2013

Social Network Opinion and Posts Mining for Community Preference Discovery

Tamanna Mumu
University of Windsor

Recommended Citation

Mumu, Tamanna, "Social Network Opinion and Posts Mining for Community Preference Discovery" (2013). *Electronic Theses and Dissertations.* Paper 4865.

Social Network Opinion and Posts Mining for Community Preference Discovery

By

Tamanna Mumu

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

2013

Social Network Opinion and Posts Mining for Community Preference Discovery

By

Tamanna Mumu

APPROVED BY:

April 25, 2013

AUTHOR’S 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

The popularity of posts, topics, and opinions on social media websites and the influence ability of users can be discovered by analyzing the responses of users (e.g., likes/dislikes, comments, ratings). Existing web opinion mining systems such as OpinionMiner is based on opinion text similarity scoring of users' review texts and product ratings to generate database table of features, functions and opinions mined through classification to identify arriving opinions as positive or negative on user-service networks or interest networks (e.g., Amazon.com). These systems are not directly applicable to user-user networks or friendship networks (e.g., Facebook.com) since they do not consider multiple posts on multiple products, users' relationships (such as influence), and diverse posts and comments.

In this thesis, we propose a new influence network (IN) generation algorithm (Opinion Based IN:OBIN) through opinion mining of friendship networks (like Facebook.com). OBIN mines opinions using extended OpinionMiner that considers multiple posts and relationships (influences) between users. Approach used includes frequent pattern mining algorithm for determining community (positive or negative) preferences for a given product as input to standard influence maximization algorithms like CELF for target marketing. Experiments and evaluations show the effectiveness of OBIN over CELF in large-scale friendship networks.

KEYWORDS

Influence Analysis, Recommendation, Ranking, Sentiment Classification, Large Scale Network, Social Network, Opinion Mining, Text Mining.

DEDICATION

To my parents Md. Rafique Uddin and Rumana Akhter, my sister Mohsina Sharmin Mouri and my husband Md. Nasir Uddin

ACKNOWLEDGEMENT

My sincere appreciation goes to my parents, husband and younger sister. Your perseverance and words of encouragement gave me the extra energy to see this work through.

I will be an ingrate without recognising the invaluable tutoring and supervision from Dr. Christie Ezeife. Your constructive criticism and advice at all times gave me the needed drive to successfully complete this work. I like to thank Dr. C. I. Ezeife for continuous research assistantship positions from NSERC and FedDev/OBI grants.

Special thanks go to my external reader, Dr. Eugene H. Kim, my internal reader, Dr. Subir Bandyopadhyay for accepting to be my thesis committee. Your decision, despite your tight schedules, to help in reading the thesis and providing valuable input is highly appreciated.

And lastly to friends and colleagues at University of Windsor, I say a very big thank you for your advice and support throughout the duration of this work.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

CHAPTER 1

INTRODUCTION

The analysis of complex networks is a new emerging research area in which networks are studied in several domains using data from a wide variety of sources. Examples of such networks are social networks, technological networks such as the Internet, biological networks such as neural networks, email networks, call detail records in telecommunications networks, transactional data in a financial institution, to learn who accessed what accounts and when (Bonchi et al., 2011). Research on social networks is being carried out using data collected from online interactions. Examples of social networks include acquaintance networks (e.g., Facebook, LinkedIn, Flickr, Instant Messenger) and collaboration networks (e.g., InnoCentive.com where companies post scientific problems, Linux open-source software community).

The rapid growth of the World Wide Web (WWW) powered by Web 2.0 (DiNucci, 1999) has made information available more than ever before and hence people now increasingly take their required information from one another rather than from corporations, media outlets, religion or political bodies. WWW has become most popular social media which covers almost all form of sharing such as experiences, photos, recommendations. To do this, people get involved in social networks formed by friend lists, by the bloggers who comment/rate on a certain topic in the blogspace, or by the users who write collaboratively in a wiki site (e.g., Wikipedia, Scholarpedia). People may give their opinions on the shared posts, those opinions may be positive, negative, or controversial to the posts. Several research (Pang et al. 2002, Turney 2002, Agrawal et al. 2003, Dave et al. 2003, Hu and Liu 2004, Mishne and Glance 2006, Nigam and Hurst 2006, Ding et al. 2008, Gomez et al. 2008, Li et al. 2008, Tang et al. 2009) have been done for analyzing users' opinions on interest networks (i.e., user-service interaction), but based on our knowledge, no work is found in friendship networks (i.e., user-user connections). Such user-service connections are domain specific and product-feature oriented. For example, these networks may be weblogs, newsgroups, bookmarks, question/answers, movie/product review domains, as opposed to friendship networks.

In this thesis a friendship network is considered, where users/groups can share their posts as an object (object may be a discussion topic or a product/service), the friends/fans of that user/group can submit their opinions in the form of likes/dislikes, thumbs-up/thumbs-down, or comments. Those comments can be the responses to the post or responses to another comment submitted by the users. In this way, the opinions can be categorized into three classes, positive or negative or neutral. We have considered the positive and negative opinions. Given a specific topic, by extracting the users who have posted such objects, and by extracting the opinions for each user we present an opinion mining based approach, named Opinion Based Influence Network (OBIN), to compute the popularity of topic and discover the community preference from the friendship network generated from the opinion mining and generate an influence network to maximize the influence spread. In this thesis, we have combined data mining techniques with information retrieval to extract relevant data, and natural language processing to analyze users' opinions. Furthermore we show that, influence spread under the new OBIN model cannot be solved with good approximation guarantee using existing methods, such as 'Lazy Forward' of Leskovec et al. (2007). This is mainly because existing works assume that the probability of a user performing an action is given and the influence spread increases if more of its neighbours perform the same action. However in our approach, this is not that case as the influence spread decreases if it performs actions for other products or against the targeted product. We conduct experiments using real-world datasets collected from Facebook.com to evaluate our approach. In the remaining of Chapter 1 we provide a brief description of social network properties, applications of social interest mining, data mining background, mining popularity and community preference, challenges over mining social networks, problem addressed and contribution for the thesis.

1.1 SOCIAL NETWORK DATA

A social network framework is represented as a graph $G = (V, E)$, where V is the set of nodes with each representing user or a customer such that $v_i, v_j \epsilon V \ and \ i, j = 1,2, \dots, N$, and E is the set of edges between nodes v_i and v_j representing a specific type of interactions between the nodes. The interactions between the users v_i and v_j can be:

1) **Explicit:** Users declare explicitly their friends or connections such as they "join" a group, "like" a page, "follow" a user or topic, accept a "friendship" request etc. These links may be incomplete and not describe all of the relationships in the network.
2) **Implicit:** Links can be identified from user's activities by analyzing broad and repeated interactions between users such as voting, sharing, bookmarking, tagging, or commenting items from a specific user or a set of users. These kinds of links also can be identified from user's similarity by using a predictive model for advertising to analyze user's visits to social network pages.

Let us consider the following social network data tables. Table 1 consists of list of users in a social network and Table 2 shows friendship relationship among these users.

User id	**Name**	**Age**	**Sex**	**Location**
519	Diana	23	F	Montreal
223	Chris	22	M	Windsor
103	Peter	45	M	Toronto
456	John	28	M	London

Table 1 User information table

User id	**Friend id**	**Date Created**
519	456	12-Mar-2007
456	103	22-Apr-2009
519	223	05-Jun-2011
223	456	02-Dec-2010

Table 2 User relationship table

Based on the data of table 1 and table 2, a social network graph can be generated as shown in figure 1.

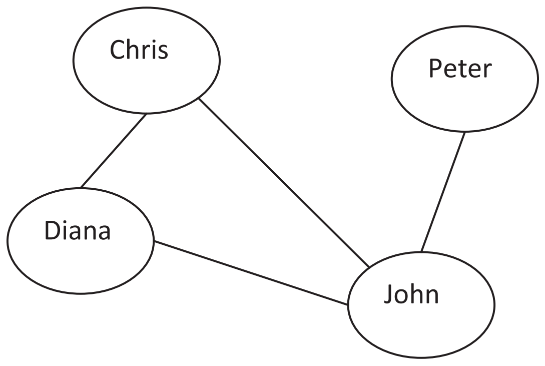
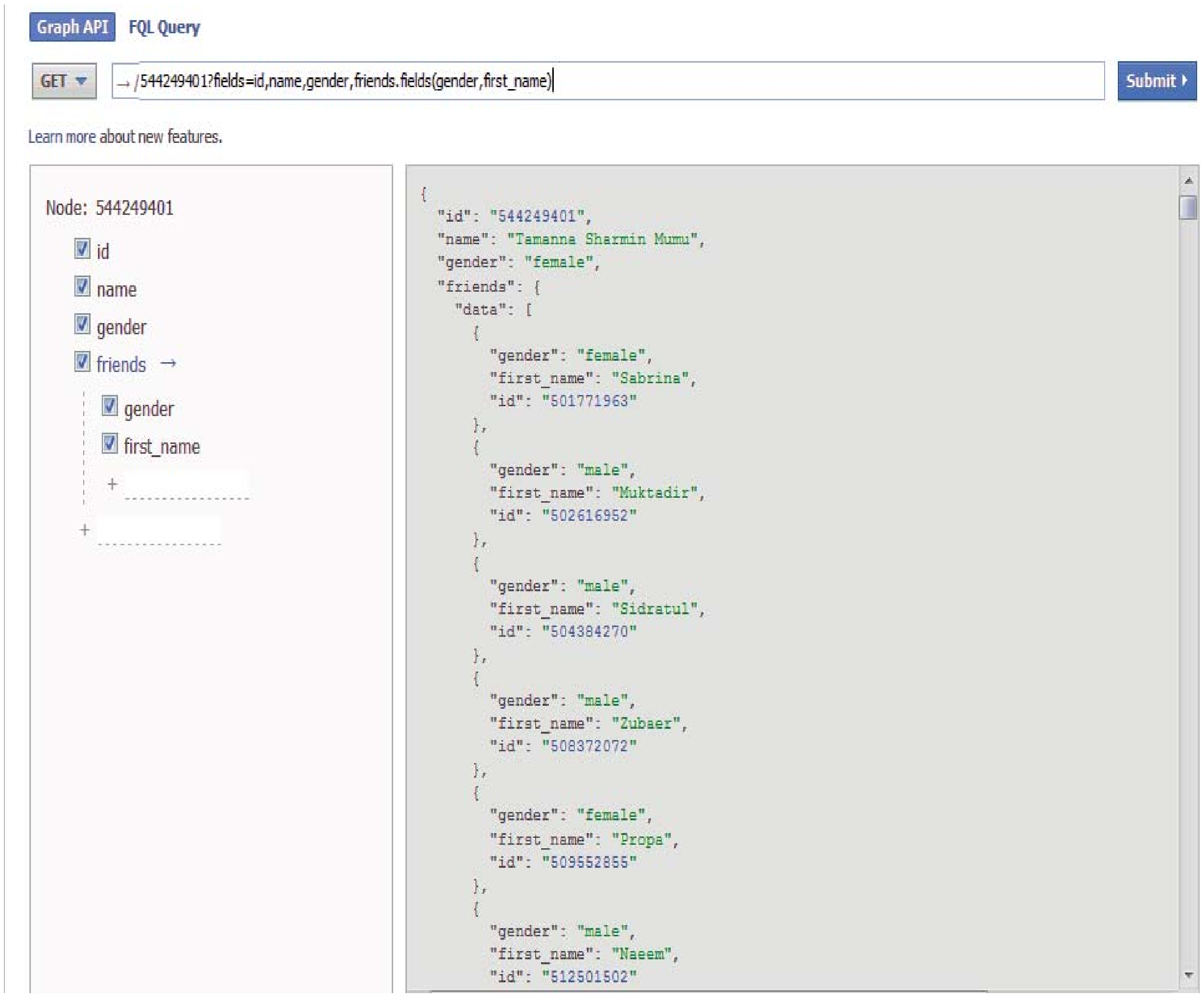


Figure 1 Social network graph generated from Table 1 & 2

Figure 1 represents a graph *G*(*V*,*E*), *V* is the set of users (or vertices) in the social network , i.e. *V*={Diana, Chris, Peter, John}. And *E* is the set of all friendship links (or edges), i.e. *E*={(Diana, Chris),(Diana,John),(Chris,John),(John,Peter)}.

Let us consider the following social network data extracted from social graph Facebook.com.



```
/544249401?fields=id,name,gender,friends.fields(gender,first_name)
```

```
Node: 544249401
  id
  name
  gender
  friends →
    gender
    first_name
```

```
{
  "id": "544249401",
  "name": "Tamanna Sharmin Mumu",
  "gender": "female",
  "friends": {
    "data": [
      {
        "gender": "female",
        "first_name": "Sabrina",
        "id": "501771963"
      },
      {
        "gender": "male",
        "first_name": "Muktadir",
        "id": "502616952"
      },
      {
        "gender": "male",
        "first_name": "Sidratul",
        "id": "504384270"
      },
      {
        "gender": "male",
        "first_name": "Zubaer",
        "id": "508372072"
      },
      {
        "gender": "female",
        "first_name": "Propa",
        "id": "509552855"
      },
      {
        "gender": "male",
        "first_name": "Naeem",
        "id": "512501502"
```

Figure 2 Extracted Social Information of a sample user (id: 544249401)

Here in figure 2,

For a sample node, we have 4 fields: $id, name, gender, friends$

From his friends list, we have 3 fields: $id, first\_name, gender$

For example,$node[id] = 544249401,\ node[name] = "Tamanna\ Sharmin\ Mumu"$, $node[gender] = "female"$

$node[friends] = data[array]$

$data[0][id] = 501771963, data[0][firs\_name] = “Sabrina”$,

$data[0][gender] = “female”$

$data[1][id] = 502616952, data[1][first\_name] = “Muktadir”$,

$data[1][gender] = “male”$

A typical social network follows certain properties:

1) *Power-law degree distributions* or exponential form (Faloutsos et al., 1999). The degree of a vertex is the number of other vertices to which it is connected. For example, the highest degree nodes are called “hubs”, and the major hubs are closely followed by smaller ones, and these ones are followed by other nodes with a much smaller degree, and so on.
2) Have *small diameter.* The diameter is defined as maximum distance between any two nodes. And the distance is measured as the minimum number of edges that must be traversed on the path from one node to another.
3) *Small world effect* (Watts and Strogatz, 1998) i.e., the average distance between vertices in a network is short. For example, how quickly one can get from one “end” of the graph to another.
4) *Clustering or network transitivity* i.e., a prediction that two vertices that are both neighbours of the same third vertex, have a keen probability of also being neighbours of one another(Girvan and Newman, 2002). For example, two of one’s friends will have a greater probability of knowing one another than two people chosen at random from the network.
5) Have *community structure* (Girvan and Newman, 2002) i.e., a group of nodes with more and/or better interactions amongst its members than between its members and the remainder of the network. The communities themselves also

connect with each other to form metacommunities, and those metacommunities are themselves connected together, and so on.

Some main types of large-scale social networks that researchers have used for research in mining social network are listed below:

1) **Friendship Network:**

Friendship network records who is friend to whom relationship among nodes. Examples of friendship networks include Facebook (www.facebook.com), MySapce (www.myspace.com), Twitter (www.twitter.com) etc. Interest mining, for example, can be applied in this area.

2) **Collaboration Network:**

Collaboration Network records who works with whom in a specific topic. Co-authorships among scientists, is an example of collaboration network. DBLP is an example of collaboration network. Expert finding method, for example, can be applied in this area (Craswell et al., 2001).

3) **Trust Network**

Trust network is a social network where both positive (e.g. *likes*) and negative (e.g. *dislikes*) types of links or edges are available. It is represented by directed graphs. Wikipedia is a good example of trust network. Trust computation or influence measurement approaches, for example, can be applied in this area (Ziegler and Lausen, 2005).

4) **Communication Network**

Communication network models the “who-talks-to-whom”, “who-emails-whom”, or “who-sell-whom” structure of social network. Such networks can be constructed from the logs of e-mail or from phone call records. ENRON dataset are an example of communication network data (Bird et al., 2006).

1.2 BUSINESS APPLICATIONS OF MINING SOCIAL INTEREST

The main way of raising the business of an organization is marketing. The traditional approach of doing marketing has been to deal with customer as individuals or to group them into segments with certain properties. These segments of customers can be referred to as “communities”. Social networks have the property of community structure. An organization can treat these communities as groups of customers. While traditional customer segmentation methods to partition a customer base are still applicable and widely used, considering communities extracted from social graphs and monitoring the aggregate trends and opinions discovered by these communities has shown its potential for a number of business applications such as marketing intelligence and competitive intelligence. This task includes identifying influential posts, influential persons and services, users' opinions analysis, and hence community detection based on shared interests. The extracted communities are interpreted as organizational units in social networks. Social network also share data with third parties for advertising purposes, and furthermore social networks also provide open APIs that allow third parties to create applications that access user profile and/or their friend's profile. Companies themselves can use social network mining to detect customers likely to purchase services that they do not intend to pay.

Some main business applications of post as well as user's interest mining are:

1) Online marketplace combines explicit community feedback that can be used effectively to compute reputation scores. For example, it has been observed that customers pay a remarkable premium for buying items from high-reputation sellers, increasing these sellers' revenue, visibility and motivation to keep high reputation scores by effectively delivering what they promise.
2) Identifying groups of customers with similar interests that supports to set up efficient recommendation systems that better guide customers through the list of items of the retailers and improve the business opportunities.
3) Delivery of products and services can be effectively done by collaborating with customers, forecasting, and creation and management of production schedules. These targets can be achieved by using insights obtained from mining customer social network data.
4) Popular search engines try to exploit as much context as possible from the query to provide relevant results such as the identities of the people executing the search as well as their connections. For example, Google has "*result from your social*

circle" feature to the search results which may have a positive impact on knowledge intensive industries.

5) In telecommunication and other industries that have rewards program for customer loyalty, community structure provides a significant role in identifying target groups and allocating policies for such rewards.
6) Security consulting companies or governments fighting criminal or terrorist organizations identify communities and network structure from social networks based on the posts and opinions published by the members of those communities.
7) The field of journalism and intelligence can have extreme help from community structure of social networks. For example, Krebs (2002) described how to mine known relationships between Al-Qaeda operative and discovering communities in that network. Identifying communities and monitoring network evolution can also be used to detect fraud.
8) Discovering positive and negative user opinions can help to assess product and service demand, tackle crisis management, foster reputation online, etc.

