Information Diffusion in Online Social Network

BY

SHUYANG LIN
B.E., Computer Science, Tsinghua University, 2010
M.S., Mathematics, University of Illinois at Chicago, 2013

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Chicago, 2015

Chicago, Illinois

Defense Committee:

Philip S. Yu, Chair and Advisor
Bing Liu, Professor at Computer Science Dept.
Brian Ziebart, Assistant Professor at Computer Science Dept.
Jing Wang, Associate Professor at Dept. of Math, Statistics, and Computer Science
Kunpeng Zhang, Assistant Professor at Dept. of Information and Decision Sciences
This thesis is dedicated to my husband Yilei Yang.
ACKNOWLEDGMENTS

First and foremost, I would like to thank my Ph.D. advisor Professor Philip S. Yu. I owe much of my growth in these five years as a scholar and as a person to his guidance. I deeply appreciate his mentoring and advising along the way, as well as his patience and kindness. Were it not for him, this dissertation would not have been possible.

I would like to thank Professor Bing Liu, Professor Jing Wang, Professor Brian Ziebart, and Professor Kunpeng Zhang for taking their valuable time to serve on my dissertation committee and for their constructive suggestions and feedbacks.

I also would like to thank my colleagues and friends that I met in UIC. They have taught me a lot through discussions and collaborations. I am really grateful to work with these great people and truly appreciate countless enjoyable moments that we have had together.

Finally, I am would like to thank my family. I truly indebted to my parents for their unconditional trust and love. I deeply appreciate my husband Yilei for his continued support, and for his weekly commute between New York and Chicago for years.

SL
CONTRIBUTION OF AUTHORS

Chapter 2 presents a published manuscript (39), for which I was the primary author. Fengjiao Wang, Qingbo Hu, and my advisor Professor Philip S. Yu contributed to revising the manuscript and discussions with respect to the work. Chapter 3 presents a published manuscript (37) and its extended version (38), for which I was the primary author. My advisor Professor Philip S. Yu contributed to revising the manuscript and discussions with respect to the work, and Xiangnan Kong contributed to revising the manuscript. Chapter 4 presents a published manuscript (35), for which I was the primary author. Qingbo Hu and my advisor Professor Philip S. Yu contributed to revising the manuscript and discussions with respect to the work, and Fengjiao Wang contributed to revising the manuscript. Chapter 5 presents published manuscript (36), for which I was the primary author. Qingbo Hu, Guan Wang and my advisor Professor Philip S. Yu contributed to revising the manuscript and discussions with respect to the work.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>CHAPTER</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>INTRODUCTION</td>
</tr>
<tr>
<td>1.1</td>
<td>Dissertation Framework</td>
</tr>
<tr>
<td>1.2</td>
<td>Learning Information Diffusion Models with Social Events</td>
</tr>
<tr>
<td>1.3</td>
<td>Predicting Trends with Information Diffusion Model</td>
</tr>
<tr>
<td>1.4</td>
<td>Steering Information Diffusion Dynamically</td>
</tr>
<tr>
<td>1.5</td>
<td>Efficient Influence Maximization with Community Effects</td>
</tr>
<tr>
<td>2</td>
<td>LEARNING INFORMATION DIFFUSION MODELS WITH SOCIAL EVENTS</td>
</tr>
<tr>
<td>2.1</td>
<td>Introduction</td>
</tr>
<tr>
<td>2.2</td>
<td>Problem Formulation</td>
</tr>
<tr>
<td>2.3</td>
<td>Proposed Model</td>
</tr>
<tr>
<td>2.3.1</td>
<td>The framework of LADP Model</td>
</tr>
<tr>
<td>2.3.2</td>
<td>Information Diffusion Model</td>
</tr>
<tr>
<td>2.3.3</td>
<td>External Trend Model</td>
</tr>
<tr>
<td>2.4</td>
<td>Parameters Estimation</td>
</tr>
<tr>
<td>2.5</td>
<td>Experiment</td>
</tr>
<tr>
<td>2.5.1</td>
<td>Baselines</td>
</tr>
<tr>
<td>2.5.2</td>
<td>Datasets</td>
</tr>
<tr>
<td>2.5.3</td>
<td>Experiment with Twitter Datasets</td>
</tr>
<tr>
<td>2.5.3.1</td>
<td>Experiment with Semi-synthetic Dataset</td>
</tr>
<tr>
<td>2.5.3.2</td>
<td>Experiment with Twitter-UIC dataset</td>
</tr>
<tr>
<td>2.5.3.3</td>
<td>Case Study</td>
</tr>
<tr>
<td>2.5.4</td>
<td>Experiment with DBLP Datasets</td>
</tr>
<tr>
<td>2.6</td>
<td>Related Work</td>
</tr>
<tr>
<td>3</td>
<td>PREDICTING TRENDS WITH INFORMATION DIFFUSION MODEL</td>
</tr>
<tr>
<td>3.1</td>
<td>Introduction</td>
</tr>
<tr>
<td>3.2</td>
<td>Preliminaries</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Notations and Definitions</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Datasets</td>
</tr>
<tr>
<td>3.3</td>
<td>Dynamic Activeness (DA) Model</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Concept of Activeness</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Framework of DA Model</td>
</tr>
<tr>
<td>3.3.3</td>
<td>Activeness Modeling</td>
</tr>
<tr>
<td>3.3.3.1</td>
<td>Activeness Propagation</td>
</tr>
<tr>
<td>3.3.3.2</td>
<td>Decay of activeness</td>
</tr>
<tr>
<td>3.3.3.3</td>
<td>Summary of Activeness Model</td>
</tr>
</tbody>
</table>
TABLE OF CONTENTS (Continued)

<table>
<thead>
<tr>
<th>CHAPTER</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3.4</td>
<td>Action Generating Process</td>
</tr>
<tr>
<td>3.4</td>
<td>Prediction Algorithm</td>
</tr>
<tr>
<td>3.4.1</td>
<td>Parameter Learning Phase</td>
</tr>
<tr>
<td>3.4.2</td>
<td>Prediction Phase</td>
</tr>
<tr>
<td>3.4.3</td>
<td>Efficient Implementation</td>
</tr>
<tr>
<td>3.4.3.1</td>
<td>Speedup of Parameter Learning Phase</td>
</tr>
<tr>
<td>3.4.3.2</td>
<td>Efficient Implementation of Prediction Phase</td>
</tr>
<tr>
<td>3.4.3.3</td>
<td>Time complexity</td>
</tr>
<tr>
<td>3.5</td>
<td>Experiment</td>
</tr>
<tr>
<td>3.5.1</td>
<td>Algorithms and Performance Measures</td>
</tr>
<tr>
<td>3.5.2</td>
<td>Trend Data for Evaluation</td>
</tr>
<tr>
<td>3.5.3</td>
<td>Coverage</td>
</tr>
<tr>
<td>3.5.4</td>
<td>Intensity</td>
</tr>
<tr>
<td>3.5.5</td>
<td>Duration</td>
</tr>
<tr>
<td>3.5.6</td>
<td>Case Study</td>
</tr>
<tr>
<td>3.5.7</td>
<td>Variations of Parameters</td>
</tr>
<tr>
<td>3.5.8</td>
<td>Summary and Discussion</td>
</tr>
<tr>
<td>3.6</td>
<td>Related Work</td>
</tr>
<tr>
<td>4</td>
<td>STEERING INFORMATION DIFFUSION DYNAMICALLY</td>
</tr>
<tr>
<td>4.1</td>
<td>Introduction</td>
</tr>
<tr>
<td>4.2</td>
<td>Motivation</td>
</tr>
<tr>
<td>4.3</td>
<td>Push-driven cascade model</td>
</tr>
<tr>
<td>4.4</td>
<td>Dynamic influence maximization problem</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Problem definition</td>
</tr>
<tr>
<td>4.4.2</td>
<td>AO* optimal search algorithm</td>
</tr>
<tr>
<td>4.4.3</td>
<td>Online search algorithm</td>
</tr>
<tr>
<td>4.5</td>
<td>Experiment</td>
</tr>
<tr>
<td>4.5.1</td>
<td>Experiment Setup</td>
</tr>
<tr>
<td>4.5.2</td>
<td>Results</td>
</tr>
<tr>
<td>4.6</td>
<td>Related Work</td>
</tr>
<tr>
<td>5</td>
<td>EFFICIENT INFLUENCE MAXIMIZATION WITH COMMUNITY EFFECTS</td>
</tr>
<tr>
<td>5.1</td>
<td>Introduction</td>
</tr>
<tr>
<td>5.2</td>
<td>Related work</td>
</tr>
<tr>
<td>5.3</td>
<td>Preliminary</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Notations</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Datasets</td>
</tr>
<tr>
<td>5.4</td>
<td>Observations</td>
</tr>
<tr>
<td>5.4.1</td>
<td>Identifying communities for information diffusion</td>
</tr>
<tr>
<td>5.4.2</td>
<td>Action homophily of communities</td>
</tr>
<tr>
<td>CHAPTER</td>
<td>PAGE</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>5.4.3 Role-based homophily of communities</td>
<td>111</td>
</tr>
<tr>
<td>5.5 Community-based Fast Influence Model</td>
<td>113</td>
</tr>
<tr>
<td>5.5.1 Influence decoupling</td>
<td>113</td>
</tr>
<tr>
<td>5.5.2 Identifying communities</td>
<td>115</td>
</tr>
<tr>
<td>5.5.3 CFI-based influence maximization algorithm</td>
<td>118</td>
</tr>
<tr>
<td>5.6 Experiment</td>
<td>122</td>
</tr>
<tr>
<td>5.6.1 Experiment setup</td>
<td>122</td>
</tr>
<tr>
<td>5.6.2 Results</td>
<td>123</td>
</tr>
<tr>
<td>6 CONCLUSIONS AND CONTRIBUTIONS</td>
<td>127</td>
</tr>
<tr>
<td>CITED LITERATURE</td>
<td>129</td>
</tr>
<tr>
<td>VITA</td>
<td>134</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>TABLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>NOTATIONS .................................................. 13</td>
</tr>
<tr>
<td>II</td>
<td>CONFERENCE LISTS OF TWO COMMUNITIES IN DBLP DATASET . 25</td>
</tr>
<tr>
<td>III</td>
<td>TOP EVENTS AND TOP SOCIAL EVENTS IDENTIFIED BY LADP AND MYERS’S IN UIC TWITTER NETWORK . ................. 30</td>
</tr>
<tr>
<td>IV</td>
<td>TOP EVENTS AND TOP SOCIAL EVENTS IDENTIFIED BY LADP AND MYERS’S IN DATA MINING COMMUNITY ................. 33</td>
</tr>
<tr>
<td>V</td>
<td>TRENDS IN THE DBLP AND TWITTER DATASET ................. 58</td>
</tr>
<tr>
<td>VI</td>
<td>ACCURACY OF DURATION PREDICTION ......................... 63</td>
</tr>
<tr>
<td>VII</td>
<td>RESULTS OF DIFFERENT USER BEHAVIOR MODELS ............ 76</td>
</tr>
<tr>
<td>VIII</td>
<td>TWO SETS OF COMMUNITIES FOR EACH NETWORK ............. 109</td>
</tr>
</tbody>
</table>
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>FIGURE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>2</td>
<td>27</td>
</tr>
<tr>
<td>3</td>
<td>28</td>
</tr>
<tr>
<td>4</td>
<td>31</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
</tr>
<tr>
<td>6</td>
<td>32</td>
</tr>
<tr>
<td>7</td>
<td>37</td>
</tr>
<tr>
<td>8</td>
<td>43</td>
</tr>
<tr>
<td>9</td>
<td>45</td>
</tr>
<tr>
<td>10</td>
<td>45</td>
</tr>
<tr>
<td>11</td>
<td>49</td>
</tr>
<tr>
<td>12</td>
<td>58</td>
</tr>
<tr>
<td>13</td>
<td>59</td>
</tr>
<tr>
<td>14</td>
<td>60</td>
</tr>
<tr>
<td>15</td>
<td>61</td>
</tr>
<tr>
<td>16</td>
<td>62</td>
</tr>
<tr>
<td>17</td>
<td>63</td>
</tr>
<tr>
<td>18</td>
<td>65</td>
</tr>
<tr>
<td>FIGURE</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>19</td>
<td>Distribution of Parameters</td>
</tr>
<tr>
<td>20</td>
<td>Relation between the “show number” and the number of retweets</td>
</tr>
<tr>
<td>21</td>
<td>An example for hypergraph of augmented states. The red-colored subgraph indicates a solution graph. The nodes with dashed outline are augmented states that cannot be reached from ((s_0, 0)). The 2-connectors to and from these nodes are omitted.</td>
</tr>
<tr>
<td>22</td>
<td>Running time for small budget</td>
</tr>
<tr>
<td>23</td>
<td>Average influence spread for small budget</td>
</tr>
<tr>
<td>24</td>
<td>Average influence spread for large budget</td>
</tr>
<tr>
<td>25</td>
<td>Running time of OnlineSearch on different datasets</td>
</tr>
<tr>
<td>26</td>
<td>Variation of (q(u_t)) and (w(u_t))</td>
</tr>
<tr>
<td>27</td>
<td>Average influence spread with varying (\theta)</td>
</tr>
<tr>
<td>28</td>
<td>Distribution of similarity between actions of user pairs</td>
</tr>
<tr>
<td>29</td>
<td>Distribution of similarity between influencer feature vectors of user pairs</td>
</tr>
<tr>
<td>30</td>
<td>Distribution of similarity between influencee feature vectors of user pairs</td>
</tr>
<tr>
<td>31</td>
<td>Inference of the CFI model</td>
</tr>
<tr>
<td>32</td>
<td>Influence of different sizes of seed set</td>
</tr>
<tr>
<td>33</td>
<td>Running time with different sizes of seed set</td>
</tr>
<tr>
<td>34</td>
<td>Effects of (\theta)</td>
</tr>
</tbody>
</table>
Recently, online social networks have become increasingly important media. From online social networks, people get all kinds of information, from popular restaurants in town, to breaking news from the other side of the world. Unlike traditional media that hold a one-to-all communication paradigm, social networks propagate information via word-of-mouth communication between users. The flourishing of social networks as new media have brought numerous applications for the researches on information diffusion. For example, viral marketing campaigns in social networks can benefit from an accurate model of information diffusion. Nevertheless, many research efforts still need to be made to fully understand the diffusion of information in online social networks.

In this thesis, we focus on information diffusion in online social networks. We study real-world diffusion data from online social networks such as Twitter, Foursquare, and Slashdot to get data-driven observations. These observations lead us to rethink some key problems with regard to information diffusion: diffusion modeling, trend predicting, and influence maximization.

First, we observe the effect of external influence on the diffusion of information, and propose a method to improve the accuracy of learning information diffusion models by excluding interferences from the external world (39). Second, we build a trend prediction model based on our observation on user activeness (37; 38). Third, we observe that social network providers play important roles in the diffusion of information, and propose a dynamic influence maximization problem from the perspective of social network providers (35). Finally, observing the community effects on information diffusion
process, we propose an efficient approximate algorithm for the influence maximization problem utilizing
the community effects (36).
CHAPTER 1

INTRODUCTION

1.1 Dissertation Framework

The recent years saw a rapid growth of online social networks. All kinds of information spread through online social networks in unprecedented speed and scale. The diffusion of information in social networks has been studied for decades by researchers in the areas of social science and physics. Recent flourishing of online social networks provides valuable data for researches on information diffusion. Online social networks make it possible to record information diffusion processes in large-scale real networks. With such data, we can get better understanding on the diffusion of information, and build more accurate models for diffusion processes. Online social networks also bring plenty of solid applications for the studies of information diffusion. For example, viral marketing through word-of-mouth in online social networks has proved to be very effective.

In this thesis, we focus on the diffusion of information in online social networks. We get insightful understandings on the diffusion of information by analyzing the information diffusion data of online social networks, and these understandings lead to more effective and efficient algorithms for applications such as predicting future information diffusion processes and finding most influential users.

Our work in this thesis covers three key problems in the study of information diffusion: diffusion modeling, trend predicting, and influence maximization.
• **First,** we observe that the information diffusion processes are often interfered by the external world outside social networks. By excluding these interferences, we are able to learn more accurate information diffusion models.

• **Second,** we build a novel diffusion model based on observations on user activeness, and the model is very suitable for the prediction of trends.

• **Third,** observing the important roles that social network service providers (websites) play in the diffusion of information, we extend the influence maximization problem to a dynamic influence maximization problem from the perspective of social network service providers.

• **Finally,** we study the effects of communities on information diffusion processes. Our understandings on the community effects make it possible for designing innovative and effective methods for influence maximization.

### 1.2 Learning Information Diffusion Models with Social Events

(Part of section was previously published (39).)

Learning information diffusion models is a fundamental problem in the study of information diffusion in social networks. Existing approaches learn the diffusion models from events in social networks. However, events in social networks may have different underlying reasons. Some of them may be caused by the social influence inside the network, while others may reflect external trends in the “real world”. Most existing work on the learning of diffusion models does not distinguish the events caused by the social influence from those caused by external trends.
In Chapter 2, we extract social events from data streams in social networks, and then use the extracted social events to improve the learning of information diffusion models. We propose a LADP (Latent Action Diffusion Path) model to incorporate the information diffusion model with the model of external trends, and then design an EM-based algorithm to infer the diffusion probabilities, the external trends and the sources of events efficiently.

1.3 Predicting Trends with Information Diffusion Model

(Part of section was previously published (37).)

With the effect of word-of-the-mouth, trends are generated in social networks when information diffuses in social networks. Predicting dynamic trends is an important problem with many useful applications. There are three dynamic characteristics of a trend that should be captured by a trend model: intensity, coverage and duration. However, existing approaches on the information diffusion are not capable of capturing these three characteristics.

In Chapter 3, we study the problem of predicting dynamic trends in social networks. We first define related concepts to quantify the dynamic characteristics of trends in social networks, and formalize the problem of trend prediction. We then propose a novel information diffusion model, Dynamic Activeness (DA) model, based on the novel concept of activeness, and design a trend prediction algorithm using the DA model. Due to the use of stacking principle, we are able to make the prediction algorithm very efficient. We examine the prediction algorithm on a number of real social network datasets, and show that it is more accurate than state-of-the-art approaches.

1.4 Steering Information Diffusion Dynamically

(Part of section was previously published (35).)
As viral marketing in online social networks flourishes recently, a lot of attention has been drawn to the study of influence maximization in social networks. However, most works in influence maximization have overlooked the important role that social network providers (websites) play in the diffusion processes. Viral marketing campaigns are usually sold by websites as services to their clients. The websites can not only select initial sets of users to start diffusion processes, but can also have impacts throughout the diffusion processes by deciding when the information should be brought to the attention of individual users. This is especially true when user attention is limited, and the websites have to notify users about an item to bring it into the attention of users.

In Chapter 4, we study the diffusion of information from the perspective of social network websites. We propose a novel push-driven cascade (PDC) model, which emphasizes the role of websites during the diffusion of information. In the PDC model, the website “pushes” items to bring them to the attention of users, and whether a user is interested in an item is decided by her preference and the social influence from her friends. Analogous to the influence maximization problem on the traditional information diffusion models, we propose a dynamic influence maximization problem on the PDC model, which is defined as a sequential decision making problem for the website. We show that the problem can be formalized as a Markov sequential decision problem, and there exists a deterministic Markovian policy that is an optimal solution for the problem. We develop an AO* algorithm that finds the optimal solution for the problem, and a heuristic online search algorithm, which has similar effectiveness, but is significantly more efficient. We evaluate the proposed algorithms on various real-world datasets, and find them significantly outperform the baselines.
1.5 Efficient Influence Maximization with Community Effects

(Part of section was previously published (36).)

In social network research, community study is one flourishing aspect which leads to insightful solutions to many practical challenges. Despite the ubiquitous existence of communities in social networks and their properties of depicting users and links, they have not been explicitly considered in information diffusion models. Previous studies on social networks discovered that links between communities function differently from those within communities. However, no information diffusion model has yet considered how the community structure affects the diffusion process.

Motivated by this important absence, in Chapter 5, we conduct exploratory studies on the effects of communities in information diffusion processes. Our observations on community effects can help to solve many tasks in the studies of information diffusion. As an example, we show its application in solving one of the most important problems about information diffusion: the influence maximization problem. We propose a community-based fast influence (CFI) model which leverages the community effects on the diffusion of information and provides an effective approximate algorithm for the influence maximization problem.
CHAPTER 2

LEARNING INFORMATION DIFFUSION MODELS WITH SOCIAL EVENTS

(This chapter was previously published as “Extracting Social Events for Learning Better Information Diffusion Models”, in Proceedings of the 19th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD ’13) ©2013 ACM, Inc. http://doi.acm.org/10.1145/2487575.2487584 (39).)

2.1 Introduction

Recently, online social networks have become a major medium for the spread of information. News, rumors, and opinions propagate in social networks. These events are usually explained by information diffusion processes driven by social influences between users of a social network. Yet each social network is not a closed world. Users obtain information not only from the social network itself, but also from other sources, such as mass media, lectures in universities, friends in real life, etc.

Most existing work on the information diffusion processes assumes that the model has been learned somehow, and focuses on exploring the properties of the learned models. A less studied, but very important topic is the learning of information diffusion models. Work on this topic usually learns the information diffusion model from events in the social network. However, it is questionable to use all the events without any distinction, since the information diffusion processes are not the only reason triggering events in social networks (2; 50; 1; 41). Some of the events are results of social influence or information diffusion processes inside the network, while others may reflect external trends in the world outside the network. For example, #DidYouKnow was a trending hashtag in the Twitter network
in the last three weeks of 2011. The hashtag was used in tweets where people talked about surprising facts. It became popular because of social influence among users in the Twitter network, while #Japan-Earthquake, another trending hashtag in the Twitter network at the same time, reflects a major event in the outside world. Most previous approaches (17; 49; 48; 13) on the learning of information diffusion models do not distinguish the two different types of events, which makes the learned models inaccurate.

In this chapter, we study a problem of learning information diffusion models. We propose a new approach that can distinguish the two different sources of events, and then use the identified social events to improve the learning of information diffusion models. Although the basic idea is straightforward, it is not easy to design a solution based on this idea. There are three key challenges:

- While the sources of some events are easy to be classified as external trends or the social influence, for most events the sources are not easy to determine. For example, when the earthquake hit Japan, a great many of Twitter users prayed for people in Japan. Some of the users did that after they saw the sad news of earthquake on TV, while others did that because they saw other users in the Twitter network do that. In this case, we cannot simply classify this event to be an externally sourced event or a socially sourced event, but need to decide the source with finer granularity. As we will show in Section 2.2, we define the influence source on the action level.

- In order to distinguish the socially sourced actions from externally sourced actions, we need a information diffusion model as well as a model of external trends, but both of them are unknown. The model of external trends can only be inferred from the externally sourced actions, while the diffusion model can only be learned from the socially sourced actions. In other words, only when we are able to decide the sources of actions, we can learn the external trend model and the
information diffusion model accurately. This leads to an inherent “chicken and egg” problem. We refer to it as “inference dependency”.

- We need to consider both external trends and information diffusion processes at the same time. It is not trivial, since the external trends are time-related, while the diffusion model depends on the structure of the social network. It requires us to integrate a temporal model and a structural model into one joint model. Besides, both of them contain plenty of parameters. It may lead to high complexity in the inference of the joint model.

We propose a novel LADP (Latent Action Diffusion Path) model to extract social events and learn diffusion models with better accuracy. Rather than classify events into external events and social events, we determine for each action in an event whether it is caused by external trends or the social influence inside the network. We use a mixture model framework to combine the external trend model and the information diffusion model together, and decide the class of each action. As the learning algorithm of the model involves the inference of external trends, diffusion probabilities, and the sources of actions at the same time, a naive implementation can lead to prohibitively high computational cost. An inference algorithm based on the expectation maximization (EM) is devised to overcome the difficulty of inference dependency, while avoiding the high computational overhead on repeated invocation of the diffusion model.

The improved accuracy can result in better performance on many applications based on information diffusion models, such as influence maximization (25) and outbreak detection (31). As we will show in the experiment, for the DBLP network, the top authors suggested by the LADP have an average H-
index and number of citations up to 20% higher than the top authors suggested by the state-of-the-art approaches.

2.2 Problem Formulation

In this section, we formally define the task of information diffusion model learning. We begin with a few key concepts as follows. The notations are summarized in Table I.

Definition 1 Social Network A social network is a graph $G = (V, E)$, where a vertex $v \in V$ corresponds to a user, and an edge $e = (v_i, v_j) \in E$ stands for a connection between the users $v_i$ and $v_j$. Edges in a social network can either be directed or undirected.

The social network itself provides nothing more than structural information. To learn the diffusion model, we also need the contents created by users in the network, for example, the tweets created by users of Twitter network, or the publications of authors in the DBLP network. We define the collection of contents as “data stream”.

Definition 2 Data Stream A data stream $S$ on a social network $G$ is defined as a chronological sequence of document sets $C_t$, i.e. $S = \{C_t\}_{t=1}^T$. A textual document $d \in C_t$ contains a set of terms. Each document is associated with a node in $V$, denoted by $v_d$, and has a time stamp, denoted by $t_d$. The $t$-th document set $C_t \in S$ contains the documents created at time step $t$, i.e. $C_t = \{d; t_d = t\}$.

If a document $d$ is contained in one of the sets $C_t$, we say that the document $d$ is contained in the data stream $S$. With a little abuse of notation, we denote it by $d \in S$.

We denote with $L$ the set of terms in the data stream $S$. Terms in the documents can be defined in various ways. For example, we can define each word in a document as a term. We can also define
each hashtag in a tweet as a term. More generally, we can define any tags or labels as terms, so that
the streams are not limited to sequences of textual documents. In this chapter, we focus on the analysis
of textual streams. Nevertheless, the proposed LADP model can be applied to more general types of
streams.

In the LADP model, we regard the generation of a document as a process that, for each term \( l \in \mathcal{L} \),
the author makes a decision whether to include it in the document or not. We call this decision an
“action”.

**Definition 3 Action** For each document \( d_i \in S \), for each term \( l \in \mathcal{L} \), there is an action \((i, l)\) taken
by the author of the document. If the document \( d_i \) contains the term \( l \), the action is a positive action,
denoted by \( x_{i,l} = 1 \). Otherwise, the action is a negative action, denoted by \( x_{i,l} = 0 \).

