increase in the number of nodes influenced. The algorithm stops when k nodes are selected.

For topic-specific influence maximization, we define the influence spread of the k nodes as the set of nodes influenced by these k nodes on a given topic. A node u is said to be influenced by another node v on a topic z if the tweets posted by u and v contain topic z. In our experiments, we use the 64,451 tweets to construct four networks using the follower-based, similarity-based, TAP and TIND method. We run the greedy algorithm to find k nodes in each of the four networks. We select two topics, namely "Obama" and "Yahoo", and determine the influence spread of the k nodes on each topic.



(a) Influence maximization on topic "Obama"



(b) Influence maximization on topic "Yahoo"

Figure 4.19: Influence maximization

Figure 4.19 shows the influence spread as we vary k from 10 to 50. We observe that the influence spread of all the four methods increases as k increases with TIND clearly in the lead. This demonstrates that topic-level influence network is effective for influence maximization.

4.6 Summary

In this chapter, we have investigated topic-level influence, e.g. the influential strength between two users at a specific topic. We take into account the temporal factor in social influence to infer the influential strength between users at topic-level. Our approach does not require the underlying network structure to be known. We propose a guided hierarchical LDA approach to automatically identify topics without using any structural information. We then construct the topic-level social influence network incorporating the temporal factor to infer the influential strength among the users for each topic. Experimental results on two real world datasets (Twitter and MemeTracker) demonstrate the effectiveness of our methods. Further, we show that the proposed topic-level influence network can improve the precision of user behavior prediction and is useful for influence maximization.

Chapter 5

Identifying k-Consistent Influencers

In this chapter, we define the notion of k-consistent influential users and devise an efficient algorithm called TCI to identify these users. Our algorithm linearizes the 2D personal-preference consistency space to construct a GridIndex. Based on the GridIndex, we can quickly obtain the k-consistent influencers for a given time interval. We conduct extensive experiments on three real world datasets to evaluate the efficiency of the proposed approach, as well as the effectiveness of using k-consistent influencers to identify information sources and experts.

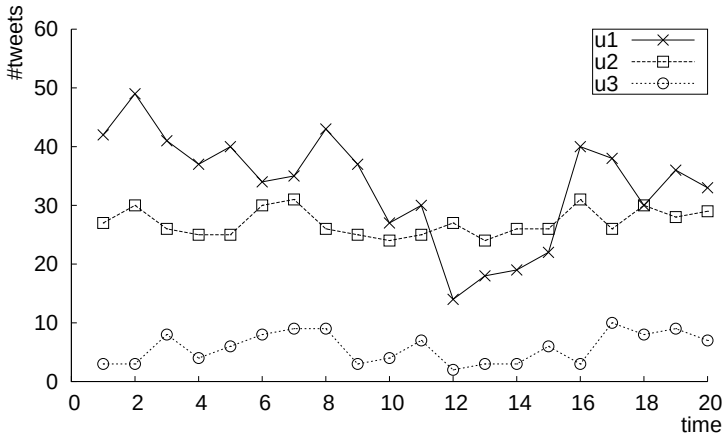
The remaining of this chapter is organized as follows: We start with the motivation of identifying k-consistent influencers in Section 5.1. In Section 5.2, we introduce some terminologies, and then give the formal problem definition. We describe the TCI algorithm in Section 5.3. We conduct experiments in Section 5.4. Finally, we summarize our work in Section 5.5.

5.1 Motivation

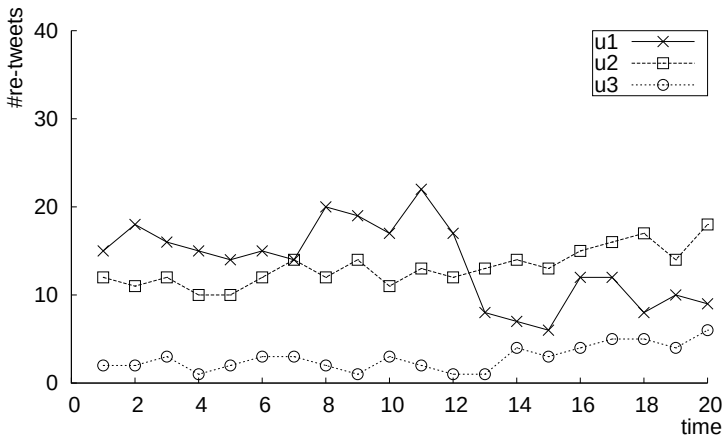
Social networking sites such as Facebook, Twitter, Delicious and YouTube have provided a platform where user can express their ideas and share information. With the prevalence of these sites, social networks now play a significant role in the spread of information. Recognizing this, researchers have focused on influence analysis to discover influential

nodes (users, entities) and influence relationships (who influences whom) among nodes in the network. Existing works on influential nodes discovery define influential user as one who posts/tweets frequently and/or with a large number of followers/friends. However, from a psychological perspective, frequency and popularity are not sufficient to develop influence and loyalty. Instead, it is consistency that builds trusts and thereby resulting in the greatest influence.

We observe that consistency comes in two forms. The first form of consistency is known as personal consistency. This refers to one who is consistent in his behavior; for example, a user could tweet regularly on the same topic over a period of time. This user tends to gain greater authority as other users' trusts in him grow, and thereby increases his influence.



(a) Personal consistency



(b) Preference consistency

Figure 5.1: Example of two forms of consistency

The second form refers to our preference for consistent behavior. We have a tendency to remain consistent with our previous actions. In the case of social networking, if a user u_2 has retweeted a post from another user u_1, there is a much higher probability that u_2 will retweet other posts from the same user u_1. In other words, u_2 has a strong preference for u_1.

Figure 5.1(a) shows an example of 3 users' tweeting frequency over 20 time points and the number of followers they have. We observe that both u_1 and u_2 have a large number of followers. However, u_1's tweeting frequency appears random whereas u_2 consistently tweets at regular interval. On the other hand, u_3 has a small number of followers but he tweets regularly. Figure 5.1(b) shows the number of followers retweeting their tweets over the 20 time points. In the beginning, u_1 appears to have the most number of followers retweeting his tweets. However, over time, the number of followers retweeting his tweets declines. In contrast, both u_2 and u_3 maintain the same number of followers retweeting their tweets. However, since u_3's base of followers is small, his influence is not as great as u_2.

Clearly, an accurate measure of degree of influence must take into account these two forms of consistency. A user is highly influential if he has high personal consistency and he has established consistent preferences to his tweets/posts in a large number of users.

We can depict the 3 users in a 2D personal-preference consistency space over 5 time points as shown in Figure 5.2. We observe that users near the top right corner are high in both personal and preference consistency. For example, u_1 has the highest personal and preference consistency at $t = 2$ and $t = 8$, but its personal consistency drops at $t = 13$ and $t = 14$. On the other hand, u_2 has the second highest personal and preference consistency at $t = 2$ and $t = 8$, and leads at time points $t = 13$, $t = 14$ and $t = 18$. Clearly, u_2 is more consistent and hence, can exert a greater influence over time compared to u_1 who seems to be more volatile.

Finding top-k consistent influencers has many interesting applications, such as targeted marketing, recommendation, experts finding, and stock market. Identifying top-k

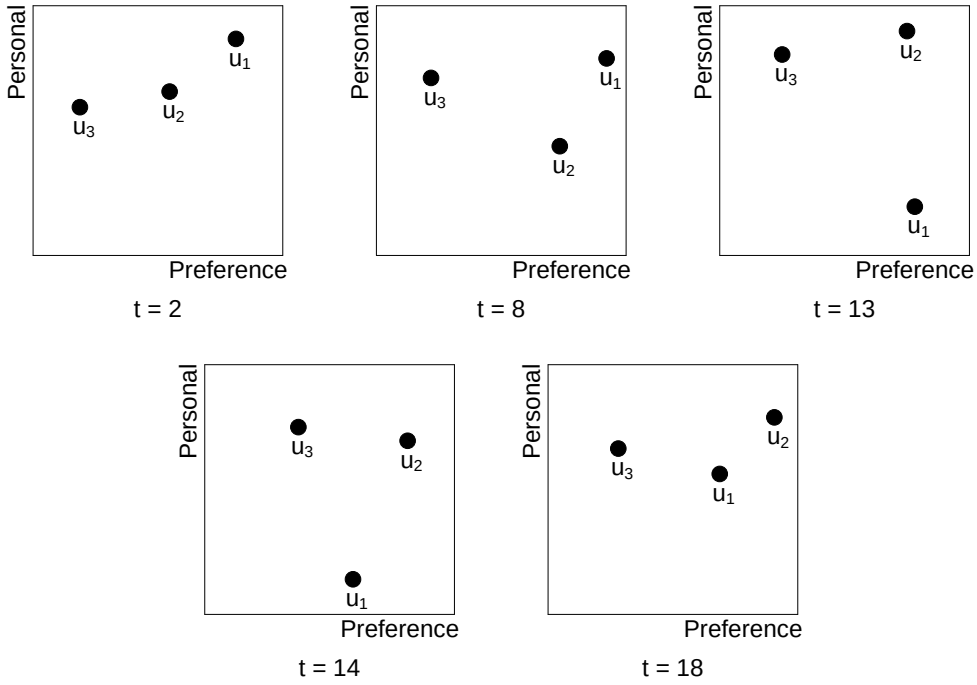


Figure 5.2: Personal-Preference 2D space

consistent influencers is a challenging task. First, we need to dynamically compute the total influence of each user at each time point from an action log. However, to find the consistent top-scorers, we need to sort and rank them at each time point. This is computationally expensive and not scalable.

5.2 Preliminaries

In this section, we first introduce some terminologies, and then give the formal problem definition.

