7-11-2015

Influence Maximization Mining for Competitive Social Networks

Xiaoni Cao
University of Windsor

Influence Maximization Mining for Competitive Social Networks

by

Xiao Ni Cao

A Thesis

Submitted to the Faculty of Graduate Studies

through the School of Computer Science

in Partial Fulfillment of the Requirements for

the Degree of Master of Science at the

University of Windsor

Windsor, Ontario, Canada

2015

Influence Maximization Mining for Competitive Social Networks

by

Xiao Ni Cao

APPROVED BY:

Dr. Animesh Sarker

Department of Mathematics and Statistics

Dr. Dan Wu

School of Computer Science

Dr. Christie Ezeife, Advisor

School of Computer Science

April 20, 2015

DECLARATION OF ORIGINALITY

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

Influence maximization (IM) is one of the fundamental problems in the area of influence propagation in social networks. Recent studies in influence maximization have primarily focused on the diffusion of single influence. In this thesis, we study the problem under a new diffusion model named Competing General Threshold (CGT) model, which discovers k most influential nodes as early adopters of technology A (e.g., Apple) in a market where a competing technology B (e.g., Blackberry) already exists along with a set of early adopters of technology B. To solve IM under the diffusion of two influences, we first define the CGT diffusion model, then estimate both A and B influence probabilities by using Maximum-Likelihood Estimation from Twitter networks. Next, we propose a new algorithm named cgtMineA to find k influential A-seeds under the CGT model. Experimental results on Twitter networks show that our approach outperforms CELF by 15%.

Keywords. Competing Ideas, General Threshold Model, Influence Maximization, Social Networks.

DEDICATION

To my parents.

-Best parents in the business.

ACKNOWLEDGEMENTS

I would like to express the deepest appreciation to my advisor *Dr. Christie Ezeife*. Without her supervision and constant help this thesis would not have been possible. Thanks also for the research assistantship supports through FedDev and NSERC grants.

I would like to thank very much my committee members, *Dr. Dan Wu*, *Dr. Animesh Sarker*, and *Dr. Tsin* for their time, patience, objectivity, and observations.

Last but not least, I would like to thank very much my predecessor colleagues, *Mumu* and *Sabbir* at University of Windsor for their valuable work and experiences which are the inspiration of this research.

Xiao Ni Cao

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Social and Information Network Analysis

Social networks such as Facebook, Twitter, Google+, and so on can be modeled as directed graphs (also known as social network graphs) where the nodes represent individuals (e.g., human being or entities such as The New York Times) and the edges represent social ties, relationships or interactions between individuals. In information networks such as arXiv.org, wordpress, and so on, vertices are information items (e.g., research papers, software engineering projects, or blog posts), edges represent interactions between items. Some main types of large-scale networks that researchers have used for social and information network analysis are listed below.

Friendship Networks. Examples of friendship networks include Facebook which has $1,280,000,000$ users as of June 2014, and Twitter which has $645,750,000$ users as of 31 August, 2014. Friendship networks can be modeled using a directed graph where vertices represent people, and there is an directed edge (v, u) from v to u if v knows and likes u. For example, on Twitter, there are two people, Peter (whose user id is 318064061) and Mark (whose user id is 317756843). Peter follows Mark on July 7, 2011 (Twitter uses "follow" to say "I want to be friends with you"). In Figure 1.1,

we represent their social relationship using a Twitter follow graph where nodes are Peter and Mark respectively, a directed edge between them means that Peter follows Mark, a value "20110707" associated to the edge $(Peter, Mark)$ indicates the follow date.

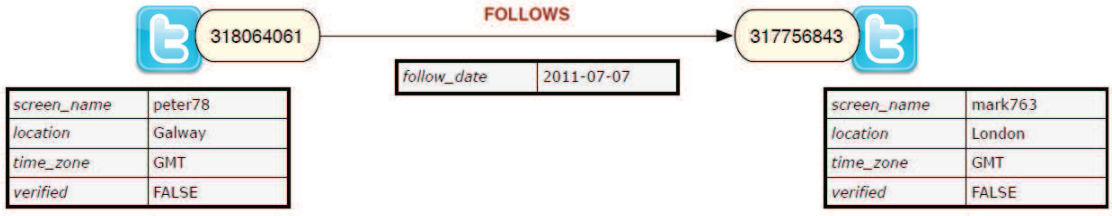


Figure 1.1: Following Graph on Twitter. Source: Figure on Page 12, Greene [2011]

Signed Networks. When two opposite relationships (such as like vs. dislike, love vs. hate, trust vs. distrust, friend vs. foe, and so on) coexist in a social network, we model this kind of social network using a weighted graph $G = (V, E, s)$, where individuals are represented by nodes, relationships between each other are represented by edges, and the sign (positive or negative) of relationships is represented by the edge weight $s \in \{-1, 1\}$:

$$s = \begin{cases} 1 & \text{if the relationship is like, trust, friend, etc.} \\ -1 & \text{if the relationship is dislike, distrust, foe, etc.} \end{cases}$$

For example, users on Wikipedia can vote for or against the nomination of others to be Wikipedia administrator, users on Epinions can express trust or distrust of other people's product reviews by rating, participants on Slashdot can declare others to be either "friends" or "foes" [Ahmed and Ezeife 2013], and users on Youtube can express like or dislike of other people's comments.

Citation Networks. Citation networks can be modeled using a citation graph where vertices represent research papers, and there is an directed edge from paper A to paper

B if A cites B. Examples of citation networks include arXiv.org.

Collaboration Networks. Collaboration networks (for example, Hollywood collaboration network or academic collaboration networks) can be modeled using a collaboration graph where vertices represent people, and there is an undirected edges between two people if they work together on at least one movie or one research project. Examples of collaboration networks include arXiv.org, Github, and DBLP.

Communication Networks. Communication networks model the "who-talks-to-whom", or "who-emails-whom" structure of social networks. Such networks can be constructed from the logs of emails or from phone call records [Mumu and Ezeife 2013]. Examples of communication networks include email communication network from Enron (as in the Enron Scandal). The Enron email network consists of $1,148,072$ emails sent between employees of Enron between 1999 and 2003 [KONECT 2014].

A number of algorithmic problems in online social and information networks analysis that researchers have been working on include (a) discovering the sentiment (positive, neutral, negative, or irrelevant attitude) toward celebrities (e.g., Obama), products (e.g., iPhone6), or topic (e.g., Super Bowl), exploring how news, opinions, or marketing information spread, predicting the trends and opinions on Twitter (b) making recommendations based on user profiles, examining friendships on Facebook, (c) processing resumes automatically and finding great new employees, clustering colleagues into circles on LinkedIn, (d) measuring document similarity, extracting frequent itemsets on Google+, (e) using natural language processing to perform sentiment analysis, mining subjective information from blog posts on the web, (f) organizing an email inbox, categorizing related emails together, detecting phishing emails, tracing how fraudulent activity diffuses within the Enron email corpus, (g) finding great software engineers, inspecting collaborative software engineering process on GitHub, (h) analyzing the emotional characteristics of the content of a video, determining the video's virality on Youtube [Russell 2013], (i) maximizing the spread of

influence through a social network, that is to find a small set of influential people (the seed set) in the online communities (the crowd) such that if we market to them by giving free samples of our products to them, the final adoption of the new products will be maximized in the crowd through word-of-mouth networks given that there are millions of users on Twitter and a company only has a limited number of free samples (budget for the advertising campaign) to distribute, or to find a small set of influential blogs in the blogsphere such that reading them allows one to gain the most engaging information and the most trending topics given that there are countless posts on the web and one only has limited attention.

1.2 Thesis Outline

The remaining of the thesis is organized as follows. The remaining of Chapter 1 provides a brief introduction on data mining, discusses diffusion of innovations and influence maximization problem, illustrates submodularity and their properties, and states thesis problem and contributions. Chapter 2 describes related work on influence maximization in great details. Chapter 3 develops a solution framework by introducing the CGT model, proving its properties, and proposing an efficient greedy mining algorithm based on its properties, cgtMineA to solve Influence Maximization under CGT model. Chapter 4 presents our experimental results. Chapter 5 concludes our study and suggests future work.

1.3 Data Mining Algorithms Used in Social and Information Networks Analysis

Data mining algorithms can be grouped into three general categories based on the objectives of the task, frequent pattern mining, classification, and clustering. In

this section, we introduce the definitions and basic concepts on these 3 categories, algorithms from each category that have been exploited by researchers for mining social and information networks.

1.3.1 Frequent Pattern Mining

Finding frequent patterns is one of the fundamental data mining problems. Frequent patterns can be a set of items, for example:

- $\{grape, mango, salmon\}$ which is a set of items bought together in many transactions in a transaction database of a grocery store, implying a frequent buying pattern
- $\{'frequent','pattern','mining'\}$ which is a set of words appearing together in many documents, implying a phrase with a particular meaning
- $\{homework1, homework2\}$ which is a set of two homework assignments such that many sentences appear in both of them, implying plagiarism [Ullman et al. 2011]

In frequent pattern (or frequent itemset) mining problem, the input is a transaction database. For example, consider the transaction database $\mathcal{D}$ in Table 1.1 which contains 5 transactions. We say an itemset (or a pattern) is frequent if the number of transactions in which the itemset (or the pattern) appears is no less than a user-defined value (called the minimum support threshold). For example, if we specify the minimum support threshold at 3, then the itemset $(2, 3, 5)$ is a frequent itemset, since it appears in 3 transactions, i.e., in transaction 200 $(1, 2, 3, 5)$, in transaction 300 $(2, 3, 5)$, and in transaction 500 $(2, 3, 4, 5)$. The output is the frequent itemsets found in the transaction databse: the frequent $1-$element itemsets $L_1 = \{(1), (2), (3), (4), (5)\}$, the frequent $2-$element itemsets $L_2 = \{(1, 2), (2, 3), (2, 4), (2, 5), (3, 5)\}$,

the frequent 3−element itemsets $L_3 = \{(2,3,5)\}$, and the frequent itemsets of all size $L = L_1 \cup L_2 \cup L_3 = \{(1),(2),(3),(4),(5),(1,2),(2,3),(2,4),(2,5),(3,5),(2,3,5)\}$.

TID	Items
100	$(1,2,3,4)$
200	$(1,2,3,5)$
300	$(2,3,5)$
400	$(1,2,4,5)$
500	$(2,3,4,5)$

Table 1.1: Transaction database with 5 transactions

Once frequent itemsets have been found, we want to find out the relationship between these frequent itemsets, i.e., the association rules generated from these frequent itemsets. An association rule is a if-then clause. For example, a rule can be like "if a basket contains items 1, 2, 3, then it probably contains items 4, 5". In order to define how likely the if-then clause is evaluated to be true, we need to introduce the definition of confidence. The confidence of an association rule is the probability that items $item_{k+1},\ldots,item_{k+q}$ are in the basket given a basket contains items $item_1,\ldots,item_k$, where the itemset $item_{k+1},\ldots,item_{k+q}$ and the itemset $item_1,\ldots,item_k$ are disjoint. The Apriori algorithm, initially proposed by Agrawal et al. [1994], is one of the most influential algorithms used to find frequent itemsets. Mumu and Ezeife [2014] exploit the Apriori algorithm to infer community preferences (positive or negative) for a given product (e.g., iPhone) as input to standard influence maximization algorithms. The ExAminer algorithm, introduced by Bonchi et al. [2003], is used to find frequent itemsets whose size is no less than a user-specified value. Goyal et al [2008] exploit the ExAminer algorithm to discover action leaders from online community, which is the first frequent pattern based algorithm for influence maximization mining.

1.3.2 Classification Methods

Classification is to classify objects to their corresponding categories. More precisely, classification is the task of learning a target function f from a training set that maps each sample x in the test set to one of the predefined class labels y. The target function is also known as a classification model. (Source: Definition 4.1 on pages 146, Pan et al. [2006].)

For example, given a training dataset in Table 1.2, and a test dataset in Table 1.3, classification is to learn a classification model from the training set, then apply the learned model to the test set to classify the noncoding RNA into two classes: pseudohairpin or pre-miRNA. The classification results are illustrated in Table 1.4.

feature 1	feature 2	class label
69.07	1.04	pseudoHairpin
53.09	8.75	pseudoHairpin
55.45	0	pseudoHairpin
72.92	0	pseudoHairpin
43.02	12.94	pseudoHairpin
69.47	0	pre-miRNA
44.19	11.76	pre-miRNA
85.11	2.17	pre-miRNA
81.97	0	pre-miRNA

Table 1.2: The tiny noncoding RNA training dataset with 5 pseudoHairpin samples and 4 pre-miRNA samples.

feature 1	feature 2	class label
56.38	2.70	unknown
42.68	12.35	unknown

Table 1.3: The tiny noncoding RNA test dataset with 2 unknown samples.

feature 1	feature 2	class label
56.38	2.70	pre-miRNA
42.68	12.35	pseudoHairpin

Table 1.4: The tiny noncoding RNA test dataset with 2 learned samples.

Classification algorithms include nearest neighbours (K-NN) which was proposed by Cover and Hart [1967], Naive Bayes classifier introduced by McCallum et al., [1998], Support Vector Machine (SVM) proposed by Cortes and Vapnik [1995], decision trees proposed by Quinlan [1986]. In [Hu et al. 2014], the authors propose an algorithm that exploits classification algorithms to tackle the Influence Maximization Problem and uses the result of a greedy algorithm to train classifiers to directly select influential nodes based on their features (Figure 1.2).

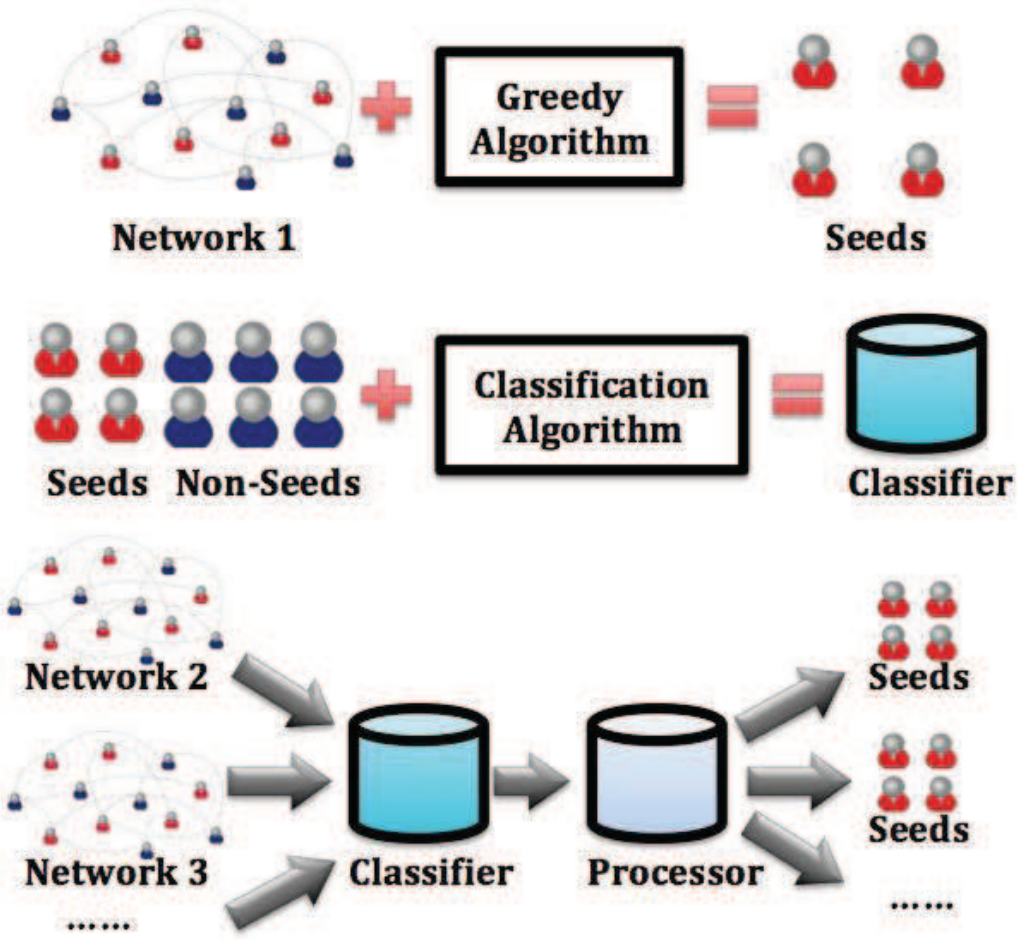


Figure 1.2: Classification in Influence Maximization. Source: Figure 1 on Page 1, Hu et al. [2014]

1.3.3 Clustering Methods

Clustering is to cluster objects in groups such that objects within a group are similar, objects between groups are different. That is, clustering techniques are trying both to maximize the similarity within a group and to maximize the difference between groups [Tan et al. 2006]. Clustering methods include the K-Means algorithm proposed by MacQueen et al., [1967] and Agglomerative Hierarchical Clustering. We will briefly discuss them below.

K-Means. The input of K-means algorithm is a set of points. The K-Means algorithm assumes there are k clusters in the point set (that is why it is called K-means.) K-means picks k points that are likely to be in different clusters as the centroid for each cluster. Then it assigns each remaining point p in the point set to a cluster such that the centroid of the cluster to which p is closest. After a point is added to a cluster, the centroid of the cluster is adjusted in order to take account of the new point [Ullman et al. 2011]. In [Soni and Ezeife 2013], the authors improve the K-means algorithm and propose a novel approach named Semantic non-parametric K-Means++ to automatically move emails from inbox to appropriate folders and sub-folders.

Hierarchical Clustering. In general, agglomerative (bottom-up) hierarchical clustering starts with a set of points and each point forms a cluster. And there is a distance matrix storing the distances between all pairs of points (i.e., clusters). Based on the distance matrix, the algorithm chooses two points (i.e., clusters) with the minimum distance in the matrix, combines them into one cluster, computes the distances between all pairs of the newly combined cluster and the old clusters, and use the resulting distances to update the distance matrix (Since we combine two clusters into one, so the distance matrix is reduced by one column and one row). The algorithm repeats this procedure (i.e., choosing two clusters with the minimum distance in the distance matrix, combining them into one cluster, and updating the distance matrix) until the minimum distance in the distance matrix is larger than a specified threshold

(which means if the points in the cluster are separated too far from each other, the algorithm would stop). In [Chen et al. 2014], the authors exploit the hierarchical clustering algorithm to improve the efficiency of mining influence maximization by discovering the community structure of the network to reduce the search space for influential nodes (Figure 1.3).

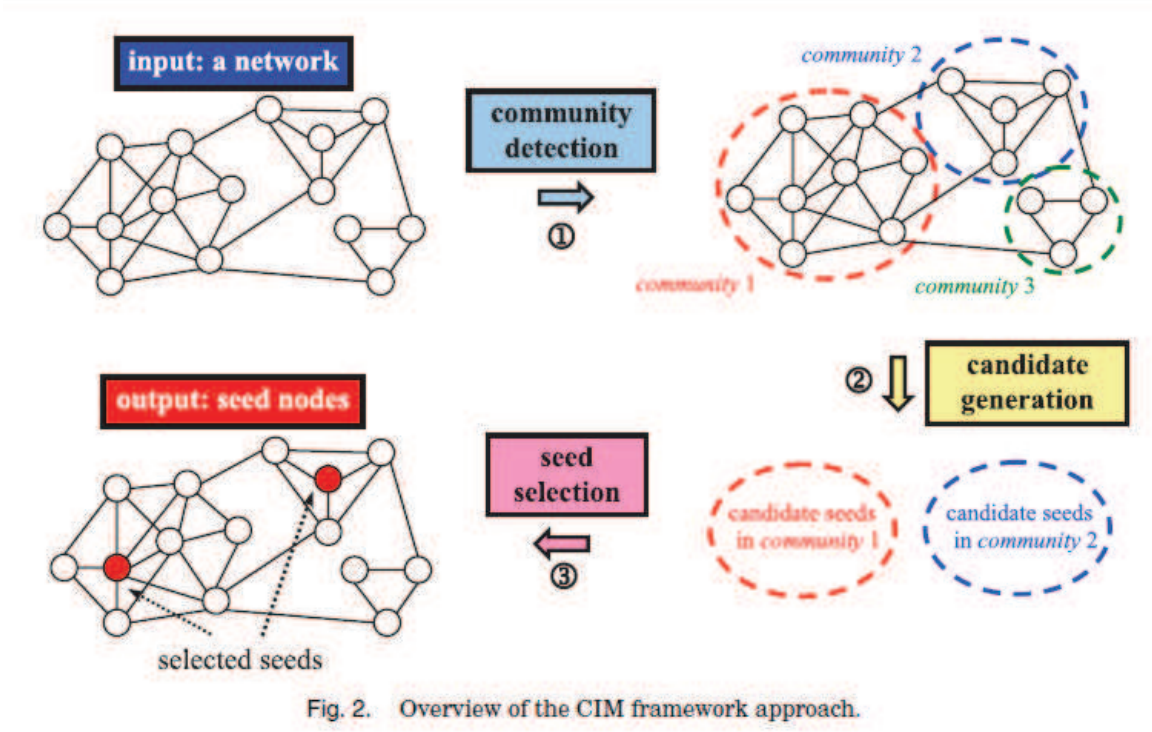


Figure 1.3: Classification in Influence Maximization. Source: Figure 2 on Page 3, Chen et al. [2014]

1.4 Diffusion of Innovations and Influence

According to Rogers [2010], one reason why the diffusion of innovations has been of so much interest to researchers is because getting an innovation adopted is often very difficult. Rogers [2010] defines that diffusion is the process by which an innovation is communicated through certain channels over time among the individuals of a social system. By innovation, he means a new idea or technology such as Google's search

engine, a new practice such as water boiling in a Peruvian village, or a new product such as Apple's iPhone. By communication channels, he means the means by which messages get from one individual to another. He compares two communication channels, mass media channels and interpersonal channels as follows. On one hand, mass media channels including radio, television, newspapers, and so on, are efficient means to inform an audience of potential adopters about the existence of an innovation. On the other hand, interpersonal channels like peer groups linking two or more individuals who are near-peers are more powerful in persuading an individual to adopt an innovation. By a social system, he means a set of individuals or organization connected to one another through relationships and interactions such as all the users on Twitter. Rogers [2010] points out that most individuals tend to be less dependent on the objective evaluations by scientific studies. Rather they adopt an innovation mainly because individuals from peers have previously adopted the innovation and conveyed a subjective evaluation of an innovation to them. Therefore, the diffusion of innovations through social networks is when individuals imitate their friends, friends of friends, colleagues in the workplace or at school, family members, acquaintances who have previously adopted an innovation by adopting the innovation as well, such adoptions will subsequently influence others who have connections with them. For example, David watches a new movie (here watching a new movie indicates an innovation.) He really likes it and blogs about the movie. David's friends Sean, Sibyl, and Eva read his blog and go watch the movie as well. After that, the action of watching the movie propagates recursively. Sean, Sibyl, and Eva influence their friends to watch the movie, and so on, creating a cascade of further watching. The diffusion process will carry on until no more adoptions are possible. Such chain reaction by words-of-mouth effect in a social network is called viral marketing (also known as direct marketing) because the adoption of the innovation will widely spread out like the way an epidemic spreads.

1.5 Influence Diffusion Models

In this section, we will introduce four main influence diffusion models. But before we do that, we will briefly introduce some terminology used in existing influence maximization research.