Finally, discovering and mining popularity and community preferences is crucial not just for offering advertising and new services, but also for growing the networks through friend suggestions and link prediction to the user to generate link recommendations for service recommendations. Link prediction is also useful to predict customer behaviour in propagating information and adopting new services.

1.3 DATA MINING

Data mining, also known as knowledge discovery, is the process of extracting interesting knowledge from large amounts of data (Han and Kamber, 2006). This large amount of data can be stored in any kind of repository such as relational databases, data warehouses, transactional databases, advanced database systems, flat files, and the World Wide Web. Data mining tasks extract interesting knowledge, regularities, patterns or high-level information from the repository and that information can be viewed or browsed from different angles. This discovered knowledge is then applied to decision making, process control, information management, prediction, and query processing (Han and Kamber, 2006).

Nowadays data mining is a very important and popular task in business applications. In business applications data mining techniques have been successfully employed for *direct marketing* i.e., the decision of whether or not to market to a particular person is based on their characteristics (Richardson and Domingos, 2002). Data mining allows an organization to ask of its data complex questions such as “what has been going on in the organization?” or “what is going to be happened next and how to profit?” the answers to first question can be provided by the data warehouse and multidimensional database systems (OLAP) that allow to browse and visualize the data easily from various perspectives (Han and Kamber, 2006). The answer to the second question can be provided by data mining tools built on classification, clustering and association rule mining. Algorithms from different research areas such as statistics, machine learning, pattern recognitions, data visualization, information retrieval, image and signal processing, and spatial data analysis can also be embedded with data mining algorithms to improve the performance of mining process.

1.3.1 Data Mining Approaches

Data mining tasks include:

1) **Classification** – a process to find the common properties among a set of objects in a database and classifies data records into different classes according to a classification model (a set of rules defined on the attributes of the data record). The objective is to first analyze the data and develop an accurate model for each class using the features (attributes) available in the data record, and then use those models to classify future data record in the database. For example, applications include medical diagnosis, performance prediction in an organization, selective marketing. Some example classification algorithms include nearest neighbours (K-NN) (Coverand Hart, 1967), Naïve Bayes classifier (McCallum et al., 1998), Support Vector Machine (SVM) (Cortes and Vapnik, 1995), decision trees (Quinlan,1986). For example, a data sample is described by the attributes age, income, student, and credit_rating in the table 3 that are called independent attributes as well as the attribute Buy_laptop which is used to classes of the data records. The class label attribute Buy_laptop is called dependent attribute and has

two decisions {yes, no}.The goal of any classification algorithm is to take training data set as input and produce a classification model (rules based on independent attributes) that place each data record in one of the two label classes of "yes" or "no" for the dependent attribute Buy_laptop. The model which is defined during training is then used to classify a new record of which the class or value for dependent attribute is unknown.

Age	**Income**	**Student**	**Credit_rating**	**Buy_laptop?**
18	Medium	Yes	Fair	Yes
20	Low	Yes	Fair	Yes
19	High	No	Good	Yes
12	Low	Yes	Unknown	No

Table 3 Example Training Data for classification

From table 3, data set is $Age \geq 18 \wedge student \wedge Credit\_rating = Fair \Rightarrow Buy\_laptop = Yes, Age < 18 \wedge Credit\_rating = Fair \Rightarrow Buy\_laptop = No.$

2) **Clustering** – this process is also known as unsupervised and unlabeled classification. The process is a measure of similarity between objects under consideration and combine similar objects into the same cluster while keeping dissimilar objects in different clusters according to a clustering algorithm. The process decomposes a large scale system into smaller components. Some clustering techniques include:
    a. Partitioning methods such as K-means algorithms (MacQueen et al., 1967). The algorithm consists of simply starting with k groups each of which consists of a single random point, and thereafter adding each new point to the group whose mean the new point is nearest. After a point is added to a group, the mean of that group is adjusted in order to take account of the new point. Thus at each stage the k-means are the means of the groups they represent (hence the term k-means).
    b. Hierarchical methods such as *agglomerative approach* where each object is placed in its own cluster and then merges these atomic clusters into larger clusters until all of the objects are in a single cluster, or *divisive*

approach where all objects are placed in one cluster and then subdivides the cluster into smaller pieces until each object forms a cluster on its own (Hastie et al., 2001)

c. Density-based methods, finds non-linear shapes structure based on the density. The method aims at identifying clusters as areas of high-point density that are separated by areas of low-point density and thus can be arbitrarily shaped in the data space (Kriegel et al., 2011)

d. Grid-based methods, cluster data elements of a data stream. Initially, the multidimensional data space of a data stream is partitioned into a set of mutually exclusive equal-size initial cells. When the support of a cell becomes high enough, the cell is dynamically divided into two mutually exclusive intermediate cells based on its distribution statistics (Park and Lee, 2004)

For example, using clustering technique in web mining Figure 3 shows "automatic storage of emails falling within a certain cluster based on email contents and senders".

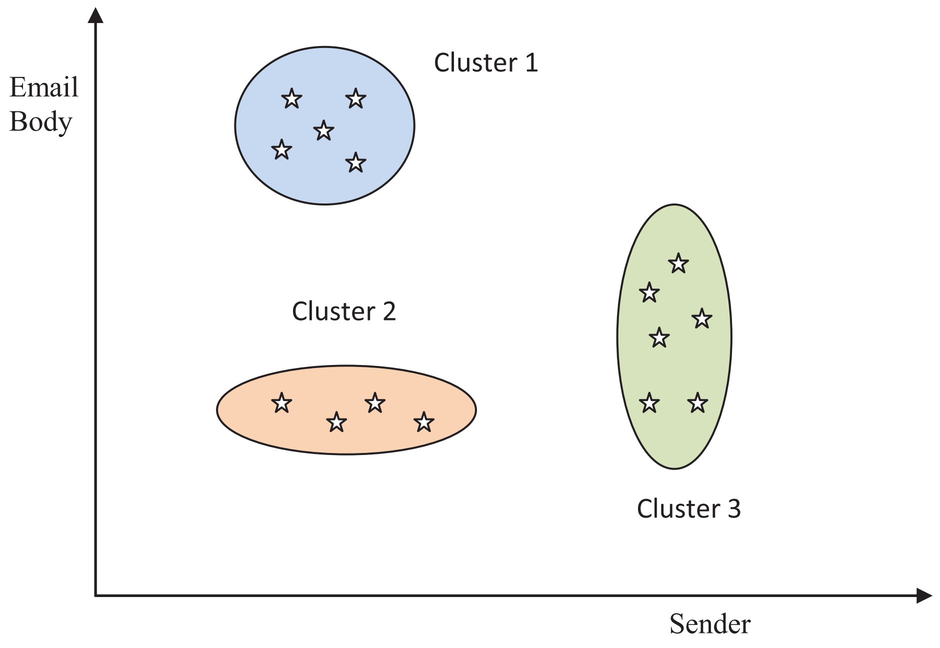


Figure 3 Clustering Example

3) **Association rule mining** – a process of finding rules from a given set of records to compute the simultaneous occurrences of various data records. Association rule mining is generally applied to databases of transactions where each transaction (record) consists of a set of items (attributes). The task is to discover all associations and correlations among data items (attributes) where the presence of one set of items in a transaction implies the presence of other items satisfying some minimum support and minimum confidence constraints (Agrawal and Srikant, 1994).

Let I = {i_1, i_2, …, i_m} be a set of items. D = a set of transactions where each transaction T is a set of items such that T ⊆ I. We can say that a transaction T contains X (a set of some items in I) if X ⊆ T. Then X => Y is an association rule where X ⊆I, Y ⊆ I, and X∩Y=Φ, and X is called antecedent and Y is called consequent.

Confidence – the probability that if the antecedent is true, then consequent will be true. Confidence = $\frac{|Rule|}{|antecedent|}$

Support – the number of records in the database that the rule applies to. Support = $\frac{|Rule|}{|Total|}$

The task of association rule mining is done in two phases. In the first phase, frequent patterns (FP) (set of attributes) that have occurred frequently not less than Minimum Support times are computed. Then, in the second phase, association rules that have confidence not less than Minimum Confidence are computed from generated frequent patterns.

For example, Table 4 describes a transaction set indicating the purchase history of customers. By analysing the transaction table rules are generated from all **frequent patterns** of the transaction data, and calculate their support (how often the rule apply) and confidence (how often is the rule correct).

Transaction ID	**Purchased Items**
1	Milk, Bread
2	Milk, Bread, Juice
3	Milk, Juice, Tea
4	Juice, Bread, Egg, Tea

Table 4 Example of Frequent patterns from Transaction table

In the above example, frequently occurred items milk, juice, and bread may lead to find association rules Milk => Bread, which means that the customers who purchase milk and may usually purchase bread. Here {Milk, Bread}, {Milk, Bread, Juice}, {Milk, Juice, Tea}, {Juice, Bread, Egg, Tea} etc., are called itemsets.

Suppose, Milk = 30 = number of transactions with Milk, Bread = 40 = number of transactions with bread, and Both = 20 = number of transaction with both milk and bread, Total = 100 = number of transactions in database

So, confidence = 20/30 = 66.67% and support = 20/100 = 20%

Rules that satisfy user-specified minimum support are called frequent items, and if the confidence is greater than a user-specified minimum confidence then we say the rule is accurate. An example of frequent pattern mining algorithm is Apriori Algorithm (Agrawal and Srikant, 1994), as described below:

4) **Apriori algorithm** - The algorithm finds the frequent itemsets and association rule in a transaction database. Apriori algorithm generates candidate itemsets by “apriori-join” and scanning the database to count the support for each candidate. The large itemset will be the itemsets whose support count is equal to or greater than a predefined Minimum Support and considered as frequent itemset. Given a transaction database in Table 5 as an example, it is known that items *Milk, Bread and Juice* appear in transaction with id 1. The task is to find all frequent itemsets whose support frequencies are equal to or greater than a predefined minimum support.

Transaction ID	Items
1	*MILK, BREAD, JUICE*
2	*BREAD, TEA*
3	*BREAD, EGG*
4	*MILK, BREAD, TEA*

Table 5 Example of Transaction database

For instance, a given minimum support is 50%, i.e., all itemsets that appear in two or more than two transactions need to be found as frequent or large itemsets. The Apriori algorithm will first find frequent 1-itemset. *MILK, BREAD, EGG, TEA*, *JUICE* are candidate 1-itemset. From scanning the database as shown in Table 5, *MILK* appears in transactions 1 and 4. Its support count is 2. *BREAD* appears in all 4 transactions, so support count is 4. *TEA* appears in transaction 2 and 4, so support count is 2. *EGG* and *JUICE* only appear in one transaction. Therefore, the large itemsets are *MILK, BREAD, TEA*. Next, candidate 2-itemsets need to be generated by applying an apriori-gen join. The apriori-gen join of large itemset L_i with L_i joins every itemset k of first L_i with every itemset n of second L_i where n > k and first (I-1) members of itemsets k and n are the same. In this example, *MILK* will join *BREAD* and *TEA*, *BREAD* will join *TEA*, but *MILK* will not join *MILK*, and *BREAD* will not join *BREAD*. Candidate 2-itemsets are *MILK-BREAD*, *MILK-TEA* and *BREAD-TEA*. Support count of these three candidate 2-itemsets need to be checked by scanning the transaction database. *MILK-BREAD* and *BREAD-TEA* are large 2-itemsets, since their support counts are 2. Candidate 3-itemsets will be generated by large 2-itemsets that is *MILK-BREAD-TEA*. The Support count of *MILK-BREAD-TEA* is 1 which is less than minimum support. Therefore, there is no large 3-itemsets. The algorithm will terminate, since the large itemset is an empty set.

5) **Sequential pattern mining** – A sequence database stores a number of records where all records are sequences of ordered events with or without time-stamp. Sequential pattern mining finds frequent subsequences as patterns in the sequence database (Ezeife and Mabroukeh, 2010).

Example – let us consider a sequence database stores customer transactions for each customer in a grocery store every week. These sequences of customer transactions can be represented as records [Tid, <ordered sequence events>], where each sequence event is a item set such as bread, milk, juice, sugar.

[T1, <(bread, milk), (bread, milk, sugar), (milk), (tea, sugar)>] is a four weeks transaction of one customer.

[T2, <(bread), (sugar, tea)>] is a two weeks transaction of another customer.

So records in the database may vary in length and each event can have one or more items in the set. A sequential pattern mining algorithm mines the sequence database looking for frequent patterns that can be used later to find association rules.

1.3.2 Web Mining

Web is a source of highly dynamic and rich collection of information that poses great challenges for knowledge discovery. Web mining tasks are classified into three categories (Cooley et al., 1997):

1) **Web content mining** – web contents involve text, images, audio, video, structured records etc. Web content mining is a process of extracting useful information from web pages. Web content mining applications include identify a specific topic represented by a web document, categorize web documents, find similar web pages located in different web servers, etc. WEBOMINER is an example of web content mining tools (Ezeife and Mutsuddy, 2012).
2) **Web structure mining** – is the process of discovering web structure information from the web document such as links between references and referents on the Web. Mining task can be applied either at the document level or at the hyperlink level to find links directed into and out of contents on the web. This inward and outward links represents the richness or importance to which the content is to a particular topic. Web structure mining application include classify web pages, ranking on web pages, create similarity measures between documents, the authority of a page on a topic, identifying web communities, etc. (Kadri and Ezeife, 2011).
3) **Web usage mining** – is a process of discovering useful patterns from web usage log data. Data set can be collected from server access logs, client side cookies,

user profiles, meta data such as page attributes or content attributes, etc., where ordered sequences of events in the sequence database are composed of single items and not sets of items, with the assumption that a web user can physically access only one web page at a time at any given point in time (Ezeife et al. 2005, Ezeife and Mabroukeh 2010). Applications of web usage mining include web crawler detection and filtering, web transaction identification, path and usage pattern discovery, web content personalization, prefetching and caching, e-commerce, and business intelligence (Facca and Lanzi, 2005).

1.4 MINING POPULARITY AND COMMUNITY PREFERENCE

The idea of popularity in social networks is when users see their social contacts performing an action, they may decide to perform the action themselves, or they may express their own opinion on that action. Influence to response to the action may come from outside the social network, or because the action is popular, or by the social contacts in the network. Due to the huge usage and rich personal information available on social media websites, business organizations or public figures have now been increasingly willing and active in maintaining pages on those websites to interact with online users, attracting a large number of fans, followers, or customers by posting interesting posts on objects such as topics or products. The popularity of that object can be discovered by analyzing the responses/feedbacks (e.g., likes/dislikes, comments/reviews) given by the users of social networks. A bulk of research has been focused on such response analysis. All of them have some general tasks: (1) identifying features of the product that users have expressed their opinions on, (2) for each feature, identifying review sentences that give positive or negative opinions, and (3) producing a summary using the discovered information. The summary helps to build a trust network and the community can be detected through surfing the trust network. Several research (Pang et al. 2002, Turney 2002, Agrawal et al. 2003, Dave et al. 2003, Hu and Liu 2004, Mishne and Glance 2006, Nigam and Hurst 2006, Ding et al. 2008, Gomez et al. 2008, Li et al. 2008, Tang et al. 2009) have been done for analyzing users responses on interest networks (i.e., user-service interaction), but there has been no previous work studying user responses in friendship networks (i.e., user-user connections). Such user-service connections are domain specific and product-feature oriented. For example, these networks may be weblogs, newsgroups, bookmarks, question/answers, movie/product review domains, etc. Due to the emerging popularity of friendship networks, discovering common interests shared by users is a fundamental problem in such friendship networks since it is the bread-and-butter function of building user communities of the same interests, finding the topic experts in different subjects, identifying hot social topics, and recommending personalized relevant contents. An efficient and scalable solution is crucial to the growth of social communities.

1.5 CHALLENGES OF MINING SOCIAL NETWORKS

A good algorithm in social network analysis should address two key problems: which groups of vertices are associated with each other (**linked data**) and when does the community structure change and how to quantify the change (**network dynamics**). Some major challenges of mining social networks are listed below:

1) Defining interactions among users where users have profiles holding heterogeneous information and complex ways of interactions between users and between user and system.
2) Scalability while dealing with real social networks with millions of users since real social networks are getting bigger. The challenge here is to develop efficient and scalable mining techniques that can process large amount of real data in shortest possible amount of time and also produce models with high accuracy.
3) Multi-aspect i.e., social influences are associated with different topics. For example, $user1$ may have high influence to $user2$ on $topic1$, but $user2$ can have a higher influence to $user1$ on $topic2$. So the challenge is to be able to differentiate those influences from multiple aspects.
4) In general, popularity of a topic/product somehow depends on how fast the posted content spreads quickly in a community (a group of users with similar interest). The spreading processes also rely on the nature of the $spreader$ and the $audience$, the structure of the $network$ through which the information is surfing, and the nature of the $content$ itself.

5) The crucial characteristic of popularity measure is the overall opinion towards the subject matter, for example, whether a product review is positive or negative. Sentiment analysis tool should efficiently extract opinions from unstructured human-authored documents.

A good algorithm should also consider some additional problems that social network data may suffer including duplicate nodes e.g., a user has two email addresses, inactive nodes e.g., the user who does not remove his profile but does not use it any more, artificial nodes e.g., automated agents. To overcome these additional problems, efficient data cleaning is also needed for social network analysis.

1.6 THESIS PROBLEM AND CONTRIBUTIONS

1.6.1 Problem Addressed

The effectiveness of an influential communication often depends on the nature (positive, negative, neutral) of responses from recipients. In a social network, people can share their interests by posting objects on their social profile/fan pages, where other people may give their opinions by showing agreements or disagreements.

All the previous works done in the area of opinion mining are through interest networks (e.g., Amazon.com) that are product specific. Our proposed opinion mining approach is the extension of OpinionMiner system (Jin et al., 2009) which will be applicable for friendship networks. In standard influence maximization (IM) systems such as CELF (Leskovec et al., 2007), takes whole social network as input to find influential users as seed set for a specific product (e.g., iPhone) for target marketing (Ahmed and Ezeife, 2013). Table 6 shows the major differences between existing systems and our proposed system.