Definition 5. Action Log. *An* action log *is a relation D where a tuple $< t, u, a > \in D$ indicates that node u has performed action a at time t.*

Figure 5.3 shows an action log and the corresponding user relation graph. For example, node u_1 performs action a at time point 0 and u_4 performs the same action a following u_1 at time point 1.

time	user	action
0	u1, u2, u3	a
0	u1, u3	b
0	u3	c
1	u1, u3, u4, u5, u7	a
1	u1, u2, u4, u7	b
1	u1, u2, u7	c
2	u1, u2, u4, u7	a
2	u2, u3, u4, u5	b
2	u2, u3, u4, u6	c
3	u2, u3, u4, u5	a
3	u1, u2, u5, u7	b
3	u2, u6, u7	c
4	u1, u2, u5, u7	a
4	u1, u2, u3, u4, u5	b
4	u1, u2, u6	c
5	u2, u3, u4, u5	a
5	u2, u3, u4, u5, u7	b
5	u2, u3, u4, u6	c

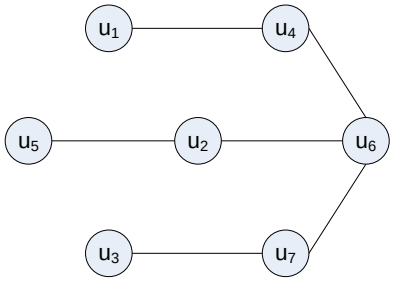


Figure 5.3: Action log and graph

Definition 6. Degree of Influence. *Let $G = (V, E)$ denote a social network where V and E are the sets of nodes and edges respectively. An edge $(u, v) \in E$ represents a relationship between node u and v. We say a node u influences node v on action a if we have $(u, v) \in E$, $< t_u, u, a >$, $< t_v, v, a > \in D$, and $t_v - t_u \leq \tau$, where τ is the time threshold. The* degree of influence *that node u has on v for action a, denoted as $p(u, v, a)$, is defined by:*

$$p(u,v,a) = \begin{cases} 0 & if \;\; t_v - t_u > \tau \\ e^{\,(t_u - t_v)} & otherwise \end{cases} \tag{5.1}$$

This implies that if node u performs an action, and shortly thereafter node v repeats the same action, then it is highly likely that u has an influence on v. On the other hand, if v repeats the action only after a long lapse, then we may conclude that it is an independent

action and that u has little influence on v.

Figure 5.4 shows the corresponding influence graph for the example action log. A directed edge from node u to v with label a denotes that node u influences v on action a. Let the time threshold τ = 1. The degree of influence that node u_1 has on u_4 for action a is $p(u_1, u_4, a) = e^{(-1)} = 0.37$.

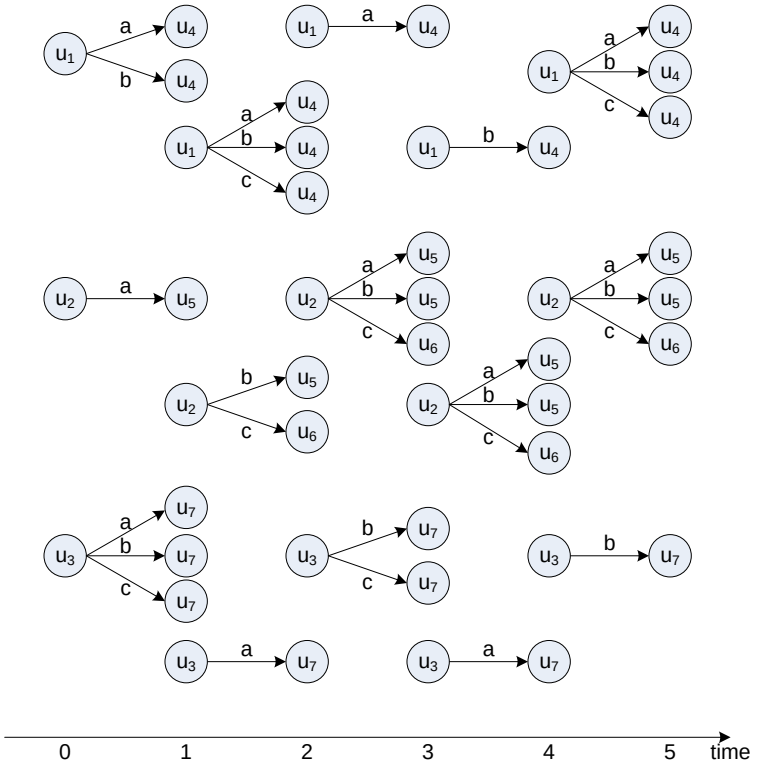


Figure 5.4: Influence graph

Definition 7. Preference Consistency. *Let A_t denote the set of actions taken by nodes u and v between the start time t_s and a given time point t. The* preference *of a node u for the node v is given by:*

$$Preference(u, v, t) = \sum_{a \in A_t} p(u, v, a) \tag{5.2}$$

The preference consistency *of node u at time point t is defined by:*

$$PrefCon(u, t) = \sum_{v \in V} Preference(u, v, t) \tag{5.3}$$

In the time interval [3,4], node u_5 performs action a and b following u_2, and node u_6 performs action c following u_2, so the preference of node u_2 for u_5 at time point 4 is $Preference(u_2, u_5, 4)$ = $p(u_2, u_5, a)$ + $p(u_2, u_5, b)$ = 0.74, and the preference of node u_2 for u_6 is $Preference(u_2, u_6, 4)$ = $p(u_2, u_6, c)$ = 0.37. The preference consistency of node u_2 at time point 4 is $PrefCon(u_2, 4)$ = $Preference(u_2, u_5, 4)$ + $Preference(u_2, u_6, 4)$ = 1.11. On the other hand, node u_4 performs action b following u_1, so the preference consistency of node u_1 at time point 4 is $PrefCon(u_1, 4)$ = $Preference(u_1, u_4, 4)$ = 0.37.

Definition 8. Personal Consistency. *Let $M = \{m_{t_s}, \cdots, m_t\}$ be the number of actions taken by user u from the start time t_s to time point t. Let μ be the mean value of M. Then the* personal consistency *of u at time point t is given by:*

$$PersonCon(u, t) = \frac{|M|}{\sum_{j \in [t_s, t]} (m_j - \mu)^2} \tag{5.4}$$

This is equivalent to the inverse value of the standard deviation of the number of posts made by u. A higher value in $PersonCon(u, t)$ implies a smaller deviation in the number of postings over time, implying that user u is more consistent. For example, the personal consistency of node u_1 and u_2 at time point 4 is as follows.

$$PersonCon(u_1, 4)$$
$$= \frac{5}{(2-2)^2 + (3-2)^2 + (1-2)^2 + (1-2)^2 + (3-2)^2}$$
$$= 1.25$$

$$PersonCon(u_2, 4)$$
$$= \frac{5}{(1-2.4)^2 + (2-2.4)^2 + (3-2.4)^2 + (3-2.4)^2 + (3-2.4)^2}$$
$$= 1.56$$

Definition 9. Overall Consistency. *The* consistency *of node u at time point t is defined as:*

$$Consistency(u,t) = \Theta(PrefCon(u,t), PersonCon(u,t)) \tag{5.5}$$

where Θ can be any function that maps the pair $(PrefCon(u,t), PersonCon(u,t))$ to a real number. In our experiment, we set Θ as the sum of the two terms.

Given the preference and personal consistency of node u_1 at time point 4, the overall consistency of u_1 is $Consistency(u_1, 4)$ = $PrefCon(u_1, 4)$ + $PersonCon(u_1, 4)$ = 0.37 + 1.25 = 1.62.

We rank the users based on their overall consistency values at each time point.

Definition 10. Rank. *Given a node u at time point t, let $S = \{v \in V \mid Consistency(v,t) > Consistency(u,t)\}$. Then, the rank of u at t is given by:*

$$rank(u,t) = |S|$$

Similarly, for node u_2 and u_3 we have: $Consistency(u_2, 4)$ = 2.67, $Consistency(u_3, 4)$ = 2.3. So the rank of node u_1, u_2 and u_3 at time point 4 is 3, 1 and 2 respectively.

Definition 11. Volatility. *Let $\mu_{rank}(u)$ denote the mean rank of u in the query interval $[q_s, q_e]$. The $volatility$ of node u in the interval $[q_s, q_e]$ is given by:*

$$Volatility(u) = \frac{\sum_{t\in[q_s,q_e]}(rank(u,t) - \mu_{rank}(u))^2}{q_e - q_s + 1} \tag{5.6}$$

For node u_1, we can get its rank at each time point in the time interval [1,5]. The volatility of u_1 is

$$Volatility(u_1)$$
$$= \frac{(1-2)^2 + (1-2)^2 + (3-2)^2 + (3-2)^2 + (2-2)^2}{5}$$
$$= 0.8$$

Similarly, for node u_2 we have

$$
\begin{aligned}
&Volatility(u_2) \\
&= \frac{(2-1.4)^2 + (2-1.4)^2 + (1-1.4)^2 + (1-1.4)^2 + (1-1.4)^2}{5} \\
&= 0.24
\end{aligned}
$$

Definition 12. Score. *The* score *of node u in the query interval $[q_s, q_e]$ is the weighted sum of consistency and volatility:*

$$Score(u) = w_1 * \sum_{t \in [q_s, q_e]} Consistency(u,t) \; - \; w_2 * Volatility(u), \tag{5.7}$$

where w_1 + w_2 = 1, $w_1 > 0$ *and* $w_2 > 0$.

Let $w_1 = w_2 = 0.5$. The score of node u_1 in the time interval [1,5] is $Score(u_1)$ = 0.5 $\times$ 12.71 $-$ 0.5 $\times$ 0.8 = 5.96. Similarly, $Score(u_2)$ = 0.5 $\times$ 14.66 $-$ 0.5 $\times$ 0.24 = 7.21. We can see that node u_2 is more consistent than u_1.

Problem Statement: Given an action log D, a social network graph G, a query time interval $[q_s, q_e]$, and time threshold τ, we want to identify a subset of users $U \subset V$ such that $|U|$ = k and $\forall u \in U$, $\nexists v \in V \setminus U$ such that $Score(v) > Score(u)$. We call the users in U the k-consistent influencers in G.

5.3 The TCI Algorithm

In this section, we first briefly review works in top-k query processing and then give the details of our TCI algorithm. Fagin et al. [40] introduce the TA and NRA algorithms for computing the top-k queries over multiple sources, where each source provides a ranking of a subset of attributes only. Variations of the threshold-based algorithms have been proposed to improve the efficiency of top-k queries [13, 21, 77, 34, 5]. Works in [36, 24] proposed several early termination algorithms for disjunctive top-k query processing,

based on a new augmented index structure called Block-Max Index. The basic idea of Block-Max Index is to store the maximum score for each block, thus enabling to skip aggressively in the index. Later, Shan et al. [95] and Dimopoulos et al. [35] proposed new algorithms to further improve the efficiency of top-k query. Note that the Block-Max Index, which skips over blocks for efficiency, is not applicable to our problem, as we need to know the ranks of users at each time point. Jestes et al. [53] study the problem of performing top-k queries on a time window. In [78], Mouratidis et al. proposed the TMA algorithm (and the more specialized SMA) for supporting multiple continuous top-k queries over data streams.

The works that are most relevant to ours are durable queries on temporal data. Lee et al. [61] were the first to study consistent top-k query. They construct a RankList for each time series to store the rank information. During query processing, they traverse the list of each time series and search for entries with rank values greater than k. The process terminates whenever an entry in the list with rank value greater than k is encountered. Wang et al. [110] proposed an efficient method called TES for durable top-k queries. TES exploits the fact that the changes in the top-k set at adjacent time stamps are usually small. TES indexes these changes and incrementally computes the snapshot top-k sets at each time stamp of the query window.