Definition 1.5.1. ***Innovation.*** *In this thesis, an "innovation" indicates a new technology, a new product, a new idea, or a new behavior/action. We use the term innovation, technology, product, idea, behavior and action interchangeably.*

Definition 1.5.2. ***Influence Diffusion.*** *Also known as influence propagation. Here, "diffusion" means "propagation". In this thesis, "diffusion" is a process by which the adoption of an innovation propagates throughout a social network from a seed set (i.e., a small number of early adopters of the innovation) to the crowd. Informally, we can think of this as an influence (for performing certain actions) propagating from the seed set to the crowd [Goyal et al. 2008]. Or diffusion is the outcome of influence [Ezeife 2014].*

Two of the most basic and influential diffusion models are the Independent Cascade model introduced by Goldenberg et al. [2001] and the Linear Threshold model introduced by Granovetter [1978]. Kempe et al. [2003] further formalized them to what they are in present and proposed the General Threshold model and the General Cascade model, which are broad generalizations of the Linear Threshold model and the Independent Cascade model respectively. The four diffusion models agree in the following aspects. The diffusion models represent a social network as a weighted, directed graph $G = (V, E)$. Each node $v \in V$ is an individual, each edge $(v, u) \in E$ is an influence relationship from node v to node u indicating that node v exerts influence on node u. Each edge $(v, u) \in E$ is assigned a non-negative probability $p_{v,u}$ or a non-negative weight $b_{v,u}$ indicating the amount of the influence that node v exerts on node u to adopt an innovation. The diffusion process is dynamic and progressive. By

dynamic we mean the diffusion process happens in discrete steps, i.e., $t = 0, 1, ..., n-1$ (where $n = |V|$, the size of V). At any time t, each node $v \in V$ has two states, active (meaning it has adopted an innovation) or inactive (meaning it has not adopted the innovation). By progressive we mean a node once becomes active at time t, it will remain active as time goes by and cannot switch back to inactive. If we use S_t to denote the set of active nodes at time t, then $S_{t-1} \subseteq S_t$ for $t \geq 1$, that is, the set of active nodes is non-decreasing as time moves in discrete steps, this is the progressive aspect of the diffusion. At time 0, there is an initial active set S_0 which represents a small set of influential nodes that adopts an innovation. The propagation process grows from there based on which diffusion model we choose. Since the set of active nodes is non-decreasing as time goes by in discrete steps, and the set V is finite, the process will stop on or before time $n-1$ when no more activations are possible. The four diffusion models differ in the way the influence of the neighborhood of a node v exerts on it and in the way a decision is made by node v to adopt a new behavior. They will be discussed briefly immediately and in details in Chapter 2.

Independent Cascade Model. The Independent Cascade model represents a social network as a weighted, directed graph $G = (V, E)$. Each edge $(v, u) \in E$ is assigned a non-negative probability $p_{v,u}$ indicating the influence that node v exerts on node u, that is if v is active, it succeeds in activating u with the probability of $p_{v,u}$. The diffusion process happens in discrete steps, i.e., $t = 0, 1, ..., n-1$ (where $n = |V|$, the size of V). At any time t, each node $v \in V$ is either active or inactive. Once v is activated, it remains active and cannot switch back to inactive. At time 0, there is an initial set S_0 that adopts a new behavior and the diffusion process unfolds as follows. If a node v is active, it is given one single chance to activate each of its inactive neighbors u with probability of $p_{v,u}$. The diffusion process will stop when no more activations are possible Kleinberg et al. [2007].

Linear Threshold Model. The Linear Threshold model represents a social network as a weighted, directed graph $G = (V, E)$. Each edge $(v, u) \in E$ is assigned a non-negative weight $b_{v,u}$ indicating the influence that v exerts on u such that the total weight of u's neighbors is no greater than 1: $\sum_{v \in N(u)} b_{v,u} \leq 1$, where $N(u)$ denotes the set of neighbors of u. Each node $v \in V$ chooses uniformly at random a threshold θ_v over the interval [0,1]. According to Granovetter [1978], in Sociology, the threshold of a node v is defined as the minimum proportion of its neighbors who have already adopted a behavior (such as joining a riot) that makes v adopt the behavior too. For example, suppose v's threshold is 25%, v has 100 neighbors, and 26 of them have joined a riot, since $26/100 = 26\% > 25\%$, v will join the riot too. A threshold of 0% means v is so radical that he will join the riot even there is no one else doing so. A threshold of 100% means v is so conservative that he will not join the riot even when everyone else around him does so. In influence maximization problems, a threshold of v, denoted as θ_v, intuitively indicates enough of its neighbors who have already adopted a behavior in order for v to do so. The threshold of each $v \in V$, denoted as θ_v being chosen uniformly at random is intended to model our lack of knowledge of the exact values [Kempe et al. 2003]. The diffusion process happens in discrete steps, i.e., $t = 0, 1, 2, ..., n-1$ (where $n = |V|$, the size of V). At any time t, each node $v \in V$ is either active or inactive. Once v is activated, it remains active and cannot switch back to inactive. At time 0, there is an initial set S_0 that adopts a new behavior. At time $t > 0$, all nodes that were active at time $t-1$ remain active, any inactive node u is activated if the total weight of its active neighbors is no less than its threshold: $\sum_{active\ v \in N(u)} b_{v,u} \geq \theta_u$. The process will stop when no more activations are possible [Kleinberg et al. 2007].

General Threshold Model. The General Threshold model represents a social network as a weighted, directed graph $G = (V, E)$. Each node $v \in V$ is associated with

a threshold function f_v. The threshold function $f_v(S)$ measures the joint influence of v's active neighbors S exerted on v, with $f_v(\emptyset) = 0$. Each node $v \in V$ chooses uniformly at random a threshold θ_v over the interval [0,1]. The diffusion process happens in discrete steps, i.e., $t = 0, 1, 2, ..., n-1$ (where $n = |V|$, the size of V). At any time t, each node $v \in V$ is either active or inactive. Once v is activated, it remains active and cannot switch back to inactive. At time 0, there is an initial set S_0 that adopts a new behavior. At time $t > 0$, all nodes that were active at time $t-1$ remain active, any inactive node v is activated if the threshold function of v is no less than the threshold of v: $f_v(S) \geq \theta_v$. The process will stop when no more activations are possible. The Linear Threshold model discussed above is a special case of the General Threshold model. In the Linear Threshold model, the threshold function of each node $u \in V$ is defined as the total weight of its active neighbors, $f_u(S) = \sum_{v \in S} b_{v,u}$, where S denotes the set of active neighbors of u, and $b_{v,u}$ is a non-negative weight on edge (v, u) indicating the influence that v exerts on u such that $\sum_{v \in N(u)} b_{v,u} \leq 1$, where $N(u)$ denotes the set of neighbors of u [Kempe et al. 2003].

General Cascade Model. The General Cascade model represents a social network as a weighted, directed graph $G = (V, E)$. Each node $u \in V$ is associated with an incremental function $p_u(v, S)$, where v is u's active neighbor who has not tried to influence u and S is the set of u's active neighbors that have tried and failed in activating u, $p_u(v, S)$ measures the influence of v on u given that the set of u's active neighbors that have tried and failed in influencing u. The diffusion process happens in discrete steps, i.e., $t = 0, 1, 2, ..., n-1$ (where $n = |V|$, the size of V). At any time t, each node $v \in V$ is either active or inactive. Once v is activated, it remains active and cannot switch back to inactive. At time 0, there is an initial set S_0 that adopts a new behavior and the diffusion process unfolds as follows. If a node v is active, it activates each of its inactive neighbors u with probability of $p_u(v, S)$. The process

will stop when no more activations are possible. The Independent Cascade model discussed above is a special case of the General Cascade model. In the Independent Cascade model, the incremental function of each node $u \in V$ is defined as the pairwise influence probability from v to u, $p_{v,u}$.

1.6 Submodular Functions and Their Properties

The diminishing returns definition of submodular function is as follows: Given a set of nodes $V = \{v_1, ..., v_n\}$, a function $f : 2^V \rightarrow \Re$ (where 2^V is the power set of V) is submodular if for any $A \subseteq B \subseteq V$ and $v \in V - B$, ($V - B$ means V except B or $V \backslash B$), we have that: $f(A \cup \{v\}) - f(A) \geq f(B \cup \{v\}) - f(B)$. The left hand side of the inequality means the marginal gain (or cost) of adding a node v in A, the right hand side of the inequality means the marginal gain (or cost) of adding a node v in B, the entire inequality says the marginal gain (or cost) of adding a node v in a larger set (i.e., B) is less than or equal to the marginal gain (or cost) of adding v in a smaller set (i.e., A). This is the diminishing return aspect of submodularity.

Example 1.6.1. ***Submodularity.*** *Consider scenario one. We have a network as shown in Figure 1.4 (a). We place two sensors S_1 and S_2 in the network to obtain a placement $A = \{S_1, S_2\}$ as shown in Figure 1.4 (b), we can see the coverage of $A = \{S_1, S_2\}$ is 8. After that, we add a new sensor S to placement A to obtain a placement $A' = \{S_1, S_2, S\}$ as shown in Figure 1.4 (c), we can see the additional (or marginal) coverage of the new sensor S is 8.*

Now, consider another scenario. We have a network as shown in Figure 1.4 (a). We place four sensors S_1, S_2, S_3, and S_4 in the network to obtain a placement $B = \{S_1, S_2, S_3, S_4\}$ as shown in Figure 1.4 (d), we can see the coverage of $B = \{S_1, S_2, S_3, S_4\}$ is 14. After that, we add a new sensor S to placement B to

obtain a placement $B' = \{S_1, S_2, S_3, S_4, S\}$ *as shown in Figure 1.4 (e), we can see the additional (or marginal) coverage of the new sensor* S *is* 4.

Figure 1.4 is trying to say that the marginal gain of adding a new node S *to a smaller sett* $A = \{S_1, S_2\}$ *is larger than the marginal gain of adding the same node* S *to a larger set* $B = \{S_1, S_2, S_3, S_4\}$. *This is the diminishing return aspect of submodularity.*

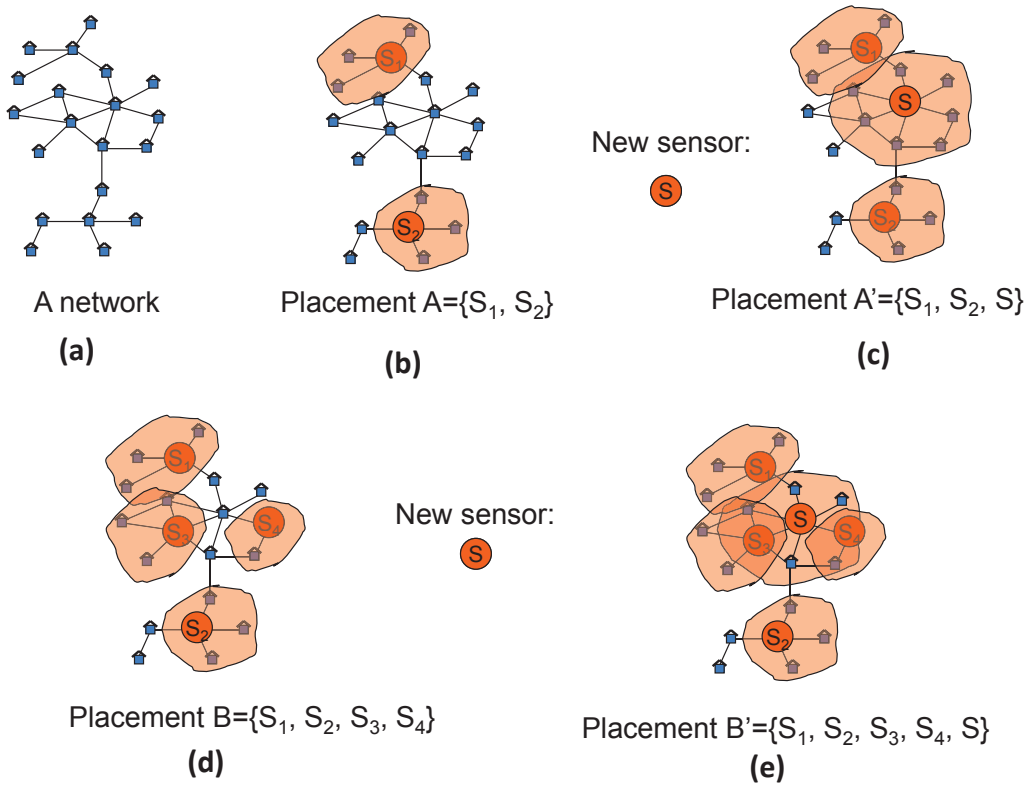


Figure 1.4: Diminishing Return of Submodular functions. Adopted from Figure on pages 8, Leskovec [2007].

Submodular functions have several properties. Of those properties, non-negative, monotone submodular functions are what we are interested in the context of influence maximization. A non-negative, monotone submodular function is defined as follows: A submodular function is monotone if it takes only non-negative values and it satisfies: $f(A \cup \{v\}) \geq f(A)$ for all elements $v \in V$ and sets $A \subseteq V$. The left hand side of the inequality means the gain (or cost) of adding a node v in A, the right hand side of

the inequality means the gain (or cost) of A, the entire inequality says the gain (or cost) of adding a node v in A) would not decrease the gain (or cost) of A.

Example 1.6.2. ***Monotonicity.*** *From Figure 1.4, it is easy to see that the coverage of* $A' = \{S_1, S_2, S\}$ *which is* 16 *(as shown in Figure 1.4 (c)) is no less than that of* $A = \{S_1, S_2\}$ *which is* 8 *(as shown in Figure 1.4 (b)).*

1.7 Influence Maximization and Its Applications

Having noticed the dynamics of spread of innovation unfold through a social network, a natural question to ask is how to maximize the spread of diffusion of the innovation, i.e., the influence maximization problem. Before further discussing the influence maximization problem, we will briefly review some terminology used in this thesis.

Definition 1.7.1. ***Influence Spread.*** *Given an initial active set* S_0*, the "influence spread" (just "influence", or just "spread") of* S_0*, denoted as* $\sigma(S_0)$*, is defined to be the expected number of final active nodes at the end of the diffusion process when no more adoptions are possible. Here,* $\sigma(\cdot)$ *is a function, defined as* $\sigma : 2^V \to \Re$*, mapping a set (the seed set* S_0*) to a real number (the expected number of final active nodes at the end of the diffusion process). On the other hand, the verb "influence" (as in node* v *influences node* u*) means "*v *activates node* u*".*

Definition 1.7.2. ***Influence Maximization.*** *Let* S_0 *denote an initial active seed set. Let* $\sigma(S_0)$ *denote the influence spread of the seed set* S_0*. Given a social graph* $G = (V, E)$*, a diffusion model, and an integer* k*, the influence maximization problem is to find a seed set* $S_0 \subseteq V$ *of size at most* k *such that* $\sigma(S_0)$ *is maximized under the diffusion model.*

Hardness of Influence Maximization Problems. The influence Maximization problem is proved to be NP-Complete, which means no polynomial-time algorithm

is known for it. If we can show the influence function $\sigma(\cdot)$ is a non-negative, monotone submodular under a diffusion process, then influence maximization problem boils down to a submodular function maximization problem. However submodular function maximization is proven to be NP-hard, therefore there is no known polynomial-time algorithm for this problem. But it can be solved approximately with guarantees in polynomial time according to Theorem 2.1 in [Kempe et al. 2003]. According to Kempe et al. [2003], if the influence function $\sigma(\cdot)$ is a non-negative, monotone submodular under a diffusion process, then we can exploit a greedy algorithm to find an approximation set S_0 of size k in polynomial time and $\sigma(S_0) \geq (1 - 1/e)\sigma(S_0^*)$, where S_0^* is the optimal set that maximizes the value of σ over all k-element sets and $e = 2.713$. In other words, the seed set S_0 found by the greedy algorithm provides a 63%-approximation to the influence maximization problem in polynomial time.

Applications of Influence Maximization. The most motivating application of influence maximization is viral marketing. Unlike mass marketing where all potential customers are targeted, viral marketing (also known as direct marketing) exploits data mining techniques to find out a handful of influential customers, by targeting them (e.g., giving them free samples of the new product), the rest of the viral marketing would take care of itself through word-of-mouth effect and the final adoption of the new product will reach a very large population of the network, like the spread of an epidemic [Domingos and Richardson 2001]. Another applications of influence maximization is outbreak detection. Suppose there are contaminants spreading over a water distribution network where nodes are pipe junctions and edges are pipes, we want to find a few locations (pipe junctions) to place sensors such that contaminants can be detected quickly and infect as few households as possible [Leskovec 2007]. Similarly, suppose an epidemic (e.g., Ebola) spreads through a social network where nodes are people and edges are the interactions between them, we want to find a small

set of contagious people to monitor such that the disease can be detected early and infects as few people as possible (or save as many lives as possible) [Leskovec 2007]. In the domain of blogsphere, where nodes are blog posts and edges are references, we want to find a few well-written quality blogs to gain as much information as possible [Leskovec 2007]. In the setting of collaboration networks, where nodes are researchers and edges are collaboration relationships, we want to find a few experts on a certain topic (e.g., database) [Tang et al. 2009]. In the setting of friendship networks, where nodes are individuals and edges are relationships, we want to find a few authoritative people on a certain product (e.g., iPhone) [Mumu and Ezeife 2013].

1.8 Learning Pairwise Influence Probabilities

In the studies of influence propagation in social networks, researchers represent a social network as a directed weighted social graph $G = (V, E)$ in which individuals are represented by nodes and there is a directed edge $(v, u) \in E$ from node v to node u indicating the propagation of influence from v to u. According to Goyal et al. [2010], real social networks do not have edge weights indicating the influence probability $p_{v,u}$ with which v influences u. Therefore, most of the researchers in this area assume the edge weights indicating the influence probabilities are given as input. In their experiments, researchers adopt primarily four models of assigning pairwise influence probabilities, i.e., the uniform model, the trivalency model, the random cascade model, and the weighted cascade model.

Uniform Model. In the uniform model, a uniform probability $p_{v,u}$ (e.g., 1%) is assigned to each edge $(v, u) \in E$ in the social graph. The uniform aspect means that all nodes exert the same amount of influence to their neighbors.

Trivalency Model. To differentiate the influence that each node $v \in V$ exerts on

their neighbors, the trivalency model assigns each edge $(v, u) \in E$ a probability $p_{v,u}$ chosen uniformly at random from the set $\{0.1, 0.01, 0.001\}$.

Random Cascade Model. Similar to the trivalency model, the random cascade model assigns each edge $(v, u) \in E$ a probability $p_{v,u}$ chosen uniformly at random from the interval [0,1] (rather than from a trilogy set).

Weighted Cascade Model. Different from the previous three models, the weighted cascade model takes the network structure into consideration. In the weighted cascade model, each edge $(v, u) \in E$ is assigned an influence probability $p_{v,u} = 1/d_u$ where d_u is the in-degree of u, i.e., the number of edges with u as their terminal vertex.

To compute the influence probabilities in a more involved way, Goyal et al. [2010] study both the network structure and user action logs. Goyal et al. [2010] tackle the problem of learning pairwise influence probabilities in social networks and define it formally as follows: Given a graph $G = (V, E, T)$ derived from a social network where $v \in V$ represents a user, an undirected edge $(u, v) \in E$ represents a social tie between user u and user v, $T : E \to N$ is a function mapping an edge to a timestamp at which the social tie is created, along with an action log $Actions(User, Action, Time)$, which is a relation containing tuples in the form of (u, a, t_u) indicating user $u \in V$ performs action $a \in A$ (where A denotes the universe of actions) at time t_u (for example, David watched the movie The Long Ranger at time 5) we want to learn a function $p : E \to [0, 1] \times [0, 1]$ such that each edge $(v, u) \in E$ is mapped to two influence probabilities $p_{v,u}$ (indicating the probability with which v influences u) and $p_{u,v}$ (indicating the probability with which u influences v). Goyal et al. [2010] use Flickr social network to construct the actions log and consider joining a group as the action. For example, to compute the pairwise influence probability $p_{v,u}$, first user u and user v need to become friends before the influence propagates from v to u, then the probability that v can influence u to adopt an action (i.e., joining a group) $p_{v,u} = \frac{\#groups\ that\ u\ joins\ after\ v\ \ joins}{\#groups\ \ that\ v\ joins}$. Based on their research, Ahmed and Ezeife [2013]

propose a new technique which mines the action log to find frequent patterns of action performed by both trusted and distrusted users and use the positive/negative patterns to learn both positive and negative influence probability under Bernoulli distribution.

1.9 Fundamental Twitter Terminology

In our solution framework, we learn influence probabilities from Twitter. Therefore, we would like to introduce fundamental twitter terminologies for the readers to better understand how we crawl Twitter to load data and perform data analysis. As a blogger, we can publish blog posts on blog platforms, to name a few, WordPress, Blogger or Tumblr. Likewise, a Twitter user can post microblogs called tweets on Twitter under their accounts. By micro, it means that each tweet consists of at most 140 characters. In addition to the 140-character text content, each tweet may consist of one or more of the following entities: mention, reply, retweet, hashtag, or URL. For example, the following tweet mentions @saradewitt, includes the hash tag #SXSWedu, and provides the URL pbskids.org/lab.

Figure 1.5: A tweet includes mention, hash tag and URL.

Figure 1.6: A reply.

Given a tweet, you can

- **Reply** it by clicking the Reply button on the tweet. And your reply will become a tweet which contains "@username" at the beginning of the tweet as shown in

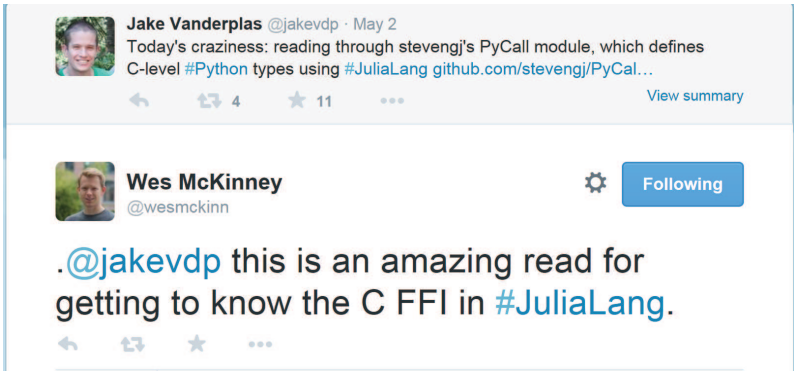


Figure 1.7: A reply.

Figure 1.8: A mention.

Figure 1.9: A retweet.

Figure 1.6. When you click on the tweet, you can find out to which tweet you replied as shown in Figure 1.7.

- **Retweet** it by clicking the Retweet button on the tweet to propagate the original tweet to all of your followers (which is an official way to quote another user's tweet). Your retweet will become a tweet which looks like the one shown in Figure 1.9.

- **Favorite** it by clicking the Favorite button on the tweet, indicating you like or are interested in the tweet.

Given a twitter account, you can

- **Follow** her/him, indicating you know, admire, or want to be friends with her/him. Intuitively, following or admiring, as a binary relation R over a universal set of Twitter users is transitive if whenever user a admires user b, and b in turn admires user c, then a also admires c. Twitter uses this transitivity to recommend Twitter users followed by those whom you are following for you to follow. Another thing to know about "follow" on Twitter is its asymmetry, i.e., you can follow anyone you like on Twitter without invitation or acceptance, but your followings do not have to follow you back, and most of the time they do not even know you exist.

- **Mention** her/him by containing "@username" anywhere in the body of your tweet, indicating you like their tweets. A tweet including mention is shown in Figure 1.8.

Remark 1.9.1. *Since a reply contains "@username" at the beginning of the tweet, a mention contains "@username" anywhere in the tweet, therefore a reply is a special instance of a mention.*

1.10 Thesis Contribution

Recent research in influence diffusion models has primarily focused on diffusion of single innovation cascade. However in the real world, there usually are multiple innovations competing within a social network [Zhang et al. 2014], for example, the launch of Apple's iPhone 6 to a market where Google's Nexus 5, Samsung's Galaxy S5, Blackberry's Q10, and so on already exist. In the setting of single influence diffusion models, there is only one technology (say technology A standing for Apple) in the network. We represent the underlying social network (the medium for the propagation of technology A) as $G = (V, E)$, where V represents individuals, E represents interactions between them. Initially (at time 0), there is only one seed set S_0 (i.e., a small number of early adopters of technology A). The adoption of technology A propagates throughout the social network from the seed set S_0 to the crowd. In the thesis, we extend the existing single influence diffusion to two influence diffusions. In the setting of two influence diffusions models, there are two technologies (technology A standing for Apple and technology B standing for Blackberry) in the network. We suppose technology B comes in the network first. There are two aspects to this extension. (1) We are studying influence maximization in the setting of two influence diffusions, the different setting determines a different input for the algorithm. The input for the thesis problem (to find an influential seed set S_0^A of size k in the network where the seed set S_0^B already exists) is the social network $G = (V, E)$, a seed set for technology B S_0^B, and a budget k for the size of a seed set for technology A S_0^A, while the input for influence maximization under single influence diffusion is the social network $G = (V, E)$ and a budget k for the size of a seed set for technology A S_0^A. (2) In the setting of two influence diffusions, the two influences propagate in a competitive way. Each node has four states in the two influence diffusion models, A meaning adopting technology A, B meaning adopting technology B, AB meaning adopting both A and B, 0 meaning adopting neither technology A nor technology B.