Existing Systems	**Type of network**	**Size of products/opinions**	**Measurements**	**Limitations**
General IM CELF (Leskovec et al., 2007)	Social network (user – user network) (e.g., Facebook.com)	All users post about multiple products on multiple posts	Probability of users performing actions after an influential user	1. Does not consider ‘opinion’ of users 2. Not product

				specific 3. Not scalable
General IM T-IK (Ahmed and Ezeife, 2013)	Social network (Trust network) (e.g., Wikipedia.com)	All users post about multiple products on multiple posts	Probability of users performing or not performing actions (+/-) by influential users	1. Does not consider 'opinion' of users 2. Positive/negative influences are explicitly given 3. Not product specific 4. Not scalable
General Opinion Mining OpinionMiner (Jin et al., 2009)	Domain specific websites (user – service network) (e.g., Amazon.com)	One user posts about one product on single post page	Comments and ratings on the product	1. Predefined product features are given 2. Ignore opinions about different products

Table 6 Differences between issues handling by existing systems and proposed system

OpinionMiner takes product features as input parameters. Features are domain-dependant, a set of features must be prepared. For example, if the system wants to extract opinion about Digital Cameras, prepare features as cover color, pixel ratio, zoom, memory, etc., and tag the reviews accordingly. The system mines opinions for reviews that have predefined product features. Moreover, the system does not consider opinions expressed on irrelevant product entities. For example, Samsung Galaxy page containing any review about iPhone will not be considered as the opinion for Galaxy.

This is also to be noted that all the previous works primarily consider one specific feature of the post popularity such as only sentiment of comments (Nigam and Hurst 2006, Ding et al. 2008, Jin et al. 2009, Dave et al. 2003, Mishne and Glance 2006, Gomez et al. 2008) only topic propagation i.e., who spreads the topic to others (Tang et al. 2009, Agrawal et al. 2003), or only rating on topic post i.e., thumbs-up/thumbs-down (Pang et al. 2002, Li et al. 2008, Turney 2002). However, in friendship networks, to analyze the popularity of a post and users, all kinds of explicit and implicit opinions need to be aggregated.

The main limitation with general IM systems like CELF is that they are not effectively product-specific because of the need to first search large social networks data for multiple product opinions. For example, the existing systems may find a very influential user to his friends over network for various products and topics, but for a specific product such as iPhone, he may not be influential at all. So considering those users as influential for a product reduces the accuracy and efficiency of such general IM algorithms.

Motivated by the issues described above, the problem we tackle is as follows:
Problem Definition – Build an influential network (IN) generation model for influence maximization based on mining users' posts and opinions (positive or negative) on a specific product and relationships from a friendship network graph $G(V, E)$ where every edge $e_{ij} \epsilon E$ connects nodes v_i and v_j ($v_i, v_j \epsilon V and\ i, j = 1,2, \dots, N$) and indicates v_i and v_j have relationships on a specific product. Also, measure influence acceptance score of each node v_i in V for a product and remove nodes that are below certain threshold before applying IM algorithm on that pruned friendship network to more effectively and efficiently compute a product-specific IM.

To solve the above problem, thesis contributions are:

1.6.2 Thesis Contribution

1. First, to consider opinions on friendship network for specific product
    a. A new influence network (IN) generation model is proposed, called OBIN, Opinion Based Influence Network.
        i. OBIN considers multiple posts by multiple users on a specific product
        ii. OBIN aggregates all kinds of users' explicit/implicit opinions (e.g., likes/dislikes, re-shares, positive/negative comments)
        iii. OBIN discovers users-users relationships
2. We propose a local search algorithm, called TPD (Topic-Post Distribution) based on network pruning strategy to discover ranked list of users and opinions, and to classify relevant and irrelevant users for specific product.

3. We propose PCP-Miner (Post-Comment Polarity Miner) algorithm, to compute the popularity scores of users by extending OpinionMiner (Jin et al., 2009) with Apriori frequent pattern mining, and to compute the influence scores of users to discover user-user relationships.
4. Experimental analysis shows that, OBIN gives relevant influential users for a product more efficiently, and the influence spread over the network is occurred more effectively than standard IM algorithms.

1.7 THESIS OUTLINE

In Chapter 2 we provide a detailed related work on opinion mining for large scale networks and also discuss limitations of these works and motivation for the thesis. In Chapter 3 we provide a proposed solution framework to solve popularity measure in friendship network with running examples and complexity analysis. In Chapter 4 we provide various experimental results including comparisons between the existing and the proposed approach. Finally, Chapter 5 provides some concluding remarks.

CHAPTER 2

RELATED WORKS

Social network analysis often focuses on macro-level research such as degree distribution, diameter, clustering coefficient, community detections, small-world effects, preferential attachments, etc. (Tang et al., 2009). Recently, many researchers have analyzed social network data to find patterns of popularity or influence in various domains. Such domains include blogging (e.g., Slashdot.org) and micro-blogging (e.g., Twitter.com) domains, bookmarking domains (e.g., Digg.com), co-authorship domains (e.g., Academia.edu), movie review domains (e.g., IMDb.com), and product review domains (e.g., Amazon.com). Weblog domains define a relationship between the writer of the blog and the readers by publishing short news posts and allowing readers to comment on them. In co-authorship domains, each author is related to some specific topic, there is no random author-topic relation. Movie review domains provide ratings and brief quotes from several reviews and generate an aggregate opinion. Product review domains are dedicated to specific types of products. All the domains are well-structured for a specific topic whereas friendship network is more complex and heterogeneous. Moreover, the great majority of research study only features related to the network itself or simple popularity matrices of the posts (e.g., number of likes/thumbs-up, number of comments), without analyzing the correlation of these aspects with the content of the posts.

Our work in this thesis is motivated by some previous studies of comments in newsgroups (Agrawal et al. 2003), bookmarking domains (Li et al. 2008), co-authorship domains (Tang et al. 2009), product-review domains (Dave et al. 2003, Hu and Liu 2004, Ding et al. 2008, Jin et al. 2009), movie-review domains (Pang et al. 2002), and weblogs (Mishne and Glance 2006, Gomez et al. 2008). In this chapter, we further discuss some of the general IM approaches such as (Leskovec et al. 2007, Ahmed and Ezeife 2013).

2.1 PRODUCT REVIEW DOMAINS

2.1.1 Feature-based Approach

Dave et al. (2003) proposed an opinion extraction and mining method based on features and scoring matrices. This approach takes structured reviews and identifies appropriate features and scoring formula to determine whether reviews are positive or negative. The results perform machine learning method called Transductive learning to classify review sentences from the web. This approach can be summarized using the following steps:

Training a classifier – starting with a portion of web document the following are applied to refine the classifier to classify the sentences mined from broad web searches. Based on the scores, the classifier can determine whether a review sentence is positive or negative.

1. Collect users' text reviews, title, thumbs-up or thumbs-down rating from the large web sites
2. Separate the document into sentences, then split sentences into single-word token.
3. Substitute numerical tokens with $NUMBER$, product's name token with $\_productname$
4. Pass the document sentence by sentence through Lin's MINIPAR linguistic parser to yield part of speech of each word and the relationships between parts of the sentence.
5. Pass the resulted words through WordNet, a database for finding synonyms.
6. Identify negative phrases and mark all words following the phrases as negated.
7. Combine sets of n adjacent tokens into $n-grams$.
8. Count frequencies of the extracted features i.e., the number of times each term occurs, the number of documents each term occurs in, and the number of categories a term occurs in. then set upper and lower limits for each of these measures, constraining the number of features looking for to determine a threshold for the classifier.
9. After selecting a set of features $f_1 \dots f_n$, assign them scores. These scores are used to place the test documents in the set of positive reviews C or negative reviews C.

$$score\ (f_i) = \frac{p(f_i|C) - p(f_i\ |C')}{p(f_i|C) + p(f_i\ |C')}$$

$$p(f_i \mid C) = \text{the normalized term frequency}$$

$$= \text{the number of times a feature } f_i \text{ occurs in C}$$

$$\text{and dividing it by total number of tokens in C}$$

Classifying – if document $d_i = f_1 \dots f_n$ then

$$class\ (d_i) = \begin{cases} C\ \ if\ eval\ (d_i) > 0 \\ C'\ if\ eval\ (d_i) < 0 \end{cases}$$

Where$eval\ (d_i) = \sum_j score\ (f_j)$

The classification result shows reviews under positive opinion reviews or negative opinion reviews.

Example:

At first this approach strip out HTML tags from the document containing reviews. Suppose example reviews are "*This bulky lens of Kodak is not useful for me*", "*The zoom view of Kodak is awesome*", "*I love the pink Kodak color*".

The substitution step converts the sentence to "*This bulky lens of X is not useful for me*", "*The zoom view of X is awesome*" and "*I love the pink X color*".

After parsing, these sentences become: $ulky(Adj){:}\,lens(N){:}\,subj{:}\,useful(Adj), zoom(N){:}\,subj{:}\,awesome(Adj)$, $I(Pro){:}\,love(V){:}\,pink(N){:}\,subj{:}\,color(N)$.

After turning into negation, the negation phrase "not useful" become NOTuseful.

Unigram	Bigram	Trigram
Positive features		
Awesome	I love	The zoom view, the pink color
Negative features		
Bulky	Not useful	

Table 7 n-grams features from extracted reviews

Table 7 shows n-adjacent token into n-grams. Now we can calculate the score for features "$lens$" appears one time in negative feature "$bulky$', "$zoom\ view$" appears one time in positive feature, "$pink\ color$" appears one time in positive feature.

$$Score\ (lens)\ = \frac{0/_4 - 3/_2}{0/_4 + 3/_2} = \frac{-1.5}{1.5} = -1\ ,$$

$Score\ (zoom\ view) = \frac{1/_4 - 0/_2}{1/_4 + 0/_2} = 1$ and similarly $Score\ (pink\ color)\ =\ 1$

Hence, $eval\ (Kodak) = \sum_3 score(f_3) = -1 + 1 + 1 = 1$

So, $class\ (Kodak)\ =\ positive,\ since\ eval\ (Kodak)\ >\ 0$

2.1.2 Opinion Summarization

Hu and Liu (2004) proposed a feature-based summarization FBS method that mine product features from customers' reviews, identifies sentiment opinion, and summarize the results. The inputs to FBS are a product name and an entry web page for all the reviews of the product. FBS method has the following task:

1) **Part-of-Speech Tagging (POS)** – NLProcessor linguistic parser (http://www.infogistics.com/textanalysis.html) is used to parse each review to split text into sentences and to produce the POS tag for each word. Output of the NLProcessor is XML. For example <W C='NN'> means a noun and <NG> means a noun group or noun phrase. Each tagged sentence is saved in the review database.

    Example – suppose a sentence "I am absolutely in awe of this camera". Output of POS steps is

    *<S><NG><W C = 'PRP' L = 'SS' T = 'w' S = 'Y'>**I**</W></NG>*

    *<VG><W C = 'VBP'>**am**</W><W C = 'RB'>**absolutely**</W></VG>*

    *<W C = 'IN'>**in**</W><NG><W C = 'NN'>**awe**</W></NG>*

    *<W C = 'IN'>**of**</W><NG><W C = 'DT'>**this**</W><W C =*

    *'NN'>**camera**</W></NG><W C = '.'>**.**</W></S>*

2) **Opinion Words Extraction**– If a sentence has one or more than one product features and one or more opinion words, then it is called opinion sentence. The opinion words are identified by the following method:

   Example – "*The auto-flash is disgusting and makes the face blur*", here *disgusting* is the effective opinion of *auto-flash*.

   "The picture quality is awesome" and "The application that is used in it is awesome" share the same opinion word awesome, and suppose there are no sentences to talk about picture quality or application. That means these two features are infrequent. In this case, the nearest noun phrases around the opinion word awesome are *picture quality* and *application*.

3) **Opinion Words Orientation**–Words that encode a desirable state (e.g., beautiful, amazing) have a positive orientation, while undesirable state (e.g., ridiculous) have negative orientation. This task has following steps:
   a. Select adjective list from WordNet store them to *seed list*. For example, *great, cool, nice, fantastic* are positive adjectives; and *bad, dull, dumb* are negative adjectives.
   b. In WordNet, adjectives are organized into bipolar cluster. For example the Figure 4 shows bipolar adjective structure for the word 'tiny'

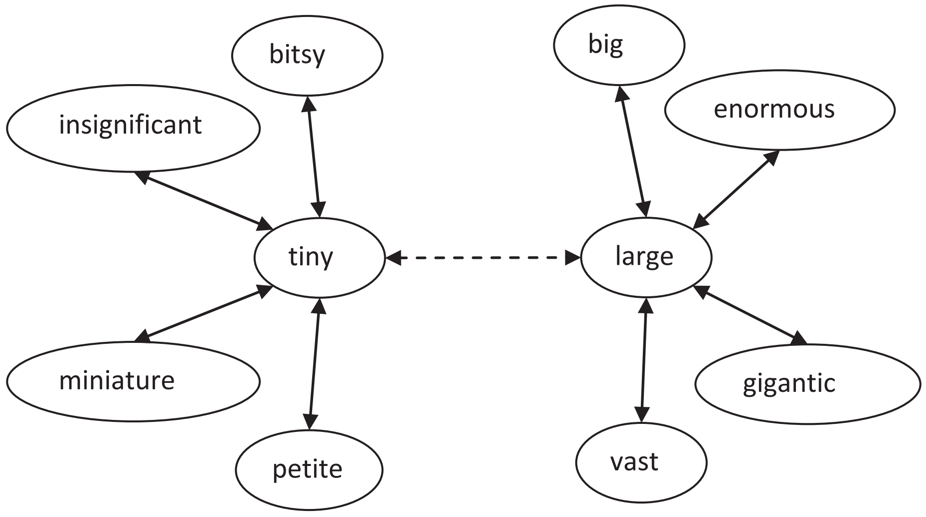


Figure 4 Bipolar adjective structure, $\rightarrow$ = synonym and $\dashrightarrow$ = antonym

4) Opinion Sentence Orientation –identification method of positive or negative sentences has following steps:

-
```
for each opinion sentence $s_i$
    orientation = 0
    for each opinion word w in $s_i$
        orientation += **wordOrientation** ( w , $s_i$ )
    if ( orientation > 0) $s_i$ is positive
    else if (orientation < 0) $s_i$ is negative
    else {for each feature f in $s_i$
        orientation +=**wordOrientation** (f's effective opinion, $s_i$ )
        if ( orientation > 0) $s_i$ is positive
        else if ( orientation < 0) $s_i$ is negative
        else $s_i$'s orientation = $s_{i-1}$' orientation}
```
-
```
**wordOrientation** (word, sentence)
orientation = orientation of word in seed list
    if ( there is negation word appears closely around word in sentence)
        orientation = opposite (orientation)
```

Example – for the feature "picture" let us take example sentences

- "*overall this is a good camera with a really good picture clearly*". This sentence is determined as s_i *positive* by fulfilling first *if* statement.
- "*the auto and manual along with movie modes are very easy to use, but the software is not intuitive*". The orientation of this sentence is determined by the last *else* statement and average orientation of effective features are used, and the average orientation is *positive*.

2.1.3 Feature-Entity Based Approach

Jin et al. (2009) also worked similar as Hu and Liu (2004). Jin et al. (2009) have defined four entity types, eight tag sets and four pattern tag sets to the feature-based approach called OpinionMiner.

Components	Physical objects of a product. e.g., LCD, viewfinder or battery of a Camera
Functions	Capabilities provided by the product. e.g., automatic flash or auto focus of a Camera.
Features	Properties of components or functions. e.g., color, size or weight.
Opinions	Thoughts expressed by reviewers on a product features, components or functions.

Table 8 Definitions of Entity Types

Tag Set	**Corresponding Entities**
<PROD_FEAT>	Features
<PROD_PARTS>	Components
<PROD_FUNCTION>	Function
<OPINION_POS_EXP>	Explicit positive opinion
<OPINION_NEG_EXP>	Explicit negative opinion
<OPINION_POS_IMP>	Implicit positive opinion
<OPINION_NEG_IMP>	Implicit negative opinion
<BG>	Background words

Table 9 Basic tag set and corresponding entity

Pattern Tag	**Corresponding Pattern**
<>	Independent entity
<-BOE>	The beginning component of an entity
<-MOE>	The middle component of an entity
<-EOE>	The end of an entity

Table 10 Pattern tag set and corresponding pattern

Each word in the review is represented by hybrid tag combining basic tag set and pattern tag set.

Example: Let us suppose an opinion sentence “I love the ease of transferring the pictures to my computer”.

Hybrid tags:*<BG>***I***</BG><OPINION_POS_EXP>***love***</OPINION_POS_EXP><BG>***the***</BG><PROD_FEAT-BOE>***ease***</PRODUCT_FEAT-BOE><PRODUCT_FEAT-MOE>***of***</PROD_FEAT-MOE><PRODUCT_FEAT-MOE>***transferring***</PROD_FEAT-MOE><PRODUCT_FEAT-MOE>***the***</PRODUCT_FEAT-MOE><PRODUCT_FEAT-EOE>***picture***</PRODUCT_FEAT-EOE><BG>***to***</BG><BG>***my***</BG><BG>***computer***</BG>*

Similar to the bipolar adjective structure represented by Hu and Liu (2004), Jin et al. (2009) also present propagation structure for all entity.

Example: Let us suppose a review sentence is "good picture quality". The propagation structure by Ji et al. (2009) is shown in Figure 5. Using Figure 5, some possible bi-gram can be "decent picture quality", "good image quality" etc.

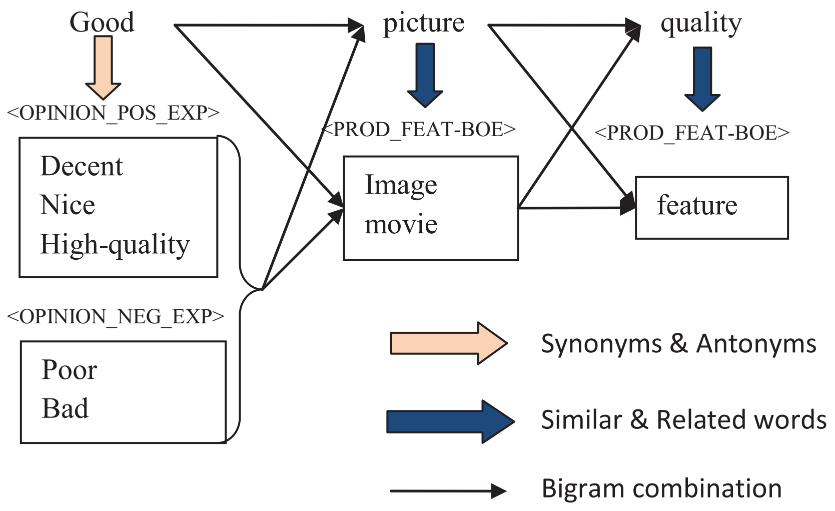


Figure 5 Example of word propagation

2.1.4 Lexicalized HMM Approach

Jin et al. (2009) proposed a bootstrapping approach for HMM as shown in Figure 6. The steps are as follows:

1. Creates two child processes. Master is responsible for co-ordinating the bootstrapping process, extracting and distributing high confidence data to each worker.