For the positive action, we also introduce the concepts of socially sourced action, and externally
sourced action. A socially sourced action is a positive action that taken by a user because she is
influenced by an information diffusion process inside the social network, while an externally sourced
action is a positive action that is triggered by an external trend. It is true that a positive action may
sometimes be triggered by both the social influence and the external trend at the same time. But in most
cases, the major source of an action can be identified, since at one point of time the user usually gets
information from only one source. In this chapter, for simplicity of the model, we assume that a positive
action can either be a socially sourced action or an externally sourced action.

**Definition 4 Event** An event in a social network is a sequence of positive actions of the same term
\( l \in \mathcal{L} \). Each event may include a socially sourced portion (or a social event), and an externally
sourced portion. The socially sourced portion contains socially sourced actions, while the externally sourced portion contains externally sourced actions.

We define the sources on the action level, rather than on the event level, on the user level, or on the document level. Although the event, the document and the user of an action are all important factors of it, each factor alone cannot perfectly capture the reasons for triggering the action. As we have discussed in Section 2.1, we cannot simply classify an event as a socially sourced event or an externally sourced event, since actions in the event may have different sources. We cannot define the classes on user level either, because each user usually obtains information from both inside and outside the network, and actions taken by a user can be triggered by the social influence or external trends. Even the classification on document level is not good enough, since each document may contain several different terms or topics. For example, a tweet in 2011 said “#DidYouKnow that #JapanEarthquake affected the underground water in Florida?”. It involves both the social event of using the hashtag “#DidYouKnow” in Twitter community and the external trend of “Japan earthquake”.

By defining the classes on the action level, our approach has greatest flexibility and can infer the underlying reasons precisely. By classifying the actions as socially sourced or externally sourced actions, the inference algorithm can split socially sourced portion and the externally sourced portion of an event. In another sense, it can extract the socially sourced portion from the event. We refer to the extracted externally sourced portion as social event and the extracting procedure as social event extraction.

Definition 5 Information Diffusion Process The information diffusion process is the process that actions of terms propagate along the edges of the social network. The process is the result of influence among users in a social network.
A diffusion model aims to predict diffusion processes. Typically, given the actions at time step $t$, the diffusion model predicts the probability for each user in the social network of taking a positive or a negative action in the next time step $t + 1$. The IC (Independent Cascade) model (25) is a widely used information diffusion model. In the IC model, when a user becomes active, she has an independent chance to make each of her neighbors become active. In the proposed LADP model, we define a mechanism of positive action propagation that extends the IC model to the action level.

**Task.** Based on the definitions of the above concepts, we can formalize the task of information diffusion model learning: Given a social network $G$ and data stream $S$ on it, we aim to learn the diffusion model on the network $G$. In varieties of information diffusion models, including the IC model and our model, the parameters of a diffusion model are the diffusion probabilities along edges, so we focus on the learning of diffusion probabilities in this chapter. Different from existing approaches, we extract social events from the data stream, and learn the diffusion model from the extracted social events.

### 2.3 Proposed Model

The LADP model extracts social events from data stream, and learns the information diffusion model from the extracted social events. The block diagram of the LADP model is shown in Figure 1(a).

To extract social events, the LADP model infers the influence source for each positive action. The distribution of socially sourced actions is decided by the information diffusion model, and the diffusion probabilities are the parameters that need to be estimated. The distribution of the externally sourced actions is decided by the external trend model, and the trend profiles are the parameters of the information diffusion model that need to be estimated. A mixture framework is proposed to integrate the information diffusion model and the external trend model.
<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{G} = (\mathcal{V}, \mathcal{E})$</td>
<td>The social network</td>
</tr>
<tr>
<td>$\mathcal{S}$</td>
<td>The data stream, defined by a temporal sequence of $C_t$</td>
</tr>
<tr>
<td>$C_t$</td>
<td>The $t$-th document collection in $\mathcal{S}$</td>
</tr>
<tr>
<td>$\mathcal{L}$</td>
<td>The set of considered terms</td>
</tr>
<tr>
<td>$N$</td>
<td>The number of documents in $\mathcal{S}$</td>
</tr>
<tr>
<td>$N_t$</td>
<td>The number of documents in $C_t$</td>
</tr>
<tr>
<td>$M$</td>
<td>The number of terms in $\mathcal{L}$</td>
</tr>
<tr>
<td>$T$</td>
<td>The number of time steps in the sequence $\mathcal{S}$</td>
</tr>
<tr>
<td>$x_{i,l}$</td>
<td>The label denoting whether the action with document $d_i$ and the term $l$ is positive or negative</td>
</tr>
<tr>
<td>$z_{i,l}$</td>
<td>The label denoting whether the action with document $d_i$ and the term $l$ is decided by the information diffusion process or not</td>
</tr>
<tr>
<td>$\theta_l$</td>
<td>The mean of $z_{i,l}$ for the term $l$</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>The Beta prior of $\theta_l$</td>
</tr>
<tr>
<td>$\mu_{l,t}$</td>
<td>The probability of an action generated from external trends about term $l$ taken at time $t$ being positive</td>
</tr>
<tr>
<td>$\beta_l$</td>
<td>The Beta prior for $\mu_{l,t}$</td>
</tr>
<tr>
<td>$q_{l,t,v}$</td>
<td>The probability of an action generated about term $l$ taken by user $v$ at time $t$ generated from the diffusion model being positive</td>
</tr>
<tr>
<td>$p_{u,v}$</td>
<td>The diffusion probability along the edge $(u, v)$</td>
</tr>
<tr>
<td>$\Lambda$</td>
<td>The collection of parameters for the model, i.e. ${\theta_l, \mu_{l,t}, p_{u,v}}^{M}_{l=1}^{T=1} \forall u, v \in \mathcal{V}$</td>
</tr>
</tbody>
</table>

**TABLE I**

**NOTATIONS**

The inference of action sources depends on trend profiles and diffusion probabilities, while the inference of trend profiles and diffusion probabilities depends on the sources of actions. We design an EM-based inference algorithm to solve this “inference dependency” problem. The algorithm estimates the parameters iteratively. In each iteration, it first infers the sources of actions based on the current estimates of trend profiles and the information diffusion probabilities, and then infers the trend profiles
and the diffusion probabilities based on the probability of each action being socially caused or externally caused.

Figure 1. LADP Model

2.3.1 The framework of LADP Model

Now we formally define the LADP model. As shown in graphical representation of the LADP model in Figure 1(b), the observation variables $x_{i,l}$ are central to the model. Each $x_{i,l}$ indicates whether the corresponding action $(i, l)$ is positive or negative. It is drawn from a mixture distribution, which integrates the external trend and the information diffusion process. The label $z_{i,l}$ decides from which component distribution the variable $x_{i,l}$ is drawn. The left part of the graphical representation ($\beta_l$ and
\( \mu_{t,t} \) shows the model of external trends, while the right part \((p_{u,v})\) represents the information diffusion model. The two of them form the two components of the distribution of \( x_{i,l} \).

Notice that, the external trend model is a temporal model, while the information diffusion model based on the structure of the network. Both of them are complicated models with a lot of parameters. It is not easy to combine these two models together without making the inference algorithm intractable. We therefore make an assumption that each action can only be generated from either the external trend model or the information diffusion model. This assumption makes it possible to integrate the two models together under a mixture model framework.

Formally, \( x_{i,l} \) is defined as:

\[
x_{i,l} = \begin{cases} 
1 & \text{if the action } (i, l) \text{ is positive} \\
0 & \text{otherwise.}
\end{cases}
\]

The hidden variables \( Z \) indicates whether the action is drawn from the external trend model or from the information diffusion model.

\[
z_{i,l} = \begin{cases} 
1 & \text{if action } (i, l) \text{ is drawn from the information diffusion component} \\
0 & \text{otherwise.}
\end{cases}
\]

Notice that \( z_{i,l} \) is defined for all actions, whether they are positive or not, though the concepts of socially sourced action and externally sourced action are defined for the positive actions only. When \( z_{i,l} = 1 \) and \( x_{i,l} = 1 \), the action is a socially sourced action. When \( z_{i,l} = 0 \) and \( x_{i,l} = 1 \), the actions is an externally source action.

For each \( i \), \( z_{i,l} \) is a random variable drawn from Bernoulli distribution with mean \( \theta_{l} \), i.e. \( z_{i,l} \sim Bernoulli(\theta_{l}) \). The mean \( \theta_{l} \) is different for each term, since different term has different potential
probability of being related to an information diffusion process or an external trend. We use a Beta prior for the parameter $\theta_l$, i.e. $\theta_l \sim Beta(\alpha)$, where $\alpha = (\alpha_1, \alpha_0)$ are fixed parameters. We choose Beta distribution because (1) it has a great flexibility of the shape, and (2) it is the conjugate prior distribution for Bernoulli distribution.

Given the corresponding hidden variable, the distribution of $x_{i,l}$ is given as:

$$x_{i,l} \sim \begin{cases} 
    \text{Bernoulli}(q_{l,t_d,v_d}) & \text{if } z_{i,l} = 1 \\
    \text{Bernoulli}(\mu_{l,t_d}) & \text{if } z_{i,l} = 0.
\end{cases}$$

where $q_{l,t_d,v_d}$ is decided by the information diffusion model, and $\mu_{l,t_d}$ is decided by the external trend model. We will discuss the two models in the following sections.

### 2.3.2 Information Diffusion Model

We use a diffusion model that can be regarded as an extension to the widely used Independent Cascade (IC) model (25). In the IC model, information propagates along the edges in the network. When a node becomes active, it attempts to activate its neighbors. For each node, the attempts to activate it from all its active neighbors are independent. Similarly, in the LADP model, $q_{l,t,v}$, the probability that a user $v$ uses a term $l$ is predicted from the actions that $v$’s neighbors took in the last time step. For each positive action about term $l$ taken by in-neighbors of $v$ at last time step $t-1$, there is an independent chance to make the action of $v$ at time $t$ to be positive. Formally, for the term $l$, the probability of an action taken by user $v$ at time $t$ being positive is given as:

$$q_{l,t,v} = 1 - \prod_{d_i \in C_{t-1}, z_{i,l} = 1} (1 - p_{v_d_i,v}) \quad (2.1)$$
where $C_{t-1}$ is the set of documents that were created at the last time step $t - 1$. $p_{vd,v}$ is the diffusion probability along with the edge $(vd, v)$. For the convenience of notation, we define $p_{vd,v} = 0$, if there is no edge between the nodes $vd$ and $v$. The product in the formula is the probability that all of the in-neighbors of $v$ fail to make the action of $v$ at time $t$ to be positive. Under the independent assumption, this probability could be calculated by multiplying together the probabilities that each attempt fails. We then can get $q_{l,t,v}$, the probability that at least one attempt succeeds, by subtracting the product from 1.

### 2.3.3 External Trend Model

For the actions that are generated by external trend model, the probabilities of being positive are not decided by the social network structure and previous actions in the network. A reasonable assumption with these actions is that the probability of being positive only depends on the time and the term, but does not depend on the user takes the action (a similar assumption was made in (41)). The reason underlying the assumption is that, if an action is decided by an external trend outside the network, we cannot make any prediction on whether the action is positive or not, based on the network structure, so the best assumption we can make is that each action is a random variable independently drawn from the same distribution.

Nevertheless, the probability of an action being positive should be depends on the term $l$ and the time $t$. That is because different terms have different levels of popularity in the external world. The more popular a term is, the more likely that an action with regard to it is positive. For a given term, its popularity changes over time. For each term $l$, the parameters $\mu_{l,t}$ forms a time sequence which we call profile of the external trend.
We therefore assume an Beta prior for $\mu_{l,t}$ the probability of being positive: $\mu_{l,t} \sim Beta(\beta_l) = Beta(\beta_{l,1}, \beta_{l,0})$. Similar to the prior of $\theta_l$, we choose Beta distribution because its shape is flexible and it is the conjugate prior for Bernoulli distribution. For each term $l$, we set the parameter $\beta_l$ in the prior distribution according to the number of positive and negative actions over all the time steps. Specifically, $\beta_{l,1}$ is set to the average number of positive actions for the term $l$ in all the steps, while $\beta_{l,0}$ is set to the average number of negative actions in all the steps.

2.4 Parameters Estimation

We have discussed the difficulty of inference dependency in the introduction. We design an EM-based algorithm to solve this difficulty by iteratively estimating the conditional distribution of hidden variables $z_{i,l}$ and the parameters. In the E-step of each iteration, the estimate for the conditional probability of $z_{i,l}$ is updated, while the estimates for parameters $\theta_l$, $\mu_{l,t}$ and $p_{u,v}$ are updated in the M-step.

The parameters $\theta_l$ and $\mu_{l,t}$ are easy to estimate using the EM algorithm of the maximum a posteriori (MAP) estimate. However, it is difficult to define and calculate the MAP estimate of the parameters $p_{u,v}$, due to the complexity of the diffusion model. To solve that difficulty, we modify the PCB model in (17) so that it can be integrated into the EM framework to provide estimate of $p_{u,v}$ efficiently. We first assume that $p_{u,v}$ are known (consequently, $q_{l,t,v}$ are known), and show the EM algorithm. Then, we discuss the estimation of $p_{u,v}$ and add it into to the EM framework.
Given the LADP model above, we want to maximize expectation of the logarithm of the posterior:

\[ Q(\Lambda|\Lambda^{(n)}) = E_{Z|X,\Lambda^{(n)}}\log Pr(\Lambda; X, Z) \]

\[ = \sum_{l=1}^{M} \left[ (\alpha_1 - 1)\log \theta_l + (\alpha_0 - 1)\log (1 - \theta_l) \right] \]

\[ + \sum_{l=1}^{M} \sum_{t=1}^{T} \left[ (\beta_{l,1} - 1)\log \mu_{l,t} + (\beta_{l,0} - 1)\log (1 - \mu_{l,t}) \right] \]

\[ + \sum_{l=1}^{M} \sum_{i=1}^{N} \left[ z_{i,l,1}^{(n)}(\log \theta_l + x_{i,l}\log (q_{l,t_d,v_d})) \right. \]

\[ \left. + (1 - x_{i,l})\log (1 - q_{l,t_d,v_d}) \right] \]

\[ + z_{i,l,0}^{(n)}(\log (1 - \theta_l) + x_{i,l}\log (\mu_{l,t_d}) + (1 - x_{i,l})\log (1 - \mu_{l,t_d})) \]

where \( z_{i,l,j}^{(n)} \) stands for \( P(z_{i,l}^{(n)} = j|X, \Lambda^{(n)}) \), for the simplicity of the equation.

The first term in the formula comes from the distribution \( \theta_l \sim Beta(\alpha) \). The second term comes from the distribution \( \mu_{l,t} \sim Beta(\beta_l) \). The last term comes from the distribution of \( X \) and \( Z \), given the parameters \( \theta, \mu \) and \( q \).

**E-step.** In the E-step, we calculate the conditional probability of hidden variables \( Z \), given the observed variables \( X \) and estimate of parameter \( \Lambda \):

\[ z_{i,l,j}^{(n)} = \frac{p(x_{i,l}|z_{i,l}^{(n)} = j, \Lambda^{(n)})}{\sum_{j'=0}^{1} p(x_{i,l}|z_{i,l}^{(n)} = j', \Lambda^{(n)})} \]

where \( j = 0, 1 \). \( p(\cdot|z_{i,l}^{(n-1)} = 1, \Lambda^{(n)}) \) is the probability mass function of \( x_{i,l} \), given it is drawn from the information diffusion component:

\[ p(x_{i,l}|z_{i,l}^{(n)} = 1, \Lambda^{(n)}) = \theta_l^{(n-1)}(q_{l,t_d,v_d})^{x_{i,l}}(1 - q_{l,t_d,v_d})^{1-x_{i,l}} \]
and \( p(\cdot | z_{i,l}^{(n)} = 0, \Lambda) \) is the probability mass function of \( x(i, l) \), given it is drawn from the external trend component:

\[
p(x_{i,l} | z_{i,l}^{(n)} = 0, \Lambda^{(n)}) = (1 - \theta_{l}^{(n)})^{x_{i,l}}(1 - \mu_{l,t_d}^{(n)})^{1-x_{i,l}}
\]

**M-step.** By taking partial derivatives of the expectation of log-likelihood, we get the new estimation of parameters.

\[
\theta_{l}^{(n+1)} = \frac{\sum_{i=1}^{N} z_{i,l}^{(n)} + \alpha_1 - 1}{N + \alpha_1 + \alpha_0 - 2}
\]

and

\[
\mu_{l,t_d}^{(n+1)} = \frac{\sum_{d_i \in C_i} z_{i,l}^{(n)} x_{i,l} + \beta_{l,1} - 1}{\sum_{d_i \in C_i} z_{i,l}^{(n)} + \beta_{l,0} + \beta_{l,1} - 2}
\]

**Estimate of Diffusion Probabilities.** We now discuss the estimate of diffusion probabilities. It is possible to formulize it as an inference problem for maximum likelihood or MAP estimate. Saito at el. defined a likelihood function for the IC model, and proposed an EM algorithm for inference problem(49). However, the number of parameters in the diffusion model is very large (one parameter for each edge in the network). The inferring of model is very time-consuming, even on a fixed set of actions. In the LADP model, the set of socially sourced actions changes in each iteration of the EM algorithm. Getting maximum likelihood estimation for the diffusion probabilities in each iteration of the EM algorithm will be an enormous computational challenge. Therefore, we follow the Partial Credit Bernoulli model (PCB) (17) to estimate the diffusion probabilities.

The original PCB model is not based on the maximum likelihood or MAP estimate, so it does not work together with the EM algorithm for inferring the LADP model. We change it so that it can be
incorporated into the M-step of the EM algorithm. We will first describe the PCB model, and then show that how we change it so that it can be incorporated into the EM algorithm.

The idea of the PCB model is as follows: if user \( v \) takes a positive action about a term \( l \) at time \( t \) after his in-neighbor \( u \)’s positive action at time \( t - 1 \), we regard there is a successful diffusion from \( u \) to \( v \). If there are more than one in-neighbors of \( v \) take positive action at the previous step, they share the credit for the one successful diffusion equally. The diffusion probability \( p_{u,v} \) is then given by the ratio of the number of successful diffusion from \( u \) to \( v \) to the number of positive actions taken by \( u \).

Formally, the diffusion probabilities can be estimated by:

\[
p_{u,v} = \frac{\sum_{(i,l) \in A_u} \sum_{d \in \text{Can}(i,l)} I(v_d=u)}{|A_u|} \tag{2.2}
\]

where \( A_v = \{(i,l) : x_{i,l} = 1, v_{d_i} = v\} \) is the set of all the positive actions taken by the node \( v \) and \( I(\cdot) \) is the indicator function. \( \text{Can}(i,l) \) is the candidate set of documents that share the credit for the successful diffusion. A document is in the candidate set if and only if it is posted by a friend of \( v_{d_i} \) and it is posted in the time step right before \( t_{d_i} \), i.e., \( \text{Can}(i,l) = \{d_j|t_{d_j} = t_{d_i} - 1, x_{j,l} = 1, u \in \text{IN}(v_{d_i})\} \), where \( \text{IN}(v) \) is the set of in-neighbors of \( v \).

According to Equation 2.2, all the positive actions are used for the learning of the diffusion model. Since in the LADP model we have the hidden variable \( z_{i,l} \) representing whether an action is drawn from diffusion processes or not, we should train the model with the socially source actions only. We then can add the hidden variable \( z_{i,l} \) to Equation 2.2, and replace it with the following equation:

\[
p_{u,v} = \frac{\sum_{(i,l) \in A_u} \sum_{d \in \text{Can}(i,l)} I(v_d=u, z_{i,l}=1)}{|A_u|} \tag{2.3}
\]

We then replace \( z_{i,l} \) in the above function with \( P(z_{i,l}^{(n)} = j|X, \Lambda^{(n-1)}) \), and integrate the learning of diffusion probabilities into the EM algorithm.
2.5 Experiment

2.5.1 Baselines

We compared the proposed LADP model with three baselines: two for the task of learning diffusion model, and the other for the analysis of extracted social events.

the PCB algorithm: We compare the LADP model against the Static PC Bernoulli algorithm (PCB) (17) for the task of learning diffusion model. The PCB algorithm is similar to the inference method described in Section 2.4, but all positive actions are regarded as socially sourced actions and are used for the probability learning. It is equivalent to the LADP model with $\theta = 1$.

Saito’s Algorithm: The Saito’s algorithm (49) is another baseline that we used for evaluating the task of learning diffusion probabilities. It is an EM algorithm for maximum likelihood estimation of the IC model.

Myers’s Algorithm: For better understanding of the LADP model, we also analyze the social events extracted by the model. The Myers’s algorithm (41) is used for comparison in the analysis of extracted social events. This algorithm is not designed for the same purpose as the LADP model. But it can divide the influence to the users into two parts: the internal part and the external part. The internal part can roughly be aligned with the socially sourced portion in our model, so we use it for a comparison in the analysis of extracted social events.

2.5.2 Datasets

The algorithms are tested on four real world datasets and one semi-synthetic dataset. Each real world dataset consists of a social network and a data stream on the network. The semi-synthetic dataset is based on real world network, but we generate the data stream synthetically. The semi-synthetic dataset
is used to evaluate the accuracy of the inference algorithm, while the real world datasets are used to test how well the LADP model works for real applications.

**Twitter-UIC dataset:** This dataset consists of 974,382 tweets on a network of 2,180 users and 14,572 links. The users in the dataset are the followers of the “UIC news” account on twitter.com. Most of them are students in University of Illinois at Chicago. Directed links in the network correspond to who-follows-whom relationships. They are directed from the one being followed to the follower. The stream data on the network are generated from the tweets posted by the users over the 52 weeks of the year 2011. Hashtags in the tweets are used as the terms. Each week is regarded as a time step.

**Twitter semi-synthetic dataset:** We first randomly crawled a network from twitter.com that consists of 40,000 users and 544,936 links, and then generate the semi-synthetic dataset using steps as follows. The diffusion probabilities with edges are randomly picked from a Beta distribution with parameters $(1,30)$. We use the same collection of terms as those in the Twitter-UIC dataset, and the Google Trend profiles of the corresponding terms are used as the external trends profile. The socially sourced actions are generated using the diffusion probabilities with the diffusion model described in 2.3.2, and the externally sourced action are generated from external trends. The synthetic data contains 223 events, and 14,170,112 positive actions. 68.1% of the positive actions are socially sourced actions, while others are externally sourced actions.

**DBLP datasets:** We extract three datasets from the DBLP database. Each of them contains the stream of publications in a certain area and the co-author network of that area. In the co-author network, each node corresponds to one author, while each undirected edge corresponds to a co-author relationship
between two authors. The titles of publications are used as the textual stream. We remove stopwords from the titles, and use bigrams as terms. The three datasets we used for the evaluations are as follows.

- **Data mining community dataset**: This community contains 14,011 publications and their authors in 10 data mining conferences over 15 years (1995-2010). Each year is regarded as a time step. Conferences are listed in Table II. The co-author network contains 6,948 nodes and 39,797 edges. Authors that have less than 3 publications are filtered out.

- **Machine learning community dataset**: It is similar to the data mining community, but are based on 10 machine learning conferences, as listed in Table II. This community contains 24,184 publications, 6,845 nodes and 34,254 edges.

- **Mixed dataset (data mining + machine learning)**: We want to test the algorithms on a more complicated community, because the above two simple communities share several desirable properties: Authors in each community are from the same area, and are strongly connected to each other; Documents in each community are from similar topics, so the events are easier to be detected. By extracting publication in the 20 listed conference and filtering out authors with less than 5 publication, we get a network that consists of 7664 nodes and 47,158 edges.

### 2.5.3 Experiment with Twitter Datasets

#### 2.5.3.1 Experiment with Semi-synthetic Dataset

To test the accuracy of the inference algorithm, we evaluate the LADP model on the semi-synthetic dataset. As the diffusion probabilities and external trends for this dataset are known, we can evaluate the inferred values directly.
### TABLE II

CONFERENCE LISTS OF TWO COMMUNITIES IN DBLP DATASET

<table>
<thead>
<tr>
<th>Data Mining</th>
<th>Machine Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>KDD</td>
<td>AAAI</td>
</tr>
<tr>
<td>SDM</td>
<td>IJCAI</td>
</tr>
<tr>
<td>ICDM</td>
<td>ICML</td>
</tr>
<tr>
<td>SIGMOD</td>
<td>ICRA</td>
</tr>
<tr>
<td>VLDB</td>
<td>ICGA</td>
</tr>
<tr>
<td>CIKM</td>
<td>UAI</td>
</tr>
<tr>
<td>ICDE</td>
<td>KR</td>
</tr>
<tr>
<td>WWW</td>
<td>IROS</td>
</tr>
<tr>
<td>PKDD</td>
<td>ATAL</td>
</tr>
<tr>
<td>PAKDD</td>
<td>AAMAS</td>
</tr>
</tbody>
</table>

**Result of diffusion model learning.** The result is shown in Figure 2(a), for the diffusion probability with each edge in the network, we calculate the prediction error as the difference between the real value and the inferred value. We plot the empirical cumulative distribution function of the prediction error in Figure 2(a). Each point in the curve shows a value and the percentage of the prediction errors that are below the given value. As shown in the figure, the diffusion probabilities inferred by LADP model have much smaller error than the ones inferred by the baselines. For example, for 92.0% edges, the prediction errors of the LADP model are smaller than 0.05, while only for 79.8% and 50.8% edges respectively the prediction errors of the PCB algorithm and the Saito’s algorithm are within this range.

The only difference between the LADP model and the PCB algorithm is that the LADP model extracts the social events from the entire events, and learns the diffusion probabilities from the social events, while the PCB algorithm learns the diffusion probabilities from all actions in the events. It implies that the improvement of accuracy by the LADP model is the result of extracting social events.

**Extraction of social events.** For better understanding of how the LADP model makes the improvement, we show the difference between inferred socially sourced portion and the ground truth in Figure
2(b). To calculate the difference, we first define the time sequence \( \{a_t\}_{t=1}^T \) of a set of actions. For each time step \( t = 1, \ldots, T \), there is an element \( a_t \) in the time sequence, which is the number of actions in the set that are created at time step \( t \). We then calculate for each event the L2 distance between the time sequence of the extracted socially sourced portion and that of the ground truth, and plot the empirical distribution of the distance in Figure 2(b).