During the two influence diffusions process, once an inactive node v becomes active, say A-active (meaning adopting technology A), it cannot switch to other states (i.e., B, AB, or 0). This is the competitive aspect of the two influence difussions. This is because once a node v becomes, say A-active, it cannot switch to B, which means it blocks the influence propagation of technology B. The reason why existing algorithms like CELF which run in the single influence diffusion model cannot be directly applied under the two influence diffusion model is that the two influence diffusions unfold in a competing and random way. If the two diffusions unfold in a non-competing way, i.e., a B-node can switch to A, then we can simply apply CELF to find A-nodes in the graph using the parameters for influence A (such as $p^A(v,u)$, θ^A, which will be explained in Chapter 3). However, the diffusions unfold in a competing way, e.g., once a node becomes B-active, it cannot switch to A. If the two diffusions unfold in a deterministic way, then we can simply apply CELF to find A-nodes in the sub-graph that does not include B-nodes. However, the two diffusions unfold in a random way (because each node chooses uniformly at random two thresholds over [0,1]), there is no way to know which nodes would become B-nodes.

Second, in the studies of influence propagation in social networks, researchers represent a social network as a directed weighted social graph in which individuals are represented by nodes and there is a directed edge from node v to node u if v can influence u with the probability indicated as the edge weight. Goyal et al. [2010] point out that most of the researchers in this area assume the influence probabilities as the edge weights are given as input and ignore how the probabilities can be derived from social network data, i.e., user action logs. Goyal et al. [2010] use Flickr social network to construct the actions log and consider joining a group as the action. For example, Goyal et al. [2012] compute the probability that v can influence u as $p_{v,u} = \frac{\#groups\,that\,u\,joins\,after\,v\,joins}{\#groups\,that\,v\,joins}$. Based on their research, Ahmed and Ezeife [2013] propose a new technique which mines the action log to find frequent patterns of

action performed by both trusted and distrusted users and use the positive/negative patterns to learn both positive and negative influence probability under Bernoulli distribution. Ahmed and Ezeife [2013] use Epinions to construct the actions log and consider rating a user's product review as the action. They learn the pairwise influence probability from Epinions and consider rating a user's review as an action. They define the probability that v influences u as $p^{+}v,u = \frac{\#reviews\ u\ rates\ the\ same\ as\ v}{\#reviews\ v\ rates}$, and the probability that u is not influenced by v as $p^{-}v,u = \frac{\#reviews\ u\ rates\ not\ the\ same\ as\ v}{\#reviews\ v\ rates}$. In this thesis, the underlying social network we are studying is Twitter. We use MLE under Bernoulli distribution (as done in [Goyal et al. 2010] and [Ahmed and Ezeife, 2013]) to estimate the probability that u retweets v, the probability that u replies v, and the probability that u mentions v. We assume the probability that u retweets/replies/mentions v's tweets is the probability that v influences u to perform an action.

Contributions. Motivated by these limitations, the formal problem definition we propose to tackle is as follows:

Thesis Problem Definition 1.10.1. *Let S_0^A be the seed set for technology A, S_0^B the seed set for technology B. The influence spread for technology A of two seed sets S_0^A and S_0^B under the CGT model, denoted as $\sigma^A(S_0^A, S_0^B)$, is defined as the expected number of A-nodes at the end of the diffusion process.*

Given a directed social network $G = (V, E)$, a non-negative budget k, a seed set of B-nodes S_0^B, and CGT model, the problem of finding influential A-nodes when technology B already exists in the network is to find a seed set S_0^A as early adopters of technology A of size at most k such that $\sigma^A(S_0^A, S_0^B)$ is maximum.

The main contributions of thesis are as follows:

1. We propose a new well-defined diffusion model named Competing General Threshold (CGT) model which allows more than one competing innovation

(e.g. Apple as A when Blackberry as B is already in the market) to propagate in social networks under the CGT model, which makes it more general and natural

2. In order to compute the pairwise influence probabilities, we use Bernoulli Maximum-Likelihood Estimation for Twitter social network to construct the formula of the pairwise influence probabilities, then we use relational algebra operators left-join and projection on Twitter datasets to retrieve the parameters in the influence probabilities formula

3. We extend the existing threshold function [Goyal et al. 2010] under the single influence diffusion to define both A and B threshold functions under the CGT model

4. We claim that the influence spread function for A under our CGT model is a monotone, non-submodular function

5. We propose a new algorithm, cgtMineA, based on the greedy algorithm [Kempe et al. 2003] and the local search algorithm [Ahmed and Ezeife 2013] to find influential A-nodes in competitive social networks under the CGT model in polynomial time

6. We perform in depth analysis of our proposed solution using real life dataset collected from Twitter. In terms of the quality of seeds selected, our experiments show that cgtMineA outperforms CELF by 15%

Chapter 2

Related Works

Influence maximization was first introduced by Domingos and Richardson [2001]. Domingos and Richardson [2001] state that unlike mass marketing where all potential customers are targeted, direct marketing exploits data mining techniques to find out a handful of influential customers and targeting them, the rest of the direct marketing would take care of itself through word-of-mouth network, like the spread of an epidemic. To do that, they propose a general framework by modeling markets as social networks, and modeling social networks as Markov random fields where the probability that each customer adopts a new product is a function of both how much a customer feels desire for the product and the influence exerted by other customers. In addition, they make an important point that influence maximization depends not only on the influential individuals but also on the structure and context of the entire network. The problem of maximizing the spread of influence through a social network was then formalized by Kempe et al. [2003]. They first discuss two basic diffusion models, i.e., the Linear Threshold model and the Independent Cascade model. They then define the influence (spread) of a seed set S_0, denoted as $\sigma(S_0)$, to be the expected number of nodes who adopt the innovation at the end of the diffusion process. They next define the influence maximization problem as follows: Given a

graph $G = (V, E)$ derived from a social network, a budget k, the task is to find a k-element seed set S_0 over all k-element set $\subseteq V$ such that $\sigma(S_0)$ is maximized, that is S_0 yields the maximum influence on all nodes $\in V - S_0$ by getting the maximum expected number of active nodes at the end of the diffusion process. They adopt a hill-climbing algorithm and propose an efficient approximation solution which runs in polynomial time under both the Linear Threshold Model and the Independent Cascade Model. Based on the greedy algorithm proposed by Kempe et al. [2003], Leskovec et al. [2007] propose an efficient greedy algorithm named CELF working under both the Linear Threshold model and the Independent Cascade model, speeding up the original greedy algorithm by 700 times. The highlight of CELF is that the authors exploit the nice properties of submodular functions to significantly prune the number of iterations needed for influence estimation of a new candidate. In the setting of blogshpere, Agarwal et al. [2008] propose a novel approach to discovering influential bloggers by defining influence score for each blogger using the number of their blogs' inlinks, the number of comments their blogs receive, the number of their blogs' outlinks, and the length of the blog post. Goyal et al. [2011] develop an algorithm called SIMPATH for influence maximization under the Linear Threshold model. SIMPATH is an iterative method, building on the CELF [Leskovec et al. 2007], i.e., it exploits the lazy forward optimization proposed by CELF to select seeds iteratively. Unlike CELF, SIMPATH optimizes the spread estimation process in three key novel ways. In addition, it enhances the quality of the selection of seed set where they measure the quality of seed set on the basis of the spread of influence, i.e., the wider its spread, the better its quality. However, neither Linear Threshold model nor Independent Cascade model tackles influence maximization problem in signed social networks. To fill the gap, Ahmed and Ezeife [2013] propose a general framework named TGT where both positive relationships and negative relationships are considered and propose a new algorithm named MineSeedLS (as CELF-like algorithms cannot be applied to TGT

model) to discover influential nodes under the TGT. In [Mumu and Ezeife 2014], the authors propose a model named OBIN, which takes as input a social network graph $G = (V, E)$ and a product z and outputs an influence graph $G_z(V, E)$ for a product z from computed community preference where V is a sub-graph of the entire social network G containing only the relevant nodes to a certain product. The authors then perform influence maximization algorithms in the sub-graph containing only relevant nodes to a certain product. According to Goyal et al. [2010], real social networks do not have edge weights indicating the influence probability $p_{v,u}$ with which v influences u. Therefore, most of the researchers in this area assume the edge weights indicating the influence probabilities are given as input. Goyal et al. [2010] point out that although the real social network do not have the pairwise influence probability $p_{v,u}$ explicitly as the edge weight on $(v, u) \in E$, the probabilities can be derived from social network data, i.e., user action logs. We will discuss each of these papers in this chapter.

2.1 Influence Maximization

2.1.1 Maximizing the Spread of Influence through a Social Network

In [Kempe et al. 2003], the authors state that the motivation for researchers to study influence maximization comes from viral marketing, a marketing technique such that if a company wants to market a new product in the population, instead of targeting all possible customers, they would like to target a small set of influential people who have the ability of spreading the adoption of the new product to the crowd. Here, "target" means giving free samples of their new product to an individual. The question is who should they target in order to trigger the maximum final adoptions, i.e., the influence maximization problem?

The authors propose several diffusion models which describe how the dynamics of adoptions propagate throughout the social network, including the Linear Threshold model, the Independent Cascade model, and the General Threshold model as follows:

Linear Threshold Model. The Linear Threshold model represents a social network as a weighted, directed graph $G = (V, E)$. Each edge $(v, u) \in E$ is assigned a non-negative weight $b_{v,u}$ indicating the influence that v exerts on u such that $\sum_{v \in N(u)} b_{v,u} \leq 1$, where $N(u)$ denotes the set of neighbors of u. Each node $v \in V$ chooses uniformly at random a threshold θ_v over the interval [0,1]. The diffusion process happens in discrete steps, i.e., $t = 0, 1, 2, ..., n-1$. At any time t, each node $v \in V$ is either active or inactive. Once v is activated, it remains active and cannot switch back to inactive. At time 0, there is an initial set S_0 that adopts a new behavior. At time $t > 0$, all nodes that were active at time $t - 1$ remain active, any inactive node u is activated if the total weight of its active neighbors is no less than its threshold: $\sum_{active\ v \in N(u)} b_{v,u} \geq \theta_u$. The process will stop when no more activations are possible [Kleinberg et al. 2007].

Example 2.1.1. ***Linear Threshold Model.*** *We use Figure 2.1 to illustrate how the Linear Threshold Model works.*

Let S_t denote the set of active nodes at time t, $t = 0, 1, 2, ..., n-1$. Then $V - S_{t-1}$ denotes the set of inactive nodes at time t. At time 0 (Figure 2.1 (a)), there is a social network $G = (V, E)$, along with an initial set of active node(s), i.e., $S_0 = \{1\}$. At time 1, node 1 activates node 2 since $p_{1,2} = 1.0$ and $\theta_2 = 0.5$, but fails to activate node 3 since $p_{1,3} = 0.1$ and $\theta_3 = 0.5$ (Figure 2.1 (b)). At time 2, nodes 1 and 2 jointly activate node 3 since $p_{1,3} + p_{2,3} = 0.1 + 0.4 = 0.5$, and $\theta_3 = 0.5$ (Figure 2.1 (c)). At this point, the diffusion stops since no more activations are possible. From Figure Figure 2.1 (c), we can see the influence spread of $\{1\}$ is 3, the number of active nodes at the end of the diffusion.

Independent Cascade Models. The Independent Cascade model represents a

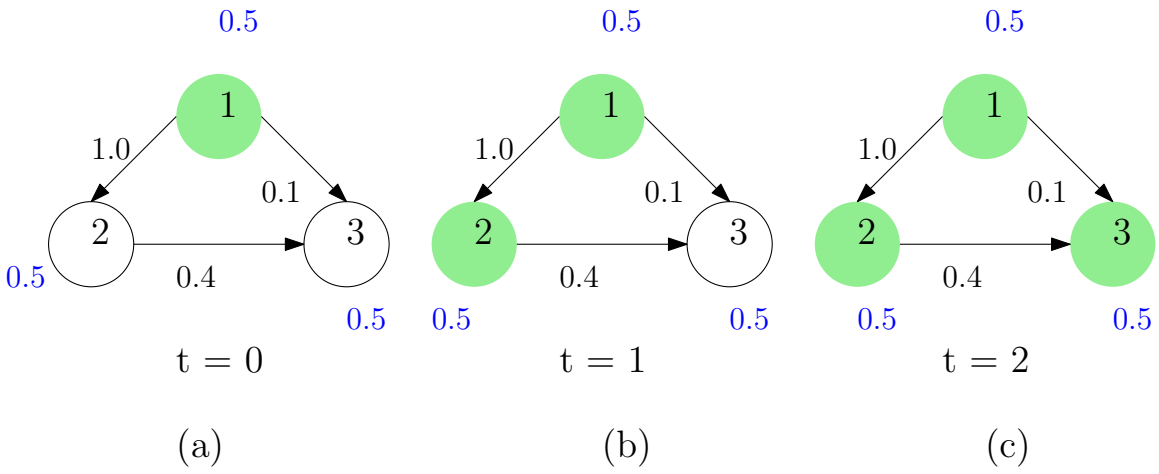


Figure 2.1: Linear Threshold Model

social network as a weighted, directed graph $G = (V, E)$. Each edge $(v, u) \in E$ is assigned a non-negative probability $p_{v,u}$ indicating the influence that node v exerts on node u, that is if v is active, it succeeds in activating u with the probability of $p_{v,u}$. The diffusion process happens in discrete steps, i.e., $t = 0, 1, ..., n-1$. At any time t, each node $v \in V$ is either active or inactive. Once v is activated, it remains active and cannot switch back to inactive. At time 0, there is an initial set S_0 that adopts a new behavior and the diffusion process unfolds as follows. If a node v is active, it is given one single chance to activate each of its inactive neighbors u with probability of $p_{v,u}$. By only one chance, we mean that if v, one of $u's$ active neighbors, attempts to active u at time t, regardless of whether v succeeds or not, v will not be granted another attempt to activate u in the following steps, i.e., v is not contagious to u anymore. If u has more than one active neighbors, each of its active neighbors will be given only one chance to activate u, one at a time and in an arbitrary order. The diffusion process will stop when no more activations are possible [Kleinberg et al. 2007].

Example 2.1.2. ***Independent Cascade Models.*** *We use Figure 2.2 to illustrate*

how the Independent Cascade model works.

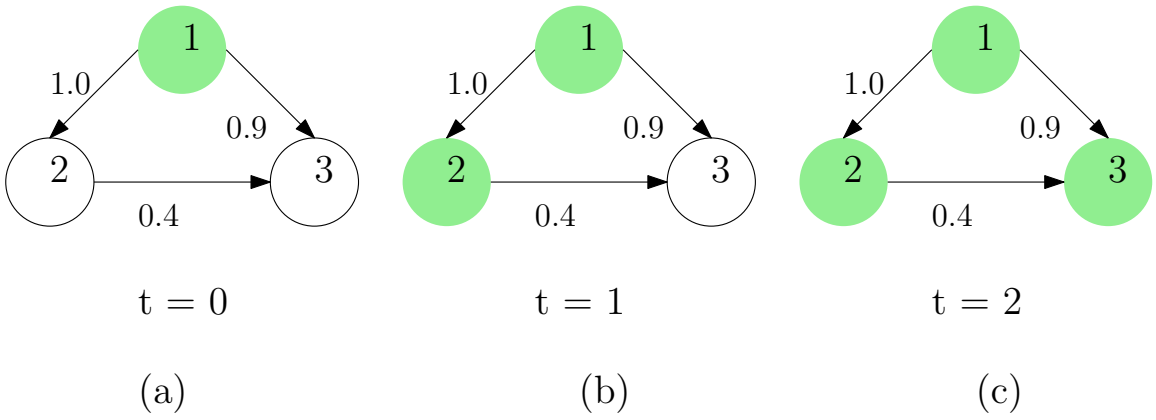


Figure 2.2: Independent Cascade Model.

Let S_t denote the set of active nodes at time t, $t = 0, 1, 2, ..., n-1$, with $S_{-1} = 0$. Then $V - S_{t-1}$ denotes the set of inactive nodes at time t. At time 0 (Figure 2.2 (a)), there is a social network $G = (V, E)$, along with an initial set of active node(s), i.e., $S_0 = \{1\}$. At time 1, node 1 activates node 2 since we flip a biased coin with the probability $p_{1,2} = 1.0$ to get a head for the influence propagation from node 1 to node 2, and we get a head, but fails to activate node 3 since we flip a biased coin with the probability $p_{1,3} = 0.9$ to get a head for the influence propagation from node 1 to node 3, and we get a tail (Figure 2.2 (b)). At time 2, nodes 2 activates node 3 since we flip a biased coin with the probability $p_{2,3} = 0.4$ to get a head for the influence propagation from node 2 to node 3, and we get a head (Figure 2.2 (c)). At this point, the diffusion stops since no more activations are possible. From Figure Figure 2.2 (c), we can see the influence spread of $\{1\}$ is 3, the number of active nodes at the end of the diffusion.

General Threshold Model. The General Threshold model represents a social network as a weighted, directed graph $G = (V, E)$. Each node $v \in V$ is associated

with a threshold function f_v. $f_v(S)$ measures the joint influence of v's active neighbors S exerted on v, with $f_v(\emptyset) = 0$. Each node $v \in V$ chooses uniformly at random a threshold θ_v over the interval [0,1]. The diffusion process happens in discrete steps, i.e., $t = 0, 1, 2, ..., n-1$. At any time t, each node $v \in V$ is either active or inactive. Once v is activated, it remains active and cannot switch back to inactive. At time 0, there is an initial set S_0 that adopts a new behavior. At time $t > 0$, all nodes that were active at time $t-1$ remain active, any inactive node v is activated if $f_v(S) \geq \theta_v$. The process will stop when no more activations are possible. The Linear Threshold model discussed above is a special case of the General Threshold model. In the Linear Threshold model, the threshold function of each node $u \in V$ is defined as the total weight of its active neighbors, $f_u(S) = \sum_{v \in S} b_{v,u}$, where S denotes the set of active neighbors of u, and $b_{v,u}$ is a non-negative weight on edge (v, u) indicating the influence that v exerts on u such that $\sum_{v \in N(u)} b_{v,u} \leq 1$, where $N(u)$ denotes the set of neighbors of u [Kempe et al. 2003].

Example 2.1.3. ***General Threshold Model.*** *We use Figure 2.3 to illustrate how the General Threshold model works.*

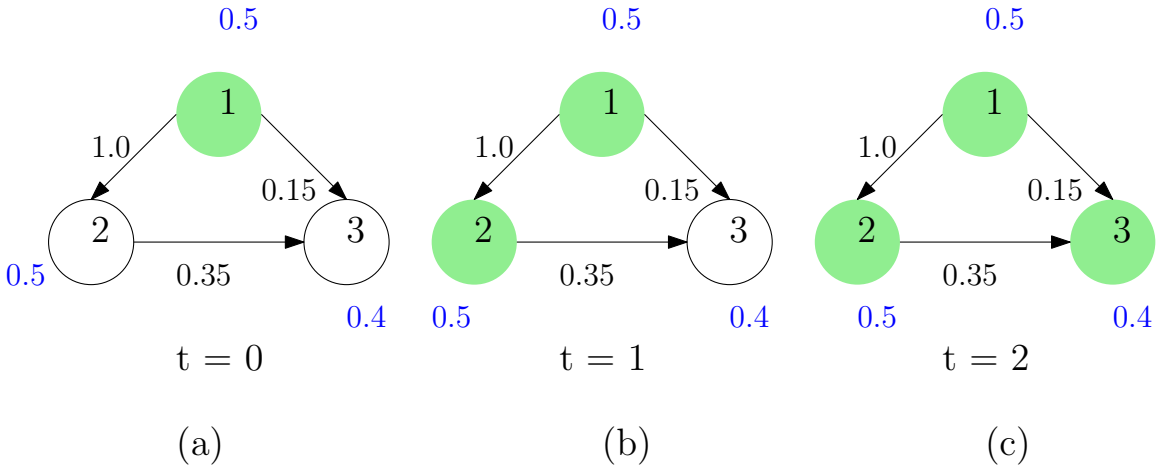


Figure 2.3: General Threshold Model

Let S_t denote the set of active nodes at time t, $t = 0, 1, 2, ..., n-1$. Then $V - S_{t-1}$ denotes the set of inactive nodes at time t. There are various ways to define the threshold function of node u, f_u. In this example, we define the threshold function of node u as $f_u = 1 - \prod_{v \in N(u) \cap S_{t-1}} (1 - p_{v,u})$, where $N(u)$ denotes the set of neighbors of node u, and $N(u) \cap S_{t-1}$ denotes the set of active neighbors of node u at time t. At time 0 (Figure 2.3 (a)), there is a social network $G = (V, E)$, along with an initial set of active node(s), i.e., $S_0 = \{1\}$. At time 1, node 1 activates node 2 since $p_{1,2} = 1.0$ and $\theta_2 = 0.5$, but fails to activate node 3 since $p_{1,3} = 0.1$ and $\theta_3 = 0.5$ (Figure 2.3 (b)). At time 2, nodes 1 and 2 jointly activate node 3 since $f_3(\{1, 2\}) = 1 - (1 - 0.15)(1 - 0.35) = 0.4475$, and $\theta_3 = 0.4$ (Figure 2.3 (c)). At this point, the diffusion stops since no more activations are possible. From Figure Figure 2.3 (c), we can see the influence spread of $\{1\}$ is 3, the number of active nodes at the end of the diffusion.

Then the authors define the problem of maximizing the influence spread through a social network formally as follows:

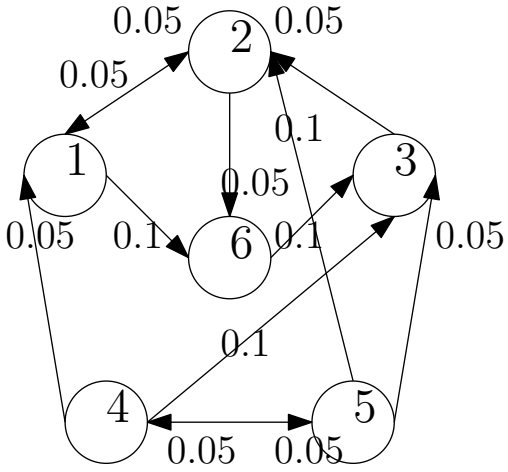
Given a graph $G = (V, E)$ derived from a social network and a budget k. Let $S_0 \subseteq V$ denote the initial seed set of active nodes. Let $\sigma(S_0)$ denote the influence spread of a seed set of nodes S_0, i.e., the expected number of active nodes at the end of the diffusion process with S_0 be the initial seed set at the beginning of the diffusion process, with $\sigma(\emptyset) = 0$. We would like to find a k-element set S_0 over all k-element set $\subseteq V$ such that $\sigma(S_0)$ is maximum.

The authors show that the influence maximization problem is NP-hard under both the Linear Threshold model and the Independent Cascade model. But it can be solved approximately with guarantees in polynomial time according to Theorem 2.1 in [Kempe et al. 2003]. According to Kempe et al. [2003], if the influence spread function $\sigma(\cdot)$ is a non-negative, monotone submodular under a diffusion process, then we can exploit a greedy algorithm to find an approximation set S_0 of size k in polynomial

time and $\sigma(S_0) \geq (1-1/e)\sigma(S_0^*)$, where S_0^* is the optimal set that maximizes the value of σ over all k-element sets and $e = 2.713$. In other words, S_0 found by the greedy algorithm provides a 63%-approximation to the influence maximization problem in polynomial time.

The authors show that the resulting influence spread function $\sigma(\cdot)$ is submodular under both the Linear Threshold Model and the Independent Cascade Model and present a Greedy Clibming Hill algorithm.