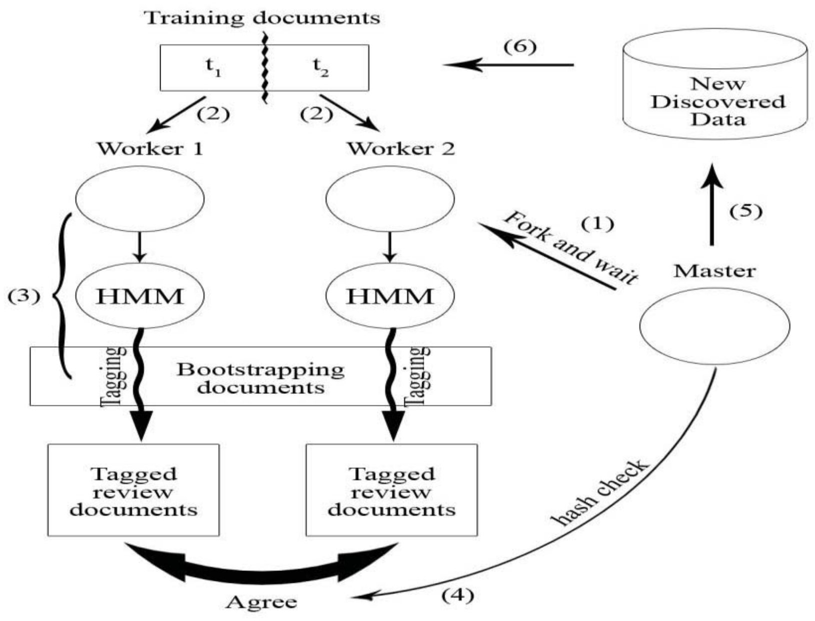


Figure 6 Bootstrapping process by Jin et al. (2009)

2. Training document is divided into two set and each is used as seeds for each worker's HMM.
3. Each worker trains its own HMM classifier based on its own training set, then each worker's trained HMM is used to tag the documents which produces a new set of tagged review documents.
4. After each tagging step, master inspects each sentence tagged by each HMM and only extracts opinion sentences.
5. A hash value is calculated for each extracted opinion sentence and compared with database. If it is a new sentence, store it to the database.
6. Master then randomly splits the newly discovered data from database into two data sets again for training.
7. Bootstrap process is continued until no more data being discovered.

2.1.5. Holistic Lexicon-based Approach

Ding et al. (2008) have proposed a holistic lexicon-based approach called Opinion Observer including an orientation score function and handling the context dependent opinion words. Ding et al. (2008) have used NLProcessor (http://www.infogistics.com/textanalysis.html) to generate part-of-speech (POS) tags and then Opinion Observer is applied to find orientations of opinions expressed on product features. The opinion orientation is identified using the following steps:

1. A positive word is assigned the semantic orientation score of +1 and a negative word is -1. A review sentence may contain opinions on multiple features. For each feature f in the sentence the total score function is computed as:

$$score(f) = \sum_{w_i : w_i \in s \,\cap\, w_i \in V} \frac{w_i \,.\, SO}{dis(w_i, f)},$$

Where $w_i = an\ opinion\ word$
$V = set\ of\ all\ opinion\ words$
$s = sentence\ containing\ f$
$dis\ (w_i, f) = distance\ between\ f\ and\ w_i$
$SO = semantic\ orientation\ of\ word\ w_i$

2. Holistic approach to handle context dependent opinions –

```
if the previous sentence exists and has an opinion then
    if there is not a "However" or "But" word to change the direction of the current
        sentence then
            orientation = the orientation of the last clause of previous sentence
else orientation = opposite orientation of the last clause of previous sentence
else if the next sentence exists and has an opinion then
    if there is a not "However" or "But" word to change the direction of the next
        sentence then
            orientation = the orientation of the first clause of next sentence
        else orientation = opposite orientation of the last clause of next sentence
else orientation = 0
```

Here the variable *orientation* is the opinion score of the current feature.

Example –

- *Intra-sentence conjunction rule* – let us suppose a sentence "*the battery life is very long*" which does not clearly show positive or negative for the word "*long*". Suppose another sentence "*This camera takes great pictures and has a long battery life*" where we can discover "*long*" is a positive for "*battery life*" since it is conjoined with "*great*".
- *Pseudo intra-sentence conjunction rule* – suppose another sentence "*The camera has a long battery life, which is great*". Here "*long*" indicates positive semantic orientation for "*battery life*" though no explicit "*and*" is used.
- *Inter-sentence conjunction rule* – if the above two rules could not decide the opinion orientation then extend the intra-sentence conjunction rule to neighbouring sentences.
- *Synonym and Antonym rule* – if a word is found to be positive then its synonyms are also positive and antonyms are negative. e.g., "*long*" is positive here for '*battery life*", so "*short*" is negative here.

2.2 OPINION MINING IN WEBLOG DOMAINS

2.2.1 Contribution of Comment Contents

Mishne and Glance (2006) have analyzed the relation between the weblog popularity and commenting patterns in it.

1) Comment extraction – Identify the "comment region" which has a sequential pattern within the HTML page.
2) Popularity – To measure weblog popularity, Mishne and Glance (2006) use the number of incoming links and the number of page view. e.g., incoming links can be the Blogpulse index, and page views can be visit counters such as Sitemeter.
3) Some exceptions – Too few comments in high-ranked weblogs due to strict moderation for spam and other form of abuse. Too many comments in low-ranked weblogs due to the usage as a forum to converse and interact by small group of

blogger's friends. Highly commented posts due to highly controversial topics (e.g., politics).

4) Disputative comments – the features used for classification are:
   1. **Frequency counts** – counts of words and word bigrams, counts of a mutually constructed small list of longer phrases, e.g., "I don't think that", "you are wrong" etc.
   2. **Level of subjectivity** – compare the language used in the encyclopaedia entries to the language used in the discussions about these entries, by building two language models for encyclopaedia and discussions. Compare them using a standard corpus divergence metric called log-likelihood proposed by Kilgarriff (2001).
   3. **Length features** – add features for the average sentence length, the average comment length in the thread, and the number of comments in the thread.
   4. **Punctuation** – frequency counts of punctuation symbols and usage of excessive punctuations.
   5. **Polarity** – sentiment analysis is used.
   6. **Referral** – references to previous content by quote, or authors by name. e.g., a direct quote as the first sentence of the comment can be a referral.

Gomez et al. (2008) have shown that to improve the quality and representativity of the generated social influence graph, filter some of the comments according to the three following criteria:

1. Anonymous comments are discarded.
2. Very low quality comments with score -1 are discarded.
3. Filter out self-replies, i.e., the replies often motivated by a forgotten aspect or error fix of the original comment.

2.2.2 Statistical Analysis

According to Gomez et al. (2008), a social network graph $G = (V, E)$ where $(i, j) \in E$ can be represented as undirected dense, undirected sparse, or directed graph based on the comment distribution.

Example: Let n_{ij} = number of times user i comments to user j

- In dense graph, an undirected edge exists between users i and j if either $n_{ij} > 0$ or $n_{ji} > 0$. the weight of that edge $w_{ij} = n_{ij} + n_{ji}$.
- In sparse graph, an undirected edge exists between users i and j if $n_{ij} > 0$ *and* $n_{ji} > 0$. $w_{ij} = \min\{n_{ij}, n_{ji}\}$.
- In directed graph, a directed edge from user i to user j exists if $n_{ij} > 0$ regardless of n_{ji}. $w_{ij} = n_{ij}$.

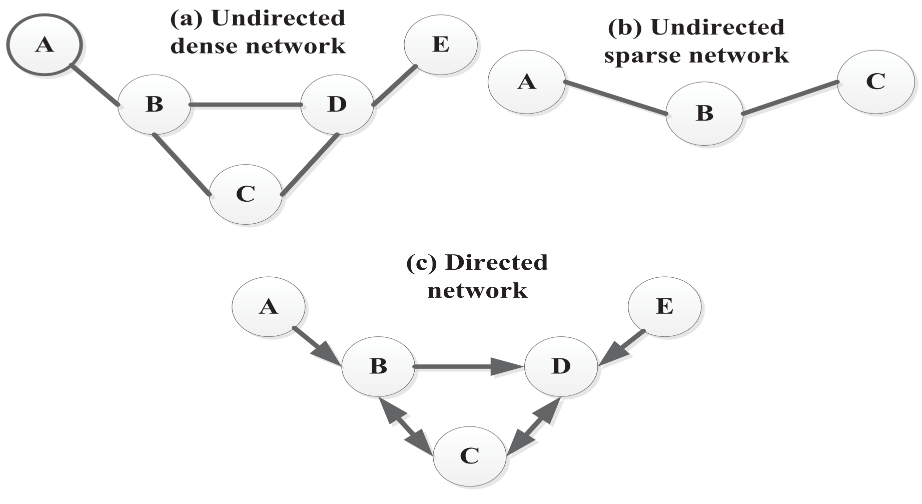


Figure 7 Example of graph generation

The structural properties of the obtained graph based on comments are as follows:

1. Degree distribution – the level of interactions between the users.
2. Small world effect – the average path length between all users in the graph. The maximal distance between two users should be very small.
3. Degree correlation/assortative mixing – detect whether highly connected users are preferentially linked to other highly connected ones or not.
4. Community structure – let λ denote the number of comments, so that users (i, j) who interchange a number of comments $w_{ij} \geq \lambda$ are included in the network, and other connections are discarded. Starting from $\lambda = \lambda_{max}$ and iteratively decreasing it by agglomerative clustering, communities can be obtained.

2.3 USER-GENERATED TAGS IN BOOKMARK DOMAINS

People use tags as a descriptive label to annotate the content that they are interested in and to share with others. The repetitive occurrence of common tags from a set of users represent their common interests. Li et al. (2008) have used vector space model (VSM), association rule mining, and Naive Baise clustering algorithm to develop an architecture for social tag-based interest discovery called $ISID$.

2.3.1 Vector Space Model (VSM) in ISID

Each URL is represented by two vectors such as all tags and all document key-words. A dataset with t terms and d documents is represented by a term-document matrix $A = (a_{ij})$ Each column vector a_j $(1 \leq j \leq d)$ corresponds to a document j. Weight a_{ij} represents the importance of term i in document j. Let f_{ij} is the frequency of term i in document j.

The tf – based weight of term i in document j is $a_{ij}^{tf} = \frac{f_{ij}}{\sqrt{\sum_{k=1}^{t} f_{kj}^2}}$

The $tf \times idf -$ based weight of term i in document j is $a_{ij}^{tf \times idf} = \frac{b_{ij}}{\sqrt{\sum_{k=1}^{t} b_{kj}^2}}$ where

$b_{ij} = f_{ij} . \log(\frac{d}{D_i})$, D_i = number of documents that contain term i, $\log(\frac{d}{D_i}) = idf$.

Example: Table 11 shows resolv.conf file in Linux OS is bookmarked by some users.

URL	http://ka1fsb.home.att.net/resolve.html
Top tf keywords	domain, name, file, resolver, server, conf, network, nameserver, ip, org, ampr
Top $tf \times idf$ keywords	ampr, domain, jnos, nameserver, conf, ka1fsb, resolver, ip, file, name, server
All tags	Linux, howto, network, sysadmin, dns

Table 11 Example of $\boldsymbol{tf}, \boldsymbol{tf} \times \boldsymbol{idf}$ keywords, and tags

2.3.2 Clustering in ISID

In ISID, for each topic (tag set), collect the posts that contain the tag set, and inserts the URLs and the users of the posts into two clusters. A naïve clustering algorithm is used. The output of the clustering algorithm is two collections of clusters identified by topics:

one for URLs, where each cluster contains all the URLs that have been saved with all the tags in the topic of the cluster, and the other for users, where each cluster contains all the users who have been used all the tags in the topic of the cluster.

Example: ISID can provide queries such as:

- For a given topic, list all URLs that contain this topic, i.e., have been tagged with all tags of the topic.
- For a given topic, list all users that are interested in this topic, i.e., have used all tags of the topic.
- For a given tags, list all topics containing the tags.
- For a given URL, list all topics the URL belongs to.
- For a given URL and a topic, list all users that are interested in the topic and have saved the URL.

2.4 OPINION MINING IN CO-AUTHORSHIP DOMAINS

2.4.1 Graphical Probabilistic Model

Tang et al. (2009) have proposed a Topical Factor Graph (TFG) model incorporate all the information into a unified probabilistic model and a method called Topical Affinity Propagation (TAP) for model learning.

TFG Model – Example: Figure 8 shows graphical representation of TFG model. $\{v_1, .., v_4\}$ are nodes in the social network; $\{y_1, \dots, y_4\}$ are hidden vectors defined on all nodes with each element representing which node has the highest probability to influence the corresponding node; $g(.)$ represents a feature function defined on a node; $f(.)$represents a feature function defined on an edge; and $h(.)$ represents a global feature function defined for each node, i.e., $k \in \{1, \dots, N\}$.

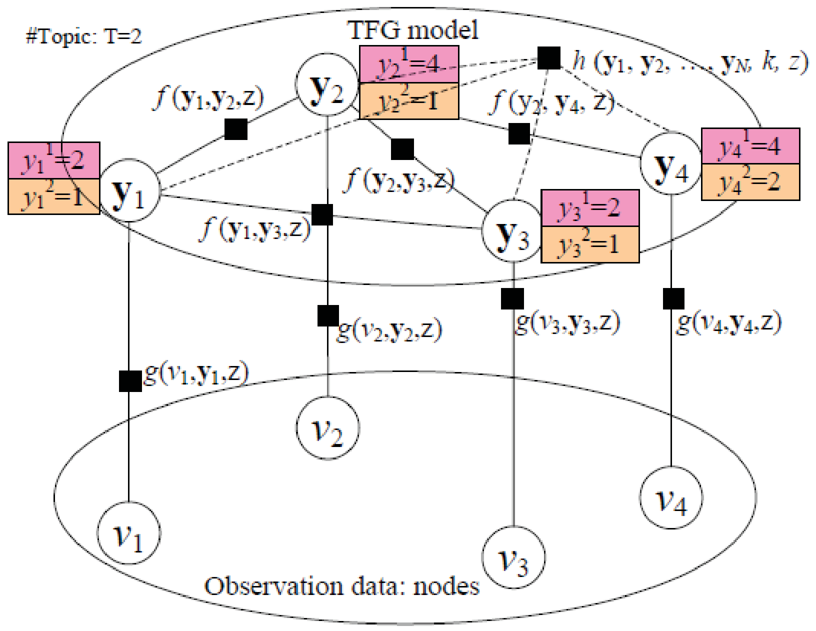


Figure 8 Graphical representation of TFG Model (Tang et al. 2009)

TAP Learning– Example: Tang et al. (2009) have used sum-product algorithm to train TFG model. In sum-product algorithm, messages are passed between nodes and functions by initiating at the leaves. For each nodev_i, once a message has arrived, it computes a message to be sent to its neighbours and wait until the replies have come. Once the replies have arrived again nodev_i computes message to be sent to each neighbours. The process runs iteratively until convergence.

Based on $Page - rank$ (to estimate the authority of candidate) and $Language - modeling$ (to estimate the relevance of a candidate with the query), TAP can provide page-rank with global Influence (PRI) and page-rank with topic-based Influence (TPRI).The combination method is to multiply or sum the Page-rank ranking score and the language model relevance score.

- In PRI, transition probability in Page-rank is replaced by the influence score.
- In TPRI, for each node v a vector of ranking scores $r[v, z]$ is introduced each of which is specific to topic z. Random walk is performed along with the coauthor relationship between authors within the same topic.

2.5 OPINION MINING IN NEWSGROUP DOMAINS

2.5.1 Graph Theoretic Approach

Agrawal et al. (2003) have developed a graph-theoretic algorithm on typical newsgroup postings. The algorithm works in following ways:

Optimum Partitioning – consider any bipartition of the vertices of a social network graph $G\ (V, E)$ into two sets F representing those for an issue and A representing those against an issue. Assume F and A to be disjoint and complementary, i.e., $F \cup A = V\ and\ F \cap A = \emptyset$. Such a pair of sets can be associated with the cut function, $f(F, A) = |E \cap (F \times A)|$, the number of edges crossing from F to A.

Constrained Graph Partitioning–Given the graph $G(V, E)$ and two sets of vertices C_F and C_A, constrained to be in the sets F and A respectively, find a bipartition of G that respects this constraint but otherwise optimizes $f(F, A)$.

Synthetic Data Generation –

1. For each author v, the number of comments p_v that v posts is a random variable drawn from a Zipf distribution (George, 1949) with mean I and theta θ. All three real datasets follow a Zipf distribution for the number of postings versus rank of author.
2. Randomly set S fraction of authors as "for" and the remaining as "against".
3. For each author, select the other users this author comments to. Let author v have p_v postings in step 1. For each of the p_v postings :
    a. With the probability P, the user is picked from the opposite side, and with probability $1 - p$ from the same side.
    b. Within the set of users on either side, a random user is picked to complete the link.

2.6 SEMANTIC ORIENTATION AND POLARITY ANALYSIS

2.6.1 Classification by Semantic Orientation of Phrases

Turney (2002) has proposed an unsupervised learning algorithm for classifying reviews. It has the following steps:

1) A part-of-speech tagger is used to identify phrases in the text that contains adjectives or adverbs. Two consecutive words are extracted from the reviews if their tags conform to any of the patterns in table 12.

First Word	**Second Word**	**Third Word (Not Extracted)**
JJ	NN or NNS	Anything
RB, RBR, or RBS	JJ	Not NN nor NNS
JJ	JJ	Not NN nor NNS
NN or NNS	JJ	Not NN nor NNS
RB, RBR, or RBS	VB, VBD, VBN, or VBG	anything

Table 12 Patterns of Tags

Example – The second pattern means that two consecutive words are extracted if the first word is an adverb and the second word is an adjective, but the third word cannot be a noun.