However, in our setting, the ranked lists correspond to users who are high in consistency values. Yet, these users may not have high scores if their rank positions differ vastly at different time points. To account for this, our proposed algorithm dynamically computes the total score that combines consistency and volatility, and output the k-consistent users.

Figure 5.5 gives an overview of our proposed approach to identify the k-consistent influencers. Given an action log and a user relationship graph, we compute the personal and preference consistency of each user u at time point t.

We compute $PrefCon(u, t)$ by examining all users who have performed the same action following u's action. If v has previously followed u and the time lapse between v's

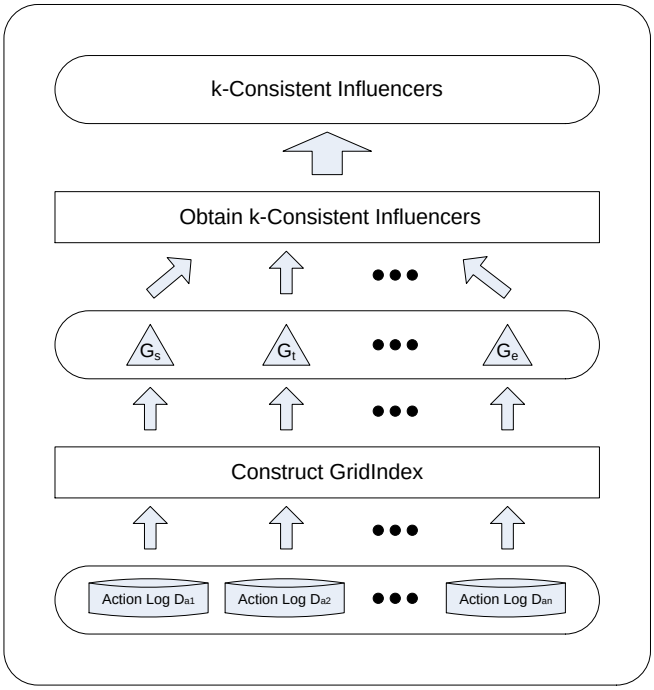


Figure 5.5: Solution overview

and u's actions is smaller than time threshold τ, we conclude that u's post has influenced v to some degree and this influence will be included in computing the preference consistency of node u for v according to Equation 5.3. Otherwise, the influence of u's post on v is said to be negligible and will be ignored.

For $PersonCon(u,t)$, we keep track of the number of posts made by user u from the start time point t_s till current time point t and obtain the variance of these numbers.

Each pair of $(PrefCon(u,t), PersonCon(u,t))$ values is a point in the personal-preference 2D space. To find users with the top-k overall consistency values $Consistency(u,t)$, the naive way of sorting users by their consistency values is computationally expensive as there may be millions of users at each time point. Given that k is typically a small fraction compared to the total number of users, this is certainly not efficient.

Instead, we partition this 2D space into cells of size $\delta \times \delta$ and assign a user u to the cell $\left(\left\lfloor \frac{PrefCon(u,t)}{\delta} \right\rfloor, \left\lfloor \frac{PersonCon(u,t)}{\delta} \right\rfloor\right)$. We observe that the top-right grid has the highest overall consistency value. As we slide the black line from this top-right cell towards the bottom-left cell, the consistency values of the users in the cells will decrease.

In other words, if we wish to find the top-k influencers, we only need to process the cells in the zig-zag order as shown by the arrows in Figure 5.6. In this manner, only the likely candidates for k-consistent users in the shaded cells are processed, resulting in great savings of computational time.

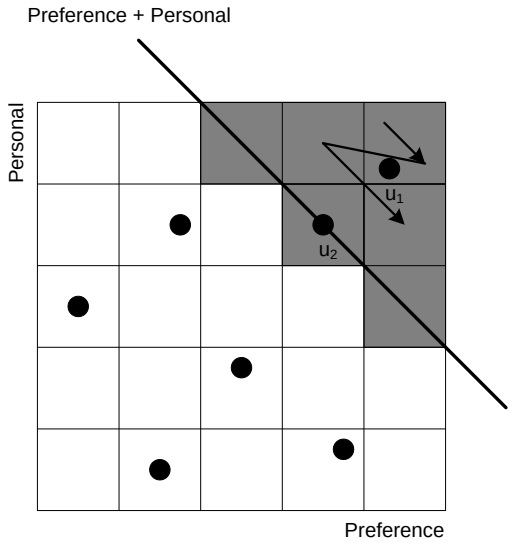


Figure 5.6: Illustration of zig-zag traversal

We map the users to the cells in a grid based on their personal and preference consistency values at time point t. Figure 5.7 shows the grids at the various time points.

Next, we design a function Φ to linearize the grids so that the cells can be processed in the desired zig-zag order as follows:

$$\Phi(i,j) = (N+M) - (\lfloor i \rfloor + \lfloor j \rfloor)$$

where N is the maximum $\lfloor \frac{PrefCon(u,t)}{\delta} \rfloor$ value and M is the maximum $\lfloor \frac{PersonCon(u,t)}{\delta} \rfloor$ value.

Note that $\Phi(N,M) = 0$, $\Phi(N-1,M) = \Phi(N,M-1) = 1$, and $\Phi(N-2,M) = \Phi(N-1,M-1) = \Phi(N,M-2) = 2$, etc. We call this set of linearized grids the GridIndex. Figure 5.8 shows the GridIndex obtained from Figure 5.7.

Based on the GridIndex, we design an algorithm called TCI to find the top-k consistent influencers. We obtain the initial lists of the top-k candidate users at each time point from

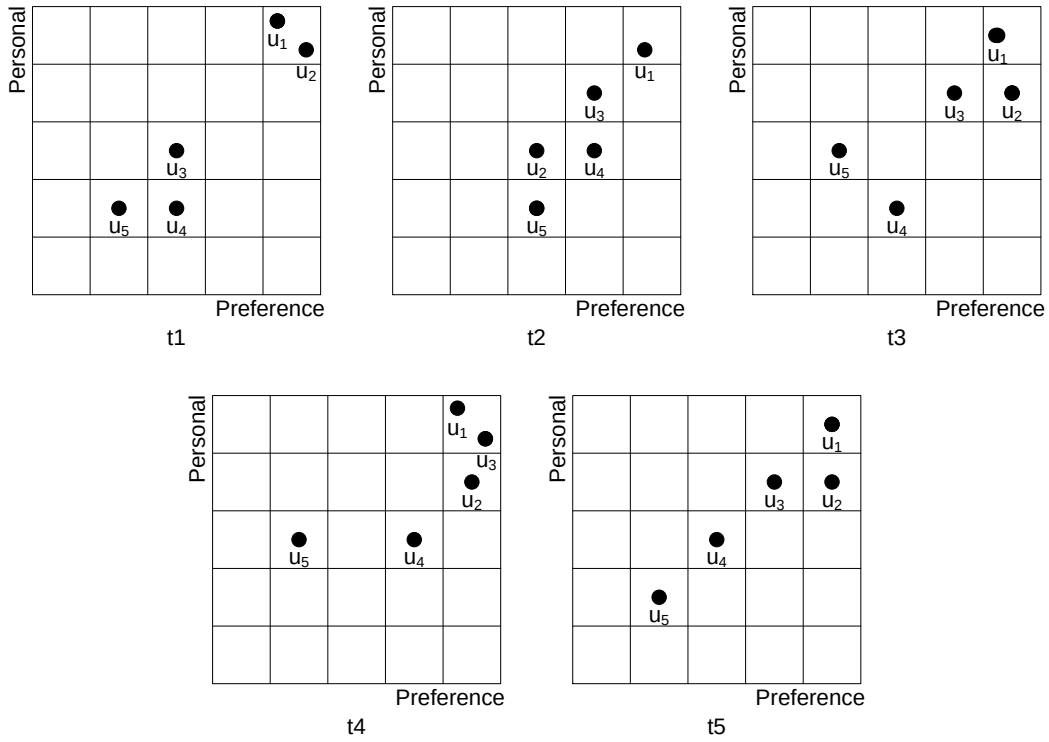


Figure 5.7: Grids at different time points

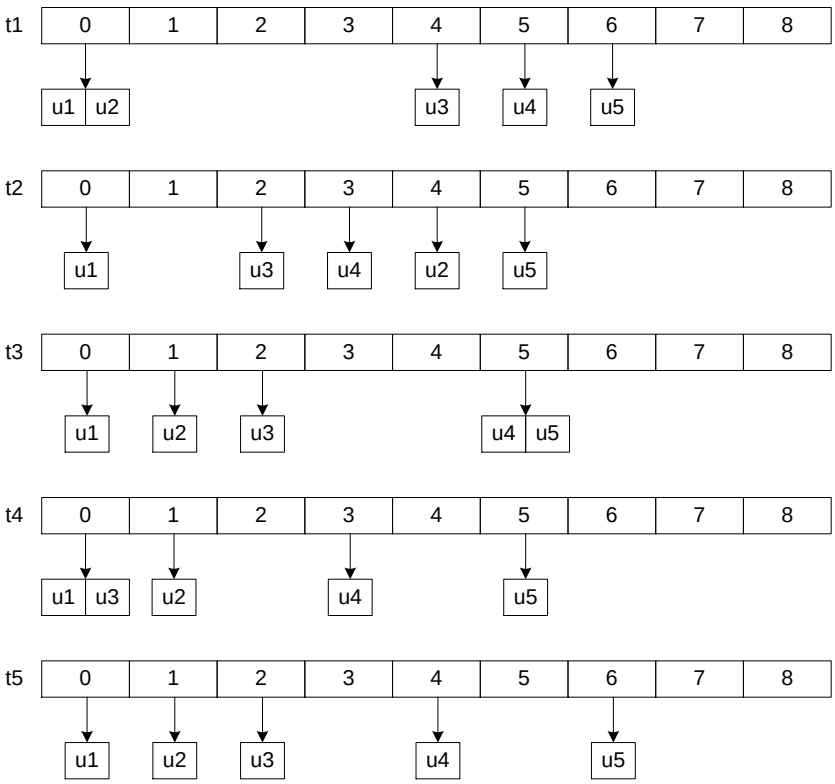


Figure 5.8: GridIndex obtained from Figure 5.7

the set of linearized grids. If a candidate user u does not appear in the lists for any of the time point, say t, then we traverse the grid at t till we find u, and compute its score $Score(u)$.

Algorithm 8 shows the details of TCI. The algorithm first scans each action log D_a backwards with a sliding window of size τ (Lines 1-8). For each tuple $< t, u, a > \in D_a$, we increment the number of posts made by user u at time point t and utilize the user relationship graph G to compute the preference of u at t.