Example 2.1.4. ***Climbing Hill Algorithm.*** *We illustrate how the Greedy Climbing Hill algorithm works under the LT model through an example shown in Figure 2.4.*



Edge Weights $p_{v,u}$

$p_{1,2} = 0.05$
$p_{1,6} = 0.1$
$p_{2,1} = 0.05$
$p_{2,6} = 0.05$
$p_{3,2} = 0.05$
$p_{4,1} = 0.05$
$p_{4,3} = 0.1$
$p_{4,5} = 0.05$
$p_{5,2} = 0.1$
$p_{5,3} = 0.05$
$p_{5,4} = 0.05$
$p_{6,3} = 0.1$

Thresholde θ_v: $\theta_1 = \theta_2 = \theta_3 = \theta_4 = \theta_5 = \theta_6 = 0.1$

Figure 2.4: A Social Network

In the social network $G = (V, E)$ shown in Figure 2.4, there are 6 nodes and 12 edges connecting them. Each node v is associated as a threshold θ_v, each edge (v, u) is assigned an edge weight $p_{v,u}$. We set our budge $k = 2$, meaning we are looking for 2 influential nodes from this network. Greedy algorithm works as follows. Initially, it sets the seed set S to $\emptyset$. In the first pass, it evaluates the marginal gain of adding

Node	Marginal Gain
1	$\sigma(\{1\} \cup \emptyset) - \sigma(\emptyset) = 4$
2	$\sigma(\{2\} \cup \emptyset) - \sigma(\emptyset) = 1$
3	$\sigma(\{3\} \cup \emptyset) - \sigma(\emptyset) = 1$
4	$\sigma(\{4\} \cup \emptyset) - \sigma(\emptyset) = 2$
5	$\sigma(\{5\} \cup \emptyset) - \sigma(\emptyset) = 2$
6	$\sigma(\{6\} \cup \emptyset) - \sigma(\emptyset) = 2$

Table 2.1: Iteration One of Greedy

Node	Marginal Gain
2	$\sigma(\{1\} \cup \{2\}) - \sigma(\{1\}) = 0$
3	$\sigma(\{1\} \cup \{3\}) - \sigma(\{1\}) = 0$
4	$\sigma(\{1\} \cup \{4\}) - \sigma(\{1\}) = 1$
5	$\sigma(\{1\} \cup \{5\}) - \sigma(\{1\}) = 1$
6	$\sigma(\{1\} \cup \{6\}) - \sigma(\{1\}) = 0$

Table 2.2: Iteration Two of Greedy

node 1 *to* $S_0 = \emptyset$*, the marginal gain of adding node* 2 *to* $S_0 = \emptyset$*,..., the marginal gain of adding node* 6 *to* $S_0 = \emptyset$*, with* $\sigma(\emptyset) = 0$*, the results are shown in Table 2.1. It picks the node with the maximum marginal gain, which is node* 1*, and adds it to the seed set. At this moment,* $S_0 = \{1\}$ *In the second pass, it evaluates the marginal gain of adding node* 2 *to* $S_0 = \{1\}$*, the marginal gain of adding node* 3 *to* $S_0 = \{1\}$*,..., the marginal gain of adding node* 6 *to* $S_0 = \{1\}$*, the results are shown in Table 2.2. It picks the node with the maximum marginal gain, which is node* 4 *(or node* 5*), and adds it to the seed set. Now,* $S_0 = \{1, 4\}$*. Since* $k = 2$*, and we have found two influential nodes* 1 *and* 4*, we are done.*

2.1.2 CELF

In [Leskovec et al. 2007], the authors proposed an efficient algorithm named CELF which achieves the same results but runs 700 times faster than the original greedy algorithm proposed by [Kempe et al. 2003]. We use the exact social network used in illustrating the Greedy algorithm in section 2.2, to show how CELF works under the LT model. In the social network $G = (V, E)$ shown in Figure 2.4, there are 6 nodes and 12 edges connecting them. Each node v is associated as a threshold θ_v, each edge (v, u) is assigned an edge weight $p_{v,u}$. We set our budge $k = 2$, meaning we are looking for 2 influential nodes from this network. CELF works as follows. Initially, it sets the seed set to $\emptyset$. In the first pass, CELF works in the same way as the Greedy algorithm. It evaluates the marginal gain of adding node 1 to $\emptyset$, the marginal gain

of adding node 2 to $\emptyset$,..., the marginal gain of adding node 6 to $\emptyset$, with $\sigma(\emptyset) = 0$, the results are shown in Table 2.3. It picks the node with the maximum marginal gain, which is node 1, and adds it to the seed set. In the second passes, it does something different from the Greedy algorithm. Instead of evaluating the influence spread of all the combinations (i.e., $\{1,2\}$, $\{1,3\}$,$\{1,4\}$,$\{1,5\}$,$\{1,6\}$), CELF sorts the nodes 2, 3, 4, 5, 6 by the marginal gain of adding them to $\emptyset$, picks the node with the maximum marginal gain which is node 4 and evaluates the marginal gain of adding node 4 to $\{1\}$, which is 1. Then it picks the node with the second maximum spread which is node 5, and evaluates the marginal gain of adding node 5 to $\{1\}$, which is 1. Then it picks the node with the third maximum spread which is node 6, and evaluates the marginal gain of adding node 6 to $\{1\}$, which is 0. At this moment, we can stop without continuing evaluating the marginal gain of adding node 2 to $\{1\}$ and the marginal gain of adding node 3 to $\{1\}$. The reason why we can stop from there is that the influence spread function $\sigma(\cdot)$ is submodular under the Linear Threshold Model. According to the diminishing return of submodularity, we know

$$\begin{aligned}\sigma(\{2\} \cup \emptyset) - \sigma(\emptyset) &= 1 \\ &\geq \sigma(\{2\} \cup \{1\}) - \sigma(\{1\})\end{aligned}$$

$$\begin{aligned}\sigma(\{3\} \cup \emptyset) - \sigma(\emptyset) &= 1 \\ &\geq \sigma(\{3\} \cup \{1\}) - \sigma(\{1\})\end{aligned}$$

Therefore, neither $\sigma(\{2\} \cup \{1\}) - \sigma(\{1\})$ nor $\sigma(\{3\} \cup \{1\}) - \sigma(\{1\})$ is greater than 1, which is the current maximum marginal gain of adding node 4 to $\{1\}$.

Node	Marginal Gain
1	$\sigma(\{1\} \cup \emptyset) - \sigma(\emptyset) = 4$
4	$\sigma(\{4\} \cup \emptyset) - \sigma(\emptyset) = 2$
5	$\sigma(\{5\} \cup \emptyset) - \sigma(\emptyset) = 2$
6	$\sigma(\{6\} \cup \emptyset) - \sigma(\emptyset) = 2$
2	$\sigma(\{2\} \cup \emptyset) - \sigma(\emptyset) = 1$
3	$\sigma(\{3\} \cup \emptyset) - \sigma(\emptyset) = 1$

Table 2.3: Iteration One of CELF

Node	Marginal Gain
4	$\sigma(\{1\} \cup \{4\}) - \sigma(\{1\}) = 1$
5	$\sigma(\{1\} \cup \{5\}) - \sigma(\{1\}) = 1$
6	$\sigma(\{1\} \cup \{6\}) - \sigma(\{1\}) = 0$

Table 2.4: Iteration Two of CELF

2.1.3 SIMPATH

In [Goyal et al. 2011], the authors state that influence maximization is one of the fundamental problems in the area of influence propagation in social networks. The authors state that the motivation for researchers to study influence maximization comes from viral marketing, a marketing technique of giving free samples of a new product to a handful of influential people who spread the adoption of the new product to the crowd. According to the authors, the problem of influence maximization is to select k nodes such that by activating them, the expected spread of influence is maximized. The input of influence maximization algorithms is a social graph with influence probabilities of edges, the output of influence maximization algorithms is a k-node seed set [Goyal et al. 2011].

Under the Linear Threshold model, the authors establish a fundamental result which serves as the basis of the SIMPATH algorithm. The result says that the spread of a set of nodes can be derived from the sum of spreads of each node in the set on appropriate induced subgraph. In order to estimate the spread of a seed set, the authors compute the spread by making a list of the simple paths starting from the seed nodes, rather than using the computationally expensive Monte Carlo simulations. In order to reduce the number of spread estimation calls in the first iteration, the authors propose a novel optimization called VERTEX COVER OPTIMIZATION, which addresses a key shortcoming of the simple greedy algorithm that CELF [Leskovec et al. 2007] does not address. In order to reduce the running time of the spread estimation

process in the subsequent iterations, the authors propose another novel optimization called LOOK AHEAD OPTIMIZATION. More precisely, at the beginning of each iteration, the optimization generates $top-l$ most promising seed candidates and shares the marginal gain of those candidate seeds.

The authors develop an algorithm called SIMPATH for influence maximization under the linear threshold model. SIMPATH is an iterative method, building on the CELF [Leskovec et al. 2007], i.e., it exploits the lazy forward optimization proposed by CELF to select seeds iteratively. Unlike CELF, SIMPATH optimizes the spread estimation process in three key novel ways. In addition, it enhances the quality of the selection of seed set where they measure the quality of seed set on the basis of the spread of influence, i.e., the wider its spread, the better its quality.

The authors first introduce the properties of Linear Threshold model, which serves as the basis of SIMPATH. Recall that in the Linear Threshold model a node v picks at most one of its incoming edge with a probability of $b_{v,w}$. Then the selected edge is considered live, the unselected edges are considered blocked. Let X denote one possible set of outcomes on the edges (for example, $\{edge1 : live, edge2 : live, edge3 : blocked...\}$) and $\sigma_X(S)$ denote the number of nodes that can be reached from S via live paths (a live path consists of only live edges) in X. Then, by the definition of the spread of S,

$$\sigma(S) = \sum_X Pr[X] \cdot \sigma_X(S) \tag{2.1}$$

$$\sigma_X(S) = \sum_{v \in V} I(S, v, X) \tag{2.2}$$

$$I(S, v, X) = \begin{cases} 1 & \text{if there is a live path in X from any node in S to v} \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$

Substitute equations 2.2 and 2.3 to 2.1, we obtain

$$\sigma(S) = \sum_{v \in V} \sum_{X} Pr[X] \cdot I(S, v, X) = \sum_{v \in V} \Upsilon_{S,v} \tag{2.4}$$

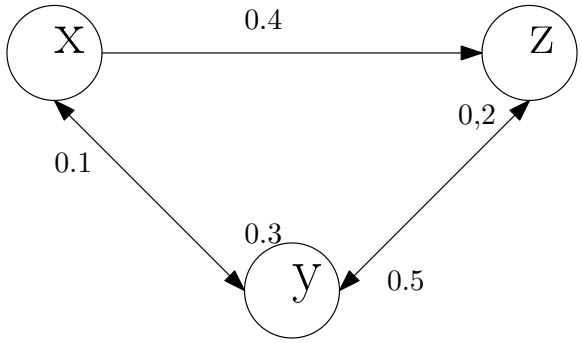


Figure 2.5: A weighted, directed graph G = (V,E) derived from a social network. Source: Figure 2 on Page 213, Goyal et al. [2011].

Theorem 2.1.1. *In the LT model, the spread of a set S is the sum of the spread of each node $u \in S$ on subgraphs induced by $V - S + u$. That is,*

$$\sigma(S) = \textstyle\sum_{u \in S} \sigma^{V-S+u}(u)$$

(Source: Theorem 1 on pages 213, Goyal et al. [2011].)

Example 2.1.5. ***The Influence Spread of a Seed Set*** S ***using SIMPATH.*** *In Figure 2.5, the influence of a node x on node z can be computed by enumerating all simple paths starting from x and ending in z.*

$$\Upsilon_{x,x} = 1$$

$$\Upsilon_{x,y} = 0.3 + 0.4 \cdot 0.5 = 0.5$$

$$\Upsilon_{x,z} = 0.4 + 0.3 \cdot 0.2 = 0.46$$

Thus, the spread of a node can be computed by enumerating simple paths starting from the node.

$$\sigma_{\{x\}} = \Upsilon_{x,x} + \Upsilon_{x,y} + \Upsilon_{x,z} = 1 + 0.5 + 0.46 = 1.96$$

The spread of a seed set $S = \{x, y\}$, accroding to therom 2.1.1, is

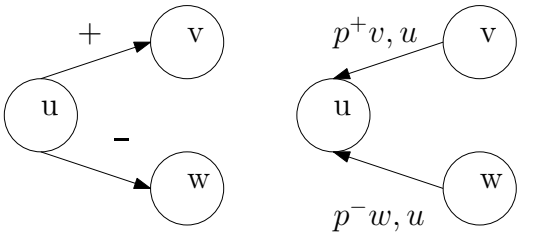
$$\sigma(S) = \sigma^{V-y}(x) + \sigma^{V-x}(y) = 1 + 0.4 + 1 + 0.2 = 2.6$$

2.1.4 Discovering Influential Nodes from Social Trust Network

In [Ahmed and Ezeife 2013], the authors state that existing influence diffusion models such as the Linear Threshold model and the Independent Cascade model [Kempe et al. 2003] consider only positive influence propagation in a social network. However, two opposite relationships (such as like vs. dislike, love vs. hate, trust vs. distrust, friend vs. foe, and so on) may coexist in a social network. For example, users on Wikipedia can vote for or against the nomination of others to be Wikipedia administrator, users on Epinions can express trust or distrust of other people's product reviews by rating, and participants on Slashdot can declare others to be either "friends" or "foes", users on Youtube can express like or dislike of other people's comments. The authors claim that we need to consider both positive influence exerted by people we trust or like and negative influence exerted by people we do not trust or dislike while studying influence diffusion process. Existing diffusion models for Influence Maximization are modeled such that a node's probability of performing an action (or adopting a product) will increase as the number of his/her friends performing the same action increases. However, the authors argue that, a node's probability of performing an action (e.g., buy an iPhobe 4S) should also decrease if its distrusted users, also buy an iPhone 4S.

The authors propose a new diffusion model named Trust-General Threshold (TGT) model which incorporates both positive and negative influence probabilities based on trust relationship among users in trust network. In a trust social network (Figure 2.6 (a)), a node u trusts node v but distrusts node w. In the corresponding influence graph (Figure 2.6 (b)), if node u trusts node v, then node v positively influences node u with the probability of p^+v, u with $p^-v, u = 0$. If node u distrusts node w, then

node w negatively influences node u with the probability of p^+w, u with $p^+w, u = 0$.



(a) Trust Graph (b) Influence Graph

Figure 2.6: Trust Graph vs Influence Graph

The authors define the positive influence probability $p^+v, u = \frac{A_{v,u}}{A_v}$ where A_v denotes the number of actions performed by node v and $A_{v,u}$ denotes the number of actions propagated from node v to node u (i.e., the number of v's actions imitated by node u). For example, the action log shows that node v (trusted by node u, in Figure 2.6 (a)) performs 3 actions in total. Among v's 3 actions, 2 actions are imitated by u. Hence, the probability of node u performing a task after node v performs the same action is $2/3 = 0.66$, which is the positive influence probability of node v on node u. Then the authors define the negative influence probability $p^-v, u = \frac{A'_{v,u}}{A_v}$ where A_v denotes the number of actions performed by node v and $A'_{v,u}$ denotes the number of actions not propagated from node v to node u (i.e., the number of v's actions not imitated by node u). For example, the action log shows node w (distrusted by node u, in Figure 2.6 (a)) performs 4 actions in total. Among w's 4 actions, only 1 action is imitated by node u, the remaining 3 actions are not imitated by node u. That is u does not perform 3 out of 4 tasks performed by w. Hence, the probability of node u not performing a task after node w performs the same action is $3/4 = 0.75$, which is the negative influence probability of node w on node u.

The authors propose an effective algorithm named MineSeedLS to discover influential nodes from trust network. T-IM takes a social network graph $G(V, E)$ and a budget k meaning to find at most k influential nodes. The algorithm returns a set of influential nodes of size at most k, also known as seed set, $S \subseteq V$. The algorithm

starts by initializing seed set S to $\emptyset$. Then the algorithm computes influence spread of each node $v \in V$. The node with highest influence spread is picked and added to S. MineSeedLS then performs the following local search operations: (1) Delete, if by removing any node v in S increases the influence spread under the T-IM model, then the node v is removed from S. (2) Add, if by adding any node v in $V - S$ increases the influence spread under tje T-IM model, then the node v is added to the set S. (3) Swap, if by swapping any node v in S with any node u in $V - S$ increases the spread under T-IM model the node v is removed from S and node u is added to S.

Example 2.1.6. ***How MineSeedLS Works.*** *We illustrate how MineSeedLS works through an example. Given a social network $G = (V, E)$ in Figure 2.7 (where each edge is assigned either positive influence probability or negative probability and for the purpose of demonstration, for each node, the positive threshold is set to 0.3 and the negative threshold is set to 0.6), and a budget $k = 2$ meaning we will discover two influential nodes. MineSeedLS will compute the influence spread for each node. The influence spread of each node is summarized in Table 2.5. The algorithm picks the node with maximum spread which is node $u1$ yielding an influence spread of 3, and adds $u1$ to the seed set S. Once we have selected one node in the seed set, MineSeedLS performs the following local search operations, delete, add and swap on the graph. Since at this moment there is only one node in the seed set S, the delete operation is skipped. Since the budget is $2 > |S| = 1$, the algorithm performs the add operation, i.e., it adds any node in $V - S$, say $u2$ to S, and computes the influence spread of $S + \{u2\}$, denoted as $\sigma_{TGT}(S + \{u2\})$. Since $\sigma_{TGT}(S + \{u2\}) = 4 > \sigma_{TGT}(S) = 3$ which is an improvement, node $u2$ is added to S. At this moment, the seed set $S = \{u1, u2\}$ with the influence spread of 4. MineSeedLS continues to check if swapping (or exchanging) any node in S and any node in $V - S$ yields any improvement in influence spread. It swaps node $u2$ and node $u3$ by removing $u2$ from and adding $u3$ to the seed set. Since $\sigma_{TGT}(S - \{u2\} + \{u3\}) = 5 > \sigma_{TGT}(S) = 4$*

which is an improvement, node $u2$ *is removed from and node* $u3$ *is added to the seed set* S*. At this moment, the seed set* $S = \{u1, u3\}$ *with the influence spread of* 5*. The algorithm will repeat the delete-add-swap procedure for any further improvement. It checks if removing any node from the seed set* S *improves the influence spread or not. It removes node* $u1$ *from* S*. Since* $\sigma_{TGT}(S - \{u1\}) = 3 < \sigma_{TGT}(S) = 5$ *which is not an improvement, it adds node* $u1$ *back to* S*. It then tries to remove* $u3$ *from* S*. Since* $\sigma_{TGT}(S - \{u3\}) = 2 < \sigma_{TGT}(S) = 5$ *which is not an improvement, it adds node* $u3$ *back to* S*. Since the budget is* $2 = |S|$*, the add operation is skipped. It will further check if swapping any node in* S *with any node in* $V - S$ *yields any improvement in spread. Since no swapping yields any improvement, the algorithm stops at this point and returns the seed set* $S = \{u1, u3\}$ *with the influence spread of* 5 *(This is a summary from [Ahmed and Ezeife 2013] on pages 126).*

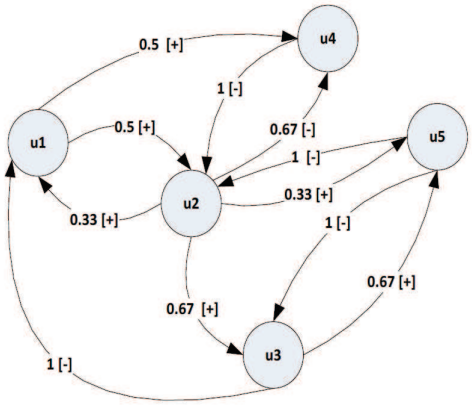


Figure 2.7: Social network graph where each edge is labeled with positive or negative influence probabilities. Source: Figure 2 on page 126 of [Ahmed and Ezeife 2013].

Node v	$u1$	$u2$	$u3$	$u4$	$u5$
$\sigma_{TGT}(\{v\})$	3	2	2	1	1

Table 2.5: Influence spread of each node. Source: Table 6 on page 126 of [Ahmed and Ezeife 2013].

2.1.5 Social Network Opinion and Posts Mining for Community Preference Discovery

In [Mumu and Ezeife 2014], the authors state that the existing influence maximization techniques such as CELF [Leskovec et al. 2007], take as input the whole social network in order to find influential nodes as seed set for a specific product (e.g., iPhone) for viral marketing. According to the authors, general influence maximization techniques like CELF do not consider multiple posts on multiple products on Facebook. Also they ignore the relationships between users. Hence the seed set found by CELF-like approaches may not be influential for that specific product (e.g., iPhone). Hence, the quality of the seed set will be reduced and the efficiency of the algorithm is slow since the search space is the entire network.

Motivated by the limitation, the authors propose a model named OBIN, which takes as input a social network graph $G = (V, E)$ and a product z and outputs an influence graph $G_z(V, E)$ for product z from computed community preference where V is a sub-graph of the entire social network G containing only the relevant nodes to the query. OBIN model consists of three main functions, TPD (Topic-Post Distribution), PCP-Miner (Post-Comment Polarity Miner), and influence network generator. (1) The first function named TPD first applies SQL queries to find all nodes, posts, and comments in the social network (i.e., Facebook) for a given product z, then separates relevant nodes from irrelevant nodes in the resulting datasets. TPD determines the relevance of a node u' on a product (e.g., iPhone) by the number of nodes linked to node u, the number of likes on u's posts, the number of shares and comments on u's posts, and if the posts of u contains the product information (e.g., iPhone screen resolution). (2) The second function named PCP-Miner identifies opinion comments among all the comments on u's posts, identifies sentiment (positive, neutral, negative, or irrelevant attitude) toward the comments, and measures the polarity score (θ_z) of the posts. The algorithm then uses the polarity score to rank relevant nodes v, and

generate a table including v's posts w, comments c on posts w, and the set of nodes who post comments on the posts w of v (which are considered influenced nodes by v.) (3) PoPGen (popularity graph generator) uses the list of ranked relevant nodes, along with their posts, the comments on their posts, the authors of the comments to compute the influence score, i.e., the extend to which the relevant nodes exert on the influenced nodes who comment their posts. PoPGen measures the influence by the number of responses. Then PoPGen generates an influence graph $G_z(V, E)$ on product z where nodes are those relevant nodes and there is an edge between two nodes if they are friends on Facebook.

Example 2.1.7. ***How OBIN Works.*** *We illustrate how OBIN model works through an example. OBIN first calls TPD to extract relevant nodes on a product z from Facebook network. It is done by executing SQL query*

```
SELECT id, name, category, likes, link
FROM search
WHERE q=iphone AND (type=page OR type = group)
```

and generating a nodes matrix as shown in Table 2.6. The first row of Table 2.6 shows that a node id is "140389060322069", the product is "iPhone", the node has $3,116,728$ *friends and the profile of the node can be viewed via the "iphone.page" link. Once having obtained relevant nodes on a product z, TPD executes SQL query*

```
SELECT post_id, message, likes.count AS A, share_count,
       created_time, comments.count, (comments.count+share_count) AS SR
FROM stream
WHERE source_id = 1 AND message != " "
ORDER BY likes.count LIMIT 100
```

in order to generate a set of posts on z of a node, say "140389060322069" as shown in Table 2.7 and Table 2.8. For example, the first row in Table 2.7 shows

that a post id is "469219579782347" posted by node "140389060322069", the post title is "Black or white", there are $61,153$ *likes on the post, and the total number of re-shares and unique comments are* $11,325$. *The first row in Table 2.8 shows that a post id is "469219579782347", a node "108936862354990" leaves a comment on the post at time "2013-01-06", the content of the comment is "this is really cool".*

Node ID V	Term	A	Link
140389060322069	iPhone	3116728	iphone.page
110018862354999	iPhone4	1435239	iPhone-4
214456561919831	iPhone Fans	261210	theappleclan

Table 2.6: Example of relevant nodes and data for z = iPhone. Source: Table 1 on page 141 of [Mumu and Ezeife 2014].

POST ID W	Term	A	SR
469219579782347	black or white	61153	11325
468646856506286	pretty amazing	33899	2213
469758623061776	Apple 5th Avenues	33041	2198

Table 2.7: Example of post data. Source: Table 2 on page 141 of [Mumu and Ezeife 2014].

POST ID W	User ID V	Time	Comment C
469219579782347	108936862354990	2013-01-06	this is really cool

Table 2.8: Example of post data. Source: Table 3 on page 141 of [Mumu and Ezeife 2014].

To determine how influential a node v *is on a certain product* z, *OBIN calls PCP-Miner to compute the polarity score* θ_z *for each post of node* v. *For example, Table 2.9 is the popularity matirix for post W "469219579782347". PCP-Miner computes*

the polarity score θ_z for post W "469219579782347" as follows:

$$\begin{aligned}\theta_z &= (\sum posrepsonses - \sum negresponses) \times 100\% \\ &= 5 - 0 \\ &= 5\end{aligned}$$

The polarity score θ_z is used to obtain a list of relevant nodes V, their posts W, comments C on posts, and the nodes who leave comments on the posts W and are therefore considered "influenced by the author of post" as shown in Table 2.10. OBIN uses post-user relationship (Table 2.10) and user-user relationship Table (2.11) to generate an influence matrix (Table 2.12) such that the element of the influence matrix is 1 if there exists a relationship in either the post-user relationship or the user-user relationship, 0 otherwise. OBIN calls PoPGen to generate an influence graph based on influence matrix (IMAT) by adding all nodes in the IMAT to the influence graph, and adding an edge between u and v if the the element $IMAT_{u,v} = 1$.