Table 13 shows a list of parts-of-speech tags according to Santorini (1990)

CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
EX	Existential there
FW	Foreign word
IN	Preposition or subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List item marker
MD	Modal
NN	Noun, singular or mass
NNS	Noun, plural
NP	Proper noun, singular
NPS	Proper noun, plural
PDT	Predeterminer
POS	Possessive ending

PP	Personal pronoun
PP$	Possessive pronoun
RB	Adverb
RBR	Adverb, comparative
RBS	Adverb, superlative
RP	Particle
SYM	Symbol
TO	to
UH	Interjection
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle
VBN	Verb, past participle
VBP	Verb, non-3rd person singular present
VBZ	Verb, 3rd person singular present
WDT	Wh-determiner
WP	Wh-pronoun
WP$	Possessive wh-pronoun
WRB	Wh-adverb

Table 13 List of part-of-speach tags (Santorini 1990)

2) Estimate the semantic orientation of each extracted phrase using PMI-IR algorithm which uses Pointwise Mutual Information as a measure of the strength of semantic association between two words as

$PMI\ (word1, word2) = \log_2 \left[\frac{P(word1\ \&\ word2)}{P(word1)p(word2)}\right]$ where $P(word1\ \&\ word2) =$ the probability that $word1$ and $word2$ co-occur, $P(word1)$ and $P(word2)$ describe the probability of $word1$ and $word2$ respectively.

In the five star review rating system, one star means "poor" and five stars mean "excellent", so the semantic orientation (SO) of a phrase is

$$SO(phrase) = PMI(phrase, \text{excellent}) - PMI(phrase, \text{poor})$$

PMI-IR estimates PMI using Information Retrieval (IR) techniques and noting the number of matching documents (hits).

Example – Let for a given query "$query$", $hits(query)$ is the number of hits returned.

So, $SO(phrase) = \log_2 \left[\frac{hits(phrase\ NEAR\ \text{excellent})hits(\text{poor})}{hits(phrase\ NEAR\ \text{poor})hits\ (\text{excellent})}\right]$

SO is positive when $phrase$ is more strongly associated with "$excellent$" and negative when $phrase$ is more strongly associated with "$poor$".

3) Assign the given review to a class "$recommended$" or "$not\ recommended$" based on the average semantic orientation of the phrases. If average SO is positive, classify the review as $recommended$, and otherwise $not\ recommended$.

2.6.2 Classification by Polar Language

According to Nigam and Hurst (2006) the identification of polar language has the following steps:

1) Set-up phase – A dictionary is developed which is tuned to the topic being explored. Each item in the dictionary is a pairing of a word and its part-of-speech.

Example – For digital camera, phrases like "blurry" may be negative and "crisp" may be positive.

2) Tokenization and Chunking phase – Input is tokenized, then segmented into discrete chunks. The input is tagged with part-of-speech information, then semantic orientation is done.

Example – Let us take an input "This car is really great".

Tokenization » {this, car, is, really, great}

POS tagging » {this_DT, car_NN, is_VB, really_RR, great_JJ}, after adding polarity lexicon {this_DT, car_NN, is_VB, really_RR, great_JJ; +}

Chunking » {(this_DT)_DET, (car_NN)_BNP, (is_VB)_BVP, (really_RR, great_JJ; +)_BADJP}. Where basic chunk categories are {DET, BNP, BADVP, BVP, OTHER}.

3) Interpretation phase – Chunked input is formed higher order grouping of a limited set of syntactic patterns that associate polarity with some topic.

Example – Syntactic patterns » Predicative modification (it is good), Attributive modification (a good car), Equality (it is a good car), and Polar clause (it broke my car).

2.7 INFLUENCE MAXIMIZATION (IM)

Kempe et al. (2003), define influence maximization as follows:

Given a network graph $G(V, E)$which is directed with influence probability or weight for each edge and an IM model M, the influence of set of vertices $A \subseteq V$, denoted $\sigma M(A)$ is the expected number of active vertices once the diffusion process is over. The goal of M is to maximize $\sigma M(A)$.

2.7.1 'Lazy Forward' Optimization

Leskovec et al. (2007) tackle the problem of outbreak detection, which is the problem of selection of nodes in a network in order to detect the spreading of virus or information as quickly as possible. Leskovec et al. (2007) develop an efficient algorithm called CELF, based on a "lazy-forward" optimization in selecting seeds.

CELF algorithm maintains a table of marginal gain, $mg(u, S)$, of each node u in current iteration sorted on $mg(u, S)$ in decreasing order, where S is the current seed set and $mg(u, S)$ is the marginal gain of u with respect to S. Table $mg(u, S)$ is re-evaluated only for the top node in next iteration. If required the table is resorted. If a node remains at the top after this, it is picked and added to the seed set. Leskovec et al. (2007) evaluated their methodology extensively on two large scale real world scenarios: a) detection of contamination in large water distribution network, and b) selection of informative blogs in a network of more than 10 million posts.

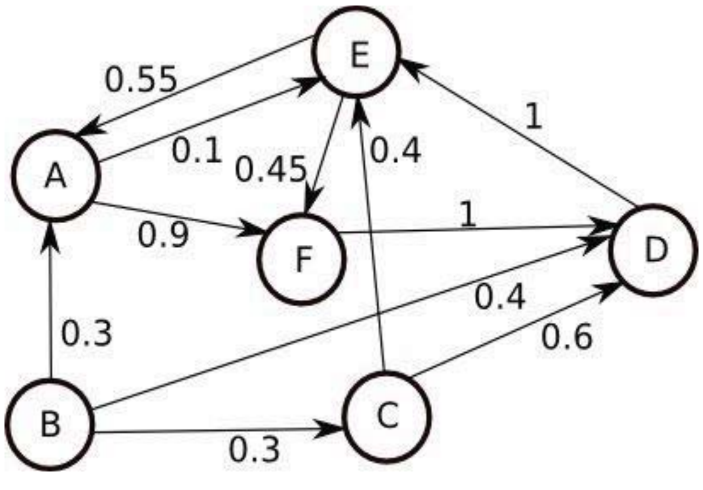


Figure 9 Social Network Graph with influence probability

Example: Consider the social network graph in figure 9 with given influence probabilities. Let us set $k = 2$, i.e., we are looking for the seed set of size 2. CELF optimization will pick node A in the first iteration and will also create a table $mg(u, S)$ as follows:

Mg(A,{})	4
Mg(B,{})	3
Mg(C,{})	3
Mg(D,{})	2
Mg(E,{})	1

CELF will pick A as its marginal gain is the highest and will be removed from the table as follows:

Mg(B,{})	3
Mg(C,{})	3
Mg(D,{})	2
Mg(E,{})	1

Now in the next iteration the CELF optimization the algorithm will only evaluate the top node, i.e., node B. The marginal gain of node B with respect to $S = \{A\}$ was 3. As there is no change then node B will be selected as next seed and added to the seed set S.

2.7.2 Trust – Influential Node Miner (T-IM) Model

Existing IM approaches assume only positive influence among users and availability of influence probability, the probability that a user is influenced by another. Ahmed et al. (2013) propose a T-IM model that computes positive and negative influences in trust network by mining frequent patterns of actions performed by users to compute the influence probabilities.

Example: Let us say a node u performs $A_{v,u}$ number of actions after its trusted neighbor v and node v performs total of A_v tasks in total. T-IM computes positive influence probability of node v on node u by dividing $A_{v,u}$ by A_v . Then extracts Negative Frequent Action Pattern, which counts the number of actions not performed by any node u after the same actions were performed by a distrusted neighbor of u. Let us say a node u does not perform $A'_{v,u}$ number of actions after its distrusted neighbor v and node v performs total of A_v tasks in total. T-IM computes negative influence probability of node v on node $u$by dividing $A'_{v,u}$by A_v. Now, let us assume that according to action log node $v$performs a total of 3 actions. And out of these 3 actions 2 actions were performed by u after node v (trusted neighbor of u) performs these same actions. So, the probability of node u performing a task after node v performs the same action is $2/3 = 0.66$.

T-IM takes social network graph $G(V, E)$ and a variable *budget*. The algorithm returns set of influential nodes (seed set), S, such that S is a subset of $V$and $|S| <= budget$. The algorithm starts by initializing seed set S to NULL. Then the algorithm computes spread of each node v in V. The node with highest spread is picked and added to S. Also, $V - S$ is the set of nodes which are not in set S but in set of all nodes V. The algorithm then performs the following local search operations:

Delete – If by removing any node *v* in S results in increasing the spread under T-IM the node is removed from *S*.

Add – If by adding any node *v* in $V - S$results in increasing the spread under T-IM model the node is added to the set S.

Swap- If by swapping any node *v* in S with any node *u* in $V - S$ results in increasing the spread under T-IM model the node *v* is removed from the set *S* and node *u* is added to the set *S*.

CHAPTER 3

PROPOSED OPINION AND POSTS MINING FOR DISCOVERING COMMUNITY PREFERENCES FROM SOCIAL NETWORKS

As discussed in Section 1.6, our goal is to identify popular posts and influential users on a given topic from the large-scale friendship network. Our task is to extract relevant topic-posts and nodes from the social graph, analyze the topic-posts and the behaviour of responses on the posts by computing the popularity and mining the users' opinion.

For example, let us consider a topic z for which we want to find relevant posts W that are popular i.e., the posts people talk about a lot. We want to identify the nodes (users) Vs who have posted such popular posts and their influential ability over the friendship network on the topic z. Suppose we have found a set of posts $\{w1, w2, w3\}$ on topic z posted by users (nodes) $\{Vs, Vx, Vy\}$. Each of the topic-post may have different data structures.

Topic-post	**Type**
w1	Text
w2	Image
w3	Video

Let us consider a topic-post $w1$ posted by Vs from the list. A set of users $\{V1, V2, V3, V4, V5\}$ may express their opinions on the post by different ways. Figure 10 shows a graph representation of the topic-post in friendship network.

Nodes	**Responses**
V1	Likes
V2	Shares
V3	Comment (reply to other comment)
V4	Comment (negative)
V5	Comment(positive)

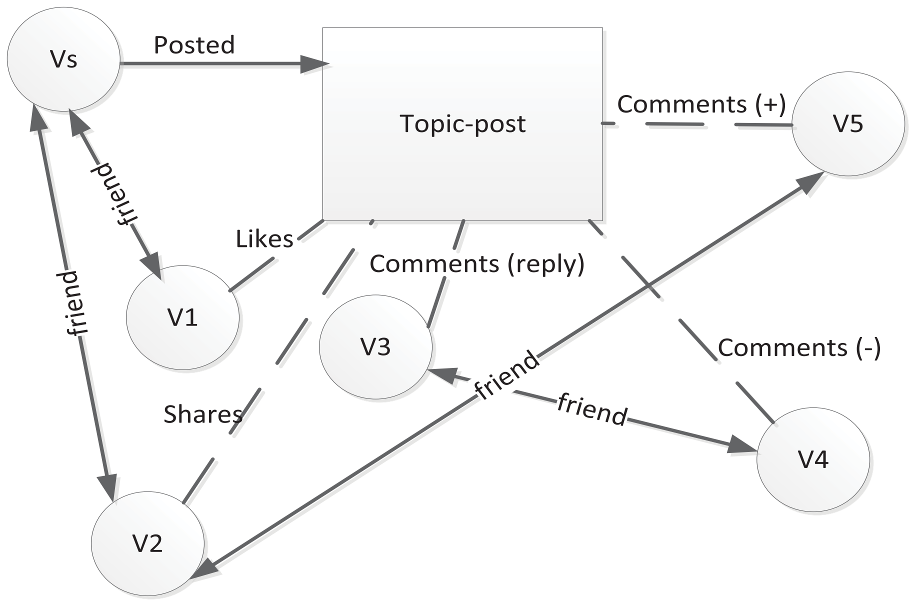


Figure 10 A heterogeneous network model for Topic-post

Figure 10 shows an example activity in friendship network where Vs posted a topic-post, and $V1, V2, V3, V4, V5$ have expressed their opinion in different ways. $V1$ likes the topic-post, $V2$ re-shares the topic-post, $V3$ replies to a previous comment on the post, $V4$ express dislike by comment, and $V5$ agrees/likes the topic post by comment. In our proposed thesis, we want to analyze all kind of responses and find out the popularity of the topic and influence ability of the node Vs. Furthermore, we want to analyze the relationships regarding the topic to discover the community preference.

Section 3.1 describes the features we have discovered by studying friendship networks and we have to analyze them. Section 3.2 describes the overall solution framework. Remaining sections describe our proposed solutions in detail with algorithms and running examples and complexity analysis.

3.1 FEATURES TO BE ANALYZED

In our thesis, we have studied friendship networks, Facebook and Google Plus, and we have classified the stories at the social networks hierarchically into two levels (1) $category$ and (2) $topic$ within the different categories. Our study has found nine common categories in recent popular friendship networks such as Facebook, Twitter, and Google Plus as follows:

(i) World Business, (ii)Technology, (iii) Science, (iv) Game, (v) Sports, (vi) Entertainment, (vii) Life Style, (viii) Politics, and (ix) Religion

Examples of topics include “iPhone”, “McBook”, “Apple”, “Google”, “Windows”, or “Linux” within the category “Technology”; “Barcelona vs Real Madrid” in category “Sports”; and “FarmVille” and “Texas HoldEm Poker” within the category “Game”.

In this thesis the popularity of a given topic post is represented by following different phenomena that we have found analyzing of friendship network.

> **Definition 3.1** ***Approve*** – We define Approve by determining how many people like a given object by clicking a $like$ button. It is the number of likes on a topic post, we denote it by $A,\ A = nL$. Where nL is the number of likes. Our proposed system extracts shared object to analyze with a specific A.

For example, if we decide $A > 50$, then our system will extract the relevant posts that have $likes$ more than 50.

> **Definition 3.2** ***Spreading*** – We define Spreading by determining how many people tend to share this object by forwarding it to other people or clicking $re - share$ button on their profile.

In a friendship network, for example Facebook or Google Plus, when a user clicks like button or comment on a specific topic-post, that post is automatically shared with friends also.

> **Definition 3.3** ***Simple response*** – We define Simple response by determining how many people tend to comment on a given post. In our proposed system, we obtain a hybrid measure of $spreading$ and $simple\,response$ by calculating the number of different user commenting on the topic-post. We denote it by SR, $SR = nC_U$, where nC_U is the number of unique comments.

For example, if we decide $SR > 50$ along with $A > 50$, then our system will extract the relevant posts that have $likes$ more than 50 and have more than 50 unique users' comments..

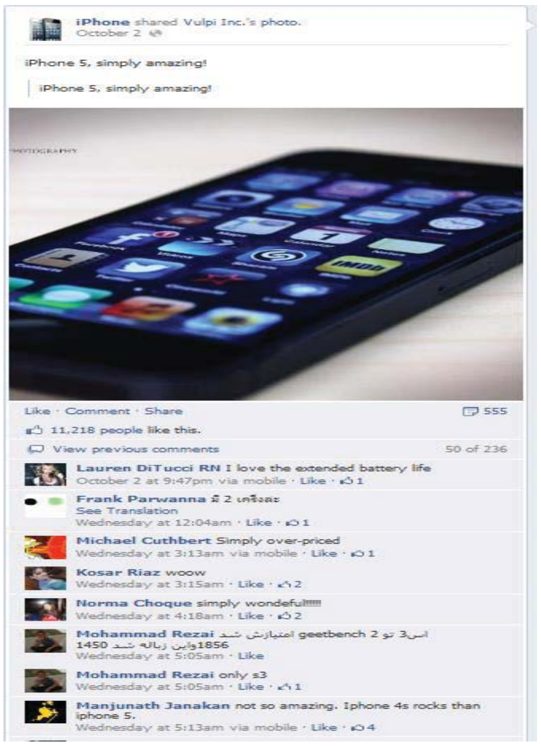
Definition 3.4 ***White responses*** – We define White responses by determining how many people tend to comment in a positive mood, for example “I love this product”.

Definition 3.5 ***Black responses*** – We define Black responses by determining how many people tend to comment in a negative mood, for example, “Buying this product is wastage of time”.

Our target is to find whether a topic-post has positive, negative, or neutral impact. We denote white response as R_W and black response as R_B.

- $R_W = (Positive > (Neutral + Negative))$
- $R_B = (Negative > (Neutral + Positive))$

Where $Positive$, $Negative$ and $Neutral$ indicate the number of comments for the topic-post categorized as positive, negative or neutral respectively.



iPhone shared Vulpi Inc.'s photo.
October 2

iPhone 5, simply amazing!

iPhone 5, simply amazing!

Like · Comment · Share 555
11,218 people like this.
View previous comments 50 of 236

Lauren DiTucci RN I love the extended battery life
October 2 at 9:47pm via mobile · Like · 1

Frank Parwanna มี 2 เครื่องละ
See Translation
Wednesday at 12:04am · Like · 1

Michael Cuthbert Simply over-priced
Wednesday at 3:13am via mobile · Like · 1

Kosar Riaz woow
Wednesday at 3:15am · Like · 2

Norma Choque simply wondeful!!!!!
Wednesday at 4:18am · Like · 2

Mohammad Rezai اس3 تو 2 geetbench امتیازش شد 1450 واین زباله شد1856
Wednesday at 5:05am · Like

Mohammad Rezai only s3
Wednesday at 5:05am · Like · 1

Manjunath Janakan not so amazing. Iphone 4s rocks than iphone 5.
Wednesday at 5:13am via mobile · Like · 4

In this example post, in figure 11,

$$Topic\ object = \text{“iPhone 5”}$$

$$Number\ of\ people\ like\ the\ post = 11{,}218$$

i.e.,$A = 11218$

$$Number\ of\ people\ re-share\ the\ post = 555$$

And so far we can see,
$number\ of\ unique\ comments = 7$

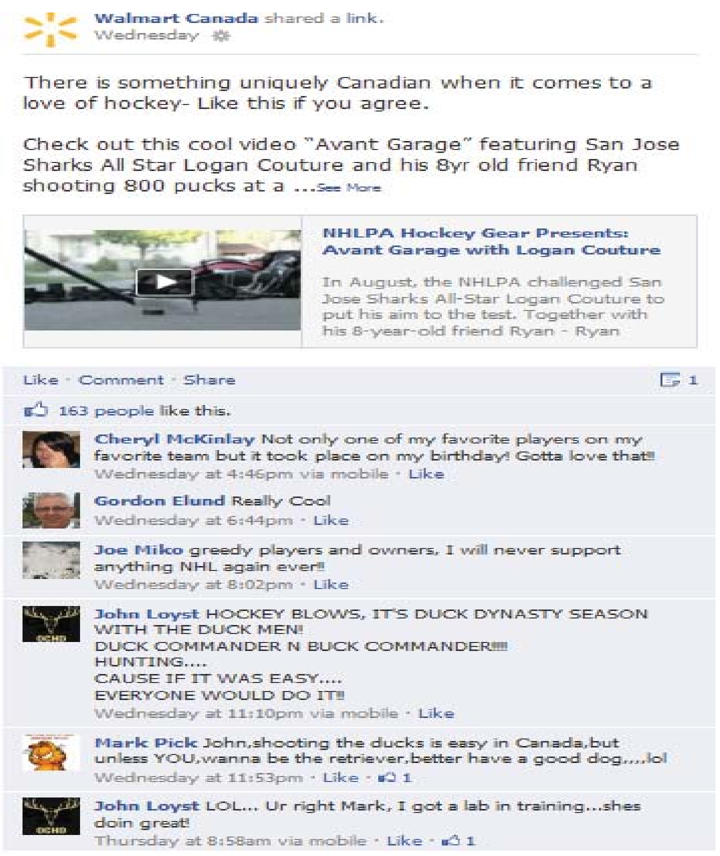
i.e.,$SR = 562$

So far we can see,
$the\ number\ of\ positive\ comments = 3$,
$number\ of\ negative\ comments = 2$, and
$number\ of\ neutral\ comments = 0$.