For the proposed LADP model, only the extracted socially sourced portion are used for the learning of diffusion model, while for the PCB and Saito’s algorithms the entire events are used for the learning, in other words, they consider the entire events to be social events, so we also show the L2 distance between the time sequence of the entire event and that of socially sourced portion.

Since the time sequence of entire events is always an overestimation of the time sequence of socially sourced portion, we can easily get a better estimation by simply scaling it. We then scale the time sequence for each event and make it have the same mean value as the time sequence of the socially sourced portion, and calculate the L2 distance between the scaled time sequence and that of socially sourced portion as well.

As shown in Figure 2(b), the distance between the time sequence of socially sourced events inferred by LADP and that of the ground truth is smaller than the distance between the time sequence of the entire events and that of socially sourced portion, even when we scale the time sequence of entire event. This reflects that the LADP model can extract social events from the entire events, and the extraction is more than divide the entire events into two parts according to the proportion of socially sourced portion and externally sourced portion. The extraction of social events by the LADP model results in better accuracy in learning the diffusion model.
2.5.3.2 Experiment with Twitter-UIC dataset

**Result of diffusion model learning.** For the real stream data, the diffusion probabilities are unknown, so we are not able to evaluate a learned diffusion model directly. Instead, we evaluate it by evaluating the most influential node suggested by the models. In the Twitter community, users can repost tweets of other users, which is called “retweet”. The more retweets a tweet gets, the more widely it spreads in the network. We can expect that users with larger number of retweets per tweet are more influential in the social network.

Given the learned diffusion model, by sampling the diffusion process for 50,000 times, we calculate the average influence of each node, i.e. the average number of activations when using each single node as the seedset. The nodes are then sorted according to the descending order of their influences. In this way, we find out the most influential nodes in the models learned by the LADP, PCB and Saito’s algorithms. We then evaluate the most influential node by the average number of retweets for each tweet. Figure 3 shows the comparison between the LADP method and the baselines for finding the most influential
nodes. For each number \( k \) on the x-axis, we calculate the average number of retweets for the top \( k \) users and plot it in the figure. In the most range of x-axis, the LADP model achieves a larger average number than the baselines. Besides, the curve of the LADP model is a monotone decreasing curve. It is more desirable than the non-monotonic curve of the PCB model, because the monotone decreasing curve suggests that higher-ranking nodes inferred by the algorithm are really more influential than lower-ranking nodes. The better performance in finding top influential users suggests that the diffusion model learned by LADP is better than those learned by the baselines.

**Analysis of top social events.** We analyze the top social events extracted by the LADP model in order to understand how the LADP model improves the learning of diffusion model. First, we rank the events according to the numbers of all actions in the events, and list the top events in the network. Then, we rank the events according to the number of actions in the inferred socially sourced portion, i.e. we rank the social events extracted by the LADP model. After excluding the externally sourced actions, the list of top social events extracted by the LADP model is different from the list of top events with the
largest number of actions. For a comparison, we also list the top events with the largest internal portion inferred by the Myers’s algorithm (41).

The results are shown in Table III. The first column in the table is the list of keywords of events with the largest number of actions. Some keywords in the list are closely related to the UIC community (Chicago, UIC, higherEd\(^1\)), but others are not (FF\(^2\), energy). The second column is the list of top five internal events returned by the Myers’s algorithm. For this dataset, the lists in the second column happen to be the same as the list in the first column, but this is not always the case, as we will show in the following experiment. In the third column, we list the top social event extracted by the LADP model. Comparing with the first column, the keywords “UIC”, “higherEd”, and “Illinois” move upward in the rankings, while the keywords “FF”, “energy” move downward. It is obvious that keywords that are related to the community get better rankings in the list returned by the LADP model. Since the keywords that are closely related to the community is more likely to suggest social events inside the network, the top events extracted by the LADP model are more likely to reflect information diffusion processes inside the social network. (Notice that though the keyword “FF” reflects an event on the Twitter website. It is not unique to the UIC community, and its propagation does not suggest an information diffusion process inside the Twitter-UIC network. Further analysis on the keywords “UIC” and “FF” will be provided in next section.) This explains that the improvement of LADP in the learning of diffusion probabilities.

\(^1\)higherEd is short for higher education in this context.

\(^2\)FF is short for FollowFriday in Twitter. It is a online event that people recommend friends for other users.
### Table III

<table>
<thead>
<tr>
<th></th>
<th>All</th>
<th>Myers’s</th>
<th>LADP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Chicago</td>
<td>Chicago (-)</td>
<td>Chicago (-)</td>
</tr>
<tr>
<td>2</td>
<td>FF</td>
<td>FF (-)</td>
<td>UIC (↑)</td>
</tr>
<tr>
<td>3</td>
<td>UIC</td>
<td>UIC (-)</td>
<td>FF (↓)</td>
</tr>
<tr>
<td>4</td>
<td>energy</td>
<td>energy (-)</td>
<td>higherEd (↑)</td>
</tr>
<tr>
<td>5</td>
<td>higherEd</td>
<td>higherEd (-)</td>
<td>Illinois (↑)</td>
</tr>
</tbody>
</table>

**TOP EVENTS AND TOP SOCIAL EVENTS IDENTIFIED BY LADP AND MYERS’S IN UIC TWITTER NETWORK**

#### 2.5.3.3 Case Study

In Figures 4, we show the results of “UIC” and “FF” inferred by the LADP model. For each of the events, we show the number of actions over time for the socially sourced portion, externally sourced portion, and the entire event. As shown in the figure, for the event of “UIC”, the socially sourced portion is the majority, while for the event of “FF”, the size of the two portions are similar. Beside, for the event of “UIC”, the socially sourced portion explains most of the peaks in the full event, while for the event of “FF”, some peaks are explained by the influence while others are explained by the external trends. This is the reason why the ranking of “UIC” moves upward in the list of top social events returned by LADP, while “FF” moves downward.

#### 2.5.4 Experiment with DBLP Datasets

**Result of diffusion model learning.** Similar to the Twitter-UIC dataset, since there is no ground truth for the diffusion probabilities, we evaluate them by looking at the most influential nodes. For the most influential users suggested by each algorithm, we evaluate the top-k influential nodes using their
H-index and the number of citations. The H-index and the number of citations of author are obtained from arnetminer.org.

On all the three datasets, we run the LADP model, the PCB algorithm and Saito’s algorithm respectively to learn the diffusion probability with each edge. These three sets of diffusion probabilities can then be used for deciding most influential nodes. By sampling the independent cascade process for 50,000 times, we calculate the average influence of each node, i.e. the average number of activations when using each single node as the seedset. We then sort nodes to the descending order of influence and plot the average H-index and the average number of citations for top-k authors.

Figures 5 and 6 show the comparison of the LADP model and the baselines on three datasets. The x-axes of these figures are the number of top authors, while the y-axes are the average H-index or the average number of citations of these top authors. On all but one figure, the curves of the LADP method are obviously above those of the baselines, which means the top-ranking authors reported by the LADP model are more influential than those reported by the baselines. Even for that Figure 6(a),
LADP performance on the top 10 authors is significantly better than the baselines. It reflects that the diffusion model learned by LADP is more accurate than the one learned by the baselines.

**Analysis of top social events.** Similar to the experiment on Twitter dataset, in order to understand how the LADP model improves the learning of diffusion model, we analyze the top social events extracted by the LADP model. Since we are more familiar with topics in the data mining, we use the data mining community for the analysis.
The results are shown in Table IV. For each network in the dataset, the first column in the table is the list of keywords of events with the largest number of actions. The second column is the list of top five internal events returned by the Myers’s algorithm. The top five social events extracted by the LADP model are listed in the third column.

The rankings of keywords related to specific research topics in the data mining community are moved upward by the LADP algorithm (data streams, time series, association rules), while keywords that are less related to specific topics move downward. That is desirable because specific topics are more likely to be connected to information diffusion processes in the network, and represent social events. The Myers’s algorithm also tends to give specific topics higher ranking, but it undesirably gives higher ranking to the topics “query processing” and “xml data”, which are related to the database community, rather than the data mining community specifically.

<table>
<thead>
<tr>
<th>All</th>
<th>Myers’s</th>
<th>LADP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1  data mining</td>
<td>query processing (↑)</td>
<td>data streams (↑)</td>
</tr>
<tr>
<td>2  data streams</td>
<td>association rules (↑)</td>
<td>time series (↑)</td>
</tr>
<tr>
<td>3  time series</td>
<td>text classification (↑)</td>
<td>data mining(↓)</td>
</tr>
<tr>
<td>4  query processing</td>
<td>xml data (↑)</td>
<td>association rules (↑)</td>
</tr>
<tr>
<td>5  association rules</td>
<td>pattern mining (↑)</td>
<td>query processing(↓)</td>
</tr>
</tbody>
</table>

**TABLE IV**

**TOP EVENTS AND TOP SOCIAL EVENTS IDENTIFIED BY LADP AND MYERS’ S IN DATA MINING COMMUNITY**
2.6 Related Work

Information diffusion has been intensively studied in social network analysis (3; 8; 7; 28; 1). Earlier work on information diffusion model does not consider the time dynamic of diffusion processes. Recent work in (31; 27; 51) consider the diffusion processes that unfold along the time, so that temporal events in the social network can be explained by the information diffusion processes. Independent Cascade (IC) model and its variants (25; 31; 10; 9; 29) form most widely used class of information diffusion models. Models in this class share two features: (1) the influences from the neighbors of a user are independent; (2) there is a diffusion probability along with each edge in the network.

The problem of estimating the diffusion probabilities has been studied in (17; 49; 48; 13). The diffusion probabilities are learned from events in social networks. Models in (17) estimate the diffusion probability for general threshold models which include the IC model and almost all of its variants. (49; 48) propose a likelihood maximization approach for the learning of diffusion probabilities. However, due to the large number of parameters and the complexity of the likelihood function, the inference algorithm is time-consuming.

Although it has long been argued that the information diffusion process is not the only reason triggering events in social networks (2; 50; 1), most existing work on the learning of diffusion probabilities neglects the propagation of information from external trends. Work in (41) explicitly models the external trends and incorporates it with the information diffusion model. While their approach adopts a simple information diffusion model and focuses on inferring the external trends, our model aims to learn the diffusion probabilities with edges in the networks, and the learned diffusion probabilities can be used in the IC model and its variants.
CHAPTER 3

PREDICTING TRENDS WITH INFORMATION DIFFUSION MODEL

(This chapter was previously published as “Predicting Trends in Social Networks via Dynamic Ac-
tiveness Model”, in Proceedings of the 22nd ACM international conference on Conference on informa-
tion & knowledge management (CIKM ’13) ©2013 ACM, Inc. http://doi.acm.org/10.1145/2505515.2505607
(37) and its extended version (38).)

3.1 Introduction

Online social networks have become increasingly important for interpersonal communication and
information sharing. Trends in online social networks now have large impacts on people’s lives. Trends
are represented by sequences of actions that are taken by users in a social network. According to the
type of the social network, an action can be posting a blog or sharing a webpage about a certain topic,
or joining an online activity.

Predicting the dynamic behavior of trends is an interesting problem with wide applications. Some
examples of such applications are as follows:

1. Online video providers may want to predict how many times a video will be played by users in
   the next month, so that they can decide the bandwidth needed for the server.

2. Disease control facilities may want to predict how many people will suffer from a contagion in
   the following week, so that they can be prepared for an outbreak.

35
3. Manufacturers may want to predict how long an existing product will continue to be popular, so that they can decide the most suitable time for the debut of a new model.

The three applications above require the prediction of trends from three different perspectives. The first example considers the **intensity** of a trend, which is the volume of actions during a fixed length of time. The second one focuses on the **coverage** of a trend, which is the number of people taking the given action during a fixed length of time. The third one considers the **duration** of a trend, which is the time span that the intensity or coverage is above a given threshold.

To better explain these three perspectives (intensity, coverage and duration), we show in Figure 7 a toy example of a trend on a social network which contains three users. Table (b) shows the intensity, coverage and duration that aggregated from actions listed in Table (a). For example, at 2008, \(v_1\) and \(v_2\) take 3 and 2 actions, respectively, while \(v_3\) taking no action, so the coverage (i.e. the number of people taking actions) is 2, and the intensity (i.e., the total number of actions taken) is 5. Though the coverage and intensity are correlated with each other, they are not interchangeable in the sense that the corresponding time series are neither similar nor synchronized. In this example, the maximum value of coverage is reached at year 2009, while the maximum value of intensity is reached at year 2008. Duration reflects how long the trend lasts. If we set the threshold to 0, duration of the trend will be 4 years, from 2007 to 2010.

Based on our observation, each of the three perspectives is useful for many real applications. A trend model should be able to characterize trends from all of these three perspectives.

Though the actions of social network users have been studied in the context of information diffusion models (e.g. the independent cascade (IC) model) (25; 17; 33; 31; 10; 9; 20; 27), the existing informa-
tion diffusion models are not suitable for modeling dynamic trends for three main reasons: **First**, most of these models assume that the diffusion processes take place in discretized time and the propagation of information between two nodes always takes one unit of time, which does not reflect the real dynamic of trends as time unfolds. Therefore, they cannot reflect the dynamic nature of intensity and coverage, or the duration of trends. **Second**, existing information diffusion models focus on the visible path of propagation, and they usually assume that the propagation can only occur between a pair of nodes that are directly linked to each other, while the trend model should focus on predicting the aggregate characteristics of trends, and the path of propagation is not important for the prediction. Besides, because of the existence of homophily (2; 50), the propagation through direct links may not be good enough to explain trends in social networks. The model of trends should be more flexible with regard to the propagation mechanism. **Third**, existing information diffusion models focus on the prediction on the individual user level, but not on the trend level. As a result, they allow the probability of influence to be

<table>
<thead>
<tr>
<th></th>
<th>2007</th>
<th>2008</th>
<th>2009</th>
<th>2010</th>
<th>2011</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_1$</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$v_2$</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$v_3$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

(a) Number of actions of the trend

<table>
<thead>
<tr>
<th></th>
<th>2007</th>
<th>2008</th>
<th>2009</th>
<th>2010</th>
<th>2011</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coverage</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Intensity</td>
<td>1</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Duration</td>
<td>2007 - 2010</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(b) Coverage, intensity, and duration of the trend
different between each pair of users, but assume that the probability remains the same for all the trends. This makes them not suitable for predicting trends based on different properties of trends.

In this chapter, we formally define the three dynamic characteristics of a trend (intensity, coverage and duration), and the problem of trend prediction. We introduce a novel concept of activeness, which reflects a user’s interest toward the given trend at a given point of time. The dynamic nature of activeness enables us to model the dynamic characteristics of trends. Due to the introduction of activeness, a more flexible propagation mechanism is made possible, so that the correlation as well as influence through direct links can be captured in our model. We propose a novel information diffusion model, Dynamic Activeness (DA) model, based on the concept of activeness. Each component of the model is built on observations on real trends, and the model is capable of capturing all the three dynamic characteristics of trends. We design a trend prediction algorithm based on the DA model. For each trend, the parameters of the model are learned specifically from the history data of that trend. The learned model can then be used to predict the dynamic characteristics of the trend in the future. For the efficiency consideration, we identify the stacking principle and utilize it to transform the effect of the sequence of actions to the sum of the individual effects by each single action in isolation. This makes the prediction based on the DA model have a similar computational complexity to the IC model. We show the performance of the DA model on real trends in social networks.
3.2 Preliminaries

3.2.1 Notations and Definitions

Let $G = (V, E)$ be a social network, in which $V = \{v_1, \cdots, v_n\}$ is the set of nodes and $E \subseteq V \times V$ is the set of edges. We consider the network to be static, since the evolution of networks is much slower than that of the trends. During the time span of a given trend, the change of the network is negligible.

A trend on the social network $G$ is defined as follows:

**Definition 6 Trend** A trend $S = [(v_1, t_1), \ldots, (v_m, t_m)]$ on the social network $G$ is a chronological sequence that consists of a given type of actions in $G$, where $t_i \leq t_j (\forall 0 \leq i < j \leq m)$ and $v_i \in V (\forall i \in \{1, \cdots, m\})$. An element $(v_i, t_i)$ in $S$ corresponds to an action of that type taken by the node $v_i$ at time $t_i$.

For simplicity of notation, we denote the time sequence of actions in trend $S$ as $T(S) = [t_1, \ldots, t_m]$, and denote the subsequence of $S$ that consists of all the actions taken by node $v$ as $S^v = [(v, t_{v,1}), \ldots, (v, t_{v,m_v})]$, where $(v, t_{v,i}) \in S (\forall 1 \leq i \leq m_v)$. We also use $S_t$ to denote the prefix of $S$ which contains all the actions taken before the time point $t$, i.e., $S_t = [(v_i, t_i) : (v_i, t_i) \in S, t_i \leq t]$.

Based on the above definition of a trend, we define intensity, coverage, and duration of a trend as follows:

**Definition 7 Intensity** The Intensity of a trend $S$ on a time interval $I = [t_{\min}, t_{\max})$ is the number of actions in $S$ that are taken during $I$. Formally, $\text{Intensity}(S, I) = |\{(v_i, t_i) : t_i \in I \land (v_i, t_i) \in S\}|.$
**Definition 8**  
*Coverage* The Coverage of a trend $S$ on a time interval $I = [t_{\text{min}}, t_{\text{max}})$ is the number of nodes in $V$ that take at least one action during $I$. Formally, $\text{Coverage}(S, I) = |\{v_i : (v_i, t_i) \in S \land t_i \in I\}|$.

**Definition 9**  
*Duration* Let $\mathcal{I} = \{I_1, \ldots, I_s\}$ be a set of intervals, where $I_i = [t_{i\text{min}}, t_{i\text{max}})$. Given a threshold $\theta$, the duration of $S$ on $\mathcal{I}$ is the number of consecutive intervals in $\mathcal{I}$ that the intensity (coverage) is above $\theta$. Formally, $\text{Duration}_{\text{cov}}(S, \mathcal{I}, \theta) = \max(j - i + 1), 1 \leq i \leq j \leq s, \text{ s.t. } \forall k, i \leq k \leq j, \text{Coverage}(S, I_k) > \theta$ and $\text{Duration}_{\text{int}}(S, \mathcal{I}, \theta) = \max(j - i + 1), 1 \leq i \leq j \leq s, \text{ s.t. } \forall k, i \leq k \leq j, \text{Intensity}(S, I_k) > \theta$.

The intensity quantifies the overall activeness of a trend within a social network. The coverage quantifies how broad a trend has impacts in a social network, i.e., the number of nodes involved within a time interval. The larger the coverage value of a trend is, the more nodes of the network are affected/involved in the trend. The duration quantifies how long a trend lasts within the social network.

For the duration, we usually want $I_1, \ldots, I_s$ to be consecutively connected intervals with equal length, i.e., $t_{max}^i = t_{min}^{i+1} (\forall i \in \{1, \ldots, s - 1\})$ and $t_{max}^i - t_{min}^i = t_{max}^j - t_{min}^j (\forall i, j \in \{1, \ldots, s\})$.

We like to point out that, given the intensity and coverage, the duration of trend can be defined in many different ways. We define it as the largest number of consecutive intervals above the threshold because this definition is most straightforward. By carefully setting the threshold $\theta$, the definition will accord with the intuitive understanding of the word “duration”.

The prediction problem of trends is defined as follows:

**Definition 10**  
*Trend Prediction Problem* Given $S_{t^*}$, the prefix of sequence $S$ before time $t^*$, the problem of trend prediction is to predict the intensity, coverage and duration of trend $S$ after time $t^*$. 
Typically, we solve the prediction problem on a set of consecutively connected equal-length intervals $\mathcal{I} = \{I_1, \ldots, I_s\}$, where $I_i = [t_{i_{\min}}^i, t_{i_{\max}}^i]$ and $t_{i_{\min}}^i = t_\lambda$. The problem is to predict $\text{Coverage}(S, I_i)$ and $\text{Intensity}(S, I_i)$ for each $I_i \in \mathcal{I}$, and $\text{Duration}_{\text{cov}}(S, \mathcal{I}, \theta_S)$ or $\text{Duration}_{\text{int}}(S, \mathcal{I}, \theta_S)$ for a given $\theta_S$.

### 3.2.2 Datasets

We take our observation and evaluation on two social networks: DBLP co-author network and Twitter user network.

**DBLP co-author network:** In this network, the nodes correspond to the authors and the edges correspond to the co-authorship. The dataset contained 934,672 nodes and 8,850,502 edges. The trend data are extracted from the DBLP data by detecting terms in the titles of publications. In each trend, an action $(v_i, t_i)$ corresponds to a publication of the author $v_i$ at time $t_i$ that contains the given term in the title. For publications with multiple authors, there is an action for each of the authors.

**Twitter user network:** In the network, the nodes correspond to the users and the edges correspond to the who-follows-whom relationships. The edges are directed from the users that are being followed to the followers. We randomly crawl a sub-network of Twitter network, which contains 40,906 nodes and 7,829,834 edges. The trends are defined by hashtags in tweets. An action $(v_i, t_i)$ in a trend corresponds to a tweet of user $v_i$ at time $t_i$ that contains the given hashtag.
3.3 Dynamic Activeness (DA) Model

3.3.1 Concept of Activeness

The DA model for trend in social network is based on the novel concept of activeness. For each trend, each node in the social network has an activeness function associated with it. The two main aspects of the concept are:

- Activeness of a node is defined as its interest toward the given trend. It is a function of time. As time goes by, activeness may increase as a result of information diffusion or social influence, or decrease as the node loses interest to the given trend.

- Activeness decides the frequency of actions taken by the node. The higher the activeness of a node is, the more actions it is likely to make in a unit time. In this sense, we can also define activeness as the “action rate” of a user.

Since activeness is dynamic in nature, we are able to design the DA model based on it, so that the model can capture the three dynamic characteristics of trends. Besides, by using activeness in the model, we are also able to design a more flexible information propagation mechanism, as we will show in Section 3.3.3.1.

3.3.2 Framework of DA Model

As shown in Figure 8, the DA model contains three elements: activeness propagation, decay of activeness and action generating process. Each of them is based on observations on real trends. Actions and activeness are connected to each other in the DA model. On the one hand, actions trigger activeness propagations in the social network. Activeness propagation, together with the decay of activeness,
decides the activeness of each user at each point of time. On the other hand, actions are generated by
the action generating process which takes the activeness as input.

As shown in Figure 2, the prediction algorithm contains two phases. In the learning phase, param-
eters of the DA model are learned from the observed part of trends. In the prediction phase, the DA
model predicts the actions in the future. Intensity, coverage and duration of trends can be predicted by
aggregating the predicted future actions.

![Figure 8. Block Diagram of the DA Model](image)

3.3.3 Activeness Modeling

For each trend $S$, let $r_v(t)$ be the activeness or action rate of node $v$ at time $t$. In this section,
we discuss the modeling of $r_v(t)$. As shown in the left most box in Figure 8, the model of activeness
contains two parts: activeness propagation and activeness decay.
3.3.3.1 Activeness Propagation

Intuitively, as a result of social influence or homophily (50; 6), the activeness of nodes are correlated with each other. The closer two nodes are, the larger the correlation is. Thus, when a node takes an action in a trend, we can expect that nodes in proximity to it have larger activeness for the trend than other nodes in the social network.

Our observation on the real trends supports this intuition. Figure 3.3.3.1 shows the curves of activeness for four trends in DBLP network, i.e., trends about “boosting”, “privacy” etc. Other trends have similar curves. For each trend, we plot the average activeness (i.e. the average number of actions per unit time) for nodes with different shortest path distances to nodes with previous actions. As we can see from the figure, as the distance increases, the activeness of nodes exponentially decreases. An important remark is that the information diffusion along direct links is not enough for explaining trends. Because if we explain trends in that way, all the nodes except for those that are directly linked to the nodes with previous actions should have the same activeness. It is also interesting to point out the exponential decreases also fits for action rates of nodes with 0-hop distance (i.e. nodes themselves have previous actions).

Based on this observation, we use the previous actions of a trend to model the activeness of nodes. Let $\text{prox}(u, v)$ be the proximity measurement from $u$ to $v$. When an action is taken by node $u$, the increase of activeness of $v$ is proportional to $\text{prox}(u, v)$, i.e., if $u$ takes an action at time $t_a$, for every node $v$ in the network we have:

$$\lim_{t \to t_a^+} r_v(t) = \lim_{t \to t_a^-} r_v(t) + \alpha \cdot \text{prox}(u, v)$$

(3.1)
We study two different measurements for proximity. The first one uses the shortest path distance from the source node to destination node. As we observed in the real trend data, proximity is defined as an exponential decreasing function of shortest distance, i.e., \( \text{prox}(u, v) = \exp(-b \cdot \text{dist}(u, v)) \), where

\[
\lim_{t \to t_a^+} r_v(t) \quad \text{and} \quad \lim_{t \to t_a^-} r_v(t)
\]

are the activeness of node \( v \) after and before the jump at time \( t_a \), and \( \alpha \) is the propagation ratio that depends on the trend.
$b \in \mathcal{R}^+$ and $\text{dist}(u, v)$ is the length of shortest path from $u$ to $v$. The second proximity measure is based on random walk, which is described as rooted PageRank in (34). To measure the proximity of nodes from a given node $u$, the random walk is started at node $u$. At each step, it has a probability of $p$ to return to node $u$, and $1 - p$ probability to move to neighbor nodes. The proximity of node $v$ from node $u$ is defined as the stationary probability for $v$. In both of the measurements, if $v$ is not reachable from $u$, we have $\text{prox}(u, v) = 0$.

While information diffusion models define the influence between nodes along the edges, the “propagation” of activeness captures a more general sense of correlation between activeness of nodes, rather than the process of information diffusion. Besides, it is different from information diffusion model in how it is parameterized. The parameter $\alpha$ of the activeness propagation depends only on the trend, but not on the node that takes the action. To the contrary, the information diffusion models usually have diffusion probabilities with individual edges as parameters, and the diffusion probabilities are constant for all the trends. We parameterize the activeness propagation in a different way for two reasons: First, it is simply not practical to make the parameters depend on trends and edges at the same time, since there will be not enough actions to be used for the learning of each parameter. Second, the main purpose of the proposed trend model is to predict the aggregate characteristics of different trends, so it is more meaningful to bind the parameter with the trends instead of the edges.