POST ID W	User ID V	Polarity	Time	Comment C
469219579782347	108936862354990	positive	2013-01-06	this is really cool
469219579782347	100002395810151	positive	2013-01-06	i want it
469219579782347	100003290108936	positive	2013-01-06	cool
469219579782347	100004582655605	null	2013-01-06	hi sakuntla
469219579782347	1850908608	positive	2013-01-06	wow
469219579782347	100002090841333	positive	2013-01-06	crazy aoubt it
469219579782347	100003365201901	null	2013-01-06	admin

Table 2.9: Example of post data. Source: Table 3 on page 141 of [Mumu and Ezeife 2014].

Node ID u	Post ID W	Node ID v
1	49823667	4
2	11250901	6

Table 2.10: Post-user relationship. Source: Table 6 on page 143 of [Mumu and Ezeife 2014].

User ID u	User ID
3	1

Table 2.11: User-user relationship. Source: Table 6 on page 143 of [Mumu and Ezeife 2014].

	1	2	3	4	5	6	7
1	0	0	1	1	0	0	0
2	0	0	0	0	0	1	0

Table 2.12: Influence Matrix (IMAT). Source: Table 7 on page 143 of [Mumu and Ezeife 2014].

2.2 Outbreak Detection

2.2.1 Identifying the Influential Bloggers in a Community

In [Agarwal et al. 2008], the authors first consider the definition of an influential blogger as follows:

Definition 2.2.1. ***Influential Blogger.*** *A blogger is considered influential if s/he has more than one influential blog post.*

Then the authors present the definition of an influential blog post as follows:

Definition 2.2.2. ***Influential Blog Post.*** *A blog post p_i is considered influential if its influence score $I(p_i)$ is greater than an influence threshold $iIndex(b_{jk})$, where the influence threshold is defined as: Given a set of U of m bloggers,*

More precisely, let $\{b_k|1 \leq k \leq m\}$ or $\{b_1, b_2, ..., b_m\}$ denote a universe set U of m bloggers, let $\{p_i|1 \leq i \leq l\}$ or $\{p_1, p_2, ..., p_l\}$ denote a set L of all the blog posts by all m bloggers, let $\{p_j|1 \leq j \leq n\}$ or $\{p_1, p_2, ..., p_n\}$ denote a set N of n blog posts by a blogger b_k. For each post $p_j \in Nwhere 1 \leq j \leq n$ by a blogger b_k, there is an influence score $I(p_j)$ associated with it. Let $max(I(p_i)) = max_{1 \leq j \leq n}(I(p_j))$ denote the maximum influence score among blogger b_k's blog posts 1 through n, let $iIndex(b_k)$ denote the influence index of blogger b_k, then $iIndex(b_k) = max(I(p_i))$.

That is, the influence of a blogger is identified by the influence of their blogs. Let V denote the set of top-k bloggers according to their influence index $iIndex$, let $min(iIndex(b_i)) = min_{1 \leq i \leq k}(iIndex(b_i))$ denote the minimum influence index among k-influential bloggers 1 through k, then $min(iIndex(b_i))$ is defined as the threshold of influential blog posts. That is, for all the blog posts $\{p_1, p_2, ..., p_l\}$ by all m bloggers, blog posts are considered influential if their influence score $I(p_j) \geq min(iIndex(b_i))$ for $1 \leq j \leq l, 1 \leq i \leq k$. Bloggers are considered influential if they posted more than one influential blog post.

According to the authors, a blog post is considered influential if (a) it is known by many people, which is measured using the number of its inlinks ι, (b) it generates follow-up activities, which is quantified by the number of comments it receives γ, (c) the ideas in the blog post are original, which is indicated by the number of its outlinks θ, (d) the content of the blog post has quality, which is measured by the length of the blog post λ. To quantify the influence of a blog post p $I(p)$, the authors exploit the four parameters jointly as follows.

$$InfluenceFlow(p) = w_{in} \sum_{m=1}^{|\iota|} I(p_m) - w_{out} \sum_{n=1}^{|\theta|} I(p_n) \tag{2.5}$$

where w_{in} and w_{out} are the weights that can be used to change the ratio of incoming and outgoing influence in the model, respectively. p_m denotes all the blog posts that refer to blog post p, for $1 \leq m \leq |\iota|$. p_n denotes all the blog posts that blog post p refers to, for $1 \leq n \leq |\theta|$. Recall that $|\iota|$ is the total numbers of inlinks of blog post p, $|\theta|$ is the total numbers of outlinks of blog post p. $InfluenceFlow(p)$ measures the recognition and the novelty simultaneously since (1) the more influential inlinks p has, the more influential p is, (2) the more influential outlinks p has, the less novel p is.

$$I(p) \propto w_{com}\gamma_p + InfluenceFlow(p) \tag{2.6}$$

where w_{com} is the weight can be exploited to change the ratio of the number of comments in the model, γ_p denotes the number of comments received by blog post p. $I(p)$ is proportional to the joint contribution by the number of comments it receives and $InfluenceFlow(p)$ since (1) the more influential comments p receives, the more influential p is, (2) the larger $InfluenceFlow(p)$ is, the more influential p is.

$$I(p) = w(\lambda) \times (w_{com}\gamma_p + InfluenceFlow(p)) \tag{2.7}$$

where $w(\lambda)$ is a weight function to measure the quality of the blog post p according to its length λ.

$$iIndex(B) = max(I(p_i)) \tag{2.8}$$

where $iIndex(B)$ is the influence index of blogger B, $max(I(p_i))$ for $1 \leq i \leq n$ is the maximum influence score among blogger B's blog posts 1 through n. That is, the influence of a blogger B is measured by their blog posts. We can sort bloggers in descending order according to their influence index, then choose top k bloggers as k most influential bloggers.

2.3 Probabilistic Models of Information Flow

2.3.1 Learning Influence Probabilities in Social Networks

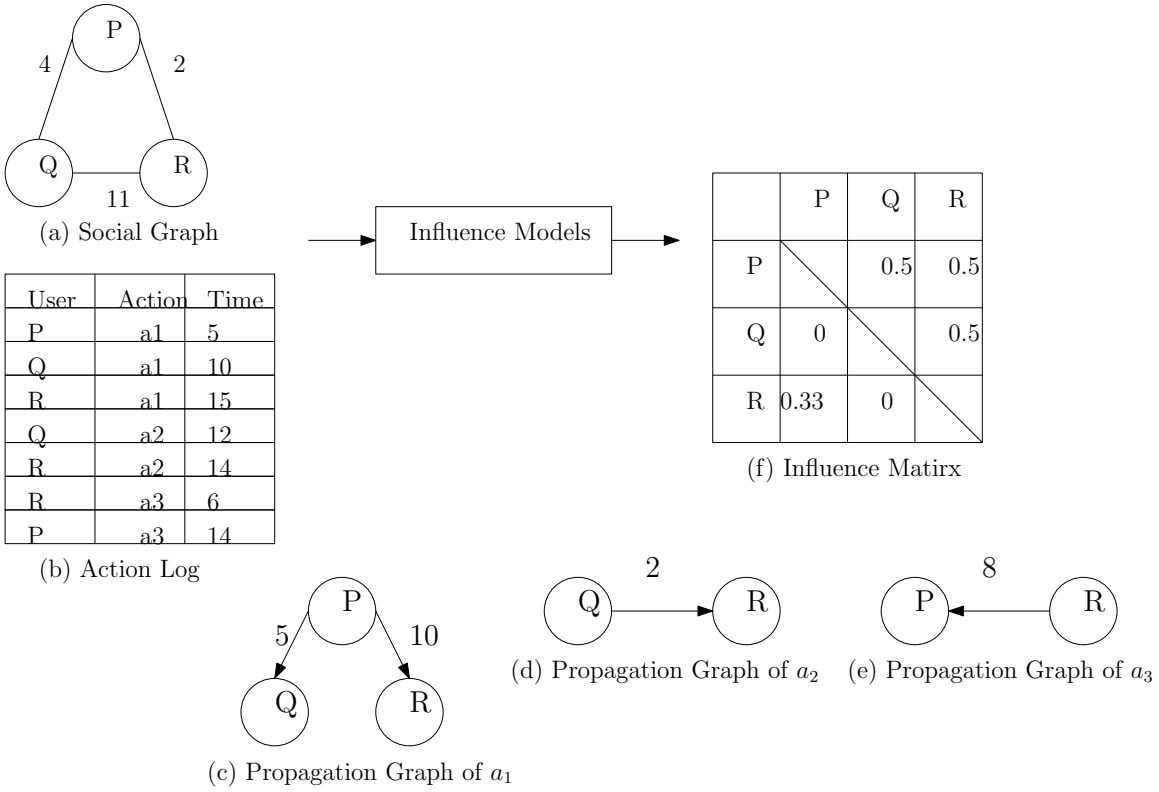
In [Goyal et al. 2010], the authors state that real social networks do not have edge weights indicating the influence probability $p_{v,u}$ with which v influences u. Therefore, most of the researchers in this area assume the edge weights indicating the influence probabilities are given as input. In their experiments, researchers adopt primarily four models of assigning pairwise influence probabilities, i.e., the uniform model, the trivalency model, the random cascade model, and the weighted cascade model (section 1.9). Goyal et al. [2010] point out that although the real social network do not have

the pairwise influence probability $p_{v,u}$ explicitly as the edge weight on $(v,u) \in E$, the probabilities can be derived from social network data, i.e., user action logs. The problem of learning probabilities in social networks is defined formally as follows:

Definition 2.3.1. ***Problem Definition*** *Given a graph $G = (V, E, T)$ derived from a social network where $v \in V$ represents a user, an undirected edge $(u, v) \in E$ represents a social tie between user u and user v, $T : E \rightarrow N$ is a function mapping an edge to a timestamp at which the social tie is created, along with an action log $Actions(User, Action, Time)$, which is a relation containing tuples in the form of (u, a, t_u) indicating user $u \in V$ performs action $a \in A$ (where A denotes the universe of actions) at time t_u, we want to learn a function $p : E \rightarrow [0,1] \times [0,1]$ such that each edge $(v,u) \in E$ is mapped to two influence probabilities $p_{v,u}$ (indicating the probability with which v influences u) and $p_{u,v}$ (indicating the probability with which u influences v).*

Input. The input of the algorithms includes an undirected social graph, an action log, and an influence model. The social graph consists of nodes representing individuals, edges indicating social ties between these individuals, and edge weights indicating when the social tie was created. For example, in the social graph shown in Figure 2.8 (a), there are 3 individuals, P, Q, and R, P and Q become friends at time 4, P and R become friends at time 2, Q and R become friends at time 11. The action log consists of tuples in the form of $(user, action, time)$ indicating user u performs action a at time t, and sorted by action and then by time in increasing order. For example, in the action log shown in Figure 2.8 (b), there are 7 tuples, indicating P performs action a1 at time 5, Q performs action a1 at time 10, and so on. The influence model includes static models, continuous time models, and discrete time models.

Action Propagation. We say an action a propagates from v to u if the social tie between u and v was created before both u and v perform action a, and v performs action a before u performs action a. For example, in Figure 2.8 (a), Q and R become



(a) Social Graph

(b) Action Log

(c) Propagation Graph of a_1

(d) Propagation Graph of a_2

(e) Propagation Graph of a_3

(f) Influence Matirx

Figure 2.8: A framework proposed by Goyal et al. for learning influence probabilities for all edges. Source: Figure 2 on Page 6, Goyal et al. [2010].

friends at time 11, Q performs action a2 at time 12, and R performs action a2 at time 14, therefore we say action a2 propagates from Q to R. On the other hand, Q and R become friends at time 11, Q performs action a1 at 10, and R performs action a1 at 15, however Q performs action a1 at 10 which is earlier than Q and R become friends, hence we say action a1 does not propagate from Q to R.

Propagation Graph. For each action $a \in A$, we have a propagation graph for it. A propagation graph for an action a is a weighted, directed graph $G(V, E)$, where a node $v \in V$ represents a user, a directed edge $(v, u) \in E$ from v to u indicating the propagation of the action a from v to u, the edge weight represents the time delay between v performing the action a and u performing the same action a. If we denote the time that u performs action a as $t_u(a)$, then the time delay on the edge is denoted as $t_u(a) - t_v(a)$. For example, Figure 2.8 (c) is the propagation graph for action a_1, the edge (P, Q) says P propagates a_1 to Q. According to the action log (shown in Figure 2.8 (b)), $t_P(a_1) = 5$, $t_Q(a_1) = 10$, hence the time delay on the edge (P, Q) is $t_Q(a_1) - t_P(a_1) = 5$.

Output. The output is an influence matrix M (shown in Figure 2.8 (f)) where $M[v, u] = p_{v,u}$, which is the pairwise influence probability of v on u. That is, we have learned $p_{v,u}$ for all edges.

The authors first introduce their solution framework which is an instance of the General Threshold Model. Recall from section 1.5, the General Threshold Model represents a social network as a weighted, directed graph $G = (V, E)$. Each node $v \in V$ is associated with a threshold function $f_v(S)$, where S is the set of v's active neighbors. $f_v(S)$ measures the joint influence probability of v's active neighbors S exerted on v, with $f_v(\emptyset) = 0$. Each node $v \in V$ chooses uniformly at random a threshold θ_v over the interval [0,1]. The diffusion process happens in discrete steps, i.e., $t = 0, 1, 2, ..., n - 1$. At any time t, each node $v \in V$ is either active or inactive. Once v is activated, it remains active and cannot switch back to inactive. At time

0, there is an initial set S_0 that adopts a new behavior. At time $t > 0$, all nodes that were active at time $t-1$ remain active, furthermore, among all the inactive nodes, any node v is activated if $f_v(S) \geq \theta_v$. The process will stop when no more activations are possible. Goyal et al. [2010] assume that the influence that each of the active neighbors of an inactive node u exerts on u is independent of each other and define the threshold function (also known as the joint influence probability of u's active neighbors exerted on u) as follows,

$$p_u(S) = 1 - \prod_{v \in S}(1 - p_{(v,u)}) \tag{2.9}$$

where u is an inactive user, S is the set of its activated neighbors, $p_u(S)$ is the joint influence probability of S exerted on u (also known as the threshold function of u), and $p_{v,u}$ is the pairwise influence probability of $v \in S$ exerted on u. If $p_u(S) \geq \theta_u$, where θ_u is the activation threshold of user u, then u is activated.

The authors then show how to estimate the pairwise influence probability $p_{(v,u)}$ in equation 2.9 in static models, continuous time models, and discrete time models respectively. We will introduce the static models on which our proposed algorithm computeInfluenceProb based (Algorithm 3 in Chapter 3). Continuous time models, and discrete time models are omitted for lack of space.

Static Model. Static models assume that the influence probabilities are static and do not change as time goes on. Three instances of static models are presented: Bernoulli distribution, Jaccard index, and partial credits.

Static Model - Bernoulli Distribution. Bernoulli distribution estimates the influence probability of v on u, $p_{v,u}$ using Maximum-Likehood Estimator (MLE) as follows:

$$p_{v,u} = \frac{A_{v2u}}{A_v} \tag{2.10}$$

where A_{v2u} denotes the number of actions propagated from v to u, A_v denotes the

number of actions performed by v.

Example 2.3.1. *In Figure 2.8, the pairwise influence probability $p_{P,Q}$ under static model estimated by Bernoulli distribution is*

$$\begin{aligned} p_{P,Q} &= \frac{A_{P2Q}}{A_P} \\ &= \frac{1}{2} \\ &= 0.5 \end{aligned}$$

$A_{P2Q} = 1$ because according to the propagation graphs (shown in Figure 2.8 (c), (d), and (e)), there is only 1 action (i.e., a_1) propagated from P to Q. $A_P = 2$ because P performs 2 actions a_1 and a_3.

Static Model - Jaccard Index. Jaccard index estimates the influence probability of v on u $p_{v,u}$ by adopting Jaccard similarity (The Jaccard similarity of two sets S and T is defined as $|S \cap T|/|S \cup T|$, i.e., the ratio of the cardinality of the intersection of S and T to the cardinality of the union of S and T [Leskovec et al. 2011]) as follows:

$$p_{v,u} = \frac{A_{v2u}}{A_{u|v}} \tag{2.11}$$

where A_{v2u} denotes the number of actions propagated from v to u, $A_{u|v}$ denotes the number of actions either performed by u or performed by v.

Example 2.3.2. *In Figure 2.8, the pairwise influence probability $p_{P,Q}$ under static model estimated by Jaccard index is*

$$\begin{aligned} p_{P,Q} &= \frac{A_{P2Q}}{A_{P|Q}} \\ &= \frac{1}{3} \\ &= 0.33 \end{aligned}$$

$A_{P2Q} = 1$ because according to the propagation graphs (shown in Figure 2.8 (c), (d), and (e)), there is only 1 action (i.e., a_1) propagated from P to Q. $A_{P|Q} = 3$ because P performs 2 actions a_1 and a_3, Q performs 3 actions a_1, a_2, and a_3. $\{a_1, a_3\} \cup \{a_1, a_2, a_3\} = \{a_1, a_2, a_3\}$, and $|\{a_1, a_2, a_3\}| = 3$.

Static Model - Partial Credits. Partial credits first estimates the credit given to each activated neighbors $v \in S$ of u who performed an action $a \in A$ before u as follows:

$$credit_{v,u}(a) = \frac{1}{\sum_{w \in S} I(t_w(a) < t_u(a))} \tag{2.12}$$

where $t_u(a)$ denotes the time at which user u performs an action $a \in A$, $t_w(a)$ denotes the time at which user w performs the action $a \in A$, S denotes the set of activated neighbors of u, I is an indicator function returning 1 if an activated neighbor $w \in S$ performs action $a \in A$ before u, returning 0 otherwise. $\sum_{w \in S} I(t_w(a) < t_u(a))$ in equation 2.4 means the number of active neighbors of u who perform the action a before user u. That is, in the partial credits model if u is influenced to adopt an action a, each of u's active neighbors who have performed the action a before u does so is given an equal credit 1/d for the action a, where d is the number of active neighbors of u who perform the action a before user u does so, or d is the number of contributors who propagate the action a to u.

Then the Bernoulli model with partial credit estimates the pairwise influence probability of v on u, $p_{v,u}$ by plugging equation 2.4 into equation 2.2 as follows:

$$p_{v,u} = \frac{\sum_{a \in A} credit_{v,u}(a)}{A_v} \tag{2.13}$$

where $\sum_{a \in A} credit_{v,u}(a)$ is the total credits given to v for propagating actions to u, and A_v denotes the number of actions performed by v.

Example 2.3.3. *In Figure 2.8, the pairwise influence probability $p_{P,Q}$ under static*

model estimated by partial credit Bernoulli distribution is

$$\begin{aligned} p_{P,Q} &= \frac{\sum_{a \in A} credit_{P,Q}(a)}{A_P} \\ &= \frac{credit_{P,Q}(a_1)}{A_P} \\ &= \frac{\frac{1}{1}}{2} \\ &= 0.5 \end{aligned}$$

$\sum_{a \in A} credit_{P,Q}(a) = 1$ *because according to the propagation graphs (shown in Figure 2.8 (c), (d), and (e)), there is only* 1 *action (i.e.,* a_1*) propagated from* P *to* Q *and* P *is the only contributor propagating action* a_1 *to* Q*, hence user* P *gets the full credit for influencing user* Q *for performing action* a_1*.* $A_P = 2$ *because* P *performs 2 actions* a_1 *and* a_3*.*

And Jaccard index model with partial credit estimates the pairwise influence probability of v on u, $p_{v,u}$ by plugging equation 2.4 into equation 2.3 as follows:

$$p_{v,u} = \frac{\sum_{a \in A} credit_{v,u}(a)}{A_{u|v}} \tag{2.14}$$

Example 2.3.4. *In Figure 2.8, the pairwise influence probability* $p_{P,Q}$ *under static model estimated by partial credit Jaccard index is*

$$\begin{aligned} p_{P,Q} &= \frac{\sum_{a \in A} credit_{P,Q}(a)}{A_{P|Q}} \\ &= \frac{credit_{P,Q}(a_1)}{A_{P|Q}} \\ &= \frac{\frac{1}{1}}{3} \\ &= \frac{1}{3} \\ &= 0.33 \end{aligned}$$

$\sum_{a \in A} credit_{P,Q}(a) = 1$ *because there is* 1 *action* (a_1) *propagated from* P *to* Q *and* P *is the only contributor propagating action* a_1 *to* Q, *user* P *gets the full credit for influencing user* Q *for performing action* a_1. $A_{P|Q} = 3$ *because* P *performs 2 actions* a_1 *and* a_3, Q *performs 3 actions* a_1, a_2, *and* a_3. $\{a_1, a_3\} \cup \{a_1, a_2, a_3\} = \{a_1, a_2, a_3\}$, *and* $|\{a_1, a_2, a_3\}| = 3$.

Chapter 3

Proposed Algorithm for Mining Influential Nodes From Competitive Social Networks

The setting of the thesis problem is the launch of technology A into a market where a competing technology B already exists along with a set of early adopters of technology B. The problem we tackle is to find k most influential nodes and convince them to adopt Technology A (e.g., giving each a free sample of Technology A) such that the final adoptions of Technology A in the crowd is maximized in the setting. Here, k is our budget for the advertising campaign meaning we have at most k free samples to distribute. If we represent the underlying social network (the medium for the propagations of two technologies) as $G = (V, E)$, where V represents individuals, E represents interactions between them, then there are two aspects related to the thesis problem. The first aspect of our problem is to study how the dynamics of adoptions of Technology A and Technology B simultaneously spread out through the network, i.e., we need a diffusion model to describe the two simultaneous influence diffusions and their resulting cascading behaviors (section 3.1), including the task of learning

the pairwise influence probabilities as the edge weights (section 3.2.2). The second aspect of our problem is to study an efficient yet effective algorithm which allows us to find the special k nodes for Technology A under the proposed diffusion model (section 3.2.4). We include analysis of the running times of all our algorithms in section 3.3.

3.1 Competing General Threshold Model

In this section, we will address the first aspect of the thesis problem, i.e., the proposed Competing General Threshold model which is an extension to the General Threshold model [Kempe et al. 2003]. Unlike the original General Threshold model which models one single influence diffusion in the network, the proposed Competing General Threshold (CGT) model is aiming to model two interfering influence diffusions in the network.

But before we do that, we will briefly review some terminology used in existing influence maximization research. In the next section, we will extend the definitions of them to our thesis problem setting. In the following definitions, the underlying social network is represented by $G = (V, E)$, where V represents individuals, E represents interactions between them, and $|V| = n$ (i.e., the cardinality of V is n).

Definition 3.1.1. ***Pairwise Influence Probability****, denoted as $p_{v,u}$, is the weight on edge $(v, u) \in E$ indicating the extent to which node v influences node u. That is, if v is active, it succeeds in activating u with the probability of $p_{v,u}$.*

Definition 3.1.2. ***Threshold Function****, also known as joint influence probability or activation function, defined as $f_v : 2^V \rightarrow [0, 1]$, where 2^V denotes the power set of V. Under the threshold model, each node $v \in V$ is associated with a threshold function $f_v(\cdot)$, $f_v(S)$ measures the joint influence of v's active neighbors S exerted on v, with $f_v(\emptyset) = 0$.*

Definition 3.1.3. ***Threshold****, or activation threshold, denoted as θ_v, is chosen uniformly at random over the interval [0,1] for each node $v \in V$ under the threshold diffusion model. Here, "uniformly" means the probability of choosing any point over [0,1] is the same, or each point is being equally likely to be chosen. Intuitively, it indicates enough (or the minimum number) of its neighbors who have already adopted a behavior in order for v to do so. θ_v being chosen uniformly at random for each $v \in V$ is intended to model our lack of knowledge of the exact values [Kempe et al. 2003].*

Competing Influence Diffusions. In this thesis, we consider the setting in which there are two competing technologies, e.g., Apple iPhone (A) vs. Blackberry (B) coexisting in the network. When there are two competing technologies, A and B coexisting in the network, there are two seed sets, the seed set that adopts innovation A, i.e., the early adopters of innovation A (denoted as S_0^A), and the seed set that adopts innovation B, i.e., the early adopters of innovation B (denoted as S_0^B). Competing influence diffusions refer to a scenario where the adoptions of two innovations propagate simultaneously throughout the network from each seed set to the crowd such that one diffusion (the propagation of one technology from its seed set to the crowd) interposes in a way that hinders or impedes the other diffusion (the propagation of the other technology from its seed set to the crowd).