After scanning all the action logs, we compute the preference consistency $PrefCon(u, t)$ and personal consistency $PersonCon(u, t)$ for each user at each time point (Lines 9-12). Then we insert the users into the various linearized grids in the GridIndex (Line 13).

Once we have constructed the GridIndex, the algorithm tries to identify the set of top-k consistent influencers, $Result$. For each time point in the given query interval, we first obtain the initial list of candidate influencers (Lines 22-27). For early termination, we compute a threshold value by summing the consistency values of the candidate users at position p in each grid G_t (Lines 28-29).

For each candidate user u who does not appear in all the lists, we expand the candidate sets corresponding to the time points that u is missing from until u is included in the candidate set (Lines 30-39). When this is completed, we obtain the rank of u at all the time points and compute the volatility of u (Lines 40-46). Finally, we compute the score of u (Line 47).

If the size of the result set is less than k, we add u to R (Lines 50-51). Otherwise, we check whether the score of u is larger than that of the k^{th} user in $Result$. If yes, we replace the k^{th} user with u (Lines 52-58). The algorithm terminates when the size of $Result$ is k and $threshold$ is smaller than $Score_{min}$.

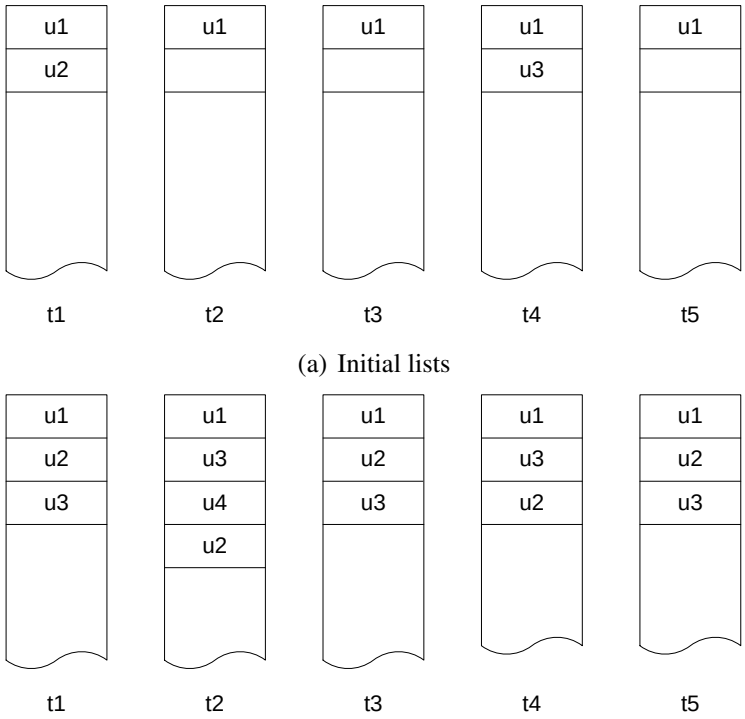
Let us illustrate how the constructed GridIndex in Figure 5.8 is used to find the 2-consistent influencers.

The initial lists obtained for the 5 time points are shown in Figure 5.9(a). We observe that u_2 has not appeared in the time points t_2 to t_5, so we proceed to traverse the GridIndex

Algorithm 8 TCI

Require: action log D, user relationship graph G, query interval $[q_s, q_e]$, time threshold τ, and integer k
Ensure: set of k-consistent influencers $Result$

for each $D_a \subset D$ where D_a is a projection of D on action a **do**
  initialize $numPost_{u,t}$ to 0 for all u and t
  for each tuple $< t, u, a > \in D_a$ **do**
    increment $numPost_{u,t}$
    $V = \{v \mid < t', v, a > \in D_a, (u, v) \in G, t' \in [t+1, t+\tau]\}$
    $Preference(u, v, t') += p(u, v, a)$
  end for
end for
let G_t be the linearized grid at time t
for each user u and time point t **do**
  $PrefCon(u, t) += Preference(u, v, t)$
  compute $PersonCon(u, t)$ from $avg(numPost_{u,t})$ and $sum(numPost_{u,t})$ using Equation 5.4
  insert u to $G_t[\Phi(\lfloor \frac{PrefCon(u,t)}{\delta} \rfloor, \lfloor \frac{PersonCon(u,t)}{\delta} \rfloor)]$
end for
$Result \leftarrow \emptyset$
initialize $threshold$, $Score_{min}$ to 0 and position p to 1
for $t = q_s$ to q_e **do**
  $ptr_t \leftarrow 0$
  $candSet_t \leftarrow G_t[ptr_t]$
end for
while ($|Result| < k$ or $threshold > Score_{min}$) **do**
  for each $t \in [q_s, q_e]$ **do**
    while $|candSet_t| < p$ **do**
      increment ptr_t
      $candSet_t = candSet_t \bigcup G_t[ptr_t]$
    end while
  end for
  let θ_t be the consistency value of the user at p in $candSet_t$
  $threshold = \sum_{t \in [q_s, q_e]} \theta_t$
  let $C = \bigcup candSet_t - \bigcap candSet_t$
  for each user $u \in C$ **do**
    let T be the set of time points that u has not appeared
    for each $t \in T$ **do**
      while $u \notin candSet_t$ **do**
        increment ptr_t
        $candSet_t = candSet_t \bigcup G_t[ptr_t]$
      end while
    end for
  end for
  for each user $u \in \bigcup candSet_t$ **do**
    $rank_u = 0$
    for each $t \in [q_s, q_e]$ **do**
      $rank_u$ += position of u in $candSet_t$
    end for
    $ave\_rank_u = \frac{rank_u}{q_e - q_s + 1}$
    compute $Volatility(u)$ using Equation 5.6
    compute $Score(u)$ using Equation 5.7
  end for
  increment position p
  if $|Result| < k$ **then**
    $Result = Result \cup \{u\}$
  else
    let u' be the user with lowest score in $Result$ and $Score_{min} = Score(u')$
    if $Score(u) > Score(u')$ **then**
      $R = R - \{u'\}$
      $R = R \cup \{u\}$
    end if
  end if
end while
return $Result$



(a) Initial lists

(b) Updated lists

Figure 5.9: Rank lists

at time points t_2 to t_5 to retrieve additional users till u_2 is found. Similarly, we traverse the GridIndex where u_3 has not appeared to retrieve additional users till u_3 is found. Figure 5.9(b) shows the updated lists.

Initially, the $threshold$ and $Score_{min}$ are set to 0. We first compute the score of u_1 and get $Score(u_1)$ = 2.25. At this time, $threshold$ is 2.25. We continue to compute the score of users until the score of the 2nd user is larger than the threshold. For u_2 and u_3, we have $Score(u_2)$ = 1.0 and $Score(u_3)$ = 1.6. We update $threshold$ to 1.2. Since the current 2nd user is u_3 and $Score(u_3) > threshold$, we can be sure that the 2-consistent influencers are u_1 and u_3.

5.4 Experimental Evaluation

In this section, we present the results of experiments conducted to evaluate the effectiveness and efficiency of our methods. We implement all the algorithms in Java. The

experiments are performed using an Intel Core 2 Quad CPU 2.83 GHz system with 3GB of main memory and running Windows XP operating system.

We use the following real world datasets in our experiments:

1. Citation dataset [105, 104]. This dataset is part of the DBLP computer science bibliography. It contains 1,397,240 papers and 3,021,489 citation relationships between these papers. Each paper is associated with attributes such as abstract, authors, year, venue, and title, etc.

2. Flixster dataset [51]. This is a social network for movies in which users to share their opinion on movies with friends by rating and reviewing movies. The Flixster dataset has 1M users, 26.7M friendship relations among users, and 8.2M ratings that range from half a star (rating 0.5) to five stars (rating 5). On average each user has 27 friends and each user has rated 8.2 movies.

3. Twitter dataset [116, 60]. This dataset consists of 17,214,780 tweets published by 1,746,259 users over a 7 month period from June 1 2009 to December 31 2009. Each tweet has the following information: user, time and content. We preprocess the tweets by removing tweets that are not in English or have no hashtags.

Table 5.1 summarizes the characteristics of these datasets. In our experiments, we set the query interval $[q_s, q_e]$ to be the whole period of the datasets. The grid size is set to 10 $\times$ 10. The default value for weight w_1 and w_2 is 0.5 respectively.

Table 5.1: Dataset statistics

Datasets	# Nodes	# Edges	Avg Edges	Max Edges
Citation	1,397,240	3,021,489	2.16	4,090
Flixster	1M	26.7M	26.70	1,045
Twitter	1,746,259	92,286,461	52.85	241,428

5.4.1 Efficiency Experiments

We first evaluate the efficiency of TCI. For comparison, we also implement TCI-NoGrid, a variant of TCI that does not utilize the GridIndex structure. TCI-NoGrid sorts all users

by their consistency values at each time point to obtain their ranks. Then it retrieves candidate users from the rank lists at each time point and computes their scores. If a retrieved user u does not appear in the lists for all time points, TCI-NoGrid will retrieve the rank lists where u does not appear to find u.

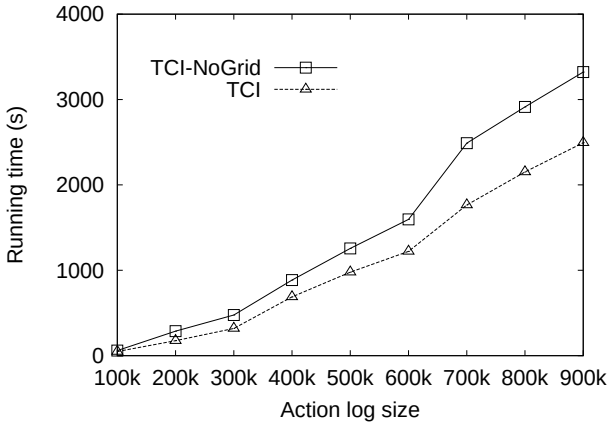
We vary the size of the action logs from 100k to 900k, and set k = 5. For the Citation dataset, we set τ to 10 years. For the Flixster dataset, τ = 10 days. For the Twitter dataset, τ is set to 10 hours.

Figure 5.10 shows the runtime for TCI and TCI-NoGrid on the three datasets. We observe that TCI outperforms TCI-NoGrid, and the gap widens as the action log size increases. This demonstrates that the grid index is effective in reducing the runtime. For the Flixster dataset, the grid index is not very beneficial. This is because the ranks of users in Flixster dataset vary greatly.