### 3.3.3.2 Decay of activeness

Intuitively, if a user is not exposed to any new information or influence from a certain trend, nor does he create any new content that belongs to that trend, the user’s interest to that trend will gradually
decay. In other words, the activeness of a node spontaneously decays if there is no new action taken by
nodes in proximity to it. This spontaneous decay is observed from real trends.

Figure 3.3.3.1 shows the average activeness for nodes as time progresses since last action in prox-
imity. Let \( R(v, k) = \{ u \in V | sp(u, v) = k \} \) represent the set of nodes that are \( k \) hops away from \( v \),
where \( sp(u, v) \) is the shortest path distance from \( u \) to \( v \). The X-axis of the figure is the time elapsed
since last action taken by nodes in \( R(v, 1) \). The Y-axis of the figure is the average activeness. As shown
in the figure, activeness decreases when the node is not exposed to a new action. Decrease of activeness
can roughly be regarded as exponential. (We only show the case of \( k = 1 \) and the curves for four trends
in this figure. But actually we have done the same test for different \( k \) values and for different trends, and
the other curves are similar.)

According to the observation, we introduce an exponential decrease to the activeness model. For
each interval \([t_0, t_1]\) when \( r_v(t) \) is not increased by the activeness propagation, \( r_v(t) = r_v(t_0)e^{-(t-t_0)/\tau} \)
for any \( t \in [t_0, t_1] \). \( \tau \) is a rate of the activeness decay. For similar reasons as to the parameter \( \alpha \), \( \tau \)
depends on the trend, but not depends on the node or the edge.

3.3.3.3 Summary of Activeness Model

Combining the two parts discussed in Section 3.3.3.1 and Section 3.3.3.2, \( r_v(t) \), the activeness of
node \( v \) at time \( t \), is given by:

\[
r_v(t) = \alpha \sum_{(v_i, t_i) \in S_t} (prox(v_i, v) e^{-(t-t_i)/\tau}) + r_v(t_0)e^{-(t-t_0)/\tau}
\]  

(3.2)
where \( t_0 \) is the start time of the trend. We set \( r_v(t_0) \) to a small value \( \epsilon \). \( r_v(t) \) is discontinuous at time points when there is a new action taken by nodes in \( R(v) \). In each interval between those discontinuous points, \( r_v(t) \) is subject to an exponential decay.

3.3.4 Action Generating Process

In this section, we discuss the generating process for actions. As we mentioned in Section 3.3.1, the activeness of a node serves as the action rate in the generating process. With the assumption that time points of actions are conditionally independent, given the activeness function, the generating process for the sequence of time points is a non-homogeneous Poisson process. This assumption keeps the model simple. We will first show that this assumption is reasonable for the action generating process, and then discuss the detail of the generating process under this assumption.

The verification is based on the fact that inter-action time of a homogeneous Poisson process, i.e. a Poisson process with constant action rate, follows exponential distribution. The sequence for each node usually contains only a few actions in a trend, which are not sufficient for the analysis. Instead, we use the global time sequence of all the actions in a trend. As a property of Poisson processes, the sum of two Poisson processes is also a Poisson process. For our case, if each individual action sequence is generated by a Poisson process, the global sequence is also generated by a Poisson process. The action rate function of this sequence is the sum of all nodes’ action functions, i.e., \( r(t) = \sum_{v \in V} r_v(t) \). Since \( r(t) \) reflects the global activeness of all nodes, it will not change too much in a short term. Therefore, the distribution of the inter-action time should roughly be an exponential distribution.

We plot the inter-action time distribution for three trends in the Twitter network in Figure 11. For each of the trends, we show the distribution of inter-action time during a week period when the trend
Figure 11. Distribution of inter-action time for three real trends. Other trends have similar curves.

is popular in the Twitter network. We divide the inter-action time into 20 bins according to the length, and plot the frequency of each bin. As showed by these figures, the inter-action time fits exponential distributions quite well. Thus, we conclude that the independent assumption is reasonable.

Under the conditional independent assumption, we derive the process of generating actions as follows. For a non-homogeneous Poisson process, the number of actions taken by this node in any time interval \([t', t)\) follows a Poisson distribution:

\[
P(\lambda = k) = e^{-\int_{t'}^{t} \lambda(t) \, dt} \frac{(\int_{t'}^{t} \lambda(t) \, dt)^k}{k!}
\]

where \(|S_{t'}^v|\) is the number of actions taken by node \(v\) before time point \(t\).

Suppose we are now at the time point \(t'\) and want to generate the next time point after \(t'\). By taking derivative with respect to \(t\), we can get the probability density function for the waiting time until next action:
\[ f_{v,t'}(t) = r_v(t) \cdot \exp\left( \int_{t'}^t r_v(t) dt \right) \] (3.3)

We then can generate the next time point in the sequence by drawing from the distribution of the waiting time.

### 3.4 Prediction Algorithm

In this section, we show prediction algorithm based on the proposed DA model. The algorithm contains two phases: the parameter learning phase and the prediction phase. In the first phase, parameters for each trend are learned from the part of the trend before \( t^* \) by maximum likelihood estimation. In the second phase, we use the learned model to predict the future trend sequence after \( t^* \).

#### 3.4.1 Parameter Learning Phase

For each trend, two parameters in the DA model need to be learned: the proportionality factor \( \alpha \) in activeness propagation and mean lifetime \( \tau \) of activeness decay. We use maximum likelihood estimation for the parameter learning.

The likelihood function is given by:

\[ L(\alpha, \tau) = \prod_{v \in V} f(T(S^v_{t^*}), |S^v_{t^*}|; \alpha, \tau) \] (3.4)
where $T(S^v_{t*})$ is the time sequence for node $v$’s actions before time $t_*$, and $|S^v_{t*}|$ is the number of actions taken by $v$ before time $t_*$. $f(\cdot)$ is the joint probability density function of the time sequence $T(S^v_{t*})$ and $|S^v_{t*}|$. According to (53), the joint probability density function is given as:

$$f(T(S^v_{t*}), |S^v_{t*}|; \alpha, \tau) = e^{-\int_{t_*}^{t} r_v(t) dt} \cdot \prod_{t_i \in T(S^v_{t*})} r_v(t_i)$$

(3.5)

Notice that we here assume that the $\epsilon$ in Equation 3.2 is small enough and negligible.

For simplicity’s sake, we introduce $H_v(\cdot)$ and $h_v(\cdot)$ as follows:

$$h_v(t, \tau) = \sum_{(v_i, t_i) \in S_t} (\text{prox}(v_i, v) e^{-(t-t_i)/\tau})$$

(3.6)

and

$$H_v(t, \tau) = \tau \sum_{(v_i, t_i) \in S_t} (\text{prox}(v_i, v)(1 - e^{-(t-t_i)/\tau}))$$

(3.7)

Take Equation 3.5, Equation 3.6 and Equation 3.7 into Equation 3.4, we get:

$$L(\alpha, \tau) = \prod_{v \in V} (e^{-\alpha H_v(t_*, \tau)} \cdot \prod_{t_i \in T(S^v_{t_*})} \alpha h_v(t_i, \tau))$$

$$= \alpha^{\left|S_{t_*}\right|} \prod_{v \in V} e^{-\alpha H_v(t_*, \tau)} \cdot \prod_{(v_i, t_i) \in S_{t_*}} h_v(t_i, \tau)$$

The log-likelihood function is then given by:

$$\log L(\alpha, \tau) = \left|S_{t_*}\right| \log \alpha - \alpha \sum_{v \in V} H_v(t_*, \tau)$$

$$+ \sum_{(v_i, t_i) \in S_{t_*}} \log(h_v(t_i, \tau))$$
By taking the partial derivative of $\log L(\alpha, \tau)$, we get the maximum-likelihood estimation of $\alpha$ and $\tau$.

$$\hat{\alpha} = \frac{|S_{t_*}|}{\sum_{v \in V} H_v(t_*, \tau)}$$ (3.8)

Fixing $\alpha$ to $\hat{\alpha}$, we get

$$\hat{\tau} = \text{argmax}[|S_{t_*}| \log \left( \frac{|S_{t_*}|}{\sum_{v \in V} H_v(t_*, \tau)} \right) - |S_{t_*}|$$

$$+ \sum_{(v_i, t_i) \in S_{t_*}} \log(h_{v_i}(t_i, \tau))]$$ (3.9)

Due to the complexity of $H(t, \tau)$ and $h(t, \tau)$ with regards to $\tau$, it is not possible to obtain a closed-form solution for the maximum-likelihood estimate of $\tau$. However, since $\tau$ is the only variable here, we can use any line search algorithm to find the $\hat{\tau}$, and techniques such as simulated annealing can be adopted to avoid falling into local optimal value.

### 3.4.2 Prediction Phase

After we learn the parameters $\tau$ and $\alpha$, we can generate the prediction of the action sequence after $t_*$. To do this, we keep track of the next action of each user in a list in the time order. Every time we pull the earliest action from the list and add it to prediction, then we update the list to capture the future actions that will be triggered by this action. The procedure is as follows:

1. Calculate $r_v(t_*)$, the activeness at time $t_*$ for each node $v$ in the network, using Equation 3.2.

2. Generate a next action for each node in the network from the pdf in Equation 3.3. Sort the actions in the ascending order of time, store them in a list $L$. 

3. While $L$ is not empty, pull the first action $(v_i, t_i)$ from it.

   i. if $t_i > t_{end}$, jump to step 4.

   ii. Add $(v_i, t_i)$ to $S'$, the predicted sequence.

   iii. For all the nodes that are reachable from $v_i$, update their activeness $r_v(t_i)$ by Equation 3.2.

   iv. Update the next action time using the pdf in Equation 3.3, and sort the list $L$ again.

4. Calculate **intensity**, **coverage**, **duration** using the predicted sequence $S'$.

   $t_{end}$ in Step 3-i is the end point of the last time interval on which we want to predict the trend. Notice that the sequence generating process is a random process. We may repeat Step 3 several times to get the average value of aggregates. If implemented in a naive manner, the calculation of $r_v(\cdot)$ can be very time-consuming. We will describe an efficient implementation in Section 3.4.3.

### 3.4.3 Efficient Implementation

As a property of the DA model, the effect of the sequence of actions can be considered as the sum of the individual effects by each single action. We call it **stacking principle**. We can reduce the complexity of prediction algorithm based on the stacking principle. The general idea is to transform a complicated formula to a summation over the sequence of actions, which is easier to calculate. The principle can be applied to both the parameter learning phase and prediction phase. We are going to show the details of the implementation in Sections 3.4.3.1 and 3.4.3.2, and discuss the time complexity in Section 3.4.3.3.

#### 3.4.3.1 Speedup of Parameter Learning Phase

For the parameter learning phase, the main complexity of computation comes from the calculation of $\sum_{v \in V} H_v(t_*, \tau)$ which is in both Equation 3.8 and Equation 3.9. It will be very inefficient if we
calculate the summation over all the nodes in the network in every step of the search algorithm. To reduce the complexity of calculation, we use the following equation:

\[
\sum_{v \in V} H_v(t_*, \tau) = \sum_{v \in V} \left[ \tau \sum_{(v_i, t_i) \in S_{t_*}} (\text{prox}(v_i, v)(1 - e^{-(t_* - t_i)/\tau})) \right]
\]

\[
= \tau \sum_{(v_i, t_i) \in S_{t_*}} [(1 - e^{-(t_* - t_i)/\tau}) \sum_{v \in V} \text{prox}(v_i, v)]
\]

The summation over all the nodes \(\sum_{v \in V} \text{prox}(v_i, v)\) in the last equation does not depend on \(\tau\). It implies that for each \(v_i\) that \((v_i, t_i) \in S_{t_*}\), we can calculate the summation \(\sum_{v \in V} \text{prox}(v_i, v)\) once and store it. Then in each step we only need to take a summation over all the actions in the trend sequence \(S_{t_*}\). Notice that the number of actions in \(S_{t_*}\) is usually much smaller compared to the total number of nodes in the graph. Therefore, it is much more efficient to be calculated than the original form.

### 3.4.3.2 Efficient Implementation of Prediction Phase

For the prediction phase, the action rate function \(r_v(t)\) in Equation 3.2 is a summation over all the actions in the network before time \(t\). According to the property of Poisson processes, we can generate an action sequence for each action \((v_i, t_i)\) before time \(t\) with the action rate function \(\alpha \cdot \text{prox}(v_i, v)e^{-(t - t_i)/\tau}\), and then adding all these sequences together to get the action sequence for \(v\).

In other words, instead of calculating \(r_v(t)\) for node \(v\) repeatedly, we can use the follow procedure to get exactly the same result: (1) For the action \((v_i, t_i)\), generate the sequence of actions that is caused by this single action for every node reachable from \(v_i\), (2) and then merge this sequence of actions with the current list of action to generate the new list. This stacking principle greatly simplifies the prediction algorithm.
3.4.3.3 Time complexity

Due to the use of stacking principle, we can reduce the time complexity of the algorithm. Let the values of $\alpha$ and $\tau$ be $n_p$-digit precision, and $m_{prox}$ be the maximum number of nodes in the proximity of a node, $|S|$ be the total number of actions in the sequence $S$. In the learning phase, calculating the intermediate result takes $O(m_{prox} \cdot |S|)$. The total number of steps of line search is $O(n_p)$, and each step of line search takes $O(|S|)$ to calculate the new value of objective function. Thus, the time complexity of learning phase is $O(n_p \cdot |S| + m_{prox} \cdot |S|)$. In the prediction phase, for each action in the $S$, the algorithm takes $O(|S|)$ time to generating the prediction sequence. The time complexity of prediction phase is $O(m_{prox} \cdot |S|)$. Thus, the total complexity is $O(n_p \cdot |S| + m_{prox} \cdot |S|)$. Usually, $n_p$ is much smaller than $m_{prox}$, we can think the time complexity as $O(m_{prox} \cdot |S|)$.

The time complexity of a naive algorithm that does not make use of the stacking principle is $O(n_p \cdot m \cdot |S| + m \cdot |S|^2)$. It is much slower than our algorithm. The time complexity of prediction model based on the IC model is $O(m_{degree} \cdot |S|)$, where $m_{degree}$ is the maximum number of nodes that are directly linked to a node. It is different from the time complexity of our algorithm in $m_{degree}$. That is because the IC model only considers the influence between nodes that are directly linked to each other. Our algorithm has similar time complexity with the IC model, though the IC model is a much more simplistic model.

3.5 Experiment

3.5.1 Algorithms and Performance Measures

As we mentioned in Section 3.3.3, for the DA model, we use two different measurements for the proximity between two nodes in the network. For the shortest path measurement (DA-sp), we set the de-
cay factor $b$ to 10 in the experiment. For the random walk measurement (DA-rw), the restart probability $p$ is set to 0.4.

We compare the DA model with three variants of the widely used IC model. All of the three variants assume that each action comes with a delay, so that they can be used to model dynamic trends:

1. **eExp** (Edge-dependent exponential delay model (48)) assumes that all the actions propagate through a certain edge are drawn independently from the same exponential distribution.

2. **tExp** (Trend-dependent exponential delay model (48)) assumes that delays for all the actions of a certain trend are drawn independently from the same exponential distribution.

3. **tEqu** (Trend-dependent equal-length delay model (31)) assumes that there is a fixed-length delay for all the actions of a certain trend.

Parameters of the three baselines model are learned by the algorithm proposed in (48). For the intensity prediction, we extend the IC variants with a “multiple actions factor” to allow a node to perform actions more than once. These extended IC variants will be explained later in Section 3.5.4.

To evaluate coverage and intensity, we use two measures: the error ratio and the coefficient of variation.

Error ratio is used to evaluate the goodness of the prediction compared with the true value. The formula of error ratio is given by:

$$ error\ ratio = \frac{|truth - prediction|}{truth} $$
Since all of the tested algorithms are stochastic algorithms, we also evaluate the variance of the outputs. For every test, we run each algorithm for multiple times and estimate the coefficient of variation. The coefficient of variation is estimated by:

\[ \hat{C}_v = \frac{s}{\bar{x}} \]

where \( s \) is the sample standard deviation, and \( \bar{x} \) is the sample mean.

### 3.5.2 Trend Data for Evaluation

As we described in Section 3.2.2, we use two social networks for the evaluation. For each network, we conduct experiments on multiple trends, as listed in Table V.

For **DBLP** dataset, we test 10 trends of hot keywords in the areas of data mining and machine learning. For each trend, we use the trend sequence before year 2005 (2005 excluded) as the training sequence, and the sequence from year 2005 to 2009 (2009 included) as the test sequence. We take each year as a time interval, on which we consider intensity and coverage.

For **Twitter** dataset, we test the algorithms on trends of 10 most popular hashtags in the last four months of year 2010. For each of the trends, we use the trend sequence from the 40th week to 47th week as training sequence, and the sequence from the 48th week to 52nd week as the test sequence. We take each week as a time interval, on which we consider intensity and coverage.

### 3.5.3 Coverage

In Figure 12, we plot the average error ratio for 5 consecutive time intervals after the end point of training sequence. As shown in the figures, error ratios for all the algorithms tend to increase as
Table V
TRENDS IN THE DBLP AND TWITTER DATASET

<table>
<thead>
<tr>
<th>DBLP</th>
<th>Twitter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concept drift</td>
<td>Boosting</td>
</tr>
<tr>
<td>Kernel methods</td>
<td>Privacy</td>
</tr>
<tr>
<td>SQL</td>
<td>Face recognition</td>
</tr>
<tr>
<td>Heterogeneous network</td>
<td>Decision tree</td>
</tr>
<tr>
<td>Active learning</td>
<td>Streams</td>
</tr>
<tr>
<td></td>
<td>Sarcasm</td>
</tr>
<tr>
<td></td>
<td>Twibbon</td>
</tr>
<tr>
<td></td>
<td>SocialMedia</td>
</tr>
<tr>
<td></td>
<td>Wikileaks</td>
</tr>
<tr>
<td></td>
<td>Geek</td>
</tr>
<tr>
<td></td>
<td>Awesome</td>
</tr>
<tr>
<td></td>
<td>SoundCloud</td>
</tr>
<tr>
<td></td>
<td>Android</td>
</tr>
<tr>
<td></td>
<td>HTML5</td>
</tr>
<tr>
<td></td>
<td>Apple</td>
</tr>
</tbody>
</table>

Figure 12. Error Ratio for Coverage Prediction

time progresses. For the DBLP dataset, the error ratio of tEqu deteriorates very quickly, while for the Twitter dataset, tExp is the worst one. In both figures, DA-rw and DA-sp have lower error ratios than the baselines for most of the ranges of X-axes, while eExp is in the middle. The difference between the error ratio of DA-rw and the error ratio of DA-sp is not large, which shows that the DA model is not sensitive to the different measurements of proximity, as long as the measurements are reasonable.
In Figure 13, we show the coefficient of variation for the coverage prediction. For the DBLP dataset, DA-rw and DA-sp have a lower coefficient of variation than the baselines, which means that the predictions got by the DA model are more stable. For the Twitter dataset, the coefficient of variation of tEqu is similar to the DA model. tExp has a strange decreasing curve, and its coefficient of variation is the lowest in the last two weeks of prediction. Considering the large error ratio of tExp in later weeks in Figure 12(b), its low coefficient of variation is most likely to be caused by a large mean value of its prediction. Thus, it is not comparable with the other three algorithms. Among all the algorithms, eExp have highest coefficient of variation, hence it is less stable than all the other algorithms. On both of the datasets, DA-sp has a slightly lower coefficient of variation than DA-rw, which mean that the shortest path distance measurement makes the result more stable than the random walk measurement.

### 3.5.4 Intensity

For the evaluation of intensity prediction, we use tEqu-mult, tExp-mult and eExp-mult instead of tEqu, tExp, and eExp as baselines. It is because the tEqu, tExp and eExp, like the standard IC model,
do not allow multiple actions being performed by the same node. To construct better baselines, we try to capture the relationship between the coverage and intensity. Figure 14 shows our observation on the relationship.

For each dataset, we calculate coverage and intensity for all the intervals in the training time, and plot each pair of coverage and intensity in this Figure 14. In both figures, the values of coverage are illustrated on the X-axes, and the values of intensity are illustrated on the Y-axes. As shown in Figure 14, for the DBLP trends, the proportional function $y = 1.1215x$ fits the relationship of intensity and coverage quite well, while for the Twitter trends, the $y = 1.2969x$ fits the relationship.

Based on this observation, we add a multiple action factor to the three baselines and get three new baselines. Figures 15(a) and 15(b) show the improvement of tEqu-mult from tEqu on the DBLP and Twitter datasets respectively. For each trend, we show the predictions of intensity made by tEqu-mult and tEqu for in the first prediction interval (year 2005 for the DBLP trends and the 48nd week for the Twitter trends) as well as the true value. As illustrated in the figure, tEqu tends to make a prediction
lower than the true value because it does not allow multiple actions taken by each node. tEqu-mult makes a better prediction by adding a multiple factor.

Figure 16 shows the error ratios for intensity prediction. As shown in the figures, for all the algorithms, error ratios tend to increase as time progresses. Among the four algorithms, tEqu-mult has the highest error ratio for the DBLP dataset, while tExp-mult is the worst one for the Twitter dataset. DA-sp performs well on the DBLP dataset, but has a high error rate on the Twitter dataset. That may due to the relatively smaller graph diameter of the Twitter dataset. DA-rw has lower error ratios than other algorithms for most of the ranges of the X-axes for both datasets.

In Figure 17, we show the coefficient of variation for the intensity prediction. Notice that the result is similar to the coefficient of variation for coverage prediction in Figure 13. It is because the result of tExp-mult is proportional to the result of tExp, their coefficients of variation should be the same. Therefore, the curves for tExp-mult in Figure 17 are exactly the same as the curves for tExp in Figure
So are the curves for tEqu-mult in Figure 17 and curves for tEqu in Figure 13. For the DBLP dataset, our algorithms have lower coefficient of variation than the baselines. For the Twitter dataset, the coefficient of variation of tExp-mult drops as time progresses, as a result of its large and inaccurate prediction for intensity, while the curves for DA-rw, DA-sp and tEqu-mult are similar. For both of the datasets, the prediction of eExp is less stable than the other four algorithms.

### 3.5.5 Duration

We test both coverage-based duration and intensity-based duration. To calculate the duration, we use the coverage and intensity at the last observed interval (year 2004 for the DBLP dataset, the 47th week for the Twitter dataset) as thresholds. Since the length of the test time is limited, coverage or intensity for about half of the trends never drops below the thresholds. We make each algorithm predict whether the duration covers all the 5 prediction intervals, and report the accuracy of this prediction.

Table VI shows the accuracy of the duration prediction. In the table, “C-D” and “I-D” are short for “Coverage-based Duration” and “Intensity-based Duration”. As shown in the table, DA-rw makes the
best predictions of accuracy in three of the four cases. The accuracy of DA-sp and tExp is similar except for the last case. tEqu has the lowest accuracy on the DBLP dataset, while eExp has lowest accuracy on Twitter dataset.

\[
\begin{array}{cccc}
| & DBLP & Twitter |
|----------------------------------|------------------|
| | C-D | I-D | C-D | I-D |
| DA-rw | 0.9 | 0.9 | 0.8 | 0.5 |
| DA-sp | 0.8 | 0.9 | 0.6 | 0.5 |
| tEqu(-mult) | 0.6 | 0.5 | 0.4 | 0.5 |
| tExp(-mult) | 0.8 | 0.9 | 0.6 | 0.7 |
| eExp(-mult) | 0.7 | 0.7 | 0.3 | 0.4 |
\end{array}
\]

**TABLE VI**
ACCURACY OF DURATION PREDICTION
3.5.6 Case Study

We select two trends for the case study: the trend for “android” in the Twitter network and the trend for “concept drift” in the DBLP network.

In Figure 18(a), we show the predicted and true coverage of the trend “android” in the last five weeks of 2011. The coverages predicted by DA-rw and DA-sp are close to the true value. Besides, only these two curves descend gently as the curve of true coverage. The coverage predicted by tExp quickly becomes much larger than the true value, while the coverage predicted by eExp is always very small. The coverage predicted by eExp fluctuates dramatically. If we choose $\theta = 86$ (the true coverage value in week 47), the coverage-based duration predicted by DA-rw, DA-sp will be 1 week, the same as the true value, while the duration predicted by three baselines will all be 0 week, since the predicted value falls below $\theta$ in the week 48.

In Figure 18(b), we show the predicted and true intensity of the trend “concept drift” in the years 2005-2009. All the curves keep increasing during the five intervals. But the intensity predicted by DA-rw and DA-sp is much closer to the true value than the value predicted by the baselines.

3.5.7 Variations of Parameters

Figure 19 shows the distributions of the parameters, mean life time $\tau$ and proportionality factor $\alpha$, for DBLP data set. As shown in the figure, each of the parameters has a large variation over trends. For example, most of the trends have $\tau$ (mean lifetime of activeness decay) around 4 years, but some trends have very small $\tau$ which is less than 1 year. The large variations show that the cascade processes for trends are different from each other in nature. As a result, it is necessary to learn the model for each individual trend.
3.5.8 Summary and Discussion

We summarize the result as follows: For the prediction of coverage, both DA-rw and DA-sp have better performance than all the three baselines. For the prediction of intensity and duration, DA-rw works better than the baselines on both datasets. While DA-sp makes better predictions than baselines on the DBLP dataset. For all the evaluations, DA-rw has comparable or better results than DA-sp.
The two datasets we used are very different in many aspects. For example, the time granularity of the DBLP dataset is one year, while in the Twitter dataset the time granularity is one millisecond. The DA model can make accurate predictions on both of dataset. It shows that the DA model is practical for various applications.

According different applications, it may not always be necessary to make predictions on all of the three measures. As shown in Figure 14, the linear relationship between intensity and coverage is very significant in the DBLP dataset, comparing to that of the Twitter dataset. That is because the intensity of trends in the Twitter dataset can easily be spammed by a small fraction of users, but it is much harder to publish a paper than posting a tweet. Therefore, for applications on the DBLP network, the measures of intensity and coverage are roughly interchangeable. For the DBLP dataset, we may only need to predict one of them. Nevertheless, each of the three measures is useful for some applications, and it is desirable to propose a model that can capture all of them at the same time.