Competing Influence Diffusions Model is the model used to describe the competing influence diffusions. In this thesis, we extend the existing General Threshold model which deals with a single influence diffusion (the propagation of a single technology) in the network (section 1.5) to the Competing General Threshold model which deals with two competing influence diffusions (two technologies propagating and competing with each other).

The Social Network under the CGT Model. The CGT model represents a social network as a weighted, directed graph $G = (V, E)$. Each node $u \in V$ is associated with two threshold functions $f_u^A(\cdot)$ and $f_u^B(\cdot)$. Let N^A denote u's active neighbors who adopt technology A, then $f_u^A(N^A)$ measures the joint A-influence of u's active neighbors who adopt technology A exerted on u, with $f_u^A(\emptyset) = 0$. Let N^B denote u's active neighbors who adopt technology B, then $f_u^B(N^B)$ measures the joint B-influence of u's active neighbors who adopt technology B exerted on u, with $f_u^B(\emptyset) = 0$. Each node $u \in V$ chooses uniformly at random over the interval [0,1] two thresholds, θ_u^A (indicates the minimum number of its A-neighbors who have already adopted technology A in order for u to do so) and θ_u^B (indicates the minimum number of its B-neighbors who have already adopted technology B in order for u to do so). That each node $u \in V$ chooses uniformly at random over the interval [0,1] two thresholds, is the random aspect of the CGT model.

The Influence Diffusions under the CGT Model. The influence diffusions happen in discrete steps, i.e., $t = 0, 1, 2, ..., n-1$. At any time t, each node $v \in V$ has one of the four states, A indicating adopting technology A or A-active, B indicating adopting technology B or B-active, AB indicating adopting technology A and technology B simultaneously or AB-active, and 0 indicating adopting neither of them or inactive. (We call A-active nodes A-nodes, B-active nodes B-nodes, and AB-active nodes AB-nodes in the rest of this thesis.) Once a node becomes active (A-active, B-active, or AB-active), it cannot change its state anymore, i.e., it cannot change back to inactive or switch to another active state. This is the competitive aspect of the two influence difussions. This is because once a node v becomes, say A-active, it cannot switch to B, which means it blocks the influence propagation of technology B [Chen et al. 2013]. At time 0, there are two seed sets, S_0^A that adopts technology A and S_0^B that adopts technology B, and $S_0^A \cap S_0^B = \emptyset$. At time $t > 0$,

all nodes that were active at time $t-1$ remain active, for each inactive node u, let N^A denote the set of u's active neighbors who adopt technology A, N^B the set of u's active neighbors who adopt technology B, then the state (whether A, B, AB, or 0) of node u is defined as follows:

Definition 3.1.4. ***The Active State of AB.*** *If $f_u^A(N^A) \geq \theta_u^A$ and $f_u^B(N^B) \geq \theta_u^B$, then u's state becomes AB meaning active in both A and B.*

Definition 3.1.5. ***The Active State of A.*** *If $f_u^A(N^A) \geq \theta_u^A$ and $f_u^B(N^B) < \theta_u^B$, then u's state becomes A meaning active in A but inactive in B.*

Definition 3.1.6. ***The Active State of B.*** *If $f_u^A(N^A) < \theta_u^A$ and $f_u^B(N^B) \geq \theta_u^B$, then u's state becomes B meaning active in B but inactive in A.*

Definition 3.1.7. ***The Active State of Inactive.*** *If $f_u^A(N^A) < \theta_u^A$ and $f_u^B(N^B) < \theta_u^B$, then u's state becomes 0 meaning inactive in both A and B.*

The process will stop before or at time $n-1$ (where n is the number of nodes in V) when no more activations are possible.

We will illustrate how two competing influence diffusions propagate under the Competing General Threshold model through an example. But before we do that, we need to define the pairwise influence probabilities $p_{v,u}^A$ and $p_{v,u}^B$ for each edge $(v,u) \in E$ under the CGT model and the threshold functions $f_u^A(\cdot)$ and $f_u^B(\cdot)$ for each node $u \in V$ under the CGT model respectively below.

Pairwise Influence Probabilities under the CGT Model. In the social network $G=(V,E)$ under the Competing General Threshold (CGT) model, each edge $(v,u) \in E$ is assigned two pairwise influence probabilities, $p_{v,u}^A$ and $p_{v,u}^B$. $p_{v,u}^A$ indicates the extent to which node v influences node u for technology A. That is, if v is A-active or AB-active, it succeeds in activating u to adopt technology A with the probability of $p_{v,u}^A$. $p_{v,u}^B$ indicates the extent to which node v influences node u for technology B. That is, if v is B-active or AB-active, it succeeds in activating u to adopt technology B

with the probability of $p^B_{v,u}$. In this thesis, we assume that a Twitter user v's influence on another Twitter user u holds across different actions, i.e.,

$$p_{v,u} = p^A_{v,u} = p^B_{v,u} \tag{3.1}$$

That is, we assume the influence probability is person-based, not product-based. If we want the influence probability to be product-based, we can assign different weights (which can be learned from past action logs) to $p_{v,u}$ to vary $p^A_{v,u}$ and $p^B_{v,u}$. We learn the pairwise influence probabilities $p_{v,u}$ from Twitter datasets (section 3.2.2) to obtain $p^A_{v,u}$ and $p^B_{v,u}$. Having obtained the pairwise influence probabilities $p^A_{v,u}$ and $p^B_{v,u}$, we compute the joint influence probabilities $f^A_u(\cdot)$ (for u's active A-neighbors to jointly affect u to adopt technology A) and $f^B_u(\cdot)$ (for u's active B-neighbors to jointly affect u to adopt technology B). The joint influence probabilities $f^A_u(\cdot)$ and $f^B_u(\cdot)$ are also known as u's threshold functions (explained next).

Threshold Functions under the CGT Model. In the social network $G = (V, E)$ under the Competing General Threshold (CGT) model, each node $u \in V$ is associated with two threshold functions $f^A_u(\cdot)$ and $f^B_u(\cdot)$. Let N^A denote u's active neighbors who adopt technology A (including those who adopt both A and B), then $f^A_u(N^A)$ measures the joint A-influence of u's active neighbors who adopt technology A exerted on u, with $f^A_u(\emptyset) = 0$. Let N^B denote u's active neighbors who adopt technology B (including those who adopt both B and A), then $f^B_u(N^B)$ measures the joint B-influence of u's active neighbors who adopt technology B exerted on u, with $f^B_u(\emptyset) = 0$. We adopt the threshold function proposed in [Goyal et al. 2010] for the General Threshold model, and define the threshold functions $f^A_u(\cdot)$ and $f^B_u(\cdot)$ under the CGT model as follows:

$$f_u^A(N^A) = 1 - \prod_{v \in N^A} (1 - p_{v,u}^A) \tag{3.2}$$

where u is an inactive node, N^A is the set of its active neighbors for technology A, $f_u^A(N^A)$ is the threshold function that measures the joint A-influence probability of N^A exerted on u, and $p_{v,u}^A$ is the pairwise A-influence probability of $v \in N^A$ exerted on u.

$$f_u^B(N^B) = 1 - \prod_{v \in N^B} (1 - p_{v,u}^B) \tag{3.3}$$

where u is an inactive node, N^B is the set of its active neighbors for technology B, $f_u^B(N^B)$ is the threshold function that measures the joint B-influence probability of N^B exerted on u, and $p_{v,u}^B$ is the pairwise B-influence probability of $v \in N^B$ exerted on u.

Example 3.1.1. ***Threshold Functions Evaluation.*** *Let us illustrate how to evaluate the threshold functions $f_u^A(N^A)$ (equation 3.2) and $f_u^B(N^B)$ (equation 3.3) for node u through an example. In the social network shown in Figure 3.1, there are 5 nodes. Of which, nodes x, y, z, v are active nodes and node u is inactive. The state of node x is AB, the state of node y is A, the state of node z is B, the state of v is B, and the state of u is 0 meaning inactive. Node u has two active neighbors who adopt technology A, i.e., the set of u's active A-neighbors $N^A = \{x, y\}$, and three active neighbors who adopt technology B, i.e., the set of u's active B-neighbors $N^B = \{x, z, v\}$. Also node u chooses uniformly at random two thresholds $\theta_u^A = 0.5$ and $\theta_u^B = 0.8$ over the interval [0,1]. Here, we assume that $p_{v,u} = p_{v,u}^A = p_{v,u}^B$.*

The threshold function $f_u^A(N^A)$ which measures the joint influence probability of N^A

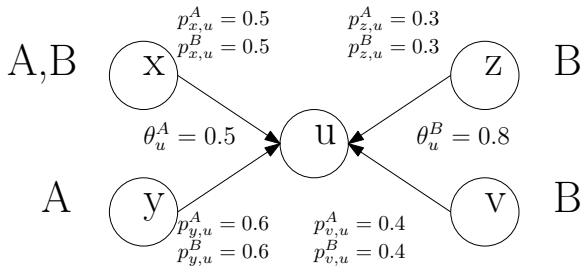


Figure 3.1: An inactive node u in the Competing General Threshold Model where the state of node x is AB, the state of node y is A, the state of node z is B, and the state of v is B.

on u is computed as follows

$$\begin{aligned} f^A_u(N^A) &= f^A_u(\{x, y\}) \\ &= 1 - \prod_{v \in \{x,y\}} (1 - p^A_{v,u}) \\ &= 1 - (1 - p^A_{x,u})(1 - p^A_{y,u}) \\ &= 1 - (1 - 0.5)\dot{(}1 - 0.6) \\ &= 0.8 \end{aligned}$$

The threshold function $f^B_u(B^A)$ which measures the joint influence probability of N^B on u is computed as follows

$$\begin{aligned} f^B_u(N^B) &= f^B_u(\{x, z, v\}) \\ &= 1 - \prod_{v \in \{x,z,v\}} (1 - p^B_{v,u}) \\ &= 1 - (1 - p^B_{x,u})(1 - p^B_{z,u})(1 - p^B_{v,u}) \\ &= 1 - (1 - 0.5)\dot{(}1 - 0.3)\dot{(}1 - 0.4) \\ &= 0.79 \end{aligned}$$

Since $f^A_u(N^A) = 0.8 > \theta^A_u = 0.5$ and $f^B_u(N^B) = 0.79 < \theta^B_u = 0.8$, then u's state becomes A based on Definition 3.1.5.

Having defined the pairwise influence probabilities $p^A_{v,u}$ and $p^B_{v,u}$ for each edge $(v,u) \in E$ under the CGT model and the threshold functions $f^A_u(\cdot)$ and $f^B_u(\cdot)$ for each node $u \in V$ under the CGT model, we now use Figure 3.2 to illustrate how the CGT model works.

Example 3.1.2. ***Two Competing Influence Diffusions under the CGT Model.*** *At time* 0 *(Figure 3.2 (a)), there is a social network* $G = (V, E)$ *(where each node is associated with two thresholds* θ_a *and* θ_b*, each edge is associated with two influence probabilities* p_a *and* p_b*), along with two seed sets, i.e.,* $S^A_0 = \{5\}$ *and* $S^B_0 = \{1\}$*. At time* 1*, node* 1 *activates node* 2 *since* $f^B_2 = 1 - (1 - p^B_{1,2}) = p^B_{1,2} = 0.5 > \theta^B_2 = 0.3$*, node* 5 *activates node* 2 *since* $f^A_2 = 1 - (1 - p^A_{5,2}) = p^A_{5,2} = 0.4 = \theta^A_2 = 0.4$*, the state of node* 2 *becomes AB based on Definition 3.1.4 (Figure 3.2 (b)). At time* 2*, nodes* 2 *and* 5 *jointly activate node* 3 *since* $f^A_3 = 1 - (1 - p^A_{2,3})(1 - p^A_{5,3}) = 1 - (1 - 0.3)(1 - 0.3) = 0.51 > \theta^A_3 = 0.5$*, the state of node* 3 *becomes A (Figure 3.2 (c)) based on Definition 3.1.5. At time* 3*, nodes* 3 *and* 5 *try to jointly activate node* 4*, but* $f^A_4 = 1 - (1 - p^A_{3,4})(1 - p^A_{5,4}) = 1 - (1 - 0.5)(1 - 0.2) = 0.6 < \theta^A_4 = 0.7$*, the state of node* 4 *becomes* 0 *(Figure 3.2 (c)) based on Definition 3.1.7. At this point, the diffusion stops since no more activations are possible.*

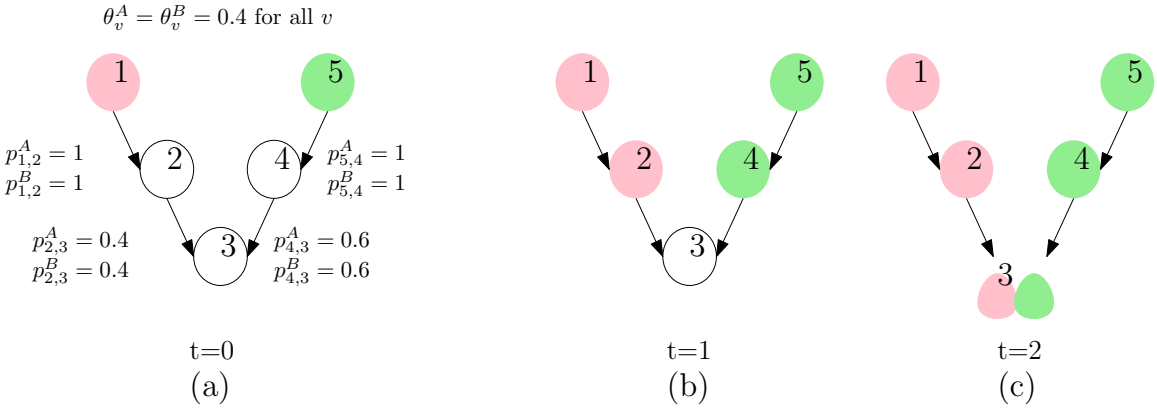


Figure 3.2: Example of Two Competing Influence Diffusions under the CGT Model

Influence Spread under the CGT Model. Let S^A_0 be the seed set for technology A, S^B_0 the seed set for technology B. The influence spread for technology A of two

seed sets S_0^A and S_0^B under the CGT model, denoted as $\sigma^A(S_0^A, S_0^B)$, is defined as the expected number of A-nodes at the end of the diffusion process. The influence spread $\sigma^A(\cdot)$ under the CGT model is monotone and non-submodular with respect to technology A.

Statement 3.1.8. *For an arbitrary instance of the Competing General Threshold model, the resulting influence function $\sigma^A(\cdot)$ is monotone with respect to technology A.*

Statement 3.1.9. *For an arbitrary instance of the Competing General Threshold model, the resulting influence function $\sigma^A(\cdot)$ is non-submodular with respect to technology A.*

We give a counter example [Chen et al. 2013] to show CGT is non-submodular. From Figure 3.3 (a), we can see $\sigma^A(\{1\} \cup \emptyset, \{6\}) - \sigma^A(\emptyset, \{6\}) = 3$. From Figure 3.3 (b), we can see
$\sigma^A(\{1\} \cup \{5\}, \{6\}) - \sigma^A(\{5\}, \{6\}) = 4$, which means the marginal gain of adding node 1 to $\emptyset \cup \{6\}$ (a small context) is smaller than the marginal gain of adding node 1 to $\{5\} \cup \{6\}$ (a large context).

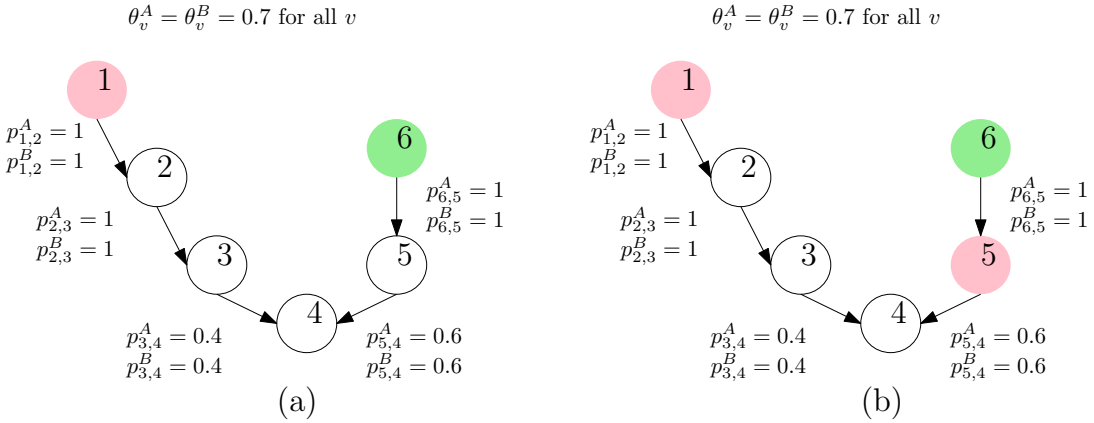


Figure 3.3: Counter example to show CGT is non-submodular

The CGT model is based on the separated-threshold model proposed by Ahmed and Ezeife [2013] where the diffusion process under the trust model is non-monotone

but submodular. It also bears resemblance to the separated-threshold model proposed by the Borodin et al. [2010] where the diffusion process is monotone but not submodular.

Thesis Problem Definition 3.1.10. *Let S_0^A be the seed set for technology A, S_0^B the seed set for technology B. The influence spread for technology A of two seed sets S_0^A and S_0^B under the CGT model, denoted as $\sigma^A(S_0^A, S_0^B)$, is defined as the expected number of A-nodes at the end of the diffusion process.*
Given a directed social network $G = (V, E)$, a non-negative budget k, a seed set of B-nodes S_0^B, and CGT model, the problem of finding influential A-seeds when technology B already exists in the network is to find a seed set S_0^A as early adopters of technology A of size at most k such that $\sigma^A(S_0^A, S_0^B)$ is maximum.

3.2 The Main CIAM System and Algorithm

The solution framework named Competing Influential A-Nodes Miner (CIAM), which is an instance of the General Competing Threshold model, is aiming to find the influential A-nodes from a social network where B-nodes already exist. The input of the overall framework is as follows:

1. Twitter Datasets - consists of 5 Twitter networks as follows,

    1.1 Twitter follow network - contains a list of edges in the form of (u, v) indicating node u follows node v (e.g., Table 3.1).

    1.2 Twitter mention network - contains a list of tuples in the form of (u,v,w) indicating node u mentions node v w times (e.g., Table 3.2)

    1.3 Twitter reply network - contains a list of tuples in the form of (u,v,w) indicating node u replies node v w times (e.g., Table 3.3)

u	v
1	4
1	5
2	1
3	2
4	3
4	5
5	6

Table 3.1: Twitter follow network

u	v	w
2	1	30
3	2	30
4	3	10

Table 3.2: Twitter mention network

u	v	w
2	1	30
3	2	30
1	4	30
1	5	15
5	6	60

Table 3.3: Twitter reply network

u	v	w
2	1	10
3	2	10
1	5	15
4	5	10

Table 3.4: Twitter retweet network

v	t
1	100
2	100
3	100
4	100
5	100
6	100

Table 3.5: Twitter tweets network

1.4 Twitter retweet network - contains a list of tuples in the form of (u,v,w) indicating node u retweets node v w times (e.g., Table 3.4)

1.5 Twitter tweets network - contains a list of tuples in the form of (u,t) indicating node u posts t tweets (e.g., Table 3.5)

2. B-seeds (denoted as S_0^B) - a list of m B-nodes in the form of $[u_1,u_2,...,u_m]$, where u_i is the node id, (e.g., [26339, 191214, ..., 503050]).

3. Budget k - an integer indicating the cardinality of seed set of A-nodes

The four main components of this system and complete flow in the **CIAM** framework are shown in Figure 3.4.

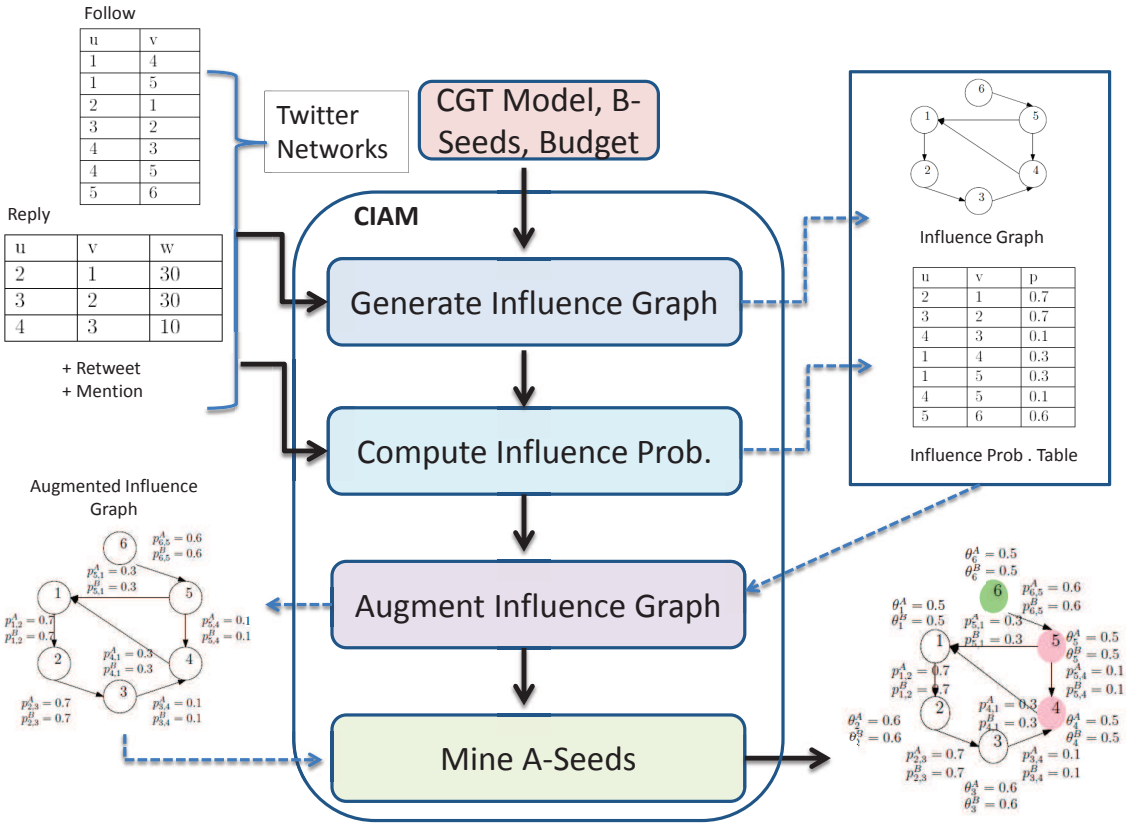


Figure 3.4: CIAM Framework

The four main steps involved in the **CIAM** are presented below, before the formal presentation of the algorithm.

Step 1. (line 1 of **CIAM** (Algorithm 1)) **CIAM** calls **convertFollowToInf** (Algorithm 2) to construct an influence graph $G = (V, E)$ from Twitter follow network, as done by existing algorithms [Kempe et al. 2003] and [Ahmed and Ezeife 2013]. Initially, the influence graph $G = (V, E)$ is empty. For each tuple (u, v) in the Twitter follow network, **convertFollowToInf** adds nodes u and v to the influence graph $G = (V, E)$ if nodes u and v have not been added to the graph yet, and adds a directed edge from v to u. Details of step 1 are presented in Section 3.2.1.

Step 2. (line 2 of **CIAM** (Algorithm 1)) **CIAM** uses Maximum-Likelihood Estimation [Fisher 1922] to construct the formula of the pairwise influence probabilities under multinomial distribution. **CIAM** calls **computeInfluenceProb** (Algorithm 3) which uses relational algebra operators left-join and projection on 5 Twitter datasets (i.e., Twitter follow network, Twitter mention network, Twitter reply network, Twitter retweet network, and Twitter tweets network) to retrieve the values of parameters in the pairwise influence probabilities formula and plug the values into the formula in order to compute the pairwise influence probabilities $p_{v,u}$ for each edge (v, u) in the influence graph which is generated from Step 1. Details of step 2 are presented in Section 3.2.2.