Here we discarded any language other than English.
Since $Positive > (Neutral + Negative)$ i.e.,
$3 > (2 + 0)$, so this post has $white\ response$

Figure 11 An example of Facebook Topic-Post

Definition 3.6 ***Raising discussion*** – It is the ability to include discussion among people, for example, people discussing the topic "Apple and Samsung battle". To determine discussions, we need to distinguish explicit replies to other comment. We denote raising discussion by RD, $RD = (nC_{L} / nC_T) \times nC_U$. Where nC_T and nC_{L} are the number of comments on the topic-post and number of comments which are replies to other comments, respectively, and nC_U is the number of unique comments on the topic-post.



Walmart Canada shared a link.
Wednesday

There is something uniquely Canadian when it comes to a love of hockey- Like this if you agree.

Check out this cool video "Avant Garage" featuring San Jose Sharks All Star Logan Couture and his 8yr old friend Ryan shooting 800 pucks at a ...See More

NHLPA Hockey Gear Presents: Avant Garage with Logan Couture

In August, the NHLPA challenged San Jose Sharks All-Star Logan Couture to put his aim to the test. Together with his 8-year-old friend Ryan - Ryan

Like · Comment · Share 1

163 people like this.

Cheryl McKinlay Not only one of my favorite players on my favorite team but it took place on my birthday! Gotta love that!!
Wednesday at 4:46pm via mobile · Like

Gordon Elund Really Cool
Wednesday at 6:44pm · Like

Joe Miko greedy players and owners, I will never support anything NHL again ever!!
Wednesday at 8:02pm · Like

John Loyst HOCKEY BLOWS, IT'S DUCK DYNASTY SEASON WITH THE DUCK MEN!
DUCK COMMANDER N BUCK COMMANDER!!!!
HUNTING....
CAUSE IF IT WAS EASY....
EVERYONE WOULD DO IT!!
Wednesday at 11:10pm via mobile · Like

Mark Pick John,shooting the ducks is easy in Canada,but unless YOU,wanna be the retriever,better have a good dog,,,,lol
Wednesday at 11:53pm · Like · 1

John Loyst LOL... Ur right Mark, I got a lab in training...shes doin great!
Thursday at 8:58am via mobile · Like · 1

In Figure 12, the last two comments can be considered as raising discussion, since Mark replies John and John again relies back to Mark by explicitly mentioning name of each other.

Figure 12 Example of Raising Discussion

Definition 3.7 ***Controversiality*** – It is the ability to split the people in different groups i.e., mostly against the given post, for example, the video game "Medal of Honor" raised a controversial opinion where some people claimed it was only a game and some people claimed it was harmful and disrespecting for fallen Allied soldiers.

1) If the highest number of positive comments is X and the highest number of negative comments is Y, then controversiality of the topic-post is denoted as C, $C = Y/X$. We consider a topic-post as controversial if the measure ranges from $0.5\ to\ 1.5$. If $X = 0$, to avoid $divide\ by\ zero$, we consider the result $C = 0$.
2) $C = 0$ means total agreement, the topic-post is either positive or negative.
   $C = 1$ means highest controversiality, the opinions split exactly into two.

Controversial posts have a tendency to be popular as seen in the analysis done in Slashdot (Gomez et al., 2008). In this thesis, Approve, Simple Responses, White and Black Responses are the main measurement, and Raising Discussion and Controversiality are used to analyze extracted information if necessary.

3.2 PROPOSED OBIN MODEL

Proposed OBIN takes a social network graph $G(V, E)$ and a product name z as input to generate an influence graph $G_z(V, E)$ on product z from computed community preference where V is the relevant nodes extracted from G. Algorithm 1 shows the algorithm for OBIN model. OBIN has 3 main functions, TPD (Topic-Post Distribution), PCP-Miner (Post-Comment Polarity Miner), and influence network generator. OBIN first executes SQL queries on social network URL to extract nodes (V_S) on a product z, and then classify relevant and irrelevant nodes. This process is done by TPD method (lines A.1-A.4 in Algorithm. 1). Then PCP-Miner (lines B.1-B.2 in Algorithm. 1) takes the ranked relevant nodes, posts, and comments from TPD to identify opinion (positive or negative) comments and compute the polarity score (θ_Z) for each relevant post. Based on the polarity score, OBIN generates an influential network that represents the community preference for the product z (line C.1-C.2 in Algorithm. 1).

```
Algorithm OBIN to generate influence network graph $G_z$ from friendship network $G$
Input: Social network URL (e.g., facebook.co), product $z$, Approve $A$, Simple response $SR$,
$Term$ in product name $z$
Output: Set of influential nodes $V_s$, influenced nodes $V_t$, influence graph $G_z$ on $z$
A. OBIN calling TPD described in Algorithm 2 to extract nodes, posts, opinions from the
   network graph to classify relevant and irrelevant nodes
   A.1. Execute SQL query on URL to find set of nodes on product z using Graph API
   A.2. Generate nodes matrix NM with 4 attributes $< node, Term, A, SR >$
   A.3. Generate relevant nodes matrix PM with 4 attributes $< node(V_S), Term, A, SR >$by
        mining $NM$ with three features $(Term + A)$, $(Term + SR)$, $(A + SR)$, to classify
        relevant and irrelevant nodes with SVM classifier. Store relevant nodes $V_S$in $PM$
   A.4. Execute SQL query on URL to find set of posts and comments on product z of $V_S$. Store
        posts $w$ in table tblPosts and comments $c$ in table tblComments.
B. OBIN calling PCP-Miner described in Algorithm 5 to identify opinion (positive/negative)
   comments and compute polarity score
   B.1. FOR each comment $c$ in tblComments table DO
        B.1.1. Execute tokenization and POS-tagging process as described in section 3.2
        B.1.2. Generate $FeqFT$ $(c, FFT)$ matrix by identifying frequent features $(FFT)$ in $c$
               through Apriori frequent pattern algorithm with minimum support 1%
        B.1.3. Identify opinion words $OW$ for extracted $FFT$ as described in section 3.2
        B.1.4. Determine semantic orientation $OR$ of $OW$as described in section 3.2
        B.1.5. Generate $OE$ $(c, FFT, OW, OR)$ matrix of $c$
        B.1.6. Store node $V_t$, who commented c, in $VT$matrix (influenced nodes matrix)
   B.2. FOR each post W in tblPosts table DO
        B.2.1. Compute the polarity score $\theta_Z$ from $OE$ matrix as $\theta_Z = (\sum positive\ c - \sum negative\ c) * 100\%$
        B.2.2. Store node $V_S$, who posted $W$, in $VS$ matrix (influential nodes matrix)
C. OBIN calling PoPGen described in Algorithm 9 to generate influence network
   C.1. Merge $VT$ and $VS$ matrices into influence matrix $IMAT$ with 3 attributes $<
        V_S, V_t, Action >$ as follows:
        C.1.1. IF node$V_t$ responds to node $V_S$
               C.1.1.1. $IMAT[Action] = \sum responses$
        C.1.2. ELSE
               C.1.2.1. $IMAT[Action] = 0$
   C.2. Generate a weighted influence graph $G_z = (V, E)$ where $V \epsilon VT, VS$ and $E =
        IMAT[Action]$ if there exist a relationship between $VT$and$VS$ matrices (section 3.3)
```
Algorithm 1 OBIN to generate influential network from friendship network

Our proposed solution framework for social and opinion posts mining for community preference discovery is illustrated in Figure 13. Following are the inputs to the framework:

1. Social network URL (e.g., Facebook.com), and topic z (e.g., iPhone).
2. Predefined threshold – Approve (A) which is the minimum number of nodes (v_t) that have to be connected to the node (v) who posted the topic-post.
3. Predefined threshold – Simple response (SR) which is the minimum number of posts (w) the node (v) has to post on topic z.

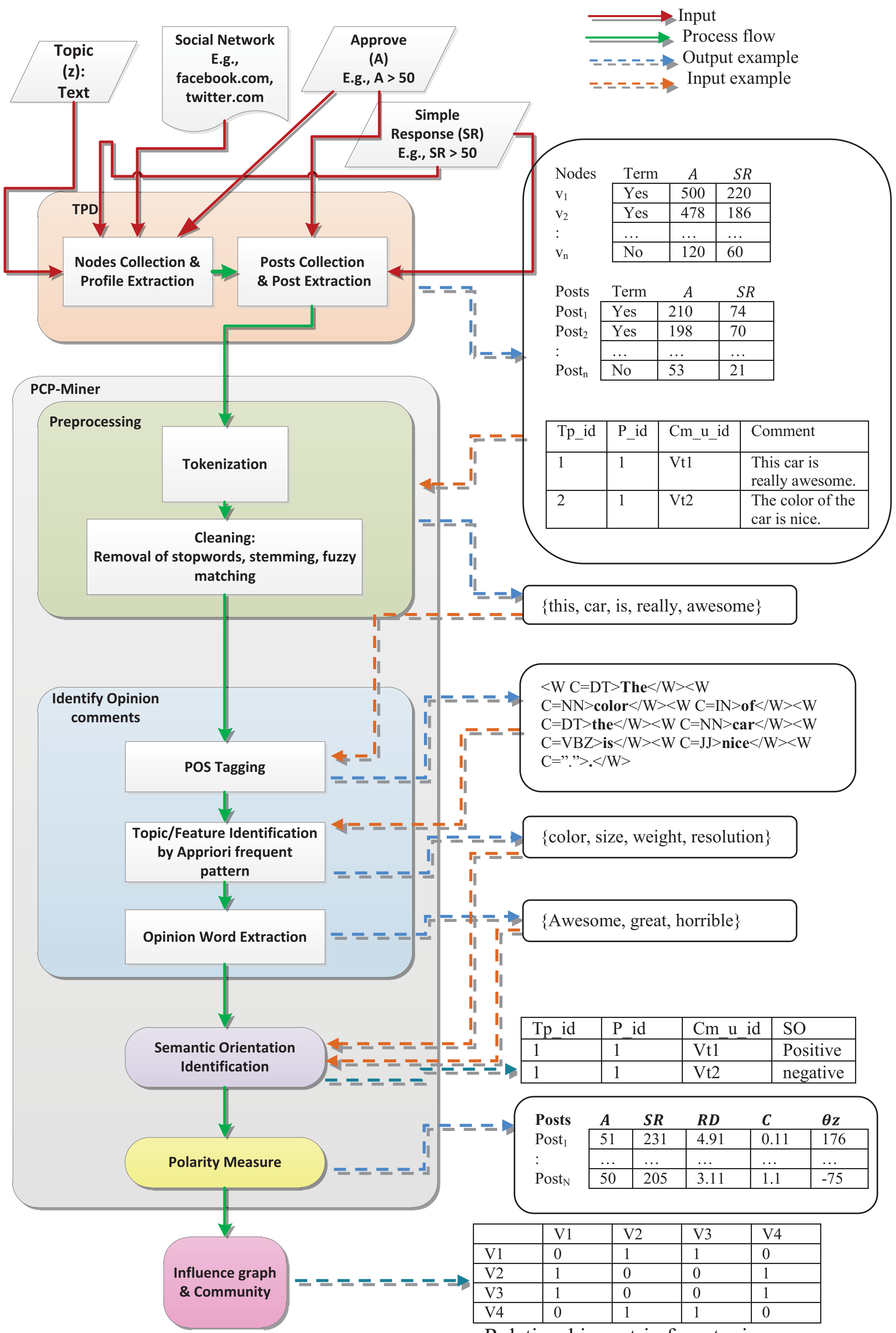
The intermediate inputs are listed below:

4. Predefined threshold – Approve (A) which is the minimum number of nodes (v_t) that have to like the topic-post (w) that is posted by node v.
5. Predefined threshold – Simple response (SR) which is the sum of the total number of unique comments (nC_U) on the topic-post (w) and the total number of re-shares of the post (w) by the nodes v_t.
6. Part-of-speech tag list – POS-tags from predefined list on Table 28 – to identify syntactic orientation of words
7. WordNet list – from (http://www.princeton.edu/wordnet/download/current-version/) to identify synonyms and antonyms

The proposed solution consists of following four steps listed below:

Step1: At first our proposed solution framework OBIN calls TPD to extract relevant nodes $v \in V$ for a topic z and filter them according to higher influential score determined by Approve A and Simple Response SR. Lines A.1 to A.4 in Algorithm 1 shows the steps for our proposed model TPD. TPD then extracts and filters relevant posts $w \in W$ for each relevant node v. Detailed steps of these processes using TPD model with algorithm and examples are given in section 3.4. The resultant data are stored into our transactional database for next steps.

Step2: In this second step, our solution framework OBIN calls PCP-Miner to fetch all the opinions for each relevant post w of each relevant node v, and apply sentence and word


Input
Process flow
Output example
Input example
Topic (z): Text
Social Network E.g., facebook.com, twitter.com
Approve (A) E.g., A > 50
Simple Response (SR) E.g., SR > 50
TPD
Nodes Collection & Profile Extraction
Posts Collection & Post Extraction
PCP-Miner
Preprocessing
Tokenization
Cleaning: Removal of stopwords, stemming, fuzzy matching
Identify Opinion comments
POS Tagging
Topic/Feature Identification by Appriori frequent pattern
Opinion Word Extraction
Semantic Orientation Identification
Polarity Measure
Influence graph & Community

Nodes	Term	*A*	*SR*
v_1	Yes	500	220
v_2	Yes	478	186
:	…	…	…
v_n	No	120	60

Posts	Term	*A*	*SR*
$Post_1$	Yes	210	74
$Post_2$	Yes	198	70
:	…	…	…
$Post_n$	No	53	21

Tp_id	P_id	Cm_u_id	Comment
1	1	Vt1	This car is really awesome.
2	1	Vt2	The color of the car is nice.

{this, car, is, really, awesome}

<W C=DT>**The**</W><W C=NN>**color**</W><W C=IN>**of**</W><W C=DT>**the**</W><W C=NN>**car**</W><W C=VBZ>**is**</W><W C=JJ>**nice**</W><W C=”.”>**.**</W>

{color, size, weight, resolution}

{Awesome, great, horrible}

Tp_id	P_id	Cm_u_id	SO
1	1	Vt1	Positive
1	1	Vt2	negative

Posts	***A***	***SR***	***RD***	***C***	***θz***
$Post_1$	51	231	4.91	0.11	176
:	…	…	…	…	…
$Post_N$	50	205	3.11	1.1	-75

	V1	V2	V3	V4
V1	0	1	1	0
V2	1	0	0	1
V3	1	0	0	1
V4	0	1	1	0

Relationship matrix for a topic

Figure 13 System Diagram of OBIN

segmentation and some cleaning such as stemming, string matching etc. Lines B.1 to B.2 in Algorithm 1 shows the processing steps for PCP-Miner. For each opinion sentence in the opinion text, our proposed PCP-Miner apply POS-tagging (Brill 1994) to identify adjective, adverb as opinion words and noun, noun phrase as features. Then identify the polarity of the comment i.e., the comment expressing positive or negative opinion. And finally compute the popularity of the relevant post w. Detailed processes are given in section 3.5 with algorithm and examples.

Step3: In this step, our solution framework store all the extracted and computed data into our data warehouse for further mining purpose.

Step4: After our previous steps, we have a ranked list of mined relevant nodes $v \in V$, their corresponding popular topic-posts $w \in W$, and aggregated opinions on each post along with their polarity (positive impact or negative impact). In this fourth step (lines C.1 to C.2 in Algorithm 1), our proposed solution framework OBIN calls PoPGen model to identify the relationships among nodes $v_i \; and \; v_j \; (v_i, v_j \; \in V \; and \; i, j = 1,2, \ldots, N$ on a topic z and how they influence to each other. Our proposed solution also identifies a global relation between nodes $v_i, and \; v_j$ for similar topic, hence discover the community preference. Details of the steps are described in Section 3.6.

3.3 DATA WAREHOUSE GENERATION

We have studied three most popular friendship networks, Facebook, Twitter, and GooglePlus, and we have classified the topic-stories into two levels:

1. Category, e.g., “World Business”, “Technology”, “Sports”, etc.
2. Topic, e.g., “iPhone”, “McBook”, “Apple” etc. in the category “Technology”.

Based on our study we found nine major categories in social networks and our extracted nodes are related to topics under those categories. Each topic that is related to a specific node, has post title, a number of likes/dislikes, re-shares, and positive/negative comments.