3.6 Related Work

Information diffusion processes in social networks have been intensively studied in (25; 17; 33; 31; 9; 20; 27). Most of this work considers trends in social network as the results of information diffusion processes. These papers focus on predicting user-level behaviors rather than the aggregated measures of trends. They usually model trends on a discretized time, which makes them not practical for predicting dynamic properties of trends. Independent Cascade (IC) model and its variants are the most widely studied models for information diffusion processes (25; 31; 9). Most of this work assumes that the models are given as an input and try to solve the influence maximization problem on the model. The work in (49; 48; 17; 13) focuses on learning the diffusion probabilities of the cascade models. The
work in (48) defines a variant of the IC model that models the diffusion delay as a random variable with an exponential distribution, and provides an inference algorithm for the model. The work in (51) proposes a microscopic social influence model, in which influence is modeled as “heat” that flows through the network. It is conceptually resemble to the activeness in our model. But the approaches are fundamentally different, and their model focuses on the prediction on each individual node.

Some existing work argues that the information diffusion inside social networks is not the only explanation of trends. Homophily is also considered to be important for the modeling of trends (50; 6; 1), and the external influence from outside the network is studied in (41).

Recently, some work studies real-life trends in online communities. Most of this work focuses on analyzing observed trends rather than predicting future trends, and does not make use of the structure of the social network. The work in (46) studies the dynamic of trends on the Twitter community. The work in (33) analyzes the cascade behaviors on blogspace, and the work in (30) analyzes trends on news website and blogs. The work in (6) provides a technique to identify popular trends, which utilizes the structural information of social networks.
CHAPTER 4

STEERING INFORMATION DIFFUSION DYNAMICALLY


4.1 Introduction

Recently, studies on information diffusion in social networks have drawn significant attention because of their promising applications on viral marketing. To maximize the impact of viral marketing, many researchers studied the problem of influence maximization. The problem is usually defined as follows: given a social network, how can we select an initial set of $k$ nodes so that the expected number of active users is maximized. This definition captures an important aspect of viral marketing, the initiation of an information diffusion process, but it fails to capture the other aspect, the control over an information diffusion process once it started.

In the real world, viral marketing campaigns are usually sold by social network websites, such as Twitter and Facebook, as services to their clients, such as Apple and Coca-Cola (40). The social network websites can not only select the initial set to start an information diffusion process for an item, but can also steer the diffusion process by deciding whether an individual user should be notified about the item, and what the proper timing of notifying that user is. The control of social network websites over
the diffusion processes is crucial for the optimization of viral marketing. However, few existing works studied this topic.

It is common practice of social network websites to notify individual users with selected items, or “push” items to selected users. In social networks, there are usually a large number of different items propagate simultaneously, including user-created posts, promoted posts, advertisements, etc. Users can easily be exposed to more items than they can pay attention to. By pushing items to users, the websites can draw the attention of users to the items that they want the users to read. To do that, many social network websites send users messages or emails about selected items. Websites also put selected items on the top of the news feeds they provide to users to draw the attention of users to those items.

When websites conduct viral marketing campaigns, they “push” advertisements or promoted items to users. A problem that they usually face is limited user attention. If websites push too many advertisements or promoted items to users, users will be not able to read all the items. Besides, too many advertisements or promoted items will hurt user experience. To deal with that problem, websites usually set a “push budget”, or the number of times each advertisement or promote item will be pushed to users. The push budgets are usually decided by the money that clients pay for the campaigns.

Steering the information diffusion process by “pushing” items to users is related to the studies of recommender systems, in the sense that both of them provide users with items that are interesting to them. Recommender system techniques such as collaborative filtering can be adopted by the diffusion steering algorithm to predict which users may potentially be interested in an item. However, steering the information diffusion is more than pushing items to users who are most interested in them. First, for an information diffusion process, the social influence between users is as important as the personal
preference of users. **Second**, the diffusion of information is a dynamic process. The website can have impact throughout the diffusion process. A decision made by the website may affect users’ interest towards an item, and then affect later decisions. **Third**, as the user interest changes over time, the proper timing of notifying a user about an item is crucial.

In this chapter, we explore the diffusion of information from the perspective of social network websites. Motivated by the observations on “limited user attention” and “co-effect of user preference and social influence”, we propose a novel push-driven cascade (PDC) model, which is natural for the scenario of viral marketing in online social networks. The dynamic influence maximization problem based on the PDC model naturally combines two important aspects of influence maximization: the initiation of a diffusion process and the control over the diffusion process. We show that the problem can be formalized as a Markov sequential decision problem, which always has an optimal deterministic Markovian solution. We develop an AO* search algorithm which finds the optimal solution for the problem. Inspired by the optimal algorithm, we also propose a heuristic search algorithm, which is orders of magnitude faster than the optimal algorithm. We evaluate the proposed algorithms on various real social network datasets. The experiment results show that proposed algorithms achieve significantly larger influence spread than the baselines.

### 4.2 Motivation

Different from information diffusions in offline social networks, diffusions in online social networks highly depend on the social network providers (websites). In this chapter, we study the diffusion of information from the perspective of social network websites.
Consider an online social network. At any point of time, there is usually a large set of various items propagate in the network. For example, there are user-created posts, links to external websites shared by users, and advertisements and promoted posts paid by clients of the website. The diffusion of each item is described by an information diffusion process. A main motivation of our work is limited user attention. Because of the huge number of items propagating in a social network, users are usually not able to read all the items, and the website has to select certain items to bring them to the attention of users by sending the users messages, or putting the items on the top of news feeds when the user visits the social network website. We say that the website “pushes” these items to the users. If a user finds any item interesting, she can take an action for the item, for example “like” the item in Facebook, “+1” the item in Google+, or “retweet” the item in Twitter. We say this user becomes active for this item when she takes an action for it. Since user attention is limited, we need to limit the number of times we pushing an item. In other words, we set a “push budget” for each item.

The problem we study in this chapter is: given a push budget, how can a website pushes an item to users to steer the diffusion processes, so that the expected number of active users (which is usually referred to as information spread) is maximized? A straightforward strategy for this problem is pushing the item to users who are most likely to be interested in it. The strategy is made possible by extensive studies in recommender systems, which provide various techniques for estimating user preference. Another possible strategy for steering the diffusion is maximizing the social influence. The studies in social influence and information diffusion find that the interest of users can be influenced by their friends, and items can propagate through the social network as a result of social influence. These studies suggest that we should help the diffusion of information by pushing the item to users with large influence to other
users. We believe that both strategies are useful for maximizing the information spread, and our solution for the dynamic influence maximization problem combines the user preference and social influence.

To better explain our motivation, we would like to show analysis results on a few datasets as examples.

**Limited user attention.** We use two social network datasets, “Twitter-friends” and “UIC-followers” as examples. Both datasets are collected from Twitter. The details of them are shown in Section 4.5.1.

We explore the relationship between the “show number”, the number of items that the website shows to users, and the “attention number”, the number of items that are actually read by the users. Every time when a user visits the website, the website will show the users a few items. If user attention is unlimited, the user will read all the items shown by the website, so the “attention number” will always be the same as the “show number”. Otherwise, if we observe that the “attention number” stays the same when the “show number” increases, it implies that user attention is limited.

Unfortunately, neither the “show number” nor the “attention number” is publicly available. However, we can infer the relationship between them from the Twitter datasets based on two observations. First, the Twitter website shows the items (tweets) to users in chronological order, in which the latest items come first. When a user visits twitter.com, the website will firstly show him the items that are created after his last visit to the website, so we can roughly use the number of items that are created by friends of a user after his last visit as the “show number”. Second, taking an action for an item (retweet) is very quick and easy in Twitter, so whether a user takes an action or not mainly depends on whether he is interested in the item or not. Assuming that the probability that a user is interested in an item is stable over time, the number of retweets is roughly proportional to the “attention number”.

In Figure 20, we show the relationship between the “show number” and the number of retweets for each visit of users to the Twitter website. Each data point in the figure corresponds to a visit of a user. The X-axes illustrate the number of tweets created by the friends of the user since his last visit to the website, while the Y-axes illustrate the number of retweets made by this user during this visit. The red curves in the figures show the average number of retweets for varying “show number”. As shown in the figure, the number of retweets does not increase as the “show number” increases. Actually, they are most likely to be independent, as the correlation coefficient between them is very close to 0 ($1.46 \times 10^{-5}$ for the Twitter-friends dataset and $7.44 \times 10^{-3}$ for the UIC-followers dataset). It implies that no matter how many items are shown to a user during his visit, the number of items he really pays attention to is almost fixed. It follows that the attention of a user is limited.

**Co-effect of user preference and social influence.** To build a reasonable model for information diffusion, we need to understand the factors that influence the behavior of users. To be specific, we would like to find out whether user preference and social influence should be included in the model of user
actions. We use a Foursquare dataset for this analysis. The details of the dataset are described in Section 4.5.1.

We study the “check-in” action of users in the Foursquare network. In Foursquare, when a user checks in at a location, her friends will get notified about her “check-in”. To explore the factors that influence the “check-in” actions of users, we use a matrix factorization model (26), a widely used technique for recommender systems, to analyze user behavior. The model maps both users and items to a joint latent factor space of dimensionality $f$. Each user $u$ is associated with an $f$-dimension vector $p_u$, while each item $i$ is associated with an $f$-dimension vector $q_i$. Elements of $p_u$ measure the strength of preference the user has to the latent factors, while elements of $q_i$ measure the extent to which the item possesses those factors. In addition to that, we model the influence from a user $v$ to a user $u$ with influence weight $w_{vu}$. Formally, the model is defined as follow:

$$
\hat{a}_{ui} = \mu + b_u + d_i + p_u^T q_i + \sum_{v \in N(u)} w_{vu} a_{vi}
$$

where $N(u)$ is the set of in-neighbors of $u$. $a_{vi}$ is set to 1, if $v$ is active for the item $i$, and 0, otherwise. $\hat{a}_{ui}$ is the estimated value of $a_{ui}$ made by the model. $\mu$, $b_u$, $d_i$ are the global bias, the user bias, and the item bias, respectively. We can learn parameters $\mu$, $b_u$, $d_i$, $p_u$, $q_i$ and $w_{vu}$ by minimizing the regularized squared error using stochastic gradient descent (26).

This model considers both the user preference and the influence from the friends of the user. We compare the model with three partial models: the bias only model, the social influence model, and the user preference models. The equations of them are listed in Table VII. Since Foursquare pushes an item
to a user only when one of his friends is active for the item, we construct an observed item set \( O_u \) for each user \( u \), which contains the locations that at least one of his friends have checked in. Each pair of \((u, i)\) such that \( i \in O_u \) is considered as a sample.

We conduct a 10-fold cross validation for each model, and evaluate them by mean squared error (MSE) and average rank (\( \text{rank} \)). (Average rank is the average rank of all the positive samples in the test set, when we rank all the test samples in the descending order of estimated value \( \hat{a}_{ui} \). A smaller average rank indicates a better prediction result (22).) As shown in Table VII, both the social influence model and the user preference model significantly outperform the bias only model, and the full model is the best one of all. This result implies that both user preference and social influence have effects on the behavior of users. A proper model for the diffusion of information should combine both two factors.

Although the user preference model performs better than the social influence model and the improvement of the full model from the user preference model is marginal, it does not mean that social influence is a less important factor than user preference. That is because of the way that the sample dataset is constructed: for each sample \((u, i)\) in the dataset, \( i \in O_u \), i.e. at least one friend of \( u \) is active for the item \( i \). If we consider all the possible pairs \((u, i)\) of users and items as samples, the social influence factor will be much more useful.

4.3 Push-driven cascade model

With limited user attention, information diffusions in online social networks highly depend on how the websites push items to users. The decisions of websites are made dynamically throughout the diffusion processes. However, existing information diffusion models ignore the role of websites during
the diffusion process. They assume that the information will automatically diffuse through the social networks, and a user will always notice all the action of his friends.

Based on the observations in last section, we propose a new information diffusion model, the push-driven cascade (PDC) model, which is more natural for the scenario of viral marketing in online social networks. In the PDC model, the website pushes an item to social network users. To avoid consume too much user attention, there is a push budget $L$ for the item, i.e. the number of times that the website can push the item to users. When being pushed to, a user may become active for the item or not. The probability that a user becomes active for an item is decided by two factors: the preference of the user and the influence from his friends. The preference of a user decides how likely the user will become active for the item when there is no social influence. Social influence increases the probability of a user becoming active, when some friends of the user are active. The model is formally defined as follows.

**Definition 11** Push-driven cascade model (PDC). For each user $u$ in the social network, there is a preference $b_u \geq 0$ for a given item. For each directed edge $e=(v,u)$ in the social network, there is an influence weight $w_{vu} \geq 0$, which determines how much $u$ is influenced by his friend $v$. For each user $u$ in the social network, the preference $b_u$, the influence from his friends, and the push budget $L$, the probability $p_u$ that a user $u$ becomes active for an item is given by:

$$p_u = \frac{\mu + b_u + d_i + \sum_{v \in N(u)} w_{vu} a_{ui}}{L}$$

where $\mu$ is a constant, $d_i$ is the diffusion factor, $a_{ui}$ is the action of user $u$ on item $i$, and $N(u)$ is the set of friends of user $u$.

In the PDC model, the website pushes an item to social network users. To avoid consume too much user attention, there is a push budget $L$ for the item, i.e. the number of times that the website can push the item to users. When being pushed to, a user may become active for the item or not. The probability that a user becomes active for an item is decided by two factors: the preference of the user and the influence from his friends. The preference of a user decides how likely the user will become active for the item when there is no social influence. Social influence increases the probability of a user becoming active, when some friends of the user are active. The model is formally defined as follows.

**Definition 11** Push-driven cascade model (PDC). For each user $u$ in the social network, there is a preference $b_u \geq 0$ for a given item. For each directed edge $e=(v,u)$ in the social network, there is an influence weight $w_{vu} \geq 0$, which determines how much $u$ is influenced by his friend $v$. For each user

<table>
<thead>
<tr>
<th>Model</th>
<th>Equation</th>
<th>MSE</th>
<th>rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>bias only</td>
<td>$\mu + b_u + d_i$</td>
<td>0.01200</td>
<td>0.24934</td>
</tr>
<tr>
<td>social influence</td>
<td>$\mu + b_u + d_i + \sum_{v \in N(u)} w_{vu} a_{ui}$</td>
<td>0.01131</td>
<td>0.20091</td>
</tr>
<tr>
<td>user preference</td>
<td>$\mu + b_u + d_i + p_u^T q_i$</td>
<td>0.00950</td>
<td>0.13805</td>
</tr>
<tr>
<td>full model</td>
<td>$\mu + b_u + d_i + p_u^T q_i + \sum_{v \in N(u)} w_{vu} a_{ui}$</td>
<td>0.00939</td>
<td>0.13777</td>
</tr>
</tbody>
</table>

TABLE VII

RESULTS OF DIFFERENT USER BEHAVIOR MODELS
$u$, the weights and the bias satisfy $b_u + \sum_{v \in N_u} w_{vu} \leq 1$, where $N_u$ is the set of in-neighbors of $u$. We denote whether $u$ is active with $a(u)$. $a(u) = 1$, if $u$ is active, and 0, otherwise. The diffusion process takes place in discrete steps $t = [1, \ldots, L]$. At the beginning of the diffusion process, $a(u) = 0$ for each user $u$. In each step $t$, the website pushes the item to a user $u_t$. The user $u_t$ becomes active with probability $q(u_t) = b_{u_t} + \sum_{v \in N_{u_t}} w_{vu_t} a(v)$. If $u_t$ does not become active at step $t$, the website may push to it again in the later steps.

The spread of information based on the PDC model depends on the sequence of users $[u_1, u_2, \ldots, u_L]$ that the website decides to push the item to. It is analogous to the initial set for traditional diffusion models. However, in the PDC model, it is not necessary for the website to decide the sequence of users before the diffusion process starts. Instead of deciding the sequence beforehand, the website may hold a policy to decide which user to push to at each step based on the outcomes of previous steps. Since the users take actions in a stochastic manner, it is beneficial if the website decides the sequence dynamically throughout the process. Similar to the influence maximization problem on the traditional diffusion model, we can define a dynamic influence maximization problem on the PDC model, which finds the best policy to maximize the number of active users given the “push budget” $L$.

Before we proceed to the dynamic influence maximization problem, we now would like to point out a few more considerations about the model definition. **First**, in online social networks, there is usually a large number of different items diffusing simultaneously. Each of them has a separate PDC model. These models may share the same set of influence weight $w_{vu}$, but have different sets of user preference $b_u$. **Second**, similar to the linear threshold model, we use a linear model for user behavior, i.e. the probability is the sum of user preference and the influence from each friends of the user. We adopt
the linear model based on the observation in Section 4.2. However, it can easily be replaced by more sophisticated user behavior models, since our solutions to the dynamic influence maximization problem in the following sections do not depend on this linear model. Third, in this chapter, we mainly focus on the dynamic influence maximization problem rather than the inference problem of the PDC model, so we assume that the parameters $b_u$ and $w_{vu}$ are given. However, the parameters can be learned from user activity data. The stochastic gradient descent method used in Section 4.2 is a possible inference algorithm for learning these parameters. Actually, a wide variety of works (26; 22) in the recommender system area can be adopted for the learning of $b_u$ and a lot of works on information diffusion models (17; 48) can be adopted for the learning of $w_{vu}$. Fourth, in the real world, since users are not always online, they may not respond immediately when the website pushes items to them. The website may not be able to wait for their responses before it pushes the item to other users. In this chapter, we assume that the website gets the responses from users immediately, and leave the more complicated situation for future work.

4.4 Dynamic influence maximization problem

4.4.1 Problem definition

Now we study the dynamic influence maximization problem on the PDC model, which is analogous to the influence maximization problem for the independent cascade model or the linear threshold model in (25). As we have discussed in Section 4.3, the dynamic influence maximization problem is different from the traditional influence maximization problem in that the website can make decision dynamically during the diffusion process, and utilize the outcome of previous steps. Thus, while the solution to an in-
fluence maximization problem is an initial set of users, the solution to a dynamic influence maximization problem is a policy which decides the user to push to given the outcome of previous steps.

Let $g_t$ be the random variable that denotes the outcome of the “push” at the $t$-th step, i.e. $g_t = 1$, if $u_t$ becomes active at step $t$, and 0 otherwise. Let $H_t = [(u_1, g_1), \ldots, (u_t, g_t)]$ be the history up to the $t$-th step ($H_0$ is an empty sequence), and $\mathcal{H}$ be the space of all possible histories. A policy decides which user to push the item to at each step $t$ based on $H_{t-1}$, the history up to the $(t - 1)$-th step. Formally, a policy is defined as follows:

**Definition 12 Policy.** A policy $\pi : (\mathcal{H}, V) \mapsto [0, 1]$ maps each pair of history and user to a probability value, such that for any $H \in \mathcal{H}$, $\sum_{u \in V} \pi(H, u) = 1$.

We adopt a general definition of policy, both deterministic and randomized policies are covered by this definition. For deterministic policies, given any $H \in \mathcal{H}$, there is a single user $u \in V$, such that $\pi(H, u) = 1$, and for any other user $v \neq u$, $\pi(H, v) = 0$. For randomized policies, for each $H \in \mathcal{H}$, $\pi(H, \cdot)$ defines a distribution over the user set $V$, which decides the probabilities that users are selected by the policy to push the item to.

Suppose there is a website that follows a policy $\pi$. At each step $t = 1, \ldots, L$, the website picks up $u_t$ from the distribution $\pi(H_{t-1}, \cdot)$. We denote with $u(\pi)$ the expected number of active users when the process ends, i.e. $u(\pi) = E(\sum_{t=1}^{L} g_t^\pi)$, where $g_t^\pi$ is the outcome at the $t$-th step.

The dynamic influence maximization problem is to find the best policy that maximizes the expected number of active users. It is formally defined as follows:
Definition 13 Dynamic influence maximization problem. Given a social network, a PDC model on the social network, and a push budget $L$, find a policy $\pi$, such that $u(\pi)$, the expected number of active users when the process ends, is maximized.

Dynamic influence maximization problem as an MDP. For the dynamic influence maximization problem on a PDC model the number of all possible policies is infinite. Fortunately, we will shortly see that the dynamic influence maximization problem actually belongs to a class of problems named Markov decision process (MDP), which have been well studied by researchers in the area of artificial intelligence for decades (43; 21). We can utilize the results about the MDPs to reduce the number of policies that we need to consider.

A Markov decision processes (MDP) is a discrete time stochastic process, which is partially controlled by a decision maker. The process takes place on a set of states. At each step, the decision maker chooses an action, gets the reward from the action, and then moves to the next state. The transition probability depends on the current state and the chosen action. The process possesses the Markov property, i.e. the transition probability is conditionally independent of the history given the current state. The solution to an MDP is a policy that maximizes the total reward. We mainly focus a special class of MDPs called finite-horizon Markov decision process (FH-MDP), which is formally defined as follows:

Definition 14 Finite-horizon Markov decision process (FH-MDP). An FH-MDP is defined on a finite set of states $S$, a set of actions $D$, and finite steps $T = [1, 2, \ldots, L]$. For each state $s \in S$, there is a set of actions $D_s \subset D$ that are available in that state. For each action $a \in D_s$, $r_s(a)$ denotes the expected immediate reward of taking action $a$ in state $s$, and $p_{ss'}(a)$ denotes the probability that taking action $a$ in state $s$ results in a transition to state $s'$. Given $s$, $s'$ and $a$, the transition probability $p_{ss'}(a)$
is conditionally independent of the history before arriving at state $s$. The process starts at a given state $s_0$. At each step $t$, the decision maker can select an action from the set of actions that are available in the current state, gets the reward, and moves to the next state that is randomly selected according to the transition probabilities.

A policy in FH-MDP is a map $\pi : (\mathcal{H}, D) \mapsto [0, 1]$ such that for any $H \in \mathcal{H}$, $\sum_{a \in D} \pi(H, a) = 1$, where $\mathcal{H} = \{[s_0, s_1, \ldots, s_t]|t \in T, s_i \in S$ for $i = 1, \ldots, t\}$ is the set of all the possible sequence of states starting from state $s_0$. The definition is similar to Definition 2, except for that the history is now defined differently. A solution to an FH-MDP is a policy that maximizes the expected total reward.

We now show that a dynamic influence maximization problem can be equivalently defined as an FH-MDP: Consider a PDC model, we can define a state by the set of users that are active. For each $A \subset V$, we denote with $s_A$ the state with the set of active users $A$. For each user $u \in V$, we define an action of pushing the item to $u$. For simplicity of notation, we denote the action as action $u$. For each state $s_A$, the set of actions that are available in that state is $V \setminus A$. Let $q(A, u) = b_u + \sum_{v \in N_u} w_{vu} I_{v \in A}$ be the probability that $u$ becomes active when being pushed to, where $I_{v \in A} = 1$ if $v \in A$, and 0 otherwise. For each successful “push”, there is a reward of value 1, so the expected immediate reward of taking action $u$ in state $s_A$ is $q(A, u)$, and the transition probability $p_{s_A s_{A'}}(u)$ is $q(A, u)$ for $A' = A \cup \{u\}$, $1 - q(A, u)$ for $A' = A$, and 0 for other cases. It is obviously that, given $s_A$, $s_{A'}$, and $u$, the transition probability is conditionally independent of the history before arriving state $s_A$. Thus it is an FH-MDP on the steps $T = [1, 2, \ldots, L]$.

For FH-MDPs, the following result exists:
Theorem 1  \textit{Optimality of deterministic Markovian policies} (44). For an FH-MDP with finite $A_i$ for each $s_i \in S$, there exists a deterministic Markovian policy which is optimal.

In the theorem, “Markovian” means that the decision is conditionally independent of the history, given the current state $s$ and the time $t$, while “deterministic” means that the decision is a fixed action, rather than a randomized action.

According to Theorem 1, for any dynamic influence maximization problem, there exists a deterministic Markovian policy that is optimal. Since we only want to find one optimal policy, we only need to consider the deterministic Markovian policies. Thus, we have reduced the number of policies to be considered from infinite to $N^{|V|}$, where $N = \binom{|V|+1}{L}$ is the number of all possible states\textsuperscript{1}.

For a deterministic Markovian policy $\pi$, given any state $s$ and time $t$, only one fixed action is taken. Thus, a policy $\pi$ can be considered as a map from the space $S \times T$ to the set of actions $A$. Since we only consider the deterministic Markovian policies now, we can redefine a policy as $\pi : S \times T \mapsto A$, where $\pi(s, t)$ is the action selected by policy $\pi$ at state $s$ and time $t$. Since $\pi(s, t)$ depends on the state $s$ and the step $t$, for the convenience of description, we regard a pair of state and time $(s, t)$ as an \textit{augmented state}, and the space $S \times T$ as the \textit{augmented state space}. In the rest of this chapter, when it does not cause confusion, we refer to augmented state $(s, t)$ as state $(s, t)$ for short. We say $(s, t)$ is an \textit{ancestor} of $(s', t + 1)$ and $(s', t + 1)$ is a \textit{successor} of $(s, t)$, if for some action $a \in D_s$ the transition probability $p_{ss'}(a) > 0$. We say an augmented state $(s, t)$ is a \textit{terminal state}, if no further action can be taken at it.

\textsuperscript{1}The number of states with $l$ active users are $\binom{|V|}{l}$. The total number of all possible states are $\sum_{l=0}^{L} \binom{|V|}{l} = \binom{|V|+1}{L}$.
For the dynamic influence maximization problem, an augmented state \((s_A, t)\) is a terminal state if \(t = L\) or \(A = V\).

Although we have reduced the number of policies that we need to consider from infinite to finite, \(N^{|V|}\) is still a huge number such that a brute-force search for the best policy is impossible. For FH-MDPs, *backward induction* is the usual method for finding the optimal policy (44). The basic idea of backward induction is using dynamic programming to calculate \(u(s, t)\), the optimal expected total reward starting from augmented state \((s, t)\), in the backward order (from \(t = L\) to 0).