Step 3. (line 3 of **CIAM** (Algorithm 1)) **CIAM** calls **augmentG** (Algorithm 4) to augment the influence graph $G = (V, E)$ (generated from Step 1) as follows. For each edge $(v, u) \in E$, **augmentG** looks up the influence probability table to find the pairwise influence probability $p_{v,u}$. It assigns the edge (v, u) two pairwise influence probabilities, $p^A_{v,u} = p_{v,u}$ (the probability that v influences u to adopt technology A) and $p^B_{v,u} = p_{v,u}$ (the probability that v influences u to adopt technology B). It stops when all the edges $(v, u) \in E$ have been visited. When it stops, it outputs the augmented influence graph $G = (V, E, P)$ where V represents Twitter users, E

represents the influence interactions between Twitter users, P represents the pairwise influence probabilities between two Twitter users (Figure 3.6), as done by [Kempe et al. 2003] and [Ahmed and Ezeife 2013]. Details of step 3 are presented in Section 3.2.3.

Step 4. (line 4 of **CIAM** (Algorithm 1)) **CIAM** calls **cgtMineA** (Algorithm 6) to finds the k most influential A-nodes in a network where there exists a seed set of B-nodes. **cgtMineA** consists of two phases. The first phase exploits the greedy algorithm [Kempe et al. 2003] such that for each node v that is not in the two seed sets (i.e., S_0^A and S_0^B), the algorithm computes the marginal gain of adding v to S_0^A and S_0^B, picks the node which yields the maximum marginal gain, and repeats this process k times to find k A-seeds. The second phase exploits the local search algorithm [Ahmed and Ezeife, 2013] such that if swapping any A-seed in S_0^A (found in the first phase) and any node not in the two seed sets (i.e., S_0^A and S_0^B) yields more A-nodes at the end of the diffusion, the algorithm will swap them. The algorithm will repeat the swapping operation until no more improvements are possible. Details of Step 4 are presented in Section 3.2.4.

The formal algorithm for the **CIAM** framework is shown in Algorithm 1.

3.2.1 Crawling Social Networks to Construct the Social Graph

The algorithm **convertFollowToInf** (Algorithm 2) presented in this section is the first step of our proposed framework **CIAM**. The input of the algorithm is Twitter follow network (Table 3.1). The Twitter follow network consists of tuples in the form of (u, v) meaning u follows v. Initially, the influence graph $G = (V, E)$ (where V is the nodes and E is the influence relationships between nodes) is set to $\emptyset$ (line 1). For each tuple (u, v) in the Twitter follow network, **convertFollowToInf** adds nodes u and v to the influence graph G if they have not been added to the G (line 2.1), then it adds a directed edge from nodes v to u (lines 2.2). After all the tuples are processed,

Algorithm 1 CIAM(TwitterData,S_0^B,k) - The main algorithm for finding k influential A-nodes in social network with existing B-nodes

Input: Twitter networks, B-seeds, budget k
Output: A-seeds of size at most k

1: convert Twitter follow network (e.g., Table 3.1) to an influence graph $G = (V, E)$ (e.g., Figure 3.5) where V represents Twitter users and E represents influence relationships between them using algorithm **convertFollowToInf** (presented in Algorithm 2)
2: learn pairwise influence probabilities from Twitter networks and output an influence probability table (e.g., Table 3.10) using algorithm **computeInfluenceProb** (presented in Algorithm 3)
3: look for the influence probability table (e.g., Table 3.10), augment the influence graph $G = (V, E)$ by assigning the pairwise influence probabilities to each edge $(v, u) \in E$, and output an augmented graph $G = (V, E, P)$ (e.g. Figure 3.6) where V represents Twitter users, E represents influence relationships between them, and P represents the influence probabilities as the edge weights using algorithm **augmentG** (presented in Algorithm 4)
4: find A-seeds in the augmented graph $G = (V, E, P)$ using algorithm **cgtMineA** (presented in Algorithm 6)

it outputs an influence graph $G = (V, E)$ (Figure 3.5), as done by existing algorithms proposed in [Kempe et al. 2003] and [Ahmed and Ezeife 2013].

Algorithm 2 convertFollowToInf(Twitter Follow Network) - Construct an influence graph from Twitter follow network

Input: Twitter follow network with tuple (u,v) meaning u follows v
Output: an influence graph $G = (V, E)$

1. Set G to $\emptyset$
2. For each tuple (u, v) in Twitter follow network
   2.1 add nodes u and v to the influence graph G
   2.2 add a directed edge (v, u) to the influence graph G
3. **return** G

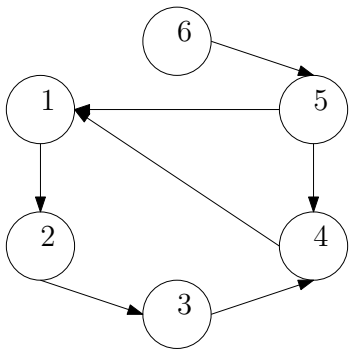


Figure 3.5: Influence Graph

3.2.2 Learning Influence Probabilities as Edge Weights from Twitter

The underlying social network we use to study influence maximization in the CGT model is Twitter network. Twitter uses retweet, reply and mention to say I like your tweets. Twitter's retweet measures how far an original tweet propagates throughout the network. Users who have a higher number of retweeted tweets can be considered more influential than users who have a few number of retweeted tweets [Russell 2013]. Twitter's reply measures how much your tweets make me feel engaged such that I want to talk something back to you [Wu et al. 2011]. Users who have a higher number of replied tweets can be considered more influential than users who have a few number of replied tweets. Twitter's mention measures the name value of the mentioned user [Cha et al. 2010]. Users who are mentioned more frequently in other users' tweets can be considered more influential than users who are mentioned infrequently in other users' tweets.

In this thesis, we assume that for each tweet of user v, there is at most one mention, one reply, or one retweet from user u. The reaction of user u to each tweet of user v can be viewed as a Bernoulli trial, responding (i.e., retweet, reply or mention) or not responding. Further, we assume that the probability that u responds (i.e., retweets, replies, or mentions) v's tweets is the pairwise influence probability $p_{v,u}$

(i.e., the probability that v influences u to perform an action once v becomes active). We use Maximum-Likelihood Estimation under Bernoulli distribution [Ahmed and Ezeife 2013] to estimate $p_{v,u}$ as follows,

$$p_{v,u} = \frac{\#\, retweets\ of\ u\ on\ v\ +\#\, replies\ of\ u\ on\ v\ +\#\, mentions\ of\ u\ on\ v}{\#\, tweets\ of\ v} \quad (3.4)$$

Having constructed the formula of the pairwise influence probability $p_{v,u}$ (equation 3.4), we now present the algorithm **computeInfluenceProb** which uses relational algebra operators left-join and projection on Twitter datasets to retrieve the numerator and denominator in equation 3.4, and compute the pairwise influence probability $p_{v,u}$. The algorithm **computeInfluenceProb** (Algorithm 3) presented in this section is the second step of our proposed framework **CIAM**. It takes as input 5 Twitter datasets, i.e., Twitter follow network (Table 3.1) which consists of tuples in the form of (u, v) meaning u follows v, Twitter tweets network (Table 3.5) which consists of tuples in the form of (v, t) meaning v posts t tweets in total, Twitter mention network (Table 3.2) which consists of tuples in the form of (u, v, w) meaning u mentions v w times, Twitter reply network (Table 3.3) which consists of tuples in the form of (u, v, w) meaning u replies v w times, and Twitter retweet network (Table 3.3) which consists of tuples in the form of (u, v, w) meaning u retweets v w times. **computeInfluenceProb** outputs the pairwise influence probabilities $p_{v,u}$ for each edge $(v, u) \in E$ (Table 3.10). There are 5 main steps in **computeInfluenceProb**.

Step 1. (line 1 of **computeInfluenceProb** (Algorithm 3)), **computeInfluenceProb** first concatenates Twitter mention network (Table 3.2), Twitter reply network (Table 3.3), and Twitter retweet network (Table 3.3) into one table named Tri, and then groups Tri by columns u and v such that each group in Tri represents node u mentions, replies, or retweets node v w times (Table 3.6).

Step 2. (line 2 of **computeInfluenceProb** (Algorithm 3)), **computeInfluenceProb** processes the grouped Tri (Table 3.6), sums up the value of w per group to obtain a summed-up Tri table (Table 3.7). Each tuple in the summed-up Tri table is in the form of (u, v, w) where w is $\#\, retweets\, of\, u\, on\, v\, +\, \#\, replies\, of\, u\, on\, v\, +\, \#\, mentions\, of\, u\, on\, v$, i.e., the numerator in equation (3.9).

Step 3. (line 3 of **computeInfluenceProb** (Algorithm 3)) **computeInfluenceProb** left-joins the summed-up Tri (Table 3.7) and TwitterTweets (Table 3.5) into one table named TriTweets (Table 3.8). Each tuple in TriTweets is in the form of (u, v, w, t) where w is $\#retweets of u on v + \#replies of u on v + \#mentions of u on v$, the numerator in equation (3.9), t is $\#\, tweets\, of\, v$, i.e., the denominator in equation (3.9).

Step 4. (line 4 of **computeInfluenceProb** (Algorithm 3)) **computeInfluenceProb** adds to TriTweets (Table 3.8) a new column named $p_{v,u}$ whose value is w/t to obtain an expended TriTweets (Table 3.9). The expended TriTweets table has tuples in the form of (u, v, w, t, p) where w is $\#retweets of u on v + \#replies of u on v + \#\, mentions\, of\, u\, on\, v$, the numerator in equation 3.4, t is $\#\, tweets\, of\, v$, the denominator in equation 3.4, and $p = w/t$ is the pairwise influence probability $p_{v,u}$ based on equation 3.4.

Step 5. (line 5 of **computeInfluenceProb** (Algorithm 3)) **computeInfluenceProb** drops unwanted columns w and t from Table 3.9 to obtain a pruned Tritweets table (with only three columns, i.e., u, v, $p_{v,u}$), and left-joins Twitter follow network (Table 3.1) and the pruned Tritweets table to obtain the final influence probability table named InfluenceProbTable (Table 3.10) where each tuple is in the form of (u, v, $p_{v,u}$) indicating the influence that node v exerts on node u, that is if v is active, it succeeds in activating u with the probability of $p_{v,u}$.

u	v	w
2	1	30
2	1	30
2	1	10
3	2	30
3	2	30
3	2	10
4	3	10
1	4	30
1	5	15
1	5	15
4	5	10
5	6	60

Table 3.6: Concatenate Twitter mention network, Twitter reply network, and Twitter retweet network into one table named Tri and group Tri by columns u and v

u	v	w	t
2	1	70	100
3	2	70	100
4	3	10	100
1	4	30	100
1	5	30	100
4	5	10	100
5	6	60	100

Table 3.8: Left-join Tri and TwitterTweets on column v to obtain a new table named TriTweets

u	v	p
2	1	0.7
3	2	0.7
4	3	0.1
1	4	0.3
1	5	0.3
4	5	0.1
5	6	0.6

Table 3.10: Drop columns w and t from TriTweets, and left-join Twitter follow network and TriTweets to obtain the influence probability table, where each tuple (u, v, p) means the probability that node v influences on node u is p.

u	v	w
2	1	70
3	2	70
4	3	10
1	4	30
1	5	30
4	5	10
5	6	60

Table 3.7: The summed-up Tri by computing the sum of w for each group

u	v	w	t	p
2	1	70	100	0.7
3	2	70	100	0.7
4	3	10	100	0.1
1	4	30	100	0.3
1	5	30	100	0.3
4	5	10	100	0.1
5	6	60	100	0.6

Table 3.9: Add a new column named p to TriTweets, where $p = w/t$

Algorithm 3 computeInfluenceProb(TwitterData) - Compute pairwise influence probabilities for each edge in the influence graph

Input: Twitter follow network (e.g., Table 3.1), Twitter tweets network (e.g., Table 3.5), Twitter mention network (e.g., Table 3.2), Twitter reply network (e.g., Table 3.3, Twitter retweet network (e.g., Table 3.4))

Output: an influence probability table (e.g., Table 3.10)

1: Concatenate Twitter mention network, Twitter reply network, and Twitter retweet network into one table named Tri and group Tri by columns u and v as shown in Table 3.6
2: Process the grouped Tri and get the sum of column w for each group as shown in Table 3.7
3: Left-join the summed-up Tri and Twitter tweets network on column v to obtain a joined table named TriTweets as shown in Table 3.8
4: Add a new column named p to the joined TriTweets, where $p = w/t$ as shown in Table 3.9
5: Drop columns w and t from TriTweets, left-join Twitter follow network and TriTweets to obtain the influence probability table named InfluenceProbTable as shown in Table 3.10
6: **return** InfluenceProbTable

3.2.3 Augment the Influence Graph with Learned Pairwise Influence Probabilities

The algorithm **augmentG** (Algorithm 4) presented in this section is the third step of our proposed framework **CIAM**. **augmentG** takes as input the influence graph $G = (V, E)$ (Figure 3.5) generated by **convertFollowToInf** (Algorithm 2), the influence probability table (Table 3.10) derived from **computeInfluenceProb** (Algorithm 3). For each edge $(v, u) \in E$, **augmentG** looks up the influence probability table to find the pairwise influence probability $p_{v,u}$ (line 1.1). It assigns the edge (v, u) two pairwise influence probabilities, $p^A_{v,u} = p_{v,u}$ (the probability that v influences u to adopt technology A) (line 1.2) and $p^B_{v,u} = p_{v,u}$ (the probability that v influences u to adopt technology B) (line 1.3). It stops when all the edges $(v, u) \in E$ have been visited. When it stops, it outputs the augmented influence graph $G = (V, E, P)$ where V represents Twitter users, E represents the influence interactions between Twitter users, P represents the pairwise influence probabilities between two Twitter

users (Figure 3.6), as done by [Kempe et al. 2003] and [Ahmed and Ezeife 2013].

Algorithm 4 augmentG(G,InfluenceProbTable) - Assign influence probabilities to each edge in the influence graph

Input: the influence graph $G = (V, E)$ without edge weights, influence probability table (i.e., InfluenceProbTable) with tuple (v,u,$p_{v,u}$)

Output: an augmented influence graph $G = (V, E, P)$ with influence probabilities as edge weights

1. For each edge $(v, u) \in E$
   - 1.1 Look up the influence probability table (Table 3.10) for $p_{v,u}$
   - 1.2 $p^A_{v,u} = p_{v,u}$
   - 1.3 $p^B_{v,u} = p_{v,u}$
2. **return** G

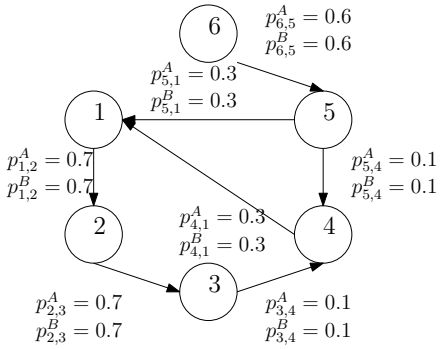


Figure 3.6: Influence graph augmented with pairwise influence probabilities for all edges

3.2.4 Discovering Influential Nodes for a Competing Action

The algorithm **cgtMineA** (Algorithm 6) presented in this section is the fourth step of our proposed framework **CIAM**. **cgtMineA** finds the k most influential A-nodes in a network where there exists a seed set of B-nodes. The algorithm takes as input the augmented influence graph $G = (V, E, P)$ (where V represents Twitter users, E represents the influence interactions between Twitter users, P represents the pairwise influence probabilities between two Twitter users) generated by **augmentG** (Algorithm 4), the seed set for B (denoted as S^B_0), and a non-negative integer k meaning

the number of influential A-nodes to be discovered. **cgtMineA** outputs a seed set for A (denoted as S_0^A) with size at most k that maximizes the expected number of final adoptions of technology A. **cgtMineA** consists of two phases. The first phase exploits the greedy algorithm [Kempe et al. 2003] such that for each node v that is not in the two seed sets (i.e., S_0^A and S_0^B), the algorithm computes the marginal gain of adding v to S_0^A and S_0^B, picks the node which yields the maximum marginal gain, and repeats this process k times to find k A-seeds. The second phase exploits the local search algorithm [Ahmed and Ezeife, 2013] such that if swapping any A-seed in S_0^A (found in the first phase) and any node not in the two seed sets (i.e., S_0^A and S_0^B) yields more A-nodes at the end of the diffusion, the algorithm will swap them. The algorithm will repeat the swapping operation until no more improvements are possible.

Before we present algorithm **cgtMineA**, we introduce an algorithm named **cgtInfA** that is called by **cgtMineA** for computing the A-influence spread of S_0^A and S_0^B (denoted as $\sigma^A(S_0^A, S_0^B)$).

cgtInfA (Algorithm 5) takes as input

1. the augmented influence graph $G = (V, E, P)$ (where V represents Twitter users, E represents the influence interactions between Twitter users, P represents the pairwise influence probabilities between two Twitter users) generated by **augmentG** (Algorithm 4)
2. two seed sets S_0^A and S_0^B

Each node $u \in V$ is associated with the following node parameters

1. float f_u^A - the threshold function of node $u \in V$ for technology A
2. float f_u^B - the threshold function of node $u \in V$ for technology B
3. float θ_u^A - the threshold for technology A, randomly chosen

4. float θ_u^B - the threshold for technology B, randomly chosen

5. string *state* - A, B, AB, or 0

cgtInfA uses the following variables

1. T^A to store the set of A-nodes activated during last step, initially T^A is set to S_0^A

2. T^B to store the set of B-nodes activated during last step, initially T^B is set to S_0^B

3. new^A to store the set of A-nodes activated during current step, new^A is set to $\emptyset$ at the beginning of the current step

4. new^B to store the set of B-nodes activated during current step, new^B is set to $\emptyset$ at the beginning of the current step

5. inf^A to store A-influence spread of S_0^A and S_0^B, initially it is set to the number of nodes in the seed set for A

cgtInfA outputs the A-influence spread of S_0^A and S_0^B (denoted as $\sigma^A(S_0^A, S_0^B)$), i.e., the expected number of A-nodes at the end of CGT diffusion process with the seed sets S_0^A and S_0^B. There are 5 main steps in **cgtInfA** (Algorithm 5).

Step 1. (line 1 of **cgtInfA** (Algorithm 5)) **cgtInfA** uses variable T^A to store the set of A-nodes activated during last step, initially T^A is set to S_0^A.

Step 2. (line 2 of **cgtInfA** (Algorithm 5)) **cgtInfA** uses variable T^B to store the set of B-nodes activated during last step, initially T^B is set to S_0^B.

Step 3. (line 3 of **cgtInfA** (Algorithm 5)) **cgtInfA** uses variable inf^A to store the A-influence spread of S_0^A and S_0^B, initially it is set to the number of nodes in the seed set for A.

Step 4. (line 4 of **cgtInfA** (Algorithm 5)) As long as there are nodes activated during last time step, those activated nodes would propagate influence during current step

through the network (line 4). **cgtInfA** uses variable new^A to store the set of A-nodes activated during current step, new^A is set to $\emptyset$ at the beginning of the current step (line 4.1), and new^B to store the set of B-nodes activated during current step, new^B is set to $\emptyset$ at the beginning of the current step (line 4.2). For each A-node v activated from last step in T^A, the algorithm will loop through each inactive out-neighbor u of v (line 4.3), compute the threshold function f_u^A for node u using equation 3.2 (line 4.3.1.1). If the threshold function f_u^A is no less than its threshold θ_u^A, the algorithm adds node u to new^A which is the set of A-nodes newly activated in current step (line 4.3.1.2), and increases A-influence spread by 1 (line 4.3.1.3). Similarly, for each B-node v activated from last step in T^B (line 4.4), the algorithm will loop through each inactive out-neighbor u of v, compute the threshold function f_u^B for node u using equation 3.3 (line 4.4.1.1). If the threshold function f_u^B is no less than its threshold θ_u^B, it adds node u to new^B which is the set of B-nodes newly activated in current step (line 4.4.1.2). After it processes all nodes in T^A (the set of A-nodes activated during last step) and T^B (the set of B-nodes activated during last step), the current diffusion step is done. At this moment, the set of A-nodes activated during current step becomes the set of A-nodes activated from last step (line 4.5), and the set of B-nodes activated during current step becomes the set of B-nodes activated from last step (line 4.6).

Step 5. (line 5 of **cgtInfA** (Algorithm 5)) When both T^A (the set of A-nodes activated during last step) and T^B (the set of B-nodes activated during last step) are empty meaning no more activations, it stops and returns the expected number A-nodes at the end of the diffusion, i.e., the A-influence spread of S_0^A and S_0^B.

Example 3.2.1. ***How cgtInfA Works.*** *In the social network shown in Figure 3.7 (b), at time* 0*, there are two seed sets,* $S_0^A = \{4, 5\}$ *and* $S_0^B = \{6\}$*. We will show how* ***cgtInfA*** *(Algorithm 5) computes the influence spread for technology A given the two seed set* S_0^A *and* S_0^B*, denoted as* $\sigma^A(S_0^A, S_0^B)$*. At time* 1*, nodes* 4 *and* 5 *jointly*

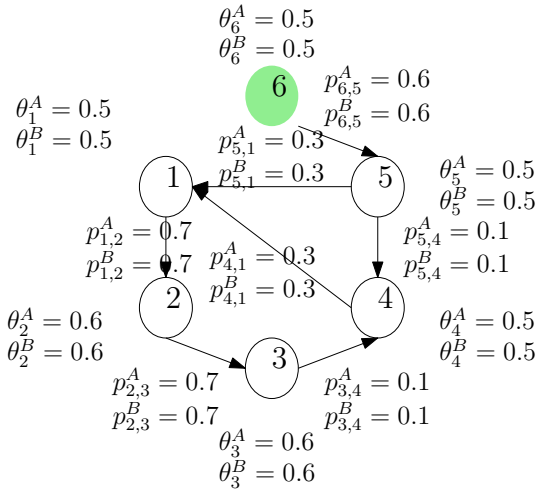
Algorithm 5 cgtInfA($G = (V, E, P)$,S_0^A,S_0^B) - compute the number of A-nodes at the end of the diffusion when the two seed sets are S_0^A and S_0^B

Input: The augmented influence graph $G = (V, E, P)$ with influence probability as edge weights, two seed sets S_0^A and S_0^B

Output: inf^A - the A-influence spread of S_0^A and S_0^B

1. Set T^A, the set of A-nodes activated during last time step, to S_0^A
2. Set T^B, the set of B-nodes activated during last time step, to S_0^B
3. Set inf^A, the A-influence spread of S_0^A and S_0^B to the number of nodes in S_0^A
4. While we have either A-nodes or B-nodes activated from last step to propagate influences
   - 4.1 Set new^A, the set of A-nodes newly activated in current step, to $\emptyset$
   - 4.2 Set new^B, the set of B-nodes newly activated in current step, to $\emptyset$
   - 4.3 For each A-node v activated from last step
     - 4.3.1 For each inactive node u in the out-neighbors of v
       - 4.3.1.1 compute the threshold function f_u^A for node u using equation 3.2
       - 4.3.1.2 Add u to new^A, the set of A-nodes newly activated in current step if the threshold function crosses the threshold
       - 4.3.1.3 Increase the number of A-nodes by 1
   - 4.4 For each B-node v activated from last step
     - 4.4.1 For each inactive node u in the out-neighbors of v
       - 4.4.1.1 compute the threshold function f_u^B for node u using equation 3.3
       - 4.4.1.2 Add u to new^B, the set of B-nodes newly activated in current step if the threshold function crosses the threshold
   - 4.5 At the end of the current step, set new^A, the set of A-nodes activated during current step to T^A, the set of A-nodes activated from last step
   - 4.6 At the end of the current step, set new^B, the set of B-nodes activated during current step to T^B, the set of B-nodes activated from last step
5. Return inf^A, the number A-nodes at the end of the diffusion

activate node 1 *since* $f_1^A = 1-(1-p_{4,1}^A)(1-p_{5,1}^A) = 1-(1-0.3)(1-0.3) = 0.51 > \theta_1^A = 0.5$, *the state of node* 1 *becomes A. At time* 2, *node* 1 *activates node* 2 *since* $f_2^A = 1-(1-p_{1,2}^A) = p_{1,2}^A = 0.7 > \theta_2^A = 0.6$, *the state of node* 2 *becomes A. At time* 3, *node* 2 *activates node* 3, *since* $f_3^A = 1-(1-p_{2,3}^A) = p_{2,3}^A = 0.7 > \theta_3^A = 0.6$, *the state of node* 3 *becomes A. At this point, the diffusion stops since no more activations are possible. The set of A-nodes at this point is* $\{1,2,3,4,5\}$, *and the number of A-nodes at this moment is* 5. *Algorithm 5 returns* 5 *which is the A-influence spread of* S_0^A *and* S_0^B.