Based on our study, we have generated a data warehouse, named OBIN_dwh as following structure:

Cat_id	**Cat_Name**
1	Games
2	Technology

Table 14 Category Table, tblCategory

Cat_id	**Tp_id**	**Tp_Name**
1	1	FirmVille
2	2	iPhone

Table 15 Topic Table, tblTopic

U_id	**Name**	**Link**
429326	Alex Brown	http://www.facebook.com/Alex.Brown
223952	Peter Pen	http://www.facebook.com/223952

Table 16 User Table, tblUser

P_id	**Approves**	**SR**	**RD**	**C**	**Score (θ_z)**	**Title**	**Link**
962538	1990	78	7.317	0.15	55	Samsung VS Apple	http://www.facebook.com/223952/posts/962538

Table 17 Posts Table, tblPost

Tp_id	**P_id**	**Lk_u_id**
2	962538	9272631
2	962538	89236063

Table 18 Likes Table, tblLikes

Tp_id	**P_id**	**Cm_u_id**	**Polarity**	**Time_posted**	**Comment**
2	962538	6932106	Positive	2012-11-02 19:04:08	I have aiphone 5, I upgraded from a 4. Theover all applications of the phone is awesome.
2	962538	40527930	Negative	2012-11-02 19:10:02	iPhone 4 was much more better than this.

Table 19 Comment Table, tblComment

U_id	Tp_id	P_id	Time_Posted	No_Likes	No_Shares	No_Comments	Com_positive	Com_negative
223952	2	962538	2012-11-02 14:02:02	1990	3	75	65	10

Table 20 Fact Table, tblFact

3.4 TOPIC-POST DISTRIBUTION (TPD) MODEL

As mentioned in the line A.1 to A.4 in algorithm 1, our proposed OBIM calls TPD for a given topic z, to filter out irrelevant nodes from the social graph that have lower influential score than a predefined threshold determined by $Approve\ (A)$ and $Simple\ Response\ (SR)$. TPD gives a set of nodes $V$and every node $v_i\ \in V$ has a topic-post distribution $\{p(z|v_i)\}$. Algorithm 2 shows the algorithm for TPD model which also connected to Algorithm 3 and Algorithm 4.

$A\ = nL$, where $nL\ = number\ of\ people\ who\ like\ the\ topic - post$

$SR\ = nC_U$, where $nC_U\ = number\ of\ unique\ comments\ on\ the\ topic - post$

Our proposed model TPD has two major tasks, first it focuses on the extraction of relevant nodes for a specific topic automatically from the given social network. Then for each node, TPD extracts relevant posts automatically. TPD consists of three steps: relevant nodes identification (**Identification**), **Preprocessing**, and **Extraction**.

3.4.1 Identification

We have a list of topics in several categories. For a given topic z, we first search if the topic is already in our database or not. If it is in our database we will take its corresponding category CT. Then we execute a search mechanism over the given social network G with the term z and CT. In our proposed approach, we focus on Facebook, Twitter, GooglePlus. For Facebook we execute Facebook Query Language (FQL) to search over the social network using Graph API.

Graph API: The Graph API presents a simple, consistent view of the Facebook social graph, uniformly representing objects in the graph (e.g., people, photos, events, pages) and the connection between them (e.g., friend relationships, share content, and photo tags). Every object in the social graph has a unique ID, that we can access. In Facebook, user's name also can be used as ID. To execute the Graph API, we need to access the URL *api('/search/q?=')* to social network website. For example, in Facebook, we need to download the Facebook SDK and run the json code "$\$facebook-> api('/search?q = '.iPhone.'\&type = page\&fields = likes, name, id, category, link\&limit = 1000')$" , that gives the nodes information for the product iPhone. Graph API also needs an APP ID and Secret code for the social network which is collected by opening an empty application in the social network website.

FQL: FQL enables SQL-style interface to query the data exposed by the Graph API. Queries are of the form "$SELECT\ [fields]\ FROM\ [table]\ WHERE\ [condition]$". FQL can handle simple math, basic Boolean operations, AND or NOT logical operators, and ORDER BY and LIMIT clauses. **Example**:

$$\begin{aligned} query1 \ = \ & SELECT\ uid, name, pic\_square\ FROM\ user\ WHERE\ uid \\ & = me()\ OR\ uid\ IN\ (SELECT\ uid2\ FROM\ friend\ WHERE\ uid1 \\ & = me()) \end{aligned}$$

This $query1$ returns all user information for the active logged-in user and friends.

Using FQL and Graph API we will get a list of nodes V relevant to the topic z or category CT. The output data has the following format:

Nodes	Term	A
v_1	…	…
v_2	…	…
:	…	…
v_n	…	…

Algorithm 3 shows the algorithm form Identification method which is called by TPD model.

Algorithm: Topic-Post Distribution (TPD) called by OBIN

Input:

1. Category Table tblCategory // with tuples $< cat\_id, cat\_name >$
2. Topic table tblTopic // with tuples $< cat\_id, tp\_id, tp\_name >$
3. URL of the Social Network // to be crawled
4. Topic z // a text

Output: Set of profiles D: $\{D_1, D_2, \ldots, D_i\}$ where $D_i = \{(w_1, v_i), \ldots, (w_N, v_i)\}$

// w_N=post_id, v_i= nodes

Other:

1. V – user who posts relevant topic-post
2. W – post that is published by V
3. tblUser – Table in transactional database to store user information
4. tblPosts – Table in transactional database to store user's posts
5. tblComments – Table in transactional database to store comments on the post W
6. $Term$ – Topic word
7. SH – number of shares of the post W
8. LK – number of likes on the post W
9. CM – number of unique comments on the post W
10. MSG = comment text

```
BEGIN

1.  $TT := z$ , $CT := cat\_name$ from tblCategory for $tp\_name = z$
2.  PM = Identification ($TT, CT$) // Algo. 3 of page 59
3.  FOR each node $V$ in PM
    3.1. TPM = Preprocessing (PM[$V$], $TT$) // Algo. 4 of page 61
    3.2. Store PM[$V$] in Tbale tblUser
    3.3. Store TPM[$W$] in Table tblPosts
    3.4. Store TPM[$C$] in Table tblComments
    3.5. D[$V$] = [$W$][$V$]
4.  END FOR
5.  D = D+D[$V$]

END
```

Algorithm 2 Topic-Post Distribution (TPD)

```
Algorithm: Identification ($TT, CT$) called by TPD – identify relevant nodes on topic z

Input:
1. Topic $TT$, Category $CT$
2. Approves $A$ // minimum number of people connected to node $V$
3. Simple Response $SR$ // minimum number of posts node $V$ has

Output: Profile Matrix PM
----------------------------------------------------------------
BEGIN
1. $Term$ := NULL
2. Execute FQL query to get $A1$ = total number of people connected to $V$ and
   $SR1$ = total number of posts V has
3. PM = $[V, Term, A1, SR1]$
4. IF PM[$A1$] <$A$ OR PM[$SR1$] <$SR$
    4.1. Remove $V$ from PM
5. END IF
6. RETURN PM
END
```

Algorithm 3 Identification of relevant nodes

3.4.1.1 Running Example

To demonstrate the entire work flow of the OBIN framework, we will use a small sample real-time dataset extracted from Facebook.com. Let us now demonstrate how we can integrate Graph API with FQL and conduct a local search in Facebook to collect all the relevant nodes $v \epsilon V$ for a given topic z.

To collect a complete list of topic categories, we use Facebook and run Javascript using jQuery and collected 146 categories. Table 21 shows a sample list of categories collected from Facebook, where Cat_id represents the category id and Cat_name represents the title of the category.

Let us suppose, z = iphone, input to Graph API: {"https: // www. facebook.com/ search/ results.php?"}, FQL = {SELECT id, name, category, likes, links FROM search WHERE q = 'iphone' AND (type = 'page' OR type = 'group')}. The results executed from Graph

API and FQL are shown in Table 22. We denote the schema of relation as $U_R = < Node\ (v), Term, A, Link >$

Cat_id	**Cat_name**
1103	Actor/Director
1105	Movie
1109	Writer
1202	Musician/Band
1300	Book
1602	Public Figure
1700	Politician
2214	Health/Beauty
2252	Food/Beverage
2603	Non-profit Organization
2201	Product/Service

Table 21 Example of topic categories

Node id v	**Term**	**A**	**Link**
130489060322069	iphone	3116728	iphone.page
110018862354999	iphone 4	1435239	Iphone-4
214456561919831	iphone Fans	261210	theappleclan
101936296565340	IPhone 4S	262165	IPhone-4S/101936296565340
144971705536847	IPhone 3G	234676	IPhone-3G/144971705536847
267282993312609	IPhone5-infocentrul	189483	iPhoneInfocentrul
159984244020234	iPhone &iPad 粉絲同好會	178115	ipad.ipod.iphone
146534208714348	iPhone 4 Society	118674	iPhoneSociety

Table 22 Example of relevant nodes and data for z = iphone

For example, in table 22, the first row shows a node with unique id "130489060322069" and name "iphone" (in Term column) that has 3116728 friends and we can visit his profile by “iphone.page” link. Note that, in this thesis we are analyzing data set with language in English. So although $v = 159984244020234$ has a good A value, we ignore it, and we index the data set according to A in descending order.

3.4.2 Preprocessing

Algorithm: Preprocessing (PM[V], TT) called by TPD – Generate Topic-post Matrix for each relevant node

Input:

```
1. Node $V$, Topic $TT$, FQL parameter $Type$
2. Access Token $AT$// access key for Graph API
3. Approve $A$ // minimum number of likes on post $W$
4. Simple responses $SR$ // minimum number of re-shares and unique comments a
   post has to have
```

Output: Post Matrix TPM and Post by Comments Matrix C

```
BEGIN
1. $Type$ := “posts”
2. Execute FQL query to get total number of likes, re-shares, comments on each
   topic-post W posted by node $V$
3. TPM1 = $[W, Term, SH, LK, CM, MSG]$ // create a temporary matrix from retrieved
   posts
4. IF TPM1$[LK]$ <$A$ AND TPM1$[(SH + CM)]$ <$SR$
   4.1. Remove $W$ from TPM1
5. ELSE
   5.1. TPM = $[W, Term, LK, (SH + CM)]$ // add the post in the matrix
   5.2. C = $[W, MSG]$ // add the comment text in the matrix
6. END IF
7. RETURN TPM
END
```

Algorithm 4 Preprocessing to generate Topic-post Matrix

In preprocessing step (Algorithm 4), each relevant node is taken and apply a local search in the whole webpage. In our proposed approach we use Graph API as a crawler to crawl the profile page. We use the crawling parameters “$type$”, “$user\_name$”, and “$access\_token$” in the Graph API.

Here, $type$: "$posts$"

$user\_name$: $node's\ ID$

$access\_token$: $graph\ API\ developer\ key$

Example: Let us take a topic z = "$iPhone\ 5$" . In our pre-processing model, $user\_name$ = "$iPhone$" will results the following data:

id: 8491

$from$: $iPhone$

$message$: "$Perfect\ Fit\ Tech\ wants\ to\ know\ which\ is\ better$?

$iPad\ Mini\ vs.\ iPad4\ (4th\ gen)$"

$picture$: "$http://fbcdn-photos-a.akamaihd.net/3842.jpg$"

$shares$: 91

$likes$: 6171

$comments$: 47

In this step, we look into four parameters: "$message$", "$shares$", "$likes$", and "$comments$". According to our problem definition,

$Approves\ (A) = likes = 6171$

$Simple\ Response\ (SR) = shares + comments = 91 + 47 = 138$

Then we apply a term matching process to find whether "$message$" contains the topic-term or not. Our resultant data have the following tabular format:

Posts	Term	A	SR
Post$_1$	Yes	…	…
Post$_2$	Yes	…	…
:	…	…	…
Post$_n$	No	…	…

A profile d is a vector w_d of N_d posts; a vector v_s of $V_z$nodes choosen from a set of nodes of size V. A collection of D profiles on topic$z$ is defined as:

$D = \{D_1, D_2, \dots, D_i\}$; $D_i = \{(w_1, v_i), (w_2, v_i), \dots, (w_N, v_i)\}$ where $w = topic - post$ and $N = number\ of\ topic - posts$.

3.4.2.1 Running Example

As we implemented Identification step, we have a set of 130 nodes with their corresponding Approve (A) and links to their profile. Note that, the nodes data table is sorted by A in descending order. Now let us set a threshold Approve (A) as 1000, meaning that we are looking for nodes having $A \geq 1000$ from this dataset. Now preprocessing step takes each node from the data set of table 18 and crawl its profile page to search relevant posts on topic z. In table 18 we have a set of 7 users $V = \{$ 130489060322069 , 110018862354999 , 101936296565340□ , 214456561919831 , 144971705536847□ , 267282993312609□ , 146534208714348□ $\}$ having $A = \{3116728, 1435239, 262165, 261210, 234676, 189483, 118674\}$. Let us take node $v = 130489060322069$ and execute query as FQL = {SELECT $post\_id$, $message$, $likes.count$, $share\_count$, $created\_time$, $comments.count$, ($comments.count + share\_count$) FROM stream WHERE $source\_id = '130489060322069'$ AND $message\ != ""$ AND $created\_time = month('2013-03-06')$ ORDER BY $likes.count$ desc LIMIT 100} that results a set of first 100 posts posted in March 2013 with total number of likes, comments, and shares sorted by number of likes. For each post we have a set of A, $Term$ i.e., the message it contains whether has the topic or not, and SR. Table 23 shows a sample data set for node $v = 130489060322069$. We denote the schema of the relation as $P_R = < Post\ (w), Term, A, SR > where\ SR = comments + shares$. For example, the first row in Table 23 shows a post with unique id “469219579782347” posted by node “130489060322069", that has the post title “black- like, white-comment, and the winner is ?” and has got 61153 likes in the post, and total number of re-shares and unique comments are 11325.

Post id$w$	Term	A	SR
469219579782347	black- like, white-comment, and the winner is ?	61153	11325
468646856506286	pretty amazing	33899	2213
469758623061776	Apple 5th Avenue	33041	2198
467263769977928	white or black?	31359	10364
465792903458348	Take it	28028	2622
472223806148591	Hero	27566	2080
466379303399708	which one?	24708	8502
180356388777720	Amazing iPhone!	20147	1880
465731800131125	iPhone 5 - The biggest thing to happen to iPhone since iPhone :)	19685	1420

Table 23 Example of Post Data

3.4.3 Extraction

After all the relevant nodes on the given topic z identified and for each node all the relevant posts are discovered, we have to extract the most relevant nodes and posts into our database for further analysis. We employ $(Term + Approves)$ features, $(Term + Simple\ Responses)$ features, and $(Approves + Simple\ Responses)$featuresto classify relevant and irrelevant nodes using Support Vector Machine (SVM). In the final step of TPD model, we store our resultant data into a transactional database for further analysis. Our transactional database has the following structure:

U_id	Name	Link
429326	Alex Brown	http://www.facebook.com/Alex.Brown
223952	Peter Pen	http://www.facebook.com/223952

Table 24 User Table tblUser

P_id	**Approves**	**SR**	**RD**	**C**	**Score (θ_z)**	**Title**	**Link**
962538	1990	78	NULL	NULL	0	Samsung VS Apple	http://www.facebook.com/223952/posts/962538

Table 25 Posts Table tblPosts

Cat_id	**Tp_id**	**P_id**	**Cm_u_id**	**Polarity**	**Time_posted**	**Comment**
1	2	962538	6932106	NULL	2012-11-02 19:04:08	I have aiphone 5, i upgraded from a 4. The overall applications of the phone is awesome.
1	2	962538	40527930	NULL	2012-11-02 19:10:02	iPhone 4 was much more better than this.

Table 26 Comments Table tblComments

In our proposed thesis, TPD keeps track of $V \times D$ (user by profile) matrix, $D \times W$ (profile by posts) matrix, and $W \times C_m$ (post by comments) matrix.

3.4.3.1 Running Example

In this step, we apply $(term + A)$, $(term + SR)$, and $(A + SR)$ features for extraction. Let us suppose $term$ = {iphone, Apple, cell, mobile, handset}, $A \geq 100$, and $SR \geq 20$, which extracts most relevant posts on the topic z = iphone. We then store the relevant nodes data, posts data, and corresponding users' comments data in our transactional database called OBIN_transaction. We denote the scema of relation as $D =< D_1, D_2, \ldots D_i >$, $D_i =< (w_1, v_i), (w_2, v_2), \ldots, (w_N, v_i) >$. For example, D = {iphone, iphone 4, iphone Fans}, D_1 = {(469219579782347, 130489060322069), (468646856506286,130489060322069), (469758623061776,130489060322069), (467263769977928,130489060322069), (465792903458348,130489060322069), (472223806148591,130489060322069), (466379303399708,130489060322069), (180356388777720,130489060322069), (465731800131125,130489060322069)}

3.5 POST-COMMENT POLARITY MINER (PCP-MINER)

In a social network, users are free to comment on any published post and express their opinion. From our proposed model TPD, we obtain a ranked list of nodes (users) who have posted relevant topic-posts. Our next task is to find useful comments on the posts, analyze the comments and decide whether the post has a good or bad impact on the topic. Our proposed model TPD gives us several topic-posts for a given topic z for each node v. For each post of each node, our proposed Post-Comment Polarity Miner (PCP-Miner) described in Algorithm 5, identifies opinion comments across all the comments on that post w, identifies the semantic orientation (SO) of the comments, and measure the polarity of the comments as well as the popularity of the post. Our proposed PCP-Miner model considers four major features on users' comments: White Responses (R_W), Black Responses (R_B), Raising Discussion (RD), and Controversiality (C). The positive, negative, or neutral polarity is determined as follows:

R_W if $(Positive\ comments\ >\ (Negative\ comments\ +\ Neutral\ comments))$

R_B if $(Negative\ comments\ >\ (Neutral\ comments\ +\ Positive\ comments))$

$RD\ =\ (nC_L\ /\ nC_T)\ \times\ nC_U$

Where nC_L= number of comments that are replies to other comments.

nC_T = total number of comments

nC_U = total number of unique comments

$C\ =\ Y/X$, where Y = total number of negative comments and X = total number of positive comments. We consider 0.5 < C < 1.5. If $C\ =\ 0$, then total agreement i.e., the post is either positive or negative. If $C\ =\ 1$, then highest controversiality, i.e., the post opinions split exactly into two sides.

Our proposed PCP-Miner has four major steps: extract comments from topic-posts, identifies opinion comments across all comments, identifies the semantic orientation of the comments, and measure the polarity of the comments.