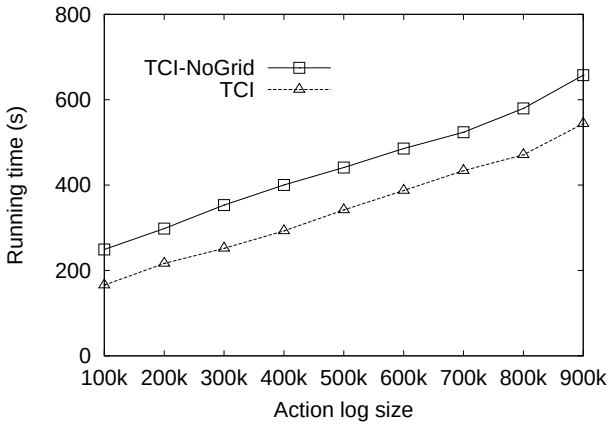
5.4.2 Sensitivity Experiments

We also examine the effect of the parameters k and τ on the performance of TCI and TCI-NoGrid. We fix the size of the action log at 100k, and vary k from 5 to 25. Figure 5.11 shows the runtime for both methods. We observe that the runtime does not change much as k increases. This is because both algorithms have to scan the action log, the time of which dominates the total running time.

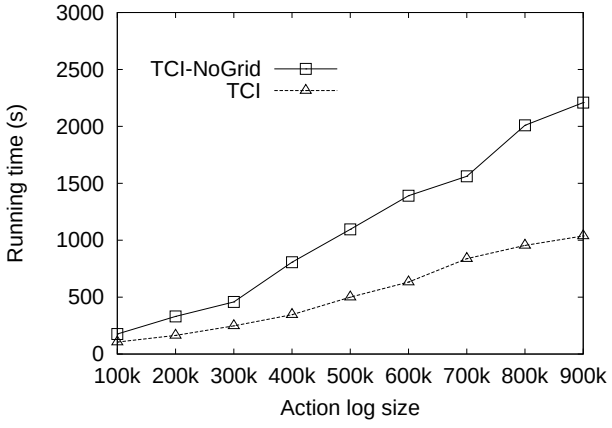
Next, we set the number of consistent influencers k to 5, action log size to 100k and vary the time threshold τ from 10 to 50. Increasing τ is equivalent to increasing the search space, i.e. the number of potential consistent influencers. Figure 5.12(a) shows that the runtime for both algorithms slightly increases as τ increases on the Citation dataset. Similar trend is observed for the Flixster dataset (see Figure 5.12(b)). However, both algorithms are sensitive on the Twitter dataset, as can be seen in Figure 5.12(c). This is because the Twitter dataset is "dense", which means in a very short time interval hundreds or thousands of tweets are posted.



(a) Citation dataset

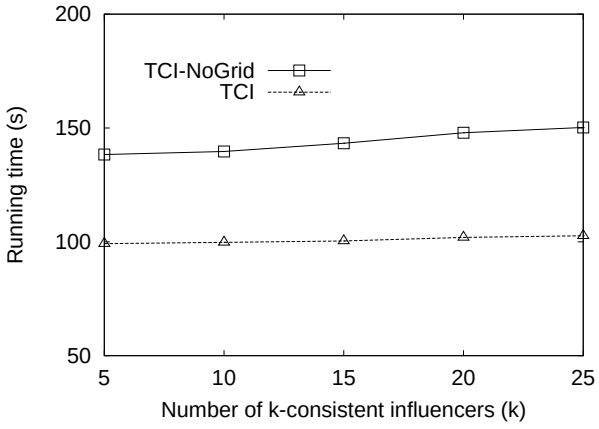


(b) Flixster dataset

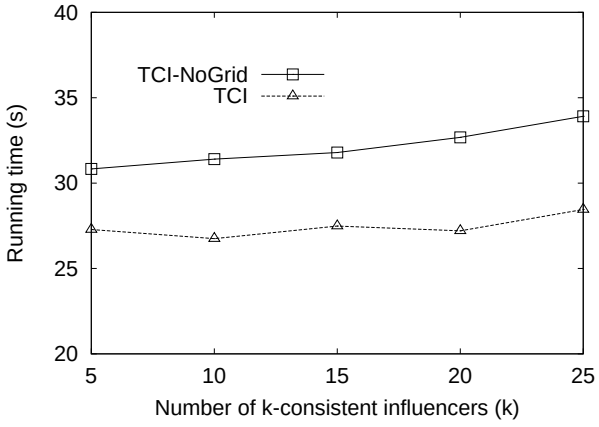


(c) Twitter dataset

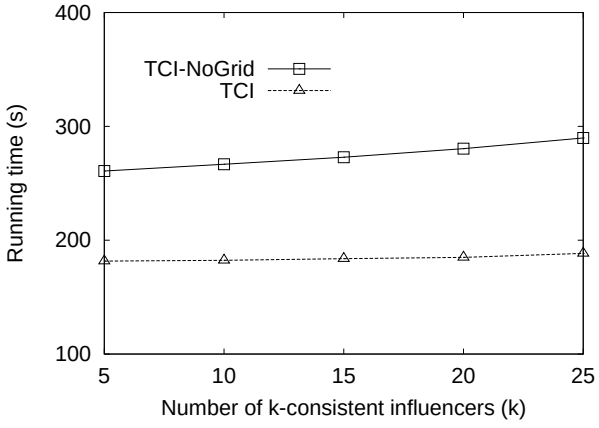
Figure 5.10: Runtime of TCI for varying action log size



(a) Citation dataset

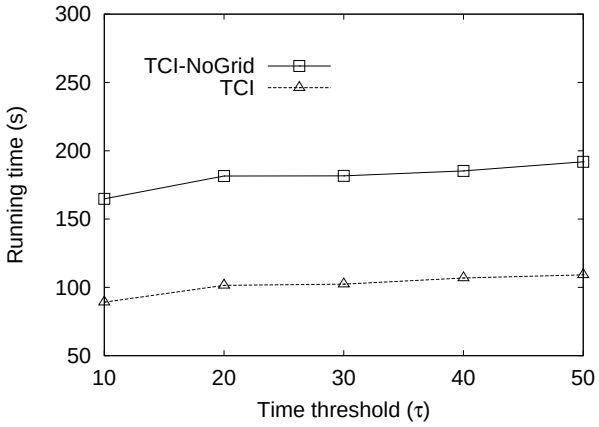


(b) Flixster dataset

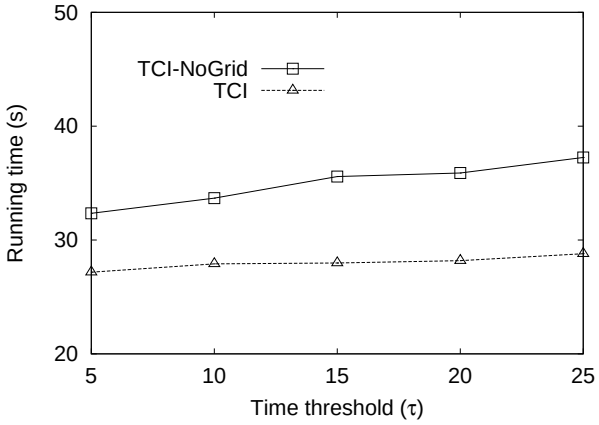


(c) Twitter dataset

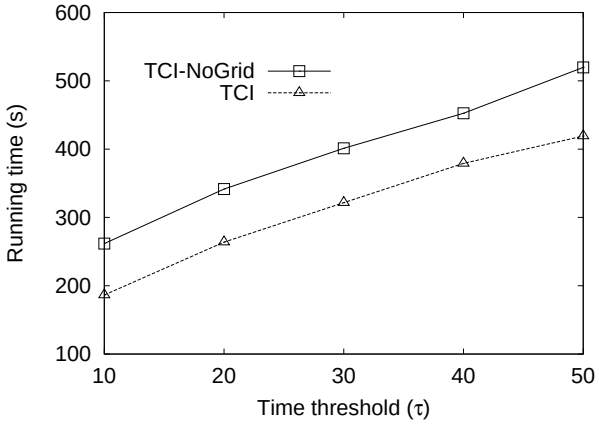
Figure 5.11: Effect of varying k



(a) Citation dataset



(b) Flixster dataset



(c) Twitter dataset

Figure 5.12: Effect of varying τ

5.4.3 Effectiveness Experiments

In this section, we demonstrate how the proposed k-consistent influencers is useful for two tasks:

1. Identifying information sources [20, 72]. Identifying information sources is useful for user recommendation. A social network user who is interested in receiving information about a particular topic would subscribe to the information sources for the same topic in order to receive up-to-date and relevant information.

2. Finding experts [11, 120]. Expert finding aims to find persons who are knowledgeable on a given topic. It has many applications in expertise search, social networks, recommendation and collaboration.

We use the Twitter dataset for the first task and the Citation dataset for the second task. We manually select the public dissemination accounts (e.g. @Yahoo) as the ground truth for Twitter dataset. For Citation dataset, we use the ground truth given in [118].

Let X be the set of ground truth, let Y be the set of predicted, then precision and recall are defined by the following equations:

$$precision = \frac{|X \cap Y|}{|Y|} \tag{5.8}$$

$$recall = \frac{|X \cap Y|}{|X|} \tag{5.9}$$

We compare the TCI algorithm with the following methods:

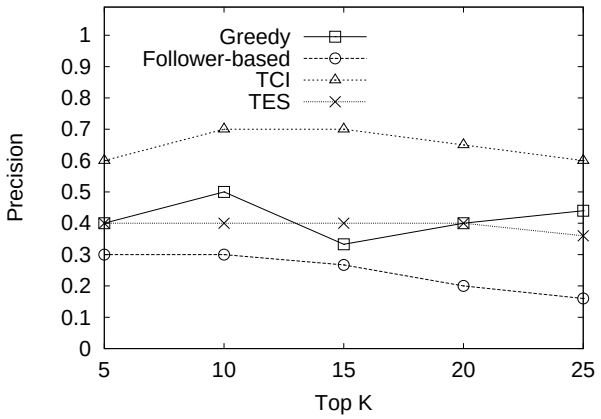
1. TES [110]. TES is designed to answer durable top-k queries. By exploiting the fact that the changes in the top-k set at adjacent time points are usually small, TES indexes these changes and incrementally computes the snapshot top-k sets at each time point of the query window.