(a) Input Graph

Figure 3.7: Example of cgtInfA(S_0^A, S_0^B)

Having introduced algorithm **cgtInfA**, we can now present the algorithm **cgtMineA** (Algorithm 6) which finds the k most influential A-nodes in a network where there exists a seed set of B-nodes. The algorithm takes as input the augmented influence graph $G = (V, E, P)$ which generated by **augmentG** (Algorithm 4), the seed set for B (denoted as S_0^B), and a non-negative integer k meaning the number of influential nodes to be discovered, and outputs a seed set for A (denoted as S_0^A) with size at most k that maximizes the expected number of final adoptions of technology A. There are 5 main steps in **cgtMineA** (Algorithm 6).

Step 1. (line 1 of **cgtMineA** (Algorithm 6)) **cgtMineA** uses variable S_0^A to store A-seeds. Initially, S_0^A is set to $\emptyset$ (line 1).

Step 2. (line 2 of **cgtMineA** (Algorithm 6)) **cgtMineA** consists of two phases. The first phase adopts the existing greedy algorithm [Kempe et al. 2003] such that for each node v that is not in the two seed sets (i.e., S_0^A and S_0^B), the algorithm computes the marginal gain of adding v to two seed sets (i.e., S_0^A and S_0^B), picks the node which yields the max and adds it to S_0^A (lines 3-4).

Step 3. (line 3 of **cgtMineA** (Algorithm 6)) **cgtMineA** repeats step 2 k times to find k seeds.

Step 4. (line 4 of **cgtMineA** (Algorithm 6)) The second phase of **cgtMineA** exploits the local search algorithm [Ahmed and Ezeife, 2013] such that if swapping any A-seed in S_0^A (found in the first phase) and any node not in the two seed sets yields larger A-influence spread (line 3.1), the algorithm will swap them.

Step 5. (line 5 of **cgtMineA** (Algorithm 6)) **cgtMineA** repeats step 4 until no more improvements in A-influence spread are possible.

Algorithm 6 cgtMineA($G = (V, E, P)$, S_0^B, k)- Find k influential A-nodes under CGT

Input: an augmented influence graph $G = (V, E, P)$ with influence probabilities as edge weights, a seed set for B (denoted as S_0^B), and a non-negative integer k

Output: a seed set for A (denoted as S_0^A) with size at most k that maximizes the expected number of final adoptions of technology A

1. Set S_0^A to $\emptyset$
2. Compute the marginal gain of adding each node $u \in V - S_0^A - S_0^B$ to S_0^A and S_0^B, pick the node u which yields the maximum marginal gain, and add node u to A-seed set S_0^A
3. Repeat Step 2 k times to find k A-seeds
4. Local Search on S_0^A to improve the selection by swapping node $u \in S_0^A$ and node $v \in V - S_0^A - S_0^B$ if $\sigma_{CGT}(S_0^A + \{v\} - \{u\}, S_0^B) > \sigma_{CGT}(S_0^A, S_0^B)$
5. Repeat step 4 until no more improvements in A-influence spread are possible
6. Return A-seed set S_0^A

Remark 3.2.1. *The marginal gain of adding a A-node u to two seed sets S_0^A and S_0^B is defined as $\sigma(S_0^A \cup \{u\}, S_0^B) - \sigma(S_0^A, S_0^B)$.*

Now, we will show how **cgtMineA** works through example 3.2.2

Example 3.2.2. *Initially, $S_0^A = \emptyset$ and $S_0^B = \{6\}$. To find the first influential A seed, the algorithm computes the marginal gain of adding each node v not in the two seed sets (i.e., $v \in V - S_0^A - S_0^B$) to S_0^A and S_0^B. The marginal gain of adding each node v into S_0^A and S_0^B is summarized in Table 3.11. The algorithm picks the node with the maximum marginal gain, which is node 1 and adds it to S_0^A. At this moment, $S_0^A = \{1\}$ and $S_0^B = \{6\}$. To find the second influential A seed, the algorithm computes the marginal gain of adding each node $v \in V - S_0^A - S_0^B$ to S_0^A and S_0^B. The marginal gain of adding each node v into S_0^A and S_0^B is summarized in Table 3.12. The algorithm picks the node with the maximum marginal gain, which is node 4 and adds it to S_0^A. So, $S_0^A = \{1, 4\}$. Since budget $k = 2$, and we have 2 nodes 1 and 4 in S_0^A, the greedy part of the algorithm is done.*

Now, the algorithm will swap any node in S_0^A with any node not in the two seed sets S_0^A and S_0^B to see if there is any improvement with the spread. At this point, $S_0^A = \{1, 4\}$, $S_0^B = \{6\}$, the set of nodes not in the two seed sets $V - S_0^A - S_0^B = \{2, 3, 5\}$, the spread $\sigma^A(S_0^A, S_0^B) = 4$. The algorithm computes the spread after swapping nodes 1 and 2, and obtains $\sigma^A(S_0^A - \{1\} + \{2\}, S_0^B) = 3 < \sigma^A(S_0^A, S_0^B) = 4$, meaning no improvements. Then, the algorithm computes the spread after swapping nodes 1 and 3, and obtains $\sigma^A(S_0^A - \{1\} + \{3\}, S_0^B) = 2 < \sigma^A(S_0^A, S_0^B) = 4$, meaning no improvements. Next, the algorithm computes the spread after swapping nodes 1 and 5, and obtains $\sigma^A(S_0^A - \{1\} + \{5\}, S_0^B) = 5 > \sigma^A(S_0^A, S_0^B) = 4$, meaning there is an improvement. Hence, the algorithm will keep the swap. At this point, $S_0^A = \{5, 4\}$, $S_0^B = \{6\}$, the set of nodes not in the two seed sets $V - S_0^A - S_0^B = \{1, 2, 3\}$, the spread $\sigma^A(S_0^A, S_0^B) = 5$. The algorithm swaps any node in S_0^A with any node not in the two seed sets S_0^A and S_0^B to see if there is any improvement with the spread. Since none

Node	Marginal Gain
1	3
2	2
3	1
4	1
5	1

Table 3.11: Marginal Gain: First Pass of cgtMineA's Greedy Phase

Node	Marginal Gain
2	0
3	0
4	1
5	1

Table 3.12: Marginal Gain: Second Pass of cgtMineA's Greedy Phase

of the swapping operations yield any improvements, the algorithm stops and returns $S_0^A = \{5, 4\}$.

3.3 Complexity Analysis

The **cgtInfA** algorithm (Algorithm 5), which is a sub-procedure of **cgtMineA** (Algorithm 6), runs in time $O(m*E)$, where m is the number of round for MC simulations, E is the number of edges in G, since for each round of MC simulation, **cgtInfA** scans the out-neighbors of each active node, and the total number of out-neighbors of all active nodes is $O(E)$.

The **cgtMineA** algorithm (Algorithm 6) runs in time $O(k * V * m * E)$, where k is the budget, i.e., the number of A-nodes to be discovered as early adopters of technology A, V is the number of nodes in G, m is the number of round for MC simulations, and E is the number of edges in G. **cgtMineA** consists of two phases. The first phase is greedy algorithm which runs in time $O(k*V*m*E)$, where k is the budget, i.e., the number of A-nodes to be discovered as early adopters of technology A, V is the number of nodes in G, m is the number of round for MC simulations, and E is the number of edges in G, since for each pass of the greedy phase, **cgtInfA** calls **cgtInfA** algorithm (Algorithm 5) $O(V)$ times, and there are k passes. The second phase is the local search based algorithm which could run for an exponential amount

of time ($O(2^n)$) until it finds an improvement in the influence spread [Ahmed and Ezeife, 2013]. To ensure the algorithm runs in polynomial time, we break the swap operation when the number of loops crosses $k * V * m * E$, which ensures the second phase runs in $O(k*V*m*E)$ time. Therefore, the overall running time of **cgtMineA** algorithm (Algorithm 6) is in $O(k * V * m * E)$.

Chapter 4

Experiments and Analysis

4.1 Dataset

On 4^{th} July 2012, two international experiments involved in searching for the elusive Higgs boson, the ATLAS and CMS collaborations announced the discovery of a Higgs boson-like partical. Domenico et al [2013] have tracked and monitored user activities on Twitter (i.e., posting tweets, retweets, mentions and replies about the discovery) before, during and after the announcement (i.e., between $00:00AM$, 1^{st} July 2012 and $11:59PM$, 7^{th} July 2012). In this research, we use their Higgs Twitter Datasets to study information diffusion under the CGT model.

The Higgs Twitter Dataset consists of four datasets, Twitter follow network, Twitter mention network, Twitter reply network, and Twitter retweet network. Twitter follow network consists of $456,631$ nodes and $14,855,875$ edges. Each line in the follower dataset is in the form of (u, v) meaning node u follows node v. Twitter mention network consists of $302,975$ nodes and $449,827$ edges. Each line in the mention dataset is in the form of (u, v, w) meaning node u mentions node v w times. Twitter reply network consists of $37,366$ nodes and $30,836$ edges. Each line in the reply dataset is in the form of (u, v, w) meaning node u replies node v w times. Twitter

retweet network consists of $425,008$ nodes and $733,647$ edges. Each line in the retweet network is in the form of (u, v, w) meaning node u retweets node v w times.

However, there are two issues with the Higgs Twitter Dataset. The first issue with the Higgs Twitter Dataset is that the follow network consists of $456,631$ nodes and $14,855,875$ edges, which is too big. Another issue with the Higgs Twitter Dataset is that it does not contain the Twitter tweets dataset. Since our main goal of this research is to show the quality of the seeds chosen by our proposed cgtMineA is better than that of CELF-like algorithms under the CGT model, to tackle the first issue, we extract a sub-graph from Twitter follow network for experiments. The sub-graph consists of $1,001$ nodes and $3,201$ edges. The extraction is done by randomly choosing a root node and performing breadth first search from the root, stopping when the number of nodes in the sub-graph is desired, as done by [He et al. 2012]. To tackle the second issue, we assign tweets count to each Twitter user by uniformly at random choosing a number over the interval $[1, 100]$ and adding the number to the total number of the user's retweets, mentions, and replies.

4.2 Algorithms Compared

In our experiments, we compare the quality of the seeds which is measured by the influence spread achieved by the following algorithms.

cgtMineA. Our proposed algorithm.

CELF. Greedy algorithm with lazy evaluation [Leskovec et al. 2007] under the CGT model that chooses k A-nodes with the largest marginal gain from the influence graph.

TGT. Local-search algorithm [Ahmed and Ezeife 2013] under the CGT model that chooses k A-nodes by two local search operations, add and swap.

4.3 Comparing Influence Spread

Figure 4.1 shows the influence spreads achieved by TGT, CELF, and our proposed cgtMineA respectively. The comparison is performed on the 1000-node sub-graph of Twitter follow network with 50 randomly chosen B-seeds. From Figure 4.1, we can see our proposed cgtMineA outperforms CELF for all A-seed set size as expected. TGT outperforms cgtMineA and CELF for all A-seed set size as expected at the cost of running time.

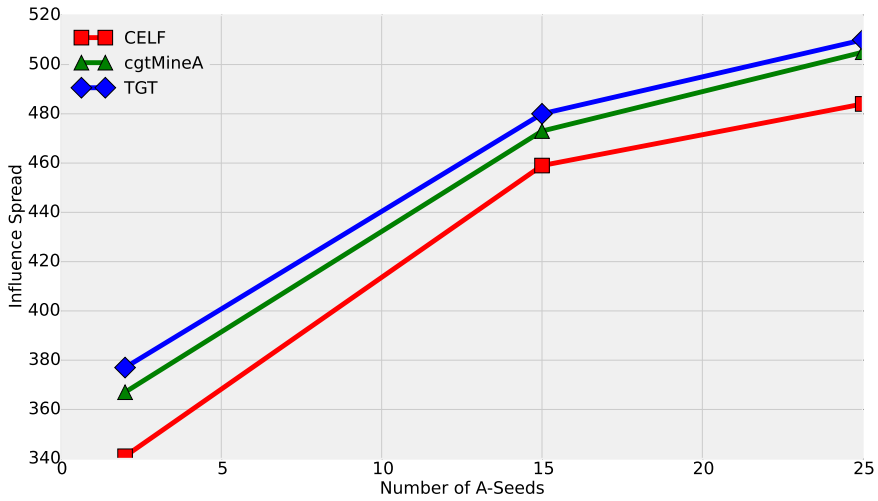


Figure 4.1: Influence spread of various algorithms in Twitter datasets

4.4 Comparing Running Time

Figure 4.2 shows the running time taken by TGT, CELF, and our proposed cgtMineA respectively. The comparison is performed on the 1000-node sub-graph of Twitter follow network with 50 randomly chosen B-seeds. From Figure 4.1, we can see that CELF performs almost in constant time when the size of A-seed set is ≤ 50. cgtMineA performs close to CELF when the size of the A-seed set is ≤ 15, takes more time than CELF as the size of the A-seed set increases but runs faster then TGT. This shows the room for improvement of cgtMineA in terms of scalability. As mentioned earlier,

scalability was not focus of this work; however there are several ways to make the approach more scalable. We discuss some of these approaches in the next section.

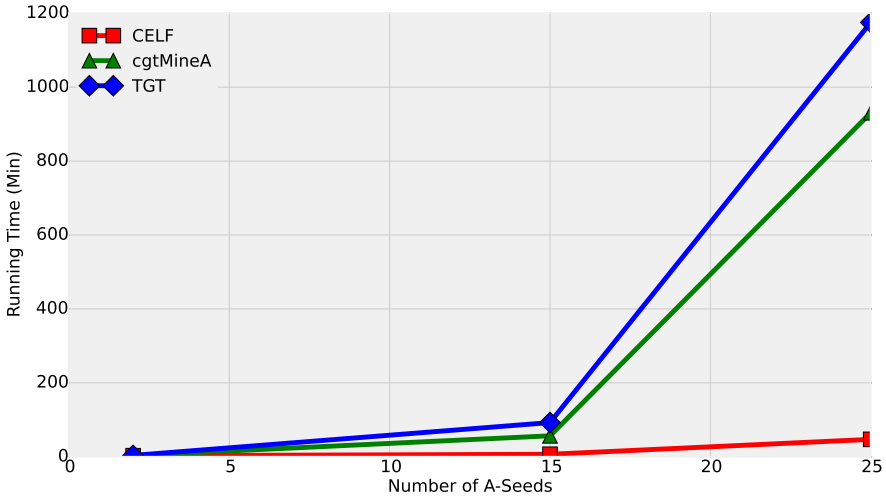


Figure 4.2: Running Time of various algorithms in Twitter datasets

From the experiments on the quality of A-influence spread and running time comparison, we can that see cgtMineA is a tradeoff solution between running time and the quality of the A-seed set because TGT under CGT model may run in exponential time.

Chapter 5

Conclusions and Future Works

Maximizing the spread of influence through a social network is to find a small set of influential people (the seed set) in the online communities (the crowd) such that if we market to them, the spread of influence will be maximized. The most motivating application of influence maximization is viral marketing. In this research, we have tackled the influence maximization problem in a network where there exist two competing influence diffusions.

First, we propose a diffusion model named Competing General Threshold (CGT) model to model how the two competing influences propagate from node to node and how a node decides to accept which influence. We show that the diffusion process under the CGT model is monotone and non-submodular, therefore the influence maximization problem under the CGT model boils down to monotone and non-submodular maximization which is proven to be NP-hard. Then, We exploit Maximum-Likelihood Estimation (MLE) to learn the two influence probabilities that v influences u to adopt each technology respectively from Twitter datasets. Based on the monotone and non-submodular property of CGT model, we propose an algorithm named cgtmineA to mine A-seeds as early adopters of technology A under the CGT model in a social network where early adopters of technology B already exist, based on the greedy al-

gorithm [Kempe et al. 2003] for the monotone property of CGT and the local search algorithm [Ahmed and Ezeife 2013] for the non-submodular property of CGT. We perform experiments on the real-world datasets from Twitter to show our proposed cgtMineA outperforms existing heuristics such as CELF by at most 15%.

In the future, to tackle the scalability of cgtMineA, we should consider the strength of weak ties [Granovetter et al. 1973] and community structure in networks [Fortunat and Santo 2009]. Another possible solution is to reduce the search space by ranking the nodes in terms of relevance as done in [Mumu and Ezeife 2014]. Also, we want to extend the Competing General Threshold network from two players to more than two players, look for more involved threshold functions, and quantify the threshold value per technology for each player. In order to design a more natural diffusion model, we should study game theory and include the idea to the model when dealing with more than one player. Other future directions include (1) to consider dynamic networks where new nodes come in, existing nodes leave, or the influence probability per edge changes as time goes on (i.e., it is not independent to time any more), (2) to consider multi-dimension network which incorporates Facebook network, Twitter network, LinkedIn, and so on.

Bibliography

Agarwal, N., Liu, H., Tang, L., and Yu, P. S. (2008). Identifying the influential bloggers in a community. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pages 207–218. ACM.

Agrawal, R., Srikant, R., et al. (1994). Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499.

Ahmed, S. and Ezeife, C. (2013). Discovering influential nodes from trust network. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, pages 121–128. ACM.

Aral, S., Muchnik, L., and Sundararajan, A. (2009). Distinguishing influence-based contagion from homophily-driven diffusion in dynamic networks. *Proceedings of the National Academy of Sciences*, 106(51):21544–21549.

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

Bishop, C. M. et al. (2006). *Pattern recognition and machine learning*, volume 4. springer New York.

Bonchi, F., Giannotti, F., Mazzanti, A., and Pedreschi, D. (2003). Examiner: Optimized level-wise frequent pattern mining with monotone constraints. In *Data*

Mining, 2003. ICDM 2003. Third IEEE International Conference on, pages 11–18. IEEE.

Borodin, A., Filmus, Y., and Oren, J. (2010). Threshold models for competitive influence in social networks. In *Internet and network economics*, pages 539–550. Springer.

Carnes, T., Nagarajan, C., Wild, S. M., and Van Zuylen, A. (2007). Maximizing influence in a competitive social network: a follower's perspective. In *Proceedings of the ninth international conference on Electronic commerce*, pages 351–360. ACM.

Cha, M., Haddadi, H., Benevenuto, F., and Gummadi, P. K. (2010). Measuring user influence in twitter: The million follower fallacy. *ICWSM*, 10:10–17.

Chen, W., Lakshmanan, L. V., and Castillo, C. (2013). Information and influence propagation in social networks. *Synthesis Lectures on Data Management*, 5(4):1–177.

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

Chen, W., Yuan, Y., and Zhang, L. (2010). Scalable influence maximization in social networks under the linear threshold model. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 88–97. IEEE.

Chen, Y.-C., Zhu, W.-Y., Peng, W.-C., Lee, W.-C., and Lee, S.-Y. (2014). Cim: community-based influence maximization in social networks. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(2):25.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1):21–27.

De Domenico, M., Lima, A., Mougel, P., and Musolesi, M. (2013). The anatomy of a scientific rumor. *Scientific reports*, 3.

Domingos, P. and Richardson, M. (2001). Mining the network value of customers. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 57–66. ACM.

Fortunato, S. (2010). Community detection in graphs. *Physics Reports*, 486(3):75–174.

Gionis, A., Terzi, E., and Tsaparas, P. (2013). Opinion maximization in social networks. In *SDM*, pages 387–395. SIAM.

Goyal, A., Bonchi, F., and Lakshmanan, L. V. (2008). Discovering leaders from community actions. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 499–508. ACM.

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

Goyal, A., Lu, W., and Lakshmanan, L. V. (2011a). Celf++: optimizing the greedy algorithm for influence maximization in social networks. In *Proceedings of the 20th international conference companion on World wide web*, pages 47–48. ACM.

Goyal, A., Lu, W., and Lakshmanan, L. V. (2011b). Simpath: An efficient algorithm for influence maximization under the linear threshold model. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 211–220. IEEE.

Granovetter, M. S. (1973). The strength of weak ties. *American journal of sociology*, pages 1360–1380.

Han, J., Pei, J., and Yin, Y. (2000). Mining frequent patterns without candidate generation. In *ACM SIGMOD Record*, volume 29, pages 1–12. ACM.

He, X., Song, G., Chen, W., and Jiang, Q. (2012). Influence blocking maximization in social networks under the competitive linear threshold model. In *SDM*, pages 463–474. SIAM.

Hill, S., Provost, F., and Volinsky, C. (2006). Network-based marketing: Identifying likely adopters via consumer networks. *Statistical Science*, pages 256–276.

Hu, Q., Wang, G., and Yu, P. S. (2014). Transferring influence: Supervised learning for efficient influence maximization across networks. In *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2014 International Conference on*, pages 45–54. IEEE.

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

Kleinberg, J. (2007). Cascading behavior in networks: Algorithmic and economic issues. *Algorithmic game theory*, 24:613–632.

Krause, A. and Guestrin, C. (2005). A note on the budgeted maximization of submodular functions.

Krause, A., Leskovec, J., Guestrin, C., VanBriesen, J., and Faloutsos, C. (2008). Efficient sensor placement optimization for securing large water distribution networks. *Journal of Water Resources Planning and Management*, 134(6):516–526.

Lappas, T., Terzi, E., Gunopulos, D., and Mannila, H. (2010). Finding effectors in social networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1059–1068. ACM.

Lee, J., Mirrokni, V. S., Nagarajan, V., and Sviridenko, M. (2009). Non-monotone submodular maximization under matroid and knapsack constraints. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 323–332. ACM.

Lerman, K. and Ghosh, R. (2010). Information contagion: An empirical study of the spread of news on digg and twitter social networks. *ICWSM*, 10:90–97.

Leskovec, J., Adamic, L. A., and Huberman, B. A. (2007a). The dynamics of viral marketing. *ACM Transactions on the Web (TWEB)*, 1(1):5.

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

MacQueen, J. et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA.

McCallum, A., Nigam, K., et al. (1998). A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on for text categorization*, volume 752, pages 41–48. Citeseer.

Mossel, E. and Roch, S. (2007). On the submodularity of influence in social networks. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 128–134. ACM.

Mumu, T. S. and Ezeife, C. I. (2014). Discovering community preference influence network by social network opinion posts mining. In *Data Warehousing and Knowledge Discovery*, pages 136–145. Springer.

Ostfeld, A., Uber, J. G., Salomons, E., Berry, J. W., Hart, W. E., Phillips, C. A., Watson, J.-P., Dorini, G., Jonkergouw, P., Kapelan, Z., et al. (2008). The battle of the water sensor networks (bwsn): A design challenge for engineers and algorithms. *Journal of Water Resources Planning and Management*, 134(6):556–568.

Pang-Ning, T., Steinbach, M., Kumar, V., et al. (2006). Introduction to data mining. In *Library of Congress*.

Park, J. S., Chen, M.-S., and Yu, P. S. (1995). *An effective hash-based algorithm for mining association rules*, volume 24. ACM.

Pastor-Satorras, R. and Vespignani, A. (2002). Immunization of complex networks. *Physical Review E*, 65(3):036104.

Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, 1(1):81–106.

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

Rogers, E. M. (2010). *Diffusion of innovations*. Simon and Schuster.

Rosen, K. (2011). *Discrete Mathematics and Its Applications 7th edition*. McGraw-Hill Science.

Russell, M. A. (2013). *Mining the Social Web: Data Mining Facebook, Twitter, LinkedIn, Google+, GitHub, and More*. " O'Reilly Media, Inc.".

Singer, Y. (2012). How to win friends and influence people, truthfully: influence maximization mechanisms for social networks. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 733–742. ACM.

Soni, G. and Ezeife, C. (2013). An automatic email management approach using data mining techniques. In *Data Warehousing and Knowledge Discovery*, pages 260–267. Springer.

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

Ullman, J. D., Leskovec, J., and Rajaraman, A. (2011). *Mining of Massive Datasets*. Cambridge University Press.

Wu, Y. and Ren, F. (2011). Learning sentimental influence in twitter. In *Future Computer Sciences and Application (ICFCSA), 2011 International Conference on*, pages 119–122. IEEE.

Zhang, H., Mishra, S., and Thai, M. T. (2014a). Recent advances in information diffusion and influence maximization of complex social networks. *Opportunistic Mobile Social Networks*, page 37.

Zhang, P., Chen, W., Sun, X., Wang, Y., and Zhang, J. (2014b). Minimizing seed set selection with probabilistic coverage guarantee in a social network. *arXiv preprint arXiv:1402.5516*.

VITA AUCTORIS

Xiao Ni Cao was born in Beijing, China. She received her Bachelor of Science in Computer Science from the University of British Columbia in May, 2012. She is currently a candidate for the Masters of Science in Computer Science at the University of Windsor, Ontario and hopes to graduate by April 2015.