Although the backward induction is a straightforward algorithm that finds the optimal policy, it is not practical for the dynamic influence maximization problem. The disadvantage of the backward induction algorithm is that it evaluates the entire augmented state space \(S \times T\), but actually only a subset of the space needs to be evaluated in order to find the best policy for the process starting from state \(s_0\). To reduce the non-necessary evaluation, we introduce the AO* search algorithm for the dynamic influence maximization in the next section.

### 4.4.2 AO* optimal search algorithm

AO* is a heuristic search algorithm that can find solutions for MDPs with acyclic state graph. Similar to the famous A* search algorithm, the AO* algorithm utilizes a heuristic function to avoid evaluating the entire search space. With an admissible heuristic function, the AO* algorithm finds the optimal solution of acyclic MDPs.

To understand how AO* search algorithm reduces the search space, we first show that the dynamic influence maximization problem can be considered as a search problem on the hypergraph of augmented states. As an example, Figure 21 shows the hypergraph for a simple dynamic influence maximization
problem with two users and \( L = 2 \). In the hypergraph, each augmented state \((s_A, t)\) is represented by a node. For each action \( v \) that is available at state \( s_A \), there is a 2-connector that directs from the node \((s_A, t)\) to its two successors \((s_A, t + 1)\) and \((s_{A \cup \{v\}}, t + 1)\). For example, when taking action \( u_1 \) at state \( s_\emptyset \), the possible resulting states are \( s_\emptyset \) and \( s_{\{1\}} \), so there is 2-connector in Figure 21 from augmented state \((s_\emptyset, 0)\) to the two successors \((s_{\{1\}}, 1)\) and \((s_\emptyset, 1)\).

A policy \( \pi \) for a dynamic influence maximization problem can be represented by a subgraph of the augmented state hypergraph, called a solution graph. For policy \( \pi \), the solution graph \( G_\pi \) is defined inductively as follows:

1. State \((s_\emptyset, 0)\) is in \( G_\pi \).
2. For any augmented state \((s_A, t)\) in \(G_\pi\) that is not a terminal state, exactly one out-going 2-connector of it is contained in \(G_\pi\), which indicates the action \(\pi(s_A, t)\). The two successors connected by the 2-connector are also contained in \(G_\pi\).

As an example, in Figure 21, the subgraph marked with the red color is a solution graph \(G_\pi\), with \(\pi(s_\emptyset, 0) = u_1\), \(\pi(s_{\{1\}}, 1) = u_2\), and \(\pi(s_\emptyset, 1) = u_1\). The augmented state \((s_\emptyset, 0)\) is in the solution graph, and its outgoing 2-connector \(u_1\) is contained in \(G_\pi\). Thus, both \((s_{\{1\}}, 1)\) and \((s_\emptyset, 1)\) are in \(G_\pi\). Similarly, for augmented states \((s_{\{1\}}, 1)\) and \((s_\emptyset, 1)\), the 2-connectors correspond to actions \(u_2\) and \(u_1\) respectively are contained in \(G_\pi\).

Notice that the solution graph representations for two different policies may be the same, as the graph representation of a policy \(\pi\) only includes the augmented states that can be reached from state \((s_\emptyset, 0)\) by following policy \(\pi\). However, since the expected number of active users of policy \(\pi\) only depends on the augmented states that can be reached from state \((s_\emptyset, 0)\) by following policy \(\pi\), any two policies \(\pi_1\) and \(\pi_2\) with the same solution graph are virtually the same with respective to the dynamic influence maximization problem. For simplicity of description, in the rest of this chapter, we do not distinguish between a policy and its solution graph representation, and consider two policies with the same solution graph representation as the same policy.

The dynamic influence maximization problem can be considered as a search problem for the solution graph with the greatest expected total reward. The AO* algorithm starts the search at state \((s_\emptyset, 0)\), and directs the search using a heuristic function. Comparing to the backward induction algorithm, the AO* algorithm reduces the number of augmented states to be evaluated in two ways: First, for any augmented states that cannot be reached from \((s_\emptyset, 0)\) by any action (marked with dashed outline in Figure 21), AO*
does not evaluate them. **Second**, for some augmented states that cannot be reached from \((s_\emptyset, 0)\) by following the optimal policy (marked with gray solid outline in Figure 21), AO* avoids evaluating them by using a heuristic function to focus the search. As long as the heuristic function is admissible, AO* can still find the optimal solution.

To describe the AO* search algorithm, we first introduce a few new notations. Given a policy \(\pi\), let \(f_\pi(s_A, t)\) be the expected total reward starting at state \(s_A\) in step \(t\) when policy \(\pi\) is followed. It is obvious that for the cases with \(t = L\) or \(A = V\), the expected total reward is 0, since no more users can become active. For any other cases, the expected total reward is the sum of the immediate reward \(r_{s_A}(u_\pi)\) and the expected future reward when policy \(\pi\) is followed. Formally, it is defined as:

\[
f_\pi(s_A, t) = \begin{cases}
0 & \text{if } t = L, \text{ or } A = V, \\
r_{s_A}(u_\pi) + q(A, u_\pi)f_\pi(S_{A \cup \{u_\pi\}}, t + 1) + (1 - q(A, u_\pi))f_\pi(S_A, t + 1), & \text{otherwise.}
\end{cases}
\]  

The dynamic influence maximization problem can then be equivalently described as finding a policy \(\pi\), such that the expected total reward \(f_\pi(s_\emptyset, 0)\) is maximized. Let \(\pi_{\text{opt}}\) be the optimal solution, we denote with \(f(s_A, t)\) the expected total reward starting at state \(s_A\) in step \(t\) when \(\pi_{\text{opt}}\) is followed, i.e. \(f(s_A, t) = f_{\pi_{\text{opt}}}(s_A, t)\).

The AO* search algorithm is a search algorithm on the hypergraph of augmented states. It starts the search from augmented state \((s_\emptyset, 0)\). It keeps a partially best solution \(\pi^*\) based on the part of
the hypergraph that it has explored, and gradually improves it, until it finds the best solution. In
the beginning, the partially best solution graph contains a single node \((s_\emptyset, 0)\). In each iteration, the
algorithm first expands the partially best solution graph \(G_{\pi^*}\) (\text{“expanding step”}), and then updates the
rewards and the best actions for nodes in \(G_{\pi^*}\) (\text{“updating step”}). The algorithm repeats the iterations
until \(G_{\pi^*}\) cannot be expanded any more and return \(G_{\pi^*}\) as the best solution graph.

In the expanding step of each iteration, the AO* search algorithm expands a nonterminal tip in \(G_{\pi^*}\)
by exploring its successors. When a non-terminal \((s_A, t)\) is explored for the first time, its estimated
reward \(\tilde{f}(s_A, 0)\) is set by a heuristic function \(h(s_A, t)\). In the updating step, for each node \((s_A, t)\)
from which the newly expanded node can be reached by following the current policy \(\pi^*\), the algorithm
updates the estimated reward and the partially best action for it with the following equations:

\[
\tilde{f}(s_A, t) = \max_{u \in V \setminus \{A\}} \left[ r_{s_A}(u) + q(A, u) \tilde{f}(s_{A \cup \{v\}}, t + 1) \right.
\]
\[
+ (1 - q(A, u)) \tilde{f}(s_A, t + 1) \]  
\[
\text{(4.2)}
\]

and

\[
\pi^*(s_A, t) = \arg \max_{u \in V \setminus \{A\}} \left[ r_{s_A}(u) + q(A, u) \tilde{f}(s_{A \cup \{v\}}, t + 1) \right.
\]
\[
+ (1 - q(A, u)) \tilde{f}(s_A, t + 1) \]  
\[
\text{(4.3)}
\]

The updates should be made in backward order, i.e. the successor of any node should be updated before
it is updated. Since for the dynamic influence maximization problem, any successor \((s', t')\) of the node
\((s, t)\) has \(t' = t + 1\), the backward order can be ensured by updating nodes in the descending order of \(t\).

The algorithm is summarized as in Algorithm 1.
Algorithm 1 AO* Search(b, w, L)

1: Initialize the partially best solution graph $G_{\pi^*}$ such that it contains a single state $(s_0, 0)$
2: while $G_{\pi^*}$ has some non-terminal tip node $(s_{A_e}, t_e)$ do
3:   Expanding step: for any successor $(s_{A'}, t_e + 1)$ of $(s_{A_e}, t_e)$ that has not been explored, if it is a terminal state, set $\tilde{f}(s_{A'}, t_e + 1) := f(s_{A'}, t_e + 1)$ as defined in Equation 4.1. Otherwise, set $\tilde{f}(s_{A'}, t_e + 1) := h_1(s_{A'}, t_e + 1)$ as defined in Equation 4.4. Mark the successors explored.
4:   Updating step:
5:     $S_{update} := \{(s_{A_e}, t_e)\}$
6:     while $S_{update}$ is not empty do
7:     $S'_{update} := \{\}$
8:     for each state $(s_A, t) \in S_{update}$ do
9:     Update $\tilde{f}(s_A, t)$ and $\pi^*(s_A, t)$ with Equations Equation 4.2 and Equation 4.3.
10:    For any explored direct ancestor $(s_{A'}, t')$ of $(s_A, t)$ from which $(s_A, t)$ is reached when following $\pi^*$, add it to $S'_{update}$.
11:   end for
12: $S_{update} := S'_{update}$
13: end while
14: end while
15: output $G_{\pi^*}$

Admissible heuristic function We have described the AO* search algorithm for the dynamic influence maximization problem. However, there is still a missing piece: how should we define the heuristic function? In the AO* search algorithm, the heuristic function $h(\cdot)$ serves as an estimator for $f(\cdot)$, the expected total reward when the optimal policy is followed. An important property of the AO* search algorithm is that, if the heuristic function is admissible, the AO* search algorithm will find the optimal policy.
Formally, for the AO* search algorithm of the dynamic influence maximization problem, a heuristic function \( h(\cdot) \) is *admissible*, if it satisfies \( h(s_A, t) \geq f(s_A, t) \) for any augmented state \((s_A, t)\)\(^1\). When using an admissible heuristic function \( h(\cdot) \), the policy returned by Algorithm 1 is an optimal policy for the dynamic influence maximization problem.

It is not difficult to find an admissible heuristic function. Actually, \( h_0(s_A, t) = L - t \) is an admissible function, since \( h_0(s_A, t) \geq f(s_A, t) \) always holds. However, a good heuristic function should be a good estimator for \( f(\cdot) \). Roughly speaking, the closer \( h(\cdot) \) to \( f(\cdot) \) is, the more efficient the AO* search algorithm is.

To estimate the best function \( h(s_A, t) \), a straightforward idea is to use \( Q(L - t, A) \), the sum of \( L - t \) largest values of the active probabilities \( q(A, u) \) for \( u \in V \setminus A \). However, this heuristic function is not admissible. Actually, it always underestimate the reward \( f(s_A, t) \), because it omits the increase of active probability caused by future actions. To make an admissible heuristic function, we calculate \( I(L - t, A) \), the upper bound of this increase, and add it to \( Q(L - t, A) \). The upper bound \( I(L - t, A) \) can be estimated by the sum of \( L - t \) largest \( w(u) \) for \( u \in V \setminus A \), where \( w(u) \) is the sum of the outgoing weights \( w_u \) of \( u \). Formally, we define the heuristic function as follows:

\[
h_1(s_A, t) = \begin{cases} 
0 & \text{if } t = L \text{ or } A = V, \\
Q(L - t, A) + I(L - t, A) & \text{otherwise.}
\end{cases}
\] (4.4)

\(^1\)The heuristic search algorithms are usually described for minimum cost problems rather than maximum reward problems. For minimum cost problems, \( h(\cdot) \) is admissible if \( h(s) \leq f(s) \).
where

\[ Q(n, A) = \sum_{\max(n): u \in V \setminus \{A\}} q(A, u) \]  \hspace{1cm} (4.5)

and

\[ I(n, A) = \sum_{\max(n): u \in V \setminus \{A\}} w(u) = \sum_{\max(n): u \in V \setminus \{A\}} \sum_{v \in V} w_{uv} \]  \hspace{1cm} (4.6)

In the above two equations, \( \sum_{\max(n): u \in V \setminus \{A\}} *u \) denotes sum of the \( n \) largest \(*u\) for \( u \in V \setminus \{A\} \).

**Theorem 2** The heuristic function \( h_1(s_A, t) \) defined in Equation 4.4 is admissible.

**Sketch of Proof:** The cases for \( t = L \) or \( A = V \) are trivial. For \( t < L \) and \( A \subset V \), let \( A_f \) be the set of active users when the process ends, when the optimal policy is followed from state \((s_A, t)\), and every push is successful. We then have

\[ h_1(s_A, t) \leq \sum_{\max(n): u \in V \setminus \{A\}} q(A_f, u) \leq \sum_{\max(n): u \in V \setminus \{A\}} q(A, u) + \sum_{\max(n): u \in V \setminus \{A\}} \sum_{v \in V} w_{uv} = h_1(s_A, t). \]

The AO* search is an offline search algorithm, i.e. it generates the entire solution graph before the diffusion process starts. For a website using AO* search algorithm, before the diffusion process starts, it first executes Algorithm 1 to get the solution graph \( G_{\pi^*} \), and sets the current state to \((s_0, 0)\) of graph \( G_{\pi^*} \). In each step of the diffusion process, it takes the best action of the current state, and moves to the successor according to the outcome of the action.

**4.4.3 Online search algorithm**

The AO* search algorithm reduces the number of augmented states that needed to be evaluated by using the heuristic function. However, when \( L \) increases, the size of the search space still grows.
exponentially. Even the solution graph itself grows exponentially as $L$ increases. As a result, the AO* search algorithm is impractical when the push budget $L$ is large. In this section, we propose an online search algorithm for the dynamic influence maximization problem. Instead of generating the entire optimal policy beforehand, the online search algorithm generates the best actions dynamically during the diffusion process. Only when an augmented state is really reached by the process, the online search algorithm generates the best action for it. Thus, online search algorithm avoids searching actions for a large number of augmented states that are not really reached in the process. Although the online search algorithm does not find the optimal solution for the dynamic influence maximization problem, it is significantly more efficient than offline search algorithms such as the AO* search algorithm, and can be applied to dynamic influence maximization problems with large push budget $L$.

The online search algorithm initializes by setting current state to $(s_0, 0)$. In each step of the diffusion process, for each successor $(s_A, t)$ of current state, the algorithm estimates $f(s_A, t)$ by the heuristic function $h_2(s_A, t)$, and then selects the best action using Equation 4.3. It takes the best action, and then moves to the successor of current state according to the outcome of the action.

The heuristic function $h_2$ is defined as follows:

$$h_2(s_A, t) = \begin{cases} 0 & \text{if } t = L \text{ or } A = V, \\ Q(L - t, A) & \text{otherwise.} \end{cases}$$  

(4.7)

where $Q(L - t, A)$ is defined the same as in Equation 4.5.
Unlike $h_1(\cdot)$ defined in Equation 4.4, $h_2(\cdot)$ is not an admissible heuristic function. However, it is a practically good estimation for $f(s_A, t)$. For real dataset, it is closer to the real value of $f(s_A, t)$.

To implement the online search efficiently, $q(A, u)$ can be calculated incrementally. Specifically, the algorithm keeps tracks of $q(A, u)$ with current set of active users $A$. When a new user $v$ becomes active, $q(A, u)$ is updated for each $u$ with $w_{vu} > 0$. With this implementation, the time complexity for the online search algorithm is $O(|V| \cdot L \cdot \log(L) \cdot d_{\text{max}})$, where $d_{\text{max}}$ is the maximum out-degree of users. Unlike the AO* search algorithm, which has an exponential time complexity with respect to $L$, the online search can be efficiently applied to dynamic influence maximization problems with large $L$, when the AO* search algorithm is practically impossible.

4.5 Experiment

4.5.1 Experiment Setup

Datasets.¹ We experiment with four real social network datasets. All the datasets are publicly available online.

- **Twitter datasets**: We use two network datasets from twitter.com. In both datasets, nodes represent users of Twitter, while edges represent “who-follows-whom” relations. Each edge is directed from the user who is being followed to the follower. In addition to the network structure, the datasets also have the timeline (tweets, retweets and replies) for each user in the two datasets throughout the year 2011. The timelines are used for the analysis in Section 4.2.

¹All datasets and algorithms are available at http://linshuyang.com/research/PDC/
– **Twitter-friends**: This dataset contains 822 users and 56,286 links. We collected this dataset at Dec. 2011. It consists of the users who are followed by the official account of Twitter (@Twitter). They are typically employees of Twitter. This dataset is a very dense social network, as the users in it are strongly connected to each others, and most of them are active Twitter users.

– **UIC-followers** (39): This dataset contains 2,187 users and 14,572 links. It consists of the users who follow the “UIC news” account in Twitter (@UICnews). Most of the users are students in the University of Illinois at Chicago.

• **Foursquare dataset** (16): This dataset contains 8,465 users and 72,978 links. Nodes in this dataset represent users of Foursquare, while indirect edges represent friendship relations. In addition to the network structure, this dataset contains 93,645 check-in actions of users. We use these check-in actions for the analysis in Section 4.2. We have remove users with less than 20 check-ins.

• **Slashdot dataset** (32): This dataset contains 77,360 users and 905,468 links. Nodes in this dataset represent users of Slashdot, while indirect edges represent friends/foes relations.

**Algorithms.** We compare our proposed algorithms with several baselines. The following is the list of algorithms we evaluate.

• **AO*Search**: the AO* search algorithm described in Section 4.4.2.

• **OnlineSearch**: the online search algorithm described in Section 4.4.3.
• **OnlineGreedy**: the greedy algorithm that keep tracks of \( q(u) \), the probabilities of becoming active for users, and in every step pushes to the user with the largest \( q(u) \) among all inactive users.

• **Greedy**: the greedy algorithm that pushes to the user with the largest \( b(u) \) among all inactive users.

• **WeightSum**: the greedy algorithm that pushes to the user with the largest \( w(u) \) among all inactive users, where \( w(u) = \sum_{v \in N_{\text{out}}(u)} \) is the sum of weight of out-edges of \( u \).

• **Random**: the random algorithm that selects each inactive user with the same probability.

**Weight and bias generating.** The PDC model is generated as follows. For each user \( u \), we generate \( b_u \) independently from the uniform distribution \( U[0, 1] \). For each incoming edge \( e = (v, u) \) to \( u \), we generate \( w_{vu} \) independently from the uniform distribution \( U[0, 1/d_{\text{in}}(u)] \), where \( d_{\text{in}}(u) \) is the in-degree of \( u \). To ensure that \( b_u + \sum_{v \in N_u} w_{vu} \leq 1 \), we then scale each \( b_u \) with a factor \( \theta \), and each \( w_{vu} \) with a factor \( 1 - \theta \), where \( \theta \) is a value between 0 and 1. In the experiment, unless it is specified, \( \theta \) is set to 0.2.

In the last part of the experiment, we experiment with different values of \( \theta \).

**Evaluation.** To evaluate an algorithm, we run the PDC process by following the policy generated by the algorithm. For each algorithm, we repeat the process 1,000 times to estimate the expected number of active users when the process ends (called influence spread of the process).

### 4.5.2 Results

**Influence spread and running time for small budget.** We first show the results of algorithms for dynamic influence maximization problems with small budget \( L \). As we discussed in Section 4.4.2, the
AO* search algorithm always finds the optimal solution, but it is only practical for problems with small budgets and moderate size of networks. The purposes of this part of the experiment are: (1) find out the practical limitation of AO*Search with respective to the complexity of problems; (2) compare the effectiveness of other algorithms with the optimal solution.
In Figure 22, we show the average running time for each algorithm with budget \( L = 1, \ldots, 5 \) for the Twitter-friends, UIC-followers, and Foursquare datasets. We do not show the figure for the Slashdot dataset, because for that dataset, AO*Search is impractical even for the case \( L = 3 \). For each dataset, we illustrate the push budget \( L \) on the X-axis, and the average running time for each step on the Y-axis. In all cases, the running time of AO*Search increases drastically as the budget \( L \) increases. For example, for the Twitter-friends dataset, when \( L = 1 \), the average running time is 0.00175 seconds, while when \( L = 5 \), the average running time for each step is 3,229 seconds. For OnlineSearch, the average running time for each step slowly increases as \( L \) increases. When \( L = 1 \), the running time of OnlineSearch is close to that of AO*Search, but when \( L = 5 \), it is orders of magnitude faster than AO*Search. For other algorithms, the average running time for every step does not have obvious increase as \( L \) increase. The running time of WeightSum is close to OnlineSearch. Other three algorithms are faster.

In Figure 23, we show the average influence spread for each algorithm with budget \( L = 1, \ldots, 5 \) for the Twitter-friends, UIC-followers, and Foursquare datasets. For each dataset, we illustrate the push budget \( L \) on the X-axis, and the average influence spread on the Y-axis. In all cases, the influence spread of OnlineSearch is very close to AO*Search and significantly greater than the baselines. The performances of OnlineGreedy and Greedy are similar, and are significantly better than Random. WeightSum outperforms Random only on the UIC-followers dataset.

We conclude from this experiment that: (1) Even for social network with moderate size, AO*Search is not practical when \( L \) grows large. (2) The influence spread achieved by OnlineSearch is close to optimal, and is significantly larger than other algorithms.
**Influence spread for large budget.** For each algorithm except for AO*Search, we evaluate them with large budget $L$ on each dataset. The result are illustrated in Figure 24. In each case, the budget $L$ is illustrated on the X-axis, while the average influence spread is illustrated on the Y-axis. In all cases, OnlineSearch has significantly larger influence spread than other algorithms. The performance for OnlineGreedy and Greedy are close when $L$ is small, but the difference becomes obvious when $L$ grows large. That is because in each step, OnlineGreedy makes the decision according to the outcome of previous steps, while Greedy selects users only based on their initial preference to the item. Among all the algorithms, WeightSum has the smallest influence spread. Even random outperforms WeightSum. It shows that the dynamic influence maximization problem on the PDC model is essentially different from the influence maximization problem on traditional diffusion model: simply selecting nodes with large influence will not help the spread of influence.

**Running time of OnlineSearch with large budget.** In Figure 25, we illustrate the running time of OnlineSearch in different datasets when the budget $L$ varies. In the figure, the budget $L$ is illustrated on the X-axis, while the average running time for each step is illustrated on the Y-axis. For each dataset, the average running time increases slowly when $L$ increases. For all the four datasets, the running time on the Slashdot is the largest, but it is still within a moderate range: when $L = 200$, the average running time for each step is 2.53 seconds. Although the number of users in the Twitter-friend is less than that in the UIC-follower dataset, the running time for the Twitter-friend dataset is slightly larger than that for the UIC-follower dataset. That is because the Twitter-friends dataset is a very dense social network, which contains more edges than the UIC-follower dataset.
Variation of $q(u_t)$ and $w(u_t)$ during the diffusion process. In this experiment, we show that the performance difference between the proposed algorithms and the greedy algorithms is caused by the different strategies underlying them. To understand the strategies, we show the variation of $q(u_t)$ and $w(u_t)$ during the diffusion process, where $u_t$ is the user who is selected at the $t$-th step, $q(u_t)$ is the probability of $u_t$ becoming active when it is selected, and $w(u_t) = \sum_{v \in N_{out}(u_t)}$ is the sum of influence weight over all out-edges of $u_t$. In Figure 26, we illustrate the step $t$ on the X-axis, and the average $q(u_t)$ or the average $w(u_t)$ in step $t$ on the Y-axis. We experiment with the Foursquare dataset with $L = 4$ and $L = 100$. As shown in Figures 26(a) and 26(c), in the early stage of the diffusion process, $AO^*Search$ and $OnlineSearch$ tend to select users with larger influence, while the $OnlineGreedy$ does not show the
Figure 25. Running time of *OnlineSearch* on different datasets

preference for users with larger influence. Comparing with Figures 26(b) and 26(d), we can find out that in the early steps of the diffusion process, *AO*Search and *OnlineSearch* sacrifice their immediate reward for the long-term reward by selecting users who are slightly less likely to become active (smaller \(q(u_t)\)), but have larger influence (larger \(w(u_t)\)). As a result, they will have candidates with larger active probability in later steps. When the process is close to the end, *AO*Search and *OnlineSearch* become more likely to select users with smaller influence but larger probability of becoming active. Another interesting observation is that, comparing with the case with small budget (\(L = 4\)), when the budget is large (\(L = 100\)), *OnlineSearch* is more willing to sacrifice the immediate reward at the beginning of the diffusion process. The results show *AO*Search and *OnlineSearch* decide to what extend they can sacrifice the immediate reward according to the budget left.

**Effect of \(\theta\).** Finally, we study how the value of \(\theta\) affects the average influence spread of algorithms. As described in Section 4.5.1, \(\theta\) is a parameter used in the problem generating. It is a value between 0 and 1. When \(\theta\) increases, the user preferences become larger, while the influence weights become smaller. To study the effect of \(\theta\), we use the Foursquare dataset with \(L = 4\) and \(L = 100\). In Figure
Figure 26. Variation of $q(u_t)$ and $w(u_t)$

27, we illustrate $\theta$ on the X-axis, and the average influence spread on the Y-axis. For both cases, when $\theta$ increases, the influence spread for all algorithms increases. When $\theta$ is larger, the difference between OnlineSearch, OnlineGreedy and Greedy tends to become smaller. That is because the social influence becomes smaller when $\theta$ increases. However, OnlineSearch consistently outperforms the baselines when $\theta$ varies from 0.2 to 0.8, and it is very close to AO*Search for the case $L = 4$. The results show that although for the previous experiments we fix $\theta$ to 0.2, their results are robust for different values of $\theta$. 
4.6 Related Work

The work in this chapter is most closely related to the studies on information diffusion and the influence maximization problems. The influence maximization problem was first proposed by Kempe et al in (25). The problem has been studied on many different information diffusion models, including the independent cascade model, the linear threshold model, and many variants of them (31; 10; 9). Most work on influence maximization does not consider the role that the social network providers play in the viral marketing. A recent work in (40) studies the influence maximization from the perspective of social network providers. In that paper, social network providers try to balance the spread of influence for different items. To the best of our knowledge, there is no existing work considers information diffusion as a sequential decision making problem for the social network providers.