2. Greedy [55]. The greedy algorithm finds k influential nodes such that the expected number of nodes influenced by these k nodes is maximized [64, 28, 27]. At each

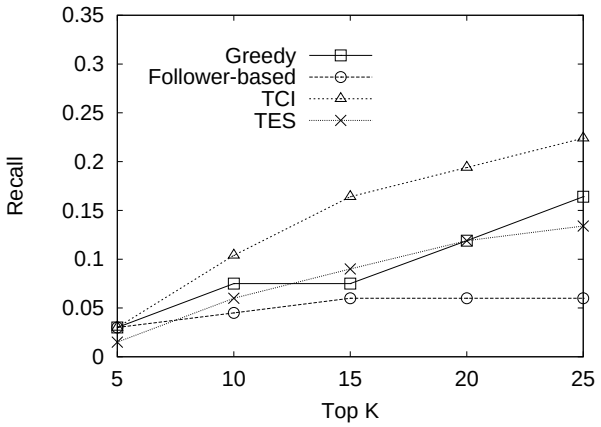
iteration, the greedy algorithm selects a node that leads to the largest increase in the number of nodes influenced. The algorithm stops when k nodes are selected.

3. Follower-based. Given the following relationships between users, the follower-based method returns the k users with the largest number of followers.

Figure 5.13 shows the precision and recall for finding information sources on Twitter dataset as we vary k from 5 to 25. We observe that the precision of TCI outperforms that of TES algorithm, the greedy algorithm and the follower-based method for all values of k. The recall for all four methods increases as k increases. Further, the gaps in recall widen as k increases. This is because all the methods will predict more information sources with the increase of k, leading to better recall.



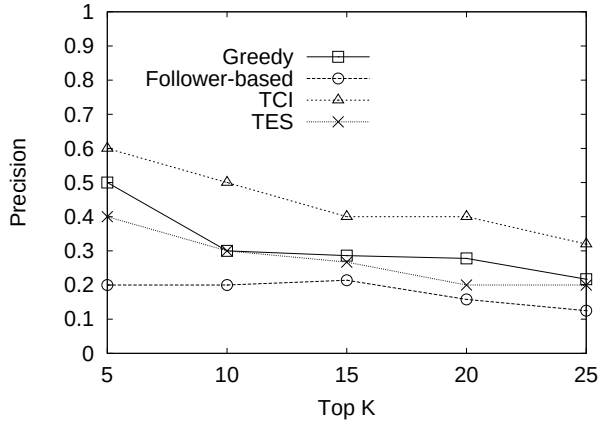
(a) Precision



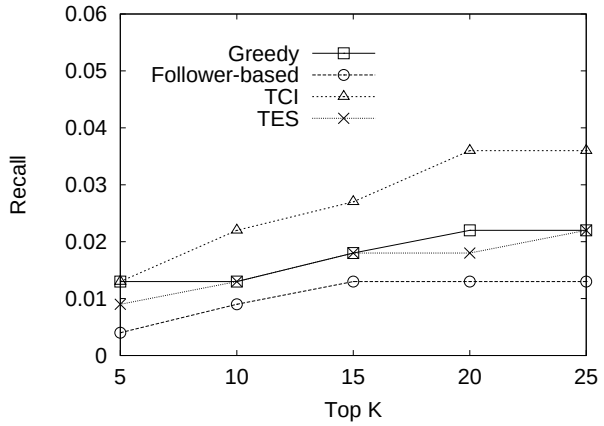
(b) Recall

Figure 5.13: Effectiveness of finding information sources on Twitter dataset

Figure 5.14 shows the precision and recall of the various methods for finding data mining experts in the Citation dataset. Again, the precision of TCI algorithm outperforms the other three methods, especially when k is large. The recall for all four methods increases as k increases, because all the methods will find more experts with larger k. Further, the gaps in recall widen as k increases. Similar results and trends are observed for information retrieval experts as shown in Figure 5.15.



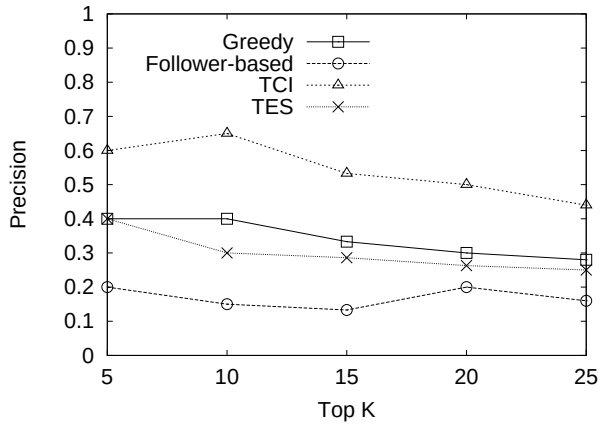
(a) Precision



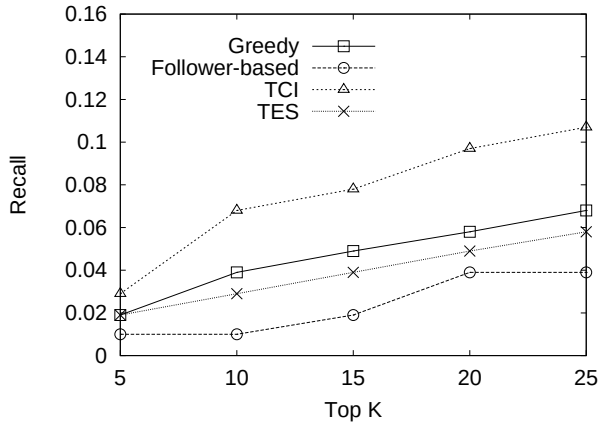
(b) Recall

Figure 5.14: Effectiveness of finding data mining experts in Citation dataset

Here we analyze why TCI outperforms the other three methods. TES algorithm is equivalent to intersect the top-k set at each time point, so the result size may be less than k. Greedy algorithm selects a node that leads to the largest increase in the number of nodes influenced at each iteration until k nodes are selected. Follower-based method returns the k users with the largest number of followers. In contrast, TCI takes into account both



(a) Precision



(b) Recall

Figure 5.15: Effectiveness of finding information retrieval experts in Citation dataset

consistency and volatility, so it is able to identify true experts.

Tables 5.2 and 5.3 show the top-5 experts on data mining and information retrieval returned by our TCI method. Among the results, some well-known authors, such as Jiawei Han and Christos Faloutsos (Data Mining), Bruce Croft and Ricardo Baeza-Yates (Information Retrieval), are all ranked among the top-5 experts. This is because these commonly ranked authors are not only highly cited, but also in the top at each time point. In our setting, high citation counts means high consistency, and high rank at each time point means little volatility. Hence, the score values of these authors are likely to be high, making them among the top-5 results.

Table 5.2: Top-5 experts on data mining

Data Mining	
consistency + volatility	consistency
Jiawei Han	Jiawei Han
Christos Faloutsos	Philip S. Yu
Philip S. Yu	Christos Faloutsos
Vipin Kumar	Mohammed J. Zaki
Mohammed J. Zaki	Rakesh Agrawal

Table 5.3: Top-5 experts on information retrieval

Information Retrieval	
consistency + volatility	consistency
Bruce Croft	Bruce Croft
Ricardo Baeza-Yates	Gerard Salton
Chengxiang Zhai	Oded Goldreich
Anil K. Jain	Michael I. Jordan
H. Garcia	Christopher D. Manning

5.5 Summary

In this chapter, we have proposed to identify top-k consistent influencers. We devise an efficient algorithm that utilizes a grid index to scan the users in the 2D personal-preference consistency space, thereby obtaining the rank of these users at a given time point. Then we design the TCI algorithm to obtain the k-consistent influencers for a given time interval. We conduct extensive experiments on three real world datasets (Citation, Flixster and Twitter) to evaluate the proposed methods. The experimental results demonstrate the effectiveness and efficiency of our methods. We show that the proposed k-consistent influencers is useful for identifying information sources and finding experts.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

Social influence plays a key role in many social networks, e.g., Facebook, Twitter and YouTube, and can benefit various applications such as viral marketing, online advertising, recommender systems, information diffusion, and experts finding. In this thesis, we have investigated three important issues in the discovery of influential nodes and influence relationships which are ignored by existing works: *influential path*, *topic-level influence* and *consistent influencer*.

First, we have focused on influential path discovery. We develop a method for inferring top-k maximal influential paths which can truly capture the dynamics of information diffusion. We propose a generative influence propagation model based on the Independent Cascade Model and Linear Threshold Model, which mathematically models the spread of certain information through a network. We design an algorithm called TIP to infer the top-k maximal influential paths. TIP utilizes the properties of top-k maximal influential paths to dynamically increase the support and prune the projected databases. In many applications, databases are updated incrementally. We also develop an incremental mining algorithm, named IncTIP, to maintain the set of top-k maximal influential paths efficiently. We evaluate the proposed algorithms on two real world datasets (MemeTracker and Twitter). The experimental results show that our algorithms are more scalable and

more efficient than the base line algorithms. In addition, influential paths can improve the precision of predicting which node will be influenced next.

Second, we have investigated topic-level influence and have taken into account the temporal factor in social influence to infer the influential strength between users at topic-level. Our approach does not require the underlying network structure to be known. We propose a guided hierarchical LDA approach to automatically identify topics without using any structural information. We then construct the topic-level social influence network incorporating the temporal factor to infer the influential strength among the users for each topic. Experimental results on two real world datasets (Twitter and MemeTracker) have demonstrated the effectiveness of our methods. Further, we show that the proposed topic-level social influence network can improve the precision of user behavior prediction and is useful for influence maximization.

Finally, we have proposed to identify k-consistent influencers. We devise an efficient algorithm that utilizes a grid index to scan the users in the 2D personal-preference consistency space, thereby obtaining the rank of these users at a given time point. Then we design the TCI algorithm to identify the k-consistent influencers for a given time interval. We conduct extensive experiments on three real world datasets (Citation, Flixster and Twitter) to evaluate the proposed methods. The experimental results demonstrate the effectiveness and efficiency of our methods. We show that the proposed k-consistent influencers is useful for identifying information sources and finding experts.

6.2 Future Work

There are several interesting directions for future work. In Chapter 3, we have focused on top-k maximal influential path discovery. We have developed a generative influence propagation model based on the Independent Cascade Model and Linear Threshold Model, which mathematically models the spread of certain information through a network. However, in the influence propagation model, we only use time difference to estimate the propagation probability; it would be more accurate if we take more informative node fea-

tures into consideration. And we will apply our TIP method to other information diffusion models.

In Chapter 4, we have investigated topic-level influence. The proposed guided hierarchical LDA typically uses Gibbs sampling for inference, a special case of Markov Chain Monte Carlo (MCMC). However, it is computationally expensive in terms of both running time and memory requirements for large datasets. First, the inference itself may take hundreds of iterations to converge. Second, the memory requirement grows linearly with data size. Therefore, it is important to scale guided hierarchical LDA for large-scale data. For future work, we will design an efficient parallel inference algorithm for guided hierarchical LDA by using a *divide-and-conquer* scheme.