We solve the dynamic influence maximization problem as a sequential decision making problem for Markov decision process (MDP). MDPs have been studied in the artificial intelligence area for decades. Many algorithms have been proposed for solving the MDPs optimally or approximately (21; 44). In
this chapter, we adopt the AO* heuristic search algorithm to solve the dynamic influence maximization problem. The AO* algorithm was first designed for solving the search problem on AND/OR graph (43). Later works show that it could also be used for solving the acyclic MDPs (21).
CHAPTER 5

EFFICIENT INFLUENCE MAXIMIZATION WITH COMMUNITY EFFECTS

(This chapter was previously published as Shuyang Lin, Qingbo Hu, Guan Wang, Philip S. Yu, “Understanding Community Effects on Information Diffusion”, in Proceedings of the 19th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD ’15) (36). With permission of Springer Science+Business Media.)

5.1 Introduction

For many years, community study is one of the hot topics in social network research. Studies in this area offer insightful solutions to many classic problems of social network research, such as network evolution (24), recommendation system (45), and expert finding (5). Communities can be potentially helpful for studies on diffusion of information in social networks. Previous studies found that links between communities function differently from those within communities: friends in the same communities have stronger links, but weaker links between friends in different communities are crucial in the diffusion of novel information, because these links provide more useful information to people (19; 14; 4).

Some key problems in the studies of information diffusion have been found difficult to solve by traditional information diffusion methods. Studies on the community structure of social networks may bring new ideas for solving these problems. For example, one of the key problems, the influence maximization problem for the independent cascade model has been proved to be NP-hard (25). By considering
the effects of community studies on the diffusion of information, we can easily come up with some intuitive heuristics to solve that problem more efficiently. For example, we may utilize the community homophily to quickly estimate the influence of users. We may also select seed nodes from different communities to minimize the overlap and maximize the influence.

However, few existing work explicitly studied the effects of communities on the diffusion of information, or use these effects to solve diffusion-related problems. Motivated by this important absence, we introduce the first exploratory study on the effects of communities on information diffusion processes. By analyzing real-world datasets, we study the diffusion of information with communities. We first observe the action homophily of communities, and then introduce the concept of role-based homophily of communities, which consists of influencee role homophily and influencer role homophily. We discover that the role-based homophily is significantly stronger than the action homophily.

Our findings on community effects can lead to insightful solutions to many problems in information diffusion studies. As an example application of these findings, we propose an approximate solution for the influence maximization problem. We design a community-based fast influence (CFI) model based on the influencee role homophily of communities. The CFI model applies a community clustering method to social networks, and makes aggregations on users’ roles as influencees. Influence maximization algorithm based on the CFI model can efficiently select seed nodes to maximize the influence. The main contributions are summarized in the following:

1. We conduct quantitative analyses on real-world datasets to explore the effects of community on the diffusion of information.
2. We get valuable findings about the community effects from quantitative analyses. We introduce the concept of role-based homophily of communities. These understandings can bring new insights to the studies of information diffusion.

3. We show an example application of our findings on the influence maximization problem. We propose a community-based fast influence (CFI) model, and an efficient approximate influence maximization algorithm based on that model.

5.2 Related work

**Information diffusion problem.** Several models have been proposed for the information diffusion processes. The independent cascade (IC) model and its variants are most widely used information diffusion models (25; 48; 17; 29). The basic idea of the IC model is: if a node in a social network becomes active, it can make its neighbors active with a probability, and for each node the attempts of its neighbors to activate it are independent. The influence maximization problem has been defined for the IC model and a few other information diffusion models (25). Given an IC model, the problem is to select a seed set with \( k \) nodes so that the expected number of active nodes are maximized. This problem has been proved to be NP-hard. The first solution to it is a greedy algorithm that repeatedly invokes a computational expensive sampling method (25). Heuristic algorithms and optimized versions of the greedy algorithm have been proposed in previous works (31; 10; 9). Work in (52) proposed an heuristic algorithm which finds influencers from communities. Different from that work, our proposed model is built on observations on real data and based on a substantially different idea. A recent work in (15) defined a group-based version of the influence maximization problem. The predefined groups studied in that work were not conceptually equivalent to the communities studied in this chapter.
Community detection. Community detection in social networks has been studied for years. Varieties of algorithms have been proposed. A good survey is available in (42). We are not going to discuss the varieties of existing community detection methods, except for those that are related to our work in this chapter. Modularity-based methods are a major class of community detection methods. Among these methods, the fast greedy method (11) is frequently used for community detection on large-scale networks. In (47), Rosvall et al. proposed the infomap method. Substantially different from modularity-based methods, the infomap method is based on flows carried by networks (47). The SHRINK algorithm in (23) is another algorithm that is related to our work. It is a parameter-free hierarchical network clustering algorithm that combines the advantages of density-based clustering and modularity optimization methods. Work in (54) utilized social influence modeling methods in the detection of communities.

5.3 Preliminary

5.3.1 Notations

A social network $G = \{V, E\}$ is a directed graph with a node set $V$ and an edge set $E$. A node $v_i \in V$ represents a user in the social network, while a directed edge $e_{ij} \in E$ represents a link from $v_i$ to $v_j$.

A community $C$ in the social network $G$ is a subset of the node set $V$. We consider non-overlapping communities in this chapter. In other words, we consider the partition of $V$ into a set of communities $C = \{C_i\}_{i=1}^m$. Each user in the network should belong to exactly one of the communities in $C$. Given a graph $G$, a community detection algorithm divides the graph $G$ into a set of communities $C$. There are a lot of different community detection algorithms. Generally, a good community detection algorithm
finds a partition, so that (1) each community is a relatively independent compartment of the graph, and
(2) nodes in the same community tend to have dense links between each other.

We follow the definition of information diffusion process in the IC model (25). An information
diffusion process starts with a set of seed nodes that are active at the first place. Active nodes can activate
their out-neighbors in the social network. Once a node is activated, it becomes active and can never
become inactive again. It is quite often for real applications that the information diffusion processes
cannot be directly observed. For example, we may observe that a person got infected by influenza, but
we do not know from whom he got infected. We define a cascade \( O = \{(v_1, t_1), \ldots, (v_m, t_m)\} \) as the
set of user actions during an information diffusion process. An action \((v_i, t_i)\) in \( O \) represents that the
user \( v_i \) becomes active at time \( t_i \). In this chapter, we focus on the scenario that the information diffusion
processes are not directly observed, but a set of cascades is observed.

5.3.2 Datasets

Foursquare (16) In this dataset, nodes represent users of the Foursquare website, while edges represent
friendship relations. Actions are defined by check-ins of users. Each cascade corresponds to a location.
When a user checks in at a location for the first time, she becomes active for the corresponding cas-
cade. This dataset contains 18,107 users, 245,034 friendship relations, and 476,482 actions of 43,063
cascades.

DBLP In this dataset, nodes represent authors, while edges represent co-author relations. We extract a
subgraph of the DBLP network with authors and papers in the areas of data mining and machine learn-
ing. We define cascades by terms (defined by bi-grams) in the titles of papers. When an author has
a paper with a certain term in the title for the first time, he becomes active for the corresponding cas-
cade. This dataset contains 6,896 users, 111,044 friendship relations, and 1,655,778 actions of 162,904 cascades.

5.4 Observations

In this section, we explore the community effects on information diffusion processes via analyses on real-world datasets. We first identify communities in social networks, and then study cascades with respect to these communities.

5.4.1 Identifying communities for information diffusion

Communities in social networks can be defined in many different ways. To understand the effects of communities on the information diffusion in general, we apply two different community detection algorithms to the two networks, and conduct community effect analyses for both algorithms.

The two community detection methods that we use to identify communities are the fast greedy (FG) method (11), and the infomap (IM) method (47). The FG method is based on the well-adopted idea of modularity maximization, while the IM method is a flow-based method, which is essentially different from the modularity maximization methods. We choose these two methods because (1) they are all widely-used community detection methods that prove to be efficient and accurate, and (2) they are based on substantially different ideas. Both methods are implemented in the igraph network analysis package (12). We use the default setting of igraph for the experiment. In Table VIII, we list the statistics for each set of communities, which include the number of communities, the largest/average size of communities, and the percentage of edges within communities.
TABLE VIII

TWO SETS OF COMMUNITIES FOR EACH NETWORK

<table>
<thead>
<tr>
<th></th>
<th>infomap</th>
<th>multilevel</th>
<th></th>
<th>infomap</th>
<th>multilevel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Foursquare</td>
<td>#Community</td>
<td>2298</td>
<td>1418</td>
<td>#Community</td>
<td>377</td>
</tr>
<tr>
<td></td>
<td>Max Size</td>
<td>994</td>
<td>2257</td>
<td>Max Size</td>
<td>928</td>
</tr>
<tr>
<td></td>
<td>Mean Size</td>
<td>7.9</td>
<td>12.8</td>
<td>Mean Size</td>
<td>18.3</td>
</tr>
<tr>
<td></td>
<td>Inside Edge</td>
<td>61.7%</td>
<td>75.5%</td>
<td>Inside Edge</td>
<td>56.9%</td>
</tr>
</tbody>
</table>

5.4.2 Action homophily of communities

We first look into the effects that communities have on the actions of users. We construct a vector for each user to keep the action information of that user, and then compare the vectors between pairs of users. We check whether the users who belong to the same community are more likely to have similar actions.

Given a set of cascades \( O = \{ O_1, \ldots, O_m \} \), we define an action vector \( a_i \) for each user \( v_i \), where \( a_i = (a_{i0}, \ldots, a_{im}) \). If the user \( v_i \) has an action in the cascade \( O_j \), we set \( a_{ij} \) to 1. Otherwise, we set \( a_{ij} \) to 0. For each pair of users \( v_i \) and \( v_j \), we calculate the cosine similarity between the action vectors \( a_i \) and \( a_j \), and then study the distribution of similarity. We consider three different cases here: (1) There is an edge \( e_{ij} \) between \( v_i \) and \( v_j \), and \( v_i \) and \( v_j \) belong to the same community; (2) There is an edge \( e_{ij} \) between \( v_i \) and \( v_j \), but \( v_i \) and \( v_j \) belong to different communities; (3) \( v_i \) and \( v_j \) is an arbitrary pair of nodes, may or may not having an edge between them. For each case, we plot the distributions of similarity, and check whether there is any difference between the distributions.
Figure 28 shows the distributions of similarity in two datasets, with two sets of communities in each dataset. In each setting, we observe a similar discrepancy among the three distributions: comparing with linked pairs in different communities, linked pairs in the same communities have larger similarity; comparing with arbitrary pairs, linked pairs have much larger similarity. Intuitively, friends in the same communities tend to have stronger link between each other, and they have more chances to influence each other indirectly via common friends. The results are quite consistent for different community detection methods.
5.4.3 Role-based homophily of communities.

We have observed the action homophily of communities. However, although the similarity between linked pairs in the same communities is relative larger than the similarity in the other two cases, it is still quite small (typically, less than 0.3). In this section, we introduce the role-based homophily of communities, and show that the role-based homophily is more significant than the action homophily.

With a set of cascades $\mathcal{O}$, we build a support matrix $S$ for the influence between users in the social networks. The element at the $i$-th row and the $j$-th column of the matrix $S$ is the number of potential influences from the user $v_i$ to the user $v_j$. We say there is a potential influence from $v_i$ to $v_j$, if both of them have actions in the same cascade, and the time of $v_i$’s action is earlier than the time of $v_j$’s action. Formally, it is defined as:

$$ s_{ij} = | \{ O_k \in \mathcal{O} \mid v_i, v_j \in V(O_k) \land t_{O_k}^i < t_{O_k}^j \} | $$

where $V(O_k)$ is the set of users that has an action in the cascade $O_k$, and $t_{O_k}^i$ is the time of $v_i$’s action in the cascade $O_k$.

We define $s_{i*}$, the $i$-th row of $S$, as the influencer feature vector of $v_i$, and $s_{*i}$, the $i$-th column of $S$, as the influencee feature vector of $v_i$. The influencer feature vector $s_{i*}$ captures the influence that $v_i$ has on other users in the social networks, while the influencee feature vector $s_{*i}$ captures the influence from other users to the user $v_i$.

Similar to what we did in Section 5.4.2, we calculate the cosine similarity between the influencer/influencee feature vectors, and compare the distributions. Figures 29 and 30 show the comparison of distributions of influencer feature vector and influencee feature vector, respectively. Similar to Figure 28, comparing with the other two cases, the similarity is larger for the case that users are linked and are in the same communities. There are a few new observations that are interesting:
First, distributions of similarity between influencer/influencee feature vector (Figures 29 and 30) show significantly larger discrepancy than the distributions of similarity of action vector (Figure 28). This observation suggests that for users in the same communities the role-based homophily is much stronger than the action homophily. The effect of community in the information diffusion process is better reflected by the roles that users play in the information diffusion process, rather than the results of information diffusion process (whether being active or inactive for a cascade).

Second, for friends in the same community, the similarity value of influencer and influencee feature vectors (typically larger than 0.5) is larger than the similarity of action vectors (typically less than 0.3).
It suggests that aggregation on the influencer/influencee feature vectors of users without significant loss of accuracy is more feasible.

Third, the influencee-based homophily is more significant than the influencer-based homophily, especially for the FG algorithm. This is easy to understand by the following example: professors and students in the same research lab have similar behaviors as influencees, because when a cascade reaches anyone in the lab, it is very likely that cascade will reach everyone in the lab quickly, but professors are probably much stronger influencers than students.

5.5 Community-based Fast Influence Model

Based on the observations in Section 5.4, we are able to design an efficient influence model which makes use of the community effects. The community-based fast influence (CFI) model we propose in this section is an approximate model for the IC model. The whole framework has three components, namely influence decoupling, community detection, and influence maximization.

5.5.1 Influence decoupling

An intuitive way to construct an approximate information diffusion model based on community effects is to consider each community as a “super-node” and make information propagates through “super-edges” between “super-nodes”. The coarse-grained information diffusion model in (15) is based on a similar idea.

Although this intuitive model is simple and seems reasonable, it may not work for our task here. When we consider a community as a “super-node”, we have to aggregate users’ roles as influencers as well as users’ roles as influencees. This may cause a problem: the influence maximization problem requires us to determine how influential each user is and find the set of seed nodes that maximizes the
influence. When we aggregate the roles of users as influencers, we lose the necessary information for solving the influence maximization problem.

To avoid this problem, the CFI model considers the roles of users as influencers and influencees separately. To be specific, we split each node $v_i$ in the network $G$ into an influencer node $v_i^{out}$ and influencee node $v_i^{in}$, and transform the network into a bipartite graph $G_b$. In the graph $G_b$, there is an edge from $v_i^{out}$ to $v_j^{in}$ if and only if the edge $e_{ij}$ exists in the original graph $G$. We call this transformation from the original network $G$ to the bipartite graph $G_b$ influence decoupling. The left part of Figure 31 shows an example of influence decoupling. In the original graph $G$, there is an edge from $v_4$ to $v_1$. Correspondingly, there is an edge from $v_4^{out}$ to $v_1^{in}$ in $G_b$. The result of influence decoupling is that we can apply the community-based aggregation to the influencee nodes only.

If we apply the original IC model directly to the decoupled graph $G_b$, we will end up with cascades with only two levels, i.e. only the nodes that are direct out-neighbors of the seed nodes can become active. This problem can be approximately solved by the community detection and the aggregation of influencee nodes. Instead of limiting influence to direct out-neighbors, the CFI model allows users to have influence on the communities that their direct out-neighbors belong to. Notice that we do omit the indirect influence from a user to the nodes that are neither his out-neighbor nor in the same communities as his out-neighbor. This is indeed a trade-off between the accuracy and efficiency, but the loss of accuracy is actually negligible. This is because the influence between nodes in different communities are smaller than the influence between nodes in the same communities, and indirect influence are generally very small. We will also show by experiment that the CFI model is a good enough approximation to the original IC model.
5.5.2 Identifying communities

We now discuss the community detection algorithm. As we have discussed in the last section, users in the same community should be similar influencees. To identify communities so that users in the same communities are similar as influencees, we design an agglomerative clustering algorithm. It starts with clusters with single users, and iteratively merges clusters together based on similarity between clusters. As shown in Figure 31, the clustering procedure is conducted on the original graph $G$, but the similarity is defined by users’ roles as influencees, and the communities detected by the algorithm will finally be applied to the influencee nodes in the decoupled graph $G_b$. 

Figure 30. Distribution of similarity between influencee feature vectors of user pairs
Similarity. The similarity between two clusters is defined as the cosine similarity between their incident influence probability vectors. Let $p_{i \rightarrow j}$ be the probability that $v_i$ influences $v_j$ directly or indirectly (i.e. the probability that $v_j$ becomes active if $v_i$ is the single seed node). For a cluster $C = \{v_{i_1}, \ldots v_{i_{nC}}\}$, we define the influence that user $v_j$ on $C$ as:

$$q_{j \rightarrow C} = \begin{cases} \frac{1}{n_C} \sum_{k=1}^{n_C} p_{j \rightarrow i_k} & \text{if } e_{j,i_k} \in E \text{ for some } i_k \in C \\ 0 & \text{otherwise.} \end{cases}$$

(5.1)

where $n_C$ is the number of nodes in the cluster $C$.

We define incident influence probability vectors of community $C$ as $q_C = (q_{1 \rightarrow C}, \ldots q_{n \rightarrow C})$, and the similarity between two clusters $C_1$ and $C_2$ as:

$$\text{sim}(C_1, C_2) = \frac{q_{C_1} \cdot q_{C_2}}{\|q_{C_1}\|\|q_{C_2}\|}$$

Influence probability estimation. Similar to the learning algorithm for the IC model in (17), given a set of cascades $\mathcal{O}$, we estimate the influence probability $p_{i \rightarrow j}$ from cascades by the equation as follows:

$$\hat{p}_{i \rightarrow j} = \frac{s_{ij}}{s_i} = \frac{|\{O_k \in \mathcal{O} \mid v_i,v_j \in V(O_k) \land t^O_k < t^O_{j}\}|}{|\{O_k \in \mathcal{O} \mid v_i \in V(O_k)\}|}$$

(5.2)

Since that $v_i$ becomes active earlier than $v_j$ does not necessarily imply that $v_i$ directly or indirectly influences $v_j$, $\hat{p}_{i \rightarrow j}$ is not an unbiased estimator of $p_{i \rightarrow j}$. However, it is still a good enough estimator for the CFI model.
**Community detection and influence aggregation.** The purpose of community detection in the CFI model learning is to aggregate users who play similar roles as influencees, while keep the accuracy of the original IC model. To serve this purpose, we adopt a community detection strategy that is similar to the algorithm in (23). By iteratively merging clusters into larger one, we get a sequence of super-graph $G_0, G_1, G_2, \ldots$. Each node in these super-graphs corresponds to a cluster. The algorithm starts with graph $G_0$, in which each cluster contains a single user. In each step $t$, we find from $G_t$ connected subgraphs that contain similar nodes, and merge these subgraphs to generate a new super-graph $G_{t+1}$.

We repeat these steps, until the similarity between any two neighbors in $G_t$ are below a threshold $\theta$.

Let $C^{(t)} = \{C_1, \ldots, C_m(t)\}$ be the set of clusters at the $t$-th iteration. We say two clusters $C_1$ and $C_2$ are neighbors, if there exist $v_i \in C_1$ and $v_j \in C_2$, s.t. edge $e_{ij}$ or $e_{ji}$ exists. For a pair of connected clusters, we say they are a **mutually most similar pair (ms-pair)** with similarity $\epsilon$ (denoted by $C_1 \leftrightarrow^\epsilon C_2$), if $\epsilon = \text{sim}(C_1, C_2) = \max_{C_i \in N(C_1)} \text{sim}(C_1, C_i) = \max_{C_i \in N(C_2)} \text{sim}(C_2, C_i)$, where $N(C_i)$ is the set of neighbors of $C_i$.

We define a **ms-subgraph** as a maximal connected subgraph of $G_t$ that are connected by ms-pairs. Formally, a graph $D$ is a ms-subgraph of $G_t$ with similarity $\epsilon$ if and only if (1) for any two nodes $C_i, C_j \in D$, there exist a path $\langle C_i, C_1 \ldots C_k, C_j \rangle$ in $D$, s.t. $C_i \leftrightarrow^\epsilon C_1, C_1 \leftrightarrow^\epsilon C_2, \ldots, C_{k-1} \leftrightarrow^\epsilon C_k, C_k \leftrightarrow^\epsilon C_j$; (2) for any nodes $C_i \notin D$ and $C_j \in D$, $C_i$ and $C_j$ are not a ms-pair. By this definition, the graph can be partitioned into ms-subgraphs (some ms-subgraphs may contain only one single node). By merging ms-subgraphs into new nodes, the original super-graph can be reduced into a smaller super-graph.
At the iteration $t$, we first find out all the ms-subgraphs of $G_t$, and then merge each ms-subgraph $D$ that contains more than one nodes and has similarity $\epsilon \geq \theta$ into a new node. The new node is a neighbor to any node that was a neighbor of any node in $D$, and the similarity between the new nodes and its neighbors need to be recalculated. The algorithm stops when the similarity between each linked nodes are less than the threshold $\theta$, and the clusters at that point of time are taken as communities. The clustering algorithm is summarized in Algorithm 2.

### 5.5.3 CFI-based influence maximization algorithm

In this subsection, we show how we can design a CFI-based algorithm for the influence maximization of the IC model. The influence maximization problem is defined as follows: Given an IC model and an integer $k > 0$, find a set of $k$ seed nodes, so that the influence of the seed nodes is maximized. The standard method to solve this problem is a greedy algorithm (25). It starts with finding one seed node that maximizes the influence, and then adds a second node to the seed node set so that the increase of influence is maximized. In this way, it repeatedly adds nodes to the seed node set, until it gets $k$ seed nodes. This greedy algorithm is very time-consuming, because in each step it uses the Monte Carlo method to evaluate all the remaining nodes. Optimized versions of the greedy algorithm have been proposed in (31) and (18), and heuristic algorithms have been proposed in (9). These algorithms also use sampling for the evaluation of nodes. We can get a new heuristic algorithm based on the CFI model. This new heuristic algorithm does not involve random sampling, so it is faster than the existing algorithms, especially when the number of seed nodes $k$ is large.

The CFI-based influence maximization algorithm also adopts a greedy framework. In each step $t$, the node that can maximize the influence increase is selected. The problem is how we can estimate the
Algorithm 2 CFIClustering($G, O, \theta$)

1: initialize $C = \{\{v_1\}, \ldots, \{v_n\}\}$
2: for each $C_i = \{V_i\}$ in $C$ do
3: 
4: 
5: end for
6: converged ← False
7: while converged = False do
8: 
9: 
10: if $C_i$ is not marked as visited then
11: 
12: 
13: if $\epsilon \geq \theta$ then
14: 
15: if $|D| > 1$ then
16: 
17: 
18: 
19: 
20: end if
21: end if
22: end if
23: end for
24: end while
25: return $C$ and $q$

influence increase using the CFI model. When $t = 1$, the problem is reduced to estimating the influence of each single node. Let $C = \{C_1, \ldots, C_m\}$ be the set of communities in the CFI model. We estimate the influence of a user $v_i$ as $\text{Inf}\left(\{v_i\}\right) = \sum_{C \in C} n_c q_{i \to C}$, where $n_c$ is the number of users in the community $C$.

Once we select the node with the greatest influence to be the first seed node, we cannot simply select the second most influential node to be the second seed node, because the nodes activated by the
Algorithm 3 BFS-MSSubgraph($C_i, C, \epsilon$)

1: initialize $D \leftarrow \{C_i\}$
2: initialize queue $Q \leftarrow \{C_i\}$
3: initialize set $V \leftarrow \emptyset$
4: while $Q$ is not empty do
5:     $C_j \leftarrow Q.dequeue()$
6:     if $\max_{C_k \in N(C_j)} \text{sim}(C_k, C_j) = \epsilon$ then
7:         $D \leftarrow D \cup \{C_j\}$
8:     for each $C_k$ in $N(C_j)$ do
9:         if $\text{sim}(C_k, C_j) = \epsilon$ then
10:            enqueue $C_k$ to $Q$
11:     end if
12: end for
13: end if
14: end while
15: return $D$

Algorithm 4 MergeClusters($D$)

1: $C_{\text{new}} \leftarrow \bigcup_{C_i \in D} C_i$
2: $N(C_{\text{new}}) \leftarrow \emptyset$
3: for each $C_i \in D$ do
4:     $N(C_{\text{new}}) \leftarrow N(C_{\text{new}}) \cup N(C_i)$
5: end for
6: for each $C_i \in N(C_{\text{new}})$ do
7:     remove clusters in $D$ from $N(C_i)$
8:     add $C_{\text{new}}$ to $N(C_i)$
9: end for
10: return $D$
first node and the second node may overlap. We need to deduct the number of nodes that has already
been activated by the first node. To do that, we decrease the number of nodes from each community by
the estimated influence of the first node $v_{i_1}$. Formally, we let $n^1_c = n_c - n_cq_{i_1\rightarrow C}$, which is the expected
number of nodes in the community $C$ that are not activated by the influence of $v_{i_1}$, and then we select the
second node $v_{i_2}$ by maximizing the increase of influence: $\Delta Inf(v) = Inf(\{v_{i_1}, v\}) - Inf(\{v_{i_1}\}) = \sum_{C\in C} n^1_c q_{i_2\rightarrow C}$. For $t = 3, \ldots, k$, we can repeat the above step to select $v_{i_3}, \ldots, v_{i_k}$. Generally, we
select $v_{i_t}$ by maximizing $\sum_{C\in C} n^{-1}_{c_t} q_{i_t\rightarrow C}$, where $n^{-1}_{c_t} = n^{t-2}_c - n^{t-2}_{c_t}q_{i_{t-1}\rightarrow C}$. The algorithm is
summarized in Algorithm 5.

\begin{algorithm}
\caption{CFIInfluenceMaximization($G, k, q, C$)}
1: initialize $S \leftarrow \emptyset$, and $n^0_C \leftarrow n_C$ for all $C \in C$
2: for $t = 1 \ldots k$ do
3: \hspace{1em} for each node $v_i \in V$ do
4: \hspace{2em} $\Delta Inf(v_i) \leftarrow \sum_{C\in C} n^{-1}_{c_t} q_{i\rightarrow C}$
5: \hspace{2em} $v^t_i \leftarrow \arg \max_{v_i \in V \setminus S} \Delta Inf(v_i)$
6: \hspace{1em} $S \leftarrow S \cup v^t_i$
7: \hspace{1em} $n^t_c \leftarrow n^{-1}_{c_t} - n^{t-1}_{c_t}q_{i_{t-1}\rightarrow C}$
8: \hspace{1em} end for
9: \hspace{1em} end for
10: output $S$
\end{algorithm}
5.6 Experiment

5.6.1 Experiment setup

We use the DBLP and Foursquare networks described in Section 5.3 for the experiment. For each network, we construct an IC model by assigning diffusion probability $1 - e^{-0.01c}$ to each edge. A similar method for model construction has been used in (10). For the DBLP network, $c$ is the number of papers coauthored by the two authors. For the Foursquare network, $c$ is the number of locations that both users visited. We do not construct the ground-truth models by learning the diffusion probabilities directly from the actions in the datasets because we want to avoid the inaccuracy caused by model learning algorithm.