In Chapter 5, we have proposed to identify top-k consistent influencers. Our TCI algorithm can identify the exact k-consistent influencers for a given time interval. However, it may not be as efficient as an approximation algorithm. For future work, we will devise an approximation algorithm to mine top-k consistent influencers. We will compare TCI algorithm with the approximation algorithm quantitatively and assess the efficiency and accuracy trade-off between the two algorithms. Another interesting direction is to deploy our TCI algorithm to the MapReduce framework.

References

[1] E. Adar, L. Adamic, L. Zhang, and R. M. Lukose. Implicit structure and the dynamics of blogspace. In *Workshop on the Weblogging Ecosystem at the 13th International World Wide Web Conference*, 2004.

[2] E. Adar and L. A. Adamic. Tracking information epidemics in blogspace. In *Web Intelligence*, pages 207–214, 2005.

[3] N. Agarwal, H. Liu, L. Tang, and P. S. Yu. Identifying the influential bloggers in a community. In *Proceedings of the international conference on Web search and web data mining*, WSDM '08, pages 207–218, 2008.

[4] A. Ahmed, L. Hong, and A. J. Smola. Hierarchical geographical modeling of user locations from social media posts. In *Proceedings of the 22nd international conference on World Wide Web*, WWW '13, pages 25–36, 2013.

[5] R. Akbarinia, E. Pacitti, and P. Valduriez. Best position algorithms for top-k queries. In *VLDB*, VLDB '07, pages 495–506, 2007.

[6] A. Anagnostopoulos, R. Kumar, and M. Mahdian. Influence and correlation in social networks. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 7–15, 2008.

[7] S. Ardon, A. Bagchi, A. Mahanti, A. Ruhela, A. Seth, R. M. Tripathy, and S. Triukose. Spatio-temporal analysis of topic popularity in twitter. *CoRR*, abs/1111.2904, 2011.

[8] J. Ayres, J. Flannick, J. Gehrke, and T. Yiu. Sequential pattern mining using a bitmap representation. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 429–435, 2002.

[9] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan. Group formation in large social networks: membership, growth, and evolution. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, pages 44–54, 2006.

[10] E. Bakshy, J. M. Hofman, W. A. Mason, and D. J. Watts. Everyone's an influencer: quantifying influence on twitter. In *Proceedings of the fourth ACM international conference on Web search and data mining*, WSDM '11, pages 65–74, 2011.

[11] K. Balog, L. Azzopardi, and M. de Rijke. Formal models for expert finding in enterprise corpora. In *SIGIR*, SIGIR '06, pages 43–50, 2006.

[12] F. Bass. A new product growth for model consumer durables. *Management Sciences*, 15(1):215–227, 1969.

[13] H. Bast, D. Majumdar, R. Schenkel, M. Theobald, and G. Weikum. Io-top-k: index-access optimized top-k query processing. In *VLDB*, VLDB '06, pages 475–486, 2006.

[14] S. Bharathi, D. Kempe, and M. Salek. Competitive influence maximization in social networks. In *Proceedings of the 3rd international conference on Internet and network economics*, WINE'07, pages 306–311, 2007.

[15] D. M. Blei, T. L. Griffiths, and M. I. Jordan. The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies. *J. ACM*, 57(2):7:1–7:30, 2010.

[16] D. M. Blei, T. L. Griffiths, M. I. Jordan, and J. B. Tenenbaum. Hierarchical topic models and the nested chinese restaurant process. In *NIPS*, 2003.

[17] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.

[18] F. Bonchi. Influence propagation in social networks: A data mining perspective. *IEEE Intelligent Informatics Bulletin*, 12(1):8–16, 2011.

[19] C. Budak, D. Agrawal, and A. El Abbadi. Limiting the spread of misinformation in social networks. In *Proceedings of the 20th international conference on World wide web*, WWW '11, pages 665–674, 2011.

[20] K. R. Canini, B. Suh, and P. L. Pirolli. Finding credible information sources in social networks based on content and social structure. In *Proceedings of the third IEEE International Conference on Social Computing (SocialCom)*, 2011.

[21] P. Cao and Z. Wang. Efficient top-k query calculation in distributed networks. In *PODC*, PODC '04, pages 206–215, 2004.

[22] M. Cha, H. Haddadi, F. Benevenuto, and K. Gummadi. Measuring user influence in twitter: The million follower fallacy. In *4th International AAAI Conference on Weblogs and Social Media (ICWSM)*, 2010.

[23] Y. Cha and J. Cho. Social-network analysis using topic models. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '12, pages 565–574, 2012.

[24] K. Chakrabarti, S. Chaudhuri, and V. Ganti. Interval-based pruning for top-k processing over compressed lists. In *ICDE*, pages 709–720, 2011.

[25] W. Chen, A. Collins, R. Cummings, T. Ke, Z. Liu, D. Rincon, X. Sun, Y. Wang, W. Wei, and Y. Yuan. Influence maximization in social networks when negative opinions may emerge and propagate. Technical Report MSR-TR-2010-137, Microsoft Research, October 2010.

[26] W. Chen, W. Lu, and N. Zhang. Time-critical influence maximization in social networks with time-delayed diffusion process. In *AAAI*, 2012.

[27] W. Chen, C. Wang, and Y. Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '10, pages 1029–1038, 2010.

[28] W. Chen, Y. Wang, and S. Yang. Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 199–208, 2009.

[29] W. Chen, Y. Yuan, and L. Zhang. Scalable influence maximization in social networks under the linear threshold model. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, ICDM '10, pages 88–97, 2010.

[30] Y. Chen, J. Guo, Y. Wang, Y. Xiong, and Y. Zhu. Incremental mining of sequential patterns using prefix tree. In *Proceedings of the 11th Pacific-Asia conference on Advances in knowledge discovery and data mining*, PAKDD'07, pages 433–440, 2007.

[31] H. Cheng, X. Yan, and J. Han. Incspan: incremental mining of sequential patterns in large database. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '04, pages 527–532, 2004.

[32] J. S. Coleman, E. Katz, and H. Menzel. *Medical innovation: A diffusion study*. Bobbs-Merrill, 1966.

[33] D. Crandall, D. Cosley, D. Huttenlocher, J. Kleinberg, and S. Suri. Feedback effects between similarity and social influence in online communities. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 160–168, 2008.

[34] G. Das, D. Gunopulos, N. Koudas, and D. Tsirogiannis. Answering top-k queries using views. In *VLDB*, VLDB '06, pages 451–462, 2006.

[35] C. Dimopoulos, S. Nepomnyachiy, and T. Suel. Optimizing top-k document retrieval strategies for block-max indexes. In *WSDM*, WSDM '13, pages 113–122, 2013.

[36] S. Ding and T. Suel. Faster top-k document retrieval using block-max indexes. In *SIGIR*, SIGIR '11, pages 993–1002, 2011.

[37] P. Domingos and M. Richardson. Mining the network value of customers. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '01, pages 57–66, 2001.

[38] W. Dong and A. Pentland. Modeling influence between experts. In *Proceedings of the ICMI 2006 and IJCAI 2007 international conference on Artifical intelligence for human computing*, ICMI'06/IJCAI'07, pages 170–189, 2007.

[39] E. Even-Dar and A. Shapira. A note on maximizing the spread of influence in social networks. In *Proceedings of the 3rd international conference on Internet and network economics*, WINE'07, pages 281–286, 2007.

[40] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. In *PODS*, PODS '01, pages 102–113, 2001.

[41] F. Giannotti, M. Nanni, D. Pedreschi, and F. Pinelli. Mining sequences with temporal annotations. In *Proceedings of the 2006 ACM symposium on Applied computing*, SAC '06, pages 593–597, 2006.

[42] J. Goldenberg and B. Libai. Using complex systems analysis to advance marketing theory development: Modeling heterogeneity effects on new product growth through stochastic cellular automata. *Academy of Marketing Science Review*, 2001.

[43] J. Goldenberg, B. Libai, and E. Muller. Talk of the network: a complex systems look at the underlying process of word-of-mouth. *Marketing Letters*, pages 211–223, 2001.

[44] M. Gomez-Rodriguez, J. Leskovec, and A. Krause. Inferring networks of diffusion and influence. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '10, pages 1019–1028, 2010.

[45] A. Goyal, F. Bonchi, and L. V. Lakshmanan. Learning influence probabilities in social networks. In *Proceedings of the third ACM international conference on Web search and data mining*, WSDM '10, pages 241–250, 2010.

[46] A. Goyal, F. Bonchi, and L. V. S. Lakshmanan. A data-based approach to social influence maximization. *Proc. VLDB Endow.*, 5(1):73–84, 2011.

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

[48] D. Gruhl, R. Guha, D. Liben-nowell, and A. Tomkins. Information diffusion through blogspace. In *WWW '04*, pages 491–501, 2004.

[49] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M.-C. Hsu. Freespan: frequent pattern-projected sequential pattern mining. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '00, pages 355–359, 2000.

[50] J. Han, J. Wang, Y. Lu, and P. Tzvetkov. Mining top-k frequent closed patterns without minimum support. In *Proceedings of the 2002 IEEE International Conference on Data Mining*, ICDM '02, pages 211–218, 2002.

[51] M. Jamali and M. Ester. A matrix factorization technique with trust propagation for recommendation in social networks. In *RecSys*, RecSys '10, pages 135–142, 2010.

[52] A. Java, P. Kolari, T. Finin, and T. Oates. Modeling the spread of influence on the blogosphere. In *World Wide Web Conference Series*, 2006.

[53] J. Jestes, J. M. Phillips, F. Li, and M. Tang. Ranking large temporal data. *Proc. VLDB Endow.*, 5(11):1412–1423, 2012.

[54] S. Jurvetson. What exactly is viral marketing? *Red Herring*, 78:110–112, 2000.

[55] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '03, pages 137–146, 2003.

[56] D. Kempe, J. M. Kleinberg, and É. Tardos. Influential nodes in a diffusion model for social networks. In *ICALP*, pages 1127–1138, 2005.

[57] M. Kimura and K. Saito. Tractable models for information diffusion in social networks. In *Principles of Data Mining and Knowledge Discovery*, pages 259–271, 2006.

[58] M. Kimura, K. Saito, and R. Nakano. Extracting influential nodes for information diffusion on a social network. In *National Conference on Artificial Intelligence*, pages 1371–1376, 2007.

[59] J. Kleinberg. Cascading behavior in networks: Algorithmic and economic issues. In *Algorithmic Game Theory*, pages 613–632, 2007.

[60] H. Kwak, C. Lee, H. Park, and S. Moon. What is twitter, a social network or a news media? In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 591–600, 2010.