We then sample each ground truth IC model to get 5,000 cascades, each with 10 seed nodes and use the sampled cascades to learn the CFI model. For the baselines, since they are all based on the IC model, we directly apply the influence maximization algorithms on the ground truth model. We evaluate the influence of seed nodes by sampling the ground truth model 10,000 times to get the average number of active nodes. We compare the following algorithms:

- **CFIGreedy** The CFI-based influence maximization algorithm with $\theta = 0.4$.

- **ICGreedy** The greedy influence maximization of IC model with the CELF++ optimization (18). We take a sample size of 10,000 to estimate the influence.

- **Degree** The heuristic algorithm that selects the nodes with the largest weighted degree. The weighted degree of a node is the sum of the diffusion probabilities over the out-going edges.
• **DegreeDiscount** The degree discount heuristics based on the degree heuristics (10). The basic idea is to discount the degree for users whose friends have been selected as seed nodes.

• **Random** Randomly selecting seed nodes.

### 5.6.2 Results

**Effectiveness results for influence maximization.** First, we present the effectiveness results of the influence maximization algorithms in terms of the number of seed nodes. We test the effectiveness of each algorithm with increasing number of seed nodes. The results of the DBLP and Foursquare datasets are illustrated in Figures 32(a) and 32(b), respectively. In each case, we illustrate the number of seed nodes on the X-axis, and the influence of seed nodes on the Y-axis. For the Foursquare data, $CFIGreedy$ performs worse than $ICGreedy$ when the size of the seed set is small, but does better than $ICGreedy$ when the size is greater than 25. This is a very interesting observation. Although the CFI model is designed to be an approximate model for the IC model, the greedy algorithm of the CFI model does not necessarily performs worse than the greedy algorithm of the IC model. This is because the CFI model considers the community structure of social networks, and the consideration of community structure may favor combinations of seed nodes that cover more communities. For the DBLP dataset, $CFIGreedy$ is less effective than $ICGreedy$. However, the difference is not very significant, especially when we consider the fact that $CFIGreedy$ is significantly faster. Besides, $CFIGreedy$ consistently outperforms $DegreeDiscount$, $Degree$ and $Random$. Notice that although $DegreeDiscount$ is a simple heuristic method, previous work showed that it is a very effective method that nearly matches the performance of $ICGreedy$ (10).
Efficiency results for influence maximization. We also tested the efficiency of influence maximization methods with varying number of seed nodes. The efficiency results for the DBLP and Foursquare datasets are illustrated in Figure 33(a) and 33(b), respectively. The X-axis denotes the number of seed nodes, whereas the Y-axis denotes the running time. Since heuristics as Random, Degree, and DegreeDiscount are obviously very fast, we only show the running time of CFIGreedy and ICGreedy. As illustrated in the figures, influence maximization based on CFIGreedy is several orders of magnitudes faster than ICGreedy with CELF++ optimization. We also add together the time spent on the learning of the CFI model and the running time of CFIGreedy to get a total time for the influence maximization on the CFI model, and illustrate the total time as “CFI(+learning time)” in the figures. Even when the learning time is added, the total running time for the CFI model is still significantly smaller the running time of ICGreedy. For example, for the DBLP dataset, it takes ICGreedy 9,079 seconds to find 60 seed nodes, while the total running time of the CFI model is 34 seconds. Notice
that, in real applications, the IC models also need to be learned from user actions, and the running time of _ICGreedy_ should also be added with the learning time of the IC model.

**Parameter sensitivity.** Finally, we tested the sensitivity of the CFI-based influence maximization with the clustering threshold \( \theta \). Figure 34(a) shows the influence of seed nodes selected by the CFI model with varying \( \theta \). We illustrate the value of \( \theta \) on the X-axis, and the influence of seed nodes on the Y-
axis. Figure 34(b) shows the total running time of influence maximization with varying $\theta$. We illustrate the value of $\theta$ on the X-axis, and the total running time of the influence maximization (the running time of model learning plus the running time of $CFIGreedy$) on the Y-axis. In each case, the number of seed nodes is set to 50. We show the results on the Foursquare dataset, while similar trends are observed on the DBLP dataset. When the threshold $\theta$ decreases, the running time increases, because the agglomerative clustering takes more steps when $\theta$ is smaller. It is an interesting observation that the influence does not monotonically increases when $\theta$ decreases. When $\theta$ is too large, the size of communities are very small, so the CFI model omits too much indirect influence. When $\theta$ is too small, the users in the same community do not have enough similarity between each other. Both cases cause loss of accuracy. Nevertheless, we notice that the variation of influence is not significant. The Y-axis of Figure 34(a) does not start at 0. When $\theta$ varies from 0.3 to 1.0, the variation of influence is within $\pm1.5\%$. 

Figure 34. Effects of $\theta$. 
CHAPTER 6

CONCLUSIONS AND CONTRIBUTIONS

(Parts of chapter were previously published (39; 37; 35; 36).)

In this thesis, we have explored information diffusion in social networks. Towards this direction, we thoroughly studied three key problems of information diffusion: diffusion modeling, trend prediction and influence maximization. We based our work on studies on real social network data and motivated our work by data-driven observations. The effectiveness of the proposed algorithms and models are evaluated by experiments on various real-world datasets. The contributions we have made are summarized as below:

- First, we studied the problem of information diffusion models on social networks. We proposed an LADP model that improves the learning by extracting social events from data stream. The LADP integrates the external trends and the information propagation process inside the social network. Evaluation on real and synthetic datasets showed that the LADP outperforms existing method on the task of learning information diffusion models. Analysis showed that the improvement is due to the extraction of social event.

- We studied the problem of trend prediction in social networks. We identify coverage, intensity and duration as the three characteristics of a trend. Though the phenomenon of trends had been widely observed and studied, none of the previous models could capture all the three important aspects of trends. We proposed a Dynamic Activeness model for trends based on the novel concept of
node activeness. The experimental result shows that the proposed DA model can predict trends more accurately than information diffusion models.

- Observing the important roles that social network providers play in the information diffusion process, we study the information diffusion process as a stochastic process that is partially controlled by the social network providers. We develop a novel push-driven cascade (PDC) model, which combines the user preference and social influence. We present the dynamic influence maximization problem on the PDC model, and design two dynamic influence maximization algorithms for the model.

- We explore the effects of communities on the information diffusion processes. We quantitatively analyze the real-world information diffusion datasets to get insightful findings on the community effects. As an application of these findings, we propose the CFI model, which is substantially different from existing approximate algorithms. Experiment shows that the CFI-based influence maximization algorithm can get comparable effectiveness as influence maximization algorithms based on the IC model, but is significantly faster. Our work sheds light on the effects of communities in the diffusion of information, and brings a new idea to the approximation of information diffusion processes.
CITED LITERATURE


VITA

NAME: Shuyang Lin

EDUCATION:

B.E. in Computer Science, Tsinghua University, 2010.

M.S. in Mathematics, University of Illinois at Chicago, 2013.

PUBLICATIONS


- Shuyang Lin, Qingbo Hu, Guan Wang, Philip S. Yu, “Understanding Community Effects on Information Diffusion”, in Proceedings of the 19th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD ’15).


• Shuyang Lin, Qingbo Hu, Fengjiao Wang, Philip S. Yu, “Steering Information Diffusion Dynamically against User Attention Limitation”, in *Proceedings of the 14th IEEE International Conference on Data Mining (ICDM ’14)*.

• Shuyang Lin, Xiangnan Kong, Philip S. Yu. “Predicting Trends in Social Networks via Dynamic Activeness Model”, in *Proceedings of the 22nd ACM International Conference on Conference on Information & Knowledge Management (CIKM ’13)*.

• Qingbo Hu, Guan Wang, Shuyang Lin, Philip S. Yu. “Silence Behavior Mining on Social Networks”, in *Proceedings of the 9th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom ’13)*.

• Shuyang Lin, Fengjiao Wang, Qingbo Hu, Philip S. Yu “Extracting Social Events for Learning Better Information Diffusion Models”, in *Proceedings of the 19th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD ’13)*.

• Sihong Xie, Guan Wang, Shuyang Lin, Philip S. Yu “Review spam detection via temporal pattern discovery”, in *Proceedings of the 18th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD ’12)*.

• Charu Aggarwal, Shuyang Lin, Philip S. Yu “Influential Node Discovery in Dynamic Social Networks”, in *Proceedings of the 12th SIAM International Conference on Data Mining (SDM ’12)*.
This is a License Agreement between Shuyang Lin ("You") and Association for Computing Machinery, Inc. ("Association for Computing Machinery, Inc.") provided by Copyright Clearance Center ("CCC"). The license consists of your order details, the terms and conditions provided by Association for Computing Machinery, Inc., and the payment terms and conditions.

<table>
<thead>
<tr>
<th>License Number</th>
<th>3599040362385</th>
</tr>
</thead>
<tbody>
<tr>
<td>License date</td>
<td>Mar 30, 2015</td>
</tr>
<tr>
<td>Licensed content publisher</td>
<td>Association for Computing Machinery, Inc.</td>
</tr>
<tr>
<td>Licensed content publication</td>
<td>Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining</td>
</tr>
<tr>
<td>Licensed content title</td>
<td>Extracting social events for learning better information diffusion models</td>
</tr>
<tr>
<td>Licensed content author</td>
<td>Shuyang Lin, et al</td>
</tr>
<tr>
<td>Licensed content date</td>
<td>Aug 11, 2013</td>
</tr>
<tr>
<td>Type of Use</td>
<td>Thesis/Dissertation</td>
</tr>
<tr>
<td>Requestor type</td>
<td>Author of this ACM article</td>
</tr>
<tr>
<td>Is reuse in the author's own new work?</td>
<td>Yes</td>
</tr>
<tr>
<td>Format</td>
<td>Print and electronic</td>
</tr>
<tr>
<td>Portion</td>
<td>Full article</td>
</tr>
<tr>
<td>Will you be translating?</td>
<td>No</td>
</tr>
<tr>
<td>Order reference number</td>
<td>None</td>
</tr>
<tr>
<td>Title of your thesis/dissertation</td>
<td>Information Diffusion in Online Social Networks</td>
</tr>
<tr>
<td>Expected completion date</td>
<td>May 2015</td>
</tr>
<tr>
<td>Estimated size (pages)</td>
<td>150</td>
</tr>
<tr>
<td>Billing Type</td>
<td>Credit Card</td>
</tr>
<tr>
<td>Credit card info</td>
<td>Visa ending in 8068</td>
</tr>
<tr>
<td>Credit card expiration</td>
<td>02/2019</td>
</tr>
<tr>
<td>Total</td>
<td>8.00 USD</td>
</tr>
</tbody>
</table>

**Rightslink Terms and Conditions for ACM Material**

1. The publisher of this copyrighted material is Association for Computing Machinery, Inc. (ACM). By clicking "accept" in connection with completing this licensing transaction, you agree that the following terms and conditions apply to this transaction (along with the Billing and Payment terms and conditions established by Copyright Clearance Center, Inc.
2. ACM reserves all rights not specifically granted in the combination of (i) the license details provided by you and accepted in the course of this licensing transaction, (ii) these terms and conditions and (iii) CCC's Billing and Payment terms and conditions.

3. ACM hereby grants to licensee a non-exclusive license to use or republish this ACM-copyrighted material* in secondary works (especially for commercial distribution) with the stipulation that consent of the lead author has been obtained independently. Unless otherwise stipulated in a license, grants are for one-time use in a single edition of the work, only with a maximum distribution equal to the number that you identified in the licensing process. Any additional form of republication must be specified according to the terms included at the time of licensing.

*Please note that ACM cannot grant republication or distribution licenses for embedded third-party material. You must confirm the ownership of figures, drawings and artwork prior to use.

4. Any form of republication or redistribution must be used within 180 days from the date stated on the license and any electronic posting is limited to a period of six months unless an extended term is selected during the licensing process. Separate subsidiary and subsequent republication licenses must be purchased to redistribute copyrighted material on an extranet. These licenses may be exercised anywhere in the world.

5. Licensee may not alter or modify the material in any manner (except that you may use, within the scope of the license granted, one or more excerpts from the copyrighted material, provided that the process of excerpting does not alter the meaning of the material or in any way reflect negatively on the publisher or any writer of the material).

6. Licensee must include the following copyright and permission notice in connection with any reproduction of the licensed material: "[Citation] © YEAR Association for Computing Machinery, Inc. Reprinted by permission." Include the article DOI as a link to the definitive version in the ACM Digital Library. Example: Charles, L. "How to Improve Digital Rights Management," Communications of the ACM, Vol. 51:12, © 2008 ACM, Inc. http://doi.acm.org/10.1145/nnnnnn.nnnnn (where nnnnn.nnnnn is replaced by the actual number).

7. Translation of the material in any language requires an explicit license identified during the licensing process. Due to the error-prone nature of language translations, Licensee must include the following copyright and permission notice and disclaimer in connection with any reproduction of the licensed material in translation: "This translation is a derivative of ACM-copyrighted material. ACM did not prepare this translation and does not guarantee that it is an accurate copy of the originally published work. The original intellectual property contained in this work remains the property of ACM."

8. You may exercise the rights licensed immediately upon issuance of the license at the end of the licensing transaction, provided that you have disclosed complete and accurate details of your proposed use. No license is finally effective unless and until full payment is received from you (either by CCC or ACM) as provided in CCC's Billing and Payment terms and conditions.

9. If full payment is not received within 90 days from the grant of license transaction, then any license preliminarily granted shall be deemed automatically revoked and shall be void as if never granted. Further, in the event that you breach any of these terms and conditions or any of CCC's Billing and Payment terms and conditions, the license is automatically revoked and shall be void as if never granted.
10. Use of materials as described in a revoked license, as well as any use of the materials beyond the scope of an unrevoked license, may constitute copyright infringement and publisher reserves the right to take any and all action to protect its copyright in the materials.

11. ACM makes no representations or warranties with respect to the licensed material and adopts on its own behalf the limitations and disclaimers established by CCC on its behalf in its Billing and Payment terms and conditions for this licensing transaction.

12. You hereby indemnify and agree to hold harmless ACM and CCC, and their respective officers, directors, employees and agents, from and against any and all claims arising out of your use of the licensed material other than as specifically authorized pursuant to this license.

13. This license is personal to the requestor and may not be sublicensed, assigned, or transferred by you to any other person without publisher's written permission.

14. This license may not be amended except in a writing signed by both parties (or, in the case of ACM, by CCC on its behalf).

15. ACM hereby objects to any terms contained in any purchase order, acknowledgment, check endorsement or other writing prepared by you, which terms are inconsistent with these terms and conditions or CCC's Billing and Payment terms and conditions. These terms and conditions, together with CCC's Billing and Payment terms and conditions (which are incorporated herein), comprise the entire agreement between you and ACM (and CCC) concerning this licensing transaction. In the event of any conflict between your obligations established by these terms and conditions and those established by CCC's Billing and Payment terms and conditions, these terms and conditions shall control.

16. This license transaction shall be governed by and construed in accordance with the laws of New York State. You hereby agree to submit to the jurisdiction of the federal and state courts located in New York for purposes of resolving any disputes that may arise in connection with this licensing transaction.

17. There are additional terms and conditions, established by Copyright Clearance Center, Inc. ("CCC") as the administrator of this licensing service that relate to billing and payment for licenses provided through this service. Those terms and conditions apply to each transaction as if they were restated here. As a user of this service, you agreed to those terms and conditions at the time that you established your account, and you may see them again at any time at http://myaccount.copyright.com

18. Thesis/Dissertation: This type of use requires only the minimum administrative fee. It is not a fee for permission. Further reuse of ACM content, by ProQuest/UMI or other document delivery providers, or in republication requires a separate permission license and fee. Commercial resellers of your dissertation containing this article must acquire a separate license.

Special Terms:

Questions? customercare@copyright.com or +1-855-239-3415 (toll free in the US) or +1-978-646-2777.

Gratis licenses (referencing $0 in the Total field) are free. Please retain this printable license for your reference. No payment is required.
This is a License Agreement between Shuyang Lin ("You") and Association for Computing Machinery, Inc. ("Association for Computing Machinery, Inc.") provided by Copyright Clearance Center ("CCC"). The license consists of your order details, the terms and conditions provided by Association for Computing Machinery, Inc., and the payment terms and conditions.

License Number: 3599030687209
License date: Mar 30, 2015
Licensed content publisher: Association for Computing Machinery, Inc.
Licensed content publication: Proceedings of the 22nd ACM international conference on Conference on information & knowledge management
Licensed content title: Predicting trends in social networks via dynamic activeness model
Licensed content author: Shuyang Lin, et al
Licensed content date: Oct 27, 2013
Type of Use: Thesis/Dissertation
Requestor type: Author of this ACM article
Is reuse in the author's own new work?: Yes
Format: Print and electronic
Portion: Full article
Will you be translating?: No
Order reference number: None
Title of your thesis/dissertation: Information Diffusion in Online Social Networks
Expected completion date: May 2015
Estimated size (pages): 150
Billing Type: Credit Card
Credit card info: Visa ending in 8068
Credit card expiration: 02/2019
Total: 8.00 USD

Terms and Conditions

Rightslink Terms and Conditions for ACM Material

1. The publisher of this copyrighted material is Association for Computing Machinery, Inc. (ACM). By clicking "accept" in connection with completing this licensing transaction, you agree that the following terms and conditions apply to this transaction (along with the Billing and Payment terms and conditions established by Copyright Clearance Center, Inc. ("CCC"), at the time that you opened your Rightslink account and that are available at any
2. ACM reserves all rights not specifically granted in the combination of (i) the license
details provided by you and accepted in the course of this licensing transaction, (ii) these
terms and conditions and (iii) CCC's Billing and Payment terms and conditions.

3. ACM hereby grants to licensee a non-exclusive license to use or republish this ACM-
copyrighted material* in secondary works (especially for commercial distribution) with the
stipulation that consent of the lead author has been obtained independently. Unless otherwise
stipulated in a license, grants are for one-time use in a single edition of the work, only with a
maximum distribution equal to the number that you identified in the licensing process. Any
additional form of republication must be specified according to the terms included at the
time of licensing.

*Please note that ACM cannot grant republication or distribution licenses for embedded
third-party material. You must confirm the ownership of figures, drawings and artwork prior
to use.

4. Any form of republication or redistribution must be used within 180 days from the date
stated on the license and any electronic posting is limited to a period of six months unless an
extended term is selected during the licensing process. Separate subsidiary and subsequent
republication licenses must be purchased to redistribute copyrighted material on an extranet.
These licenses may be exercised anywhere in the world.

5. Licensee may not alter or modify the material in any manner (except that you may use,
within the scope of the license granted, one or more excerpts from the copyrighted material,
provided that the process of excerpting does not alter the meaning of the material or in any
way reflect negatively on the publisher or any writer of the material).

6. Licensee must include the following copyright and permission notice in connection with
any reproduction of the licensed material: "[Citation] © YEAR Association for Computing
Machinery, Inc. Reprinted by permission." Include the article DOI as a link to the definitive
version in the ACM Digital Library. Example: Charles, L. "How to Improve Digital Rights
http://doi.acm.org/10.1145/nnnnnn.nnnnnn (where nnnnnn.nnnnnn is replaced by the actual
number).

7. Translation of the material in any language requires an explicit license identified during
the licensing process. Due to the error-prone nature of language translations, Licensee must
include the following copyright and permission notice and disclaimer in connection with any
reproduction of the licensed material in translation: "This translation is a derivative of ACM-
copyrighted material. ACM did not prepare this translation and does not guarantee that it is
an accurate copy of the originally published work. The original intellectual property
contained in this work remains the property of ACM."

8. You may exercise the rights licensed immediately upon issuance of the license at the end
of the licensing transaction, provided that you have disclosed complete and accurate details
of your proposed use. No license is finally effective unless and until full payment is received
from you (either by CCC or ACM) as provided in CCC's Billing and Payment terms and
conditions.

9. If full payment is not received within 90 days from the grant of license transaction, then
any license preliminarily granted shall be deemed automatically revoked and shall be void as
if never granted. Further, in the event that you breach any of these terms and conditions or
any of CCC's Billing and Payment terms and conditions, the license is automatically revoked
and shall be void as if never granted.
10. Use of materials as described in a revoked license, as well as any use of the materials beyond the scope of an unrevoked license, may constitute copyright infringement and publisher reserves the right to take any and all action to protect its copyright in the materials.

11. ACM makes no representations or warranties with respect to the licensed material and adopts on its own behalf the limitations and disclaimers established by CCC on its behalf in its Billing and Payment terms and conditions for this licensing transaction.

12. You hereby indemnify and agree to hold harmless ACM and CCC, and their respective officers, directors, employees and agents, from and against any and all claims arising out of your use of the licensed material other than as specifically authorized pursuant to this license.

13. This license is personal to the requestor and may not be sublicensed, assigned, or transferred by you to any other person without publisher's written permission.

14. This license may not be amended except in a writing signed by both parties (or, in the case of ACM, by CCC on its behalf).

15. ACM hereby objects to any terms contained in any purchase order, acknowledgment, check endorsement or other writing prepared by you, which terms are inconsistent with these terms and conditions or CCC's Billing and Payment terms and conditions. These terms and conditions, together with CCC's Billing and Payment terms and conditions (which are incorporated herein), comprise the entire agreement between you and ACM (and CCC) concerning this licensing transaction. In the event of any conflict between your obligations established by these terms and conditions and those established by CCC's Billing and Payment terms and conditions, these terms and conditions shall control.

16. This license transaction shall be governed by and construed in accordance with the laws of New York State. You hereby agree to submit to the jurisdiction of the federal and state courts located in New York for purposes of resolving any disputes that may arise in connection with this licensing transaction.

17. There are additional terms and conditions, established by Copyright Clearance Center, Inc. ("CCC") as the administrator of this licensing service that relate to billing and payment for licenses provided through this service. Those terms and conditions apply to each transaction as if they were restated here. As a user of this service, you agreed to those terms and conditions at the time that you established your account, and you may see them again at any time at http://myaccount.copyright.com

18. Thesis/Dissertation: This type of use requires only the minimum administrative fee. It is not a fee for permission. Further reuse of ACM content, by ProQuest/UMI or other document delivery providers, or in republication requires a separate permission license and fee. Commercial resellers of your dissertation containing this article must acquire a separate license.

Special Terms:

Questions? customercare@copyright.com or +1-855-239-3415 (toll free in the US) or +1-978-646-2777.

Gratis licenses (referencing $0 in the Total field) are free. Please retain this printable license for your reference. No payment is required.
Permission Letter

April 17, 2015

Springer reference

Advances in Knowledge Discovery and Data Mining
Lecture Notes in Computer Science Volume 9077, 2015, pp 82-95
Date: 17 Apr 2015
Understanding Community Effects on Information Diffusion
© Springer International Publishing Switzerland 2015
Shuyang Lin, Qingbo Hu, Guan Wang, Philip S. Yu
DOI 10.1007/978-3-319-18038-0_7
Print ISBN 978-3-319-18037-3
Online ISBN 978-3-319-18038-0

Your project
Requestor: Shuyang Lin
linshuy@gmail.com
University: University of Illinois at Chicago
Purpose: Dissertation/Thesis

With reference to your request to reuse material in which Springer Science+Business Media controls the copyright, our permission is granted free of charge under the following conditions:

Springer material
- represents original material which does not carry references to other sources (if material in question refers with a credit to another source, authorization from that source is required as well);
- requires full credit (Springer and the original publisher, book/journal title, chapter/article title, volume, year of publication, page, name(s) of author(s), original copyright notice) to the publication in which the material was originally published by adding: "With permission of Springer Science+Business Media";
- may not be altered in any manner. Abbreviations, additions, deletions and/or any other alterations shall be made only with prior written authorization of the author and/or Springer Science+Business Media;
- Springer does not supply original artwork or content.

This permission
- is non-exclusive;
- is valid for one-time use only for the purpose of defending your thesis and with a maximum of 100 extra copies in paper. If the thesis is going to be published, permission needs to be reobtained.
- includes use in an electronic form, provided it is an author-created version of the thesis on his/her own website and his/her university’s repository, including UMI (according to the definition on the Sherpa website: http://www.sherpa.ac.uk/romeo/);
- is subject to courtesy information to the co-author or corresponding author;
- is personal to you and may not be sublicensed, assigned, or transferred by you to any other person without Springer’s written permission;
- is only valid if no personal rights, trademarks, or competitive products are infringed.

This license is valid only when the conditions noted above are met.

Permission free of charge does not prejudice any rights we might have to charge for reproduction of our copyrighted material in the future.
**Title:** Steering Information Diffusion Dynamically against User Attention Limitation

**Conference:** Data Mining (ICDM), 2014 IEEE International Conference on

**Author:** Shuyang Lin; Qingbo Hu; Fengjiao Wang; Yu, P.S.

**Publisher:** IEEE

**Date:** 14-17 Dec. 2014

---

**Thesis / Dissertation Reuse**

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

*Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:*

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.

2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.

3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

*Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:*

1) The following IEEE copyright/credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]

2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.

3) In placing the thesis on the author’s university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to [http://www.ieee.org/publications_standards/publications/rights/rights_link.html](http://www.ieee.org/publications_standards/publications/rights/rights_link.html) to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

---