[61] M. L. Lee, W. Hsu, L. Li, and W. H. Tok. Consistent top-k queries over time. In *DASFAA*, DASFAA '09, pages 51–65, 2009.

[62] J. Leskovec, L. A. Adamic, and B. A. Huberman. The dynamics of viral marketing. In *Proceedings of the 7th ACM conference on Electronic commerce*, EC '06, pages 228–237, 2006.

[63] J. Leskovec, L. Backstrom, and J. Kleinberg. Meme-tracking and the dynamics of the news cycle. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 497–506, 2009.

[64] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '07, pages 420–429, 2007.

[65] J. Leskovec, M. Mcglohon, C. Faloutsos, N. Glance, and M. Hurst. Cascading behavior in large blog graphs. In *SDM*, 2007.

[66] G. Li, S. Chen, J. Feng, K.-L. Tan, and W.-S. Li. Efficient location-aware influence maximization. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 87–98, 2014.

[67] B. Liu, G. Cong, D. Xu, and Y. Zeng. Time constrained influence maximization in social networks. In *ICDM*, pages 439–448, 2012.

[68] J. Liu, S. Yan, Y. Wang, and J. Ren. Incremental mining algorithm of sequential patterns based on sequence tree. In *Advances in Intelligent Systems*, volume 138 of *Advances in Intelligent and Soft Computing*, pages 61–67, 2012.

[69] L. Liu, J. Tang, J. Han, M. Jiang, and S. Yang. Mining topic-level influence in heterogeneous networks. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, CIKM '10, pages 199–208, 2010.

[70] S. Liu, L. Chen, L. M. Ni, and J. Fan. Cim: Categorical influence maximization. In *Proceedings of the 5th International Conference on Ubiquitous Information Management and Communication*, ICUIMC '11, pages 124:1–124:10, 2011.

[71] V. Mahajan, E. Muller, and F. M. Bass. New product diffusion models in marketing: A review and directions for research. *Journal of Marketing*, 54(1):1–26, 1990.

[72] D. Mahata and N. Agarwal. What does everybody know? identifying event-specific sources from social media. In *CASoN*, pages 63–68, 2012.

[73] C. D. Manning and H. Schütze. *Foundations of statistical natural language processing*. MIT Press, 1999.

[74] F. Masseglia, P. Poncelet, and M. Teisseire. Incremental mining of sequential patterns in large databases. *Data Knowl. Eng.*, 46(1):97–121, 2003.

[75] M. Mathioudakis, F. Bonchi, C. Castillo, A. Gionis, and A. Ukkonen. Sparsification of influence networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '11, pages 529–537, 2011.

[76] Y. Matsubara, Y. Sakurai, B. A. Prakash, L. Li, and C. Faloutsos. Rise and fall patterns of information diffusion: model and implications. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '12, pages 6–14, 2012.

[77] S. Michel, P. Triantafillou, and G. Weikum. Klee: a framework for distributed top-k query algorithms. In *VLDB*, VLDB '05, pages 637–648, 2005.

[78] K. Mouratidis, S. Bakiras, and D. Papadias. Continuous monitoring of top-k queries over sliding windows. In *SIGMOD*, SIGMOD '06, pages 635–646, 2006.

[79] R. Narayanam and Y. Narahari. A shapley value-based approach to discover influential nodes in social networks. *IEEE T. Automation Science and Engineering*, 8(1):130–147, 2011.

[80] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions-i. *Mathematical Programming*, 14:265–294, 1978.

[81] S. N. Nguyen, X. Sun, and M. E. Orlowska. Improvements of incspan: incremental mining of sequential patterns in large database. In *Proceedings of the 9th Pacific-Asia conference on Advances in Knowledge Discovery and Data Mining*, PAKDD'05, pages 442–451, 2005.

[82] S. Parthasarathy, M. J. Zaki, M. Ogihara, and S. Dwarkadas. Incremental and interactive sequence mining. In *Proceedings of the eighth international conference on Information and knowledge management*, CIKM '99, pages 251–258, 1999.

[83] J. Pei, J. Han, B. Mortazavi-asl, H. Pinto, Q. Chen, U. Dayal, and M. chun Hsu. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *ICDE*, pages 215–224, 2001.

[84] D. Ramage, S. Dumais, and D. Liebling. Characterizing microblogs with topic models. In *ICWSM*, 2010.

[85] D. Ramage, D. Hall, R. Nallapati, and C. D. Manning. Labeled lda: a supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, EMNLP '09, pages 248–256, 2009.

[86] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '02, pages 61–70, 2002.

[87] E. Rogers. *Diffusion of Innovations, Fourth Edition*. Free Press, 1995.

[88] D. M. Romero, W. Galuba, S. Asur, and B. A. Huberman. Influence and passivity in social media. In *Proceedings of the 20th international conference companion on World wide web*, WWW '11, pages 113–114, 2011.

[89] M. Rosen-Zvi, T. Griffiths, M. Steyvers, and P. Smyth. The author-topic model for authors and documents. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, UAI '04, pages 487–494, 2004.

[90] K. Saito, M. Kimura, K. Ohara, and H. Motoda. Learning continuous-time information diffusion model for social behavioral data analysis. In *Proceedings of the 1st Asian Conference on Machine Learning: Advances in Machine Learning*, ACML '09, pages 322–337, 2009.

[91] K. Saito, M. Kimura, K. Ohara, and H. Motoda. Selecting information diffusion models over social networks for behavioral analysis. In *Proceedings of the 2010 European conference on Machine learning and knowledge discovery in databases: Part III*, ECML PKDD'10, pages 180–195, 2010.

[92] K. Saito, R. Nakano, and M. Kimura. Prediction of information diffusion probabilities for independent cascade model. In *Proceedings of the 12th international conference on Knowledge-Based Intelligent Information and Engineering Systems, Part III*, KES '08, pages 67–75, 2008.

[93] K. Saito, K. Ohara, Y. Yamagishi, M. Kimura, and H. Motoda. Learning diffusion probability based on node attributes in social networks. In *Proceedings of the 19th international conference on Foundations of intelligent systems*, ISMIS'11, pages 153–162, 2011.

[94] T. C. Schelling. *Micromotives and Macrobehavior*. Norton, 1978.

[95] D. Shan, S. Ding, J. He, H. Yan, and X. Li. Optimized top-k processing with global page scores on block-max indexes. In *WSDM*, WSDM '12, pages 423–432, 2012.

[96] S. Shang, P. Hui, S. R. Kulkarni, and P. W. Cuff. Wisdom of the crowd: Incorporating social influence in recommendation models. In *Proceedings of the 2011 IEEE 17th International Conference on Parallel and Distributed Systems*, ICPADS '11, pages 835–840, 2011.

[97] P. Singla and M. Richardson. Yes, there is a correlation: - from social networks to personal behavior on the web. In *Proceedings of the 17th international conference on World Wide Web*, WWW '08, pages 655–664, 2008.

[98] X. Song, Y. Chi, K. Hino, and B. L. Tseng. Information flow modeling based on diffusion rate for prediction and ranking. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 191–200, 2007.

[99] X. Song, B. L. Tseng, C.-Y. Lin, and M.-T. Sun. Personalized recommendation driven by information flow. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '06, pages 509–516, 2006.

[100] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *Proceedings of the 5th International Conference on Extending Database Technology: Advances in Database Technology*, EDBT '96, pages 3–17, 1996.

[101] N. R. Suri and Y. Narahari. Determining the top-k nodes in social networks using the shapley value. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems - Volume 3*, AAMAS '08, pages 1509–1512, 2008.

[102] J. Tang, J. Sun, C. Wang, and Z. Yang. Social influence analysis in large-scale networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 807–816, 2009.

[103] J. Tang, S. Wu, and J. Sun. Confluence: conformity influence in large social networks. In *KDD*, pages 347–355, 2013.

[104] J. Tang, J. Zhang, R. Jin, Z. Yang, K. Cai, L. Zhang, and Z. Su. Topic level expertise search over heterogeneous networks. *Machine Learning Journal*, 82(2):211–237, 2011.

[105] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su. Arnetminer: Extraction and mining of academic social networks. In *KDD*, pages 990–998, 2008.

[106] P. Tzvetkov, X. Yan, and J. Han. Tsp: Mining top-k closed sequential patterns. In *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM '03)*, pages 347–354, 2003.

[107] T. Valente. *Network models of the diffusion of innovations*. Quantitative methods in communication. Hampton Press, 1995.

[108] H. Wallach, D. Mimno, and A. McCallum. Rethinking lda: Why priors matter. In *Advances in Neural Information Processing Systems 22*, pages 1973–1981. 2009.

[109] C. Wang, J. Tang, J. Sun, and J. Han. Dynamic social influence analysis through time-dependent factor graphs. In *Proceedings of the 2011 International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '11, pages 239–246, 2011.

[110] H. Wang, Y. Cai, Y. Yang, S. Zhang, and N. Mamoulis. Durable queries over historical time series. *IEEE TKDE*, 26(3), 2014.

[111] J. Wang and J. Han. BIDE: Efficient mining of frequent closed sequences. In *Proceedings of the 20th International Conference on Data Engineering*, pages 79–90, 2004.

[112] Y. Wang, G. Cong, G. Song, and K. Xie. Community-based greedy algorithm for mining top-k influential nodes in mobile social networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '10, pages 1039–1048, 2010.

[113] J. Weng, E.-P. Lim, J. Jiang, and Q. He. Twitterrank: finding topic-sensitive influential twitterers. In *Proceedings of the third ACM international conference on Web search and data mining*, WSDM '10, pages 261–270, 2010.

[114] X. Yan, J. Han, and R. Afshar. Clospan: Mining closed sequential patterns in large datasets. In *SDM*, pages 166–177, 2003.

[115] J. Yang and J. Leskovec. Modeling information diffusion in implicit networks. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, ICDM '10, pages 599–608, 2010.

[116] J. Yang and J. Leskovec. Patterns of temporal variation in online media. In *Proceedings of the fourth ACM international conference on Web search and data mining*, WSDM '11, pages 177–186, 2011.

[117] M. J. Zaki. Spade: An efficient algorithm for mining frequent sequences. In *Machine Learning*, pages 31–60, 2001.

[118] J. Zhang, J. Tang, and J.-Z. Li. Expert finding in a social network. In *DASFAA*, pages 1066–1069, 2007.

[119] M. Zhang, B. Kao, D. W.-L. Cheung, and C. L. Yip. Efficient algorithms for incremental update of frequent sequences. In *Proceedings of the 6th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, PAKDD '02, pages 186–197, 2002.

[120] J. Zhu, D. Song, S. Rüger, and X. Huang. Modeling document features for expert finding. In *CIKM*, CIKM '08, pages 1421–1422, 2008.