Algorithm: Post-Comment Polarity Miner (PCP-Miner) called by OBIN

Input:

1. Topic z // Topic ID from Table tblTopic
2. Comment C // Table tblComments from transactional database with tuples $< tp\_id, p\_id, com\_u\_id, c0mments >$
3. Post W // with tupples $< tp\_id, p\_id >$ from Table tblPosts in transactional database

Output:

1. Features set
2. Polarity matrix for each comment
3. Polarity matrix for each post

Other:

1. TK – list of tokens with XML tags
2. OW – opinion words in comment c
3. FT – frequent features in comment c
4. OR – orientation of opinion words
5. SO – semantic orientation of comment
6. nCL – comment replies, CM – unique comments, nCT – total number of comments
7. Qz – popularity score, $Cont$ – controversiality score, RD – discussion score
8. POS, NEG, NUT – integer variable to count number of positive, negative and neutral comments respectively

```
BEGIN

1. Matrix C = create a matrix for comments from tblComments
2. FOR each $c$ in C // each comment text $c$ in the matrix C
   2.1. TOK $(c, TK)$ = Tokenization $(c)$ // Algo. 6 of page 69
   2.2. OE $(c, OW, FT, OR)$ = OpinionExtraction $(c, TK)$ //Algo. 7 of page 76
   2.3. CSO $(c, SO)$ = SemanticOrientation $(c, OW)$ //Algo. 8 of page 77
   2.4. IF$c$ is reply of $[c-1]$
        2.4.1. $nCL = nCL + 1$ // count total number of replies
   2.5. ELSE
        2.5.1. $CM = CM + 1$ // count total number of unique comments
   2.6. END IF
   2.7. $nCT = nCT + 1$ // count total number of comments
3. END FOR
4. FOR each $c$ in CSO // calculate total number of positive, negative and
   neutral comments
   4.1. IF CSO$[SO]$ = positive // semantic orientation of comment c
        4.1.1. $POS = POS + 1$
   4.2. ELSE IF CSO$[SO]$ = negative
        4.2.1. $NEG = NEG + 1$
   4.3. ELSE
        4.3.1. $NUT = NUT + 1$
   4.4. END IF
5. END FOR
6. $Qz = (POS - NEG) * 100/CM$ // popularity score of post W
7. $Cont = NEG / POS$// controversiality of post W
8. $RD = (nCL * CM) / nCT$// raising discussion score of post W
9. PCP$[W]$ = $[RD, Cont, Qz]$ // insert information into popularity matrix
10. Store PCP matrix to Data Warehouse

END
```

Algorithm 5 Post-Comment Polarity Miner (PCP-Miner)

3.5.1 Extraction

From our proposed model TPD, we have a list of comments stored in our transactional database. The extraction step of PCP-Miner contains data collection from the transactional database and pre-processing. PCP-Miner takes all the comments for each post on topic z. Data preprocessing is done by sentence segmentation and cleaning.

3.5.1.1 Tokenization

Tokenization is a straightforward Natural Language Processing task for languages like English and other languages, where words are delimited by blank spaces and punctuations. We divide each comment text into sentences and each sentence into meaningful units i.e., words. For example " $This\ car\ is\ really\ great$ " results $\{this, car, is, really, great\}$. In our proposed method, tokenization is done by scanning the comment text and identifies word and sentence boundaries. Words are delimited by punctuation (,) and sentences are delimited by question marks (?).

The input to the tokenization process is list of comment texts, and output is the marking text with XML markup: tokens are represented as "W" elements, word-class information is provided in their "T" attribute, and sentences are marked with "S" elements.

```
Algorithm: Tokenization (c) called by PCP-Miner – to segment comment
text to sentences and sentences to words

Input: Comment $c$

Output: TOK matrix with comment text and all the tokens

Other:$Index, Boundary$

BEGIN

1.  Set word $Boundary$: $\{punctuations, spaces\}$
2.  FOR$Index\ =\ 0$ to Lengthe of $c$
    2.1. IF$c[Index]\ = Boundary$
         2.1.1.  TOK$[Index]$ = $c[Index - 1]$
         2.1.2.  Apply XML parser
    2.2. END IF
3.  END FOR

END
```

Algorithm 6 Tokenization

Example:

Input text: This car is really great, latest technologies are included.

Output: <S><W T=w>**This**</W><W T=w>**car**</W><W T=w>**is**</W><W T=w>**really**</W><W T=w>**great**</W><W T=P>**,**</W><W T=w>**latest**</W><W T=w>**technologies**</W><W T=w>**are**</W><W T=w>**included**></W><W T=”.”>**.**</W></S>

Here each word token is marked as “W”, “$T = w$” means standard word, “$T = P$” means punctuation. When we find “$T = '.'$” we consider them as the sentence end and replace them by “?”, and when we find $</W>$ means end of word and replace them by “,”. Table 27 shows the list of token tags (http://www.infogistics.com/textanalysis.html).

Flag	**Meaning**	**Explanation**
w	Regular word	Such words are written in the middle of a sentence and capitalized in sentence-starting positions.
W	Proper noun	Such words are written capitalized regardless whether they are sentence starting or middle of the sentence. We may consider them also as mentioning another user in the comment , i.e., reply of a previous comment
N	Numerical	Includes real numbers
P	Punctuation	Commas, semicolons
.	Sentence end	A period, question mark, exclamation mark
URL	Links	Link to another page or user. If a user, the comment is considered as the reply of a previous comment.

Table 27 Classes of Tokens

3.5.1.2 Cleaning

Data cleaning is a complex set of tasks that takes as input one or more sets of data and produces as output a single, clean data set (Golab and Ozsu, 2010). In our thesis, cleansing tasks include removal of stopwords, stemming (Willett, 2006), and fuzzy matching (Hu and Liu, 2004) to deal with word variations and misspelling. Along with these cleaning mechanism, we also employ fuzzy duplicates removal i.e., those comments are not exact replicas but exhibit slight or even large differences in the individual data values, removal of comments having suspicious links in the content to prevent from spam, removal of comments containing languages other than English. If a post contains most of the comments having any of the data described above, we ignore the comments and take Approve (A) as polarity measure.

3.5.2 Identification of Opinion Words

In the identification of opinion words step, our proposed approach takes list of comments as input and produces a list of opinion words as output. The identification process of PCP-Miner has three steps. The first step is to use a part-of-speech tagger to identify phrases in the input text that contains adjectives or adverbs (Brill, 1994). The second step is to identify product features on which many people have expressed their opinions. The third step is to extract opinion words from the comment. For example, "This picture quality is awesome", where "awesome" is the effective opinion of picture quality.

3.5.2.1 Part of Speech Tagging (POS-tagging)

A comment text is a combination of noun, verb, adjective, etc. To identify opinion comments, we need to identify each word belongs to which part-of-speech. In our proposed thesis, POS tagging is the part-of-speech tagging (Manning and Schutze 1999) from Natural Language Processing (NLP) which reflects word's syntactic categories and helps to find opinion words. Common POS categories in English are: noun, pronoun, verb, adverb, adjective, preposition, conjunction, and interjection. In our proposed model, we use a list of POS tags from Santorini (1990).

POS Tag	**Description**	**Example**
CC	Coordinating conjunction	and
CD	Cardinal number	1, second
DT	Determiner	the
EX	Existential there	*there* is
FW	Foreign word	d'hoevre
IN	Preposition or subordinating conjunction	in, of, like
JJ	Adjective	green
JJR	Adjective, comparative	greener
JJS	Adjective, superlative	greenest
LS	List item marker	1)
MD	Modal	could, will
NN	Noun, singular or mass	table
NNS	Noun, plural	tables
NP	Proper noun, singular	Robert

NPS	Proper noun, plural	Johnsons
PDT	Predeterminer	*both* the tools
POS	Possessive ending	friend's
PP	Personal pronoun	I, he, she, it
PP$	Possessive pronoun	my, her
RB	Adverb	however, usually, generally
RBR	Adverb, comparative	better
RBS	Adverb, superlative	best
RP	Particle	give *up*
SYM	Symbol	☺, ☹
TO	to	*to* do, *to* me
UH	Interjection	wow, OMG, LOL
VB	Verb, base form	take
VBD	Verb, past tense	took, was, were
VBG	Verb, gerund or present participle	taking
VBN	Verb, past participle	taken
VBP	Verb, non-3rd person singular present	take, am, are
VBZ	Verb, 3rd person singular present	takes, is
WDT	Wh-determiner	which
WP	Wh-pronoun	who, what
WP$	Possessive wh-pronoun	whose
WRB	Wh-adverb	when, where

Table 28 List of POS tags (Santorini 1990) with example

In this step, the tokenized input is tagged with POS information and formed basic groups (noun, adjective, adverb, and verb).

Example:

Input text: "*The color of the car is nice*."

Output:

Tokenization:<S><W T=w>**The**</W><W T=w>**color**</W><W T=w>**of**</W><W T=w>**the**</W><W T=w>**car**</W><W T=w>**is**</W><W T=w>**nice**</W><W T="."">**.**</W></S>

Tokens: {the, color, of, the, car, is, nice}

POS-tags to words:<W C=DT>**The**</W><W C=NN>**color**</W><W C=IN>**of**</W><W C=DT>**the**</W><W C=NN>**car**</W><W C=VBZ>**is**</W><W C=JJ>**nice**</W><W C=".">**.**</W>

Here the token "*The*" is tagged as a determiner (DT), the token "*color*" is tagged as a noun (NN) and so on. After POS-tagging, we apply syntactic grouping to identify groups of words in same part-of-speech. The output of POS-tagging is XML markup. The POS-tags along with tokens of each comment are stored into our transactional database.

3.5.2.2 Topic/Feature Identification

In this step, our proposed approach identifies topic/product features on which users have expressed their opinions on their comments. For example, if the comment about iPhone is "The sound system is very sophisticated", then "sound system" is the feature of topic "iPhone" that the user is satisfied with. In our proposed thesis, we focus on finding features that appear explicitly as noun or noun phrases in the comments. We mainly focus on finding frequent features in comments, i.e., those features that are talked about by many users. In general, a user's comment may contain many things that are not directly related to product/topic features. Different users usually have different perceptions. However, when users comment on product features, the words that they use converge. **For example**, in the case of "iPhone", some users may use "resolution" as a feature for "camera", some use as "screen", some use as "video call", etc. To identify which itemsets are product features, we use Association rule mining (Agrawal and Srikant 1994) to identify frequent itemsets, because those itemsets are likely to be product/topic features.

Comment Id	**Features**
1	Camera {resolution, camera}
2	Picture {resolution, camera, picture}
3	Screen resolution {resolution, screen}
4	Picture {resolution, video_call, camera, picture}

Table 29 Example of Frequent features

In our thesis, an itemset is a set of words or a phrase that occurs together in some sentences. From our POS-tagging step, we have a transactional set of nouns or noun phrases. Using those set, we apply association rule miner based on the Apriori algorithm (Agrawal and Srikant 1994) to find association rules.

The input to the Apriori algorithm is the set of nouns or noun phrases from POS-tagging, and the output itemset is topic/product features. We define an itemset frequent if it appears in more than 1% (minimum support) of the comment sentences, because we don not want to lose any important comment.

The Apriori algorithm finds the set of frequent patterns (large itemsets, L_i) iteratively by computing the support of each itemset in the candidate set C_i . **In our above example**,
Candidate set $C_1 = \{resolution, camera, picture, screen, video\_call\}$
$L_1 = \{resolution, video\_call, camera, picture\}$

$$C_2 = L_1 \ apriori\ gen\ L_1$$
$$= \{resolution{:}\, video\_call, resolution{:}\, camera, resolution{:}\, picture,$$
$$video\_call{:}\, camera, video\_call{:}\, picture, camera{:}\, picture\}$$

$And\ so\ on.$

$$Confidence = \frac{|Rule|}{|antecedent|}$$

Where example of $Rule$ can be $resolution -> \ video$_call or $resolution -> \ camera$, etc. and $antecedent$ can be $resolution$ in rule $resolution -> \ camera$
Rules are formed from these large itemsets and only strong rules with confidence greater than or equal to minimum confidence are kept.

3.5.2.3 Opinion Words Extraction

Opinion words are those words used to express opinions about the topic. For example, awesome, horrible, great, etc. are opinion words. In our proposed thesis, our opinion words extraction phase has two major tasks: extract opinion words around frequent features, and extract opinion words expressed in general form. Presence of adjectives in comment text is useful for predicting whether a text expressing opinion or not. In the opinion words extraction phase, our proposed method takes the list of tokens with corresponding POS-tags from our transactional database, and search for if it contains adjective words and/or frequent features.

The inputs to the extraction phase are tuples containing tokens, POS-tags and frequent features. The output from the extraction phase is the list of opinion words nearby features. For example, “awesome” is the opinion word of feature “picture quality” in the comment “Its picture quality is awesome”.

In our proposed thesis, we use WordNet (Miller et a. 1990) to utilize the adjective synonym set and antonym set to identify the opinion expressed by the word (i.e., positive or negative opinion). To do this, we use a list of known opinion adjectives called seed list, and progressively grow this list by searching in the WordNet for each adjective identified in our comment text. For example, positive adjectives are great, nice, good, awesome, cool, fine; and negative adjectives are bad, awful, terrible, horrible. Then for each extracted adjective, we search to WordNet for synonym and antonym of that adjective, and add to our known seed list. The seed list will result the desired opinion words. Algorithm 7 shows the algorithm for opinion words extraction.

Algorithm: OpinionExtraction (c, TK) called by PCP-Miner – Opinion words extraction

Input:

1. TOK matrix // List of tokens in the comment c
2. Tag list // part-of-speech tag list
3. WordNet list

Output: Set of opinion words OW along with features FT

Other:

1. OR – positive or negative orientation or opinion words
2. $PO -$ POS tagging with XML tags
3. FFT – frequent features

```
BEGIN
1. POS ($c, PO$) = NLProcessor($TK, Tag$) // POS-tagging for comment $c$
   (http://www.infogistics.com/textanalysis.html)
2. FeqFT ($c, FFT$) = AssociationRule($PO$) // Identify frequent features in
   comment c (Agrawal and Srikant, 19994)
3. FOR each c in POS matrix
   3.1. FOR each $c$ in FeqFT
        3.1.1. IFFeqFT[$FFT$] = POS[$c$] // opinion word extraction for frequent
               features
               3.1.1.1.   OE[$OW$] = POS[$PO$]
               3.1.1.2.   OE[$FT$] = FeqFT[$FFT$]
        3.1.2. END IF
   3.2. END FOR
4. END FOR
5. FOR each $c$ in POS matrix
   5.1. FOR each $OW$ in OE matrix
        5.1.1. IF POS[$c$] = OE[$OW$] // opinion word extraction for infrequent
               features
               5.1.1.1.   OE[$OW$] = POS[$PO$]
               5.1.1.2.   OE[$FT$] = POS[$PO$] // add noun-phrase or NULL as
                          infrequent features
        5.1.2. END IF
   5.2. END FOR
6. END FOR
7. FOR each $OW$ in OE list
   7.1. IF$OW$ has synonym $s$ in WordNet list // identify semantic orientation of
        opinion words
        7.1.1. OE($OW, OR$) = $s$'s orientation
        7.1.2. ADD $OW$ with orientation in OE
   7.2. ELSE IF$OW$ has antonym $a$ in Wordnet list
        7.2.1. OE($OW, OR$) = $a$'s opposite orientation
        7.2.2. ADD $OW$ with orientation in OE
   7.3. END IF
8. END FOR
9. RETURN OE($c, OW, FT, OR$)
END
```

Algorithm 7 Opinion Word Extraction

3.5.3 Semantic Orientation Identification

From our previous steps, we have a list of extracted opinion words in comment text. Now we need to identify the semantic orientation of each extracted phrase which will be used to predict the semantic orientation of each comment. The extracted word represents a positive semantic orientation when it has good association (e.g., “great *experience*”) and a negative semantic orientation when it has bad associations (e.g., “terrible incidence”).

```
Algorithm: SemanticOrientation (c, OW) called by PCP-Miner
Input: List of opinion words OE // opinion words with corresponding features
Output: Semantic orientation $SO$ of comment $c$
------------------------------------------------------------------------
Other: $orientation$ // $positive = 1, negative = -1, neutral = 0$
BEGIN
1. Set $orientation = 0$
2. FOR each opinion word $OW$ in OE
   2.1. $orientation1$ = orientation of $OW$
   2.2. IF any negation word appear closely to $OW$
        2.2.1. $orientation1$ = opposite $orientation1$
   2.3. END IF
   2.4. $orientation = orientation + orientation1$
3. END FOR
4. IF $orientation > 0$
   4.1. CSO[$SO$] = $positive$
5. ELSE IF $orientation < 0$
   5.1. CSO[$SO$] = $negative$
6. ELSE
   6.1. CSO[$SO$] = $neutral$
7. END IF
8. RETURN CSO($c, SO$)
END
```

Algorithm 8 Semantic Orientation

The inputs to the semantic orientation identification step are extracted frequent features, extracted opinion words. The outputs from the step are the semantic orientation of comments with frequent features and comments without frequent features. Algorithm 8 shows the algorithm to identify semantic orientation of opinion words.

3.5.3.1 Opinion Words with Frequent Features

In this step, we want to identify whether the frequent features has positive semantic orientation or negative orientation. For each frequent feature we find the nearest opinion word and its orientation, the orientation of the opinion word becomes the orientation of frequent features. Then we check for the negation words (e.g., not, never, did not, do not, etc) within five-word distance in front of an opinion word (Jin et al. 2009). We define the rules for negation words are:

Rule1: A negation word appears in front of a conjunction (e.g., and, or, but). Example – "This color is good but expires soon". This sentence mainly expresses negative opinion. So if opinion word is infront of the corresponding feature and conjunction "but/except" appears between opinion word and feature, then the opinion orientation for the feature is updated with the opposite of its initial orientation.

Rule2: Negation of negative opinion word is positive, e.g., "no problem". Negation of positive opinion word is negative, e.g., "not good". Negation of neutral opinion word is negative, e.g., "does not work" where "work" is a neutral verb.

If a comment sentence contains a set of features, then for each feature, we compute an orientation score for the feature. Positive opinion word has score (+1) and negative opinion word has score (-1). All the scores are then summed up. If the final score is positive, the semantic orientation of the comment is positive. If the final score is negative, then the semantic orientation of the comment is negative.

3.5.3.2 Opinion Words without Frequent Features

Frequent features are the hot features that users comment most about the topic-post. There can be some features that only few users talked about or some user may express their opinion directly to the topic post (e.g., "It's really nice", in this comment no feature mentioned, but user directly express his opinion on the topic-post). In such case, we just measure the semantic orientation of the comment from the orientation of opinion words in the comment.

3.5.4 Polarity Measure

To estimate the popularity of a topic-post, we have to aggregate all the polarities of the topic-post. In our proposed thesis, polarity of a topic-post comes from the polarity of comments measured by White Responses (R_W) and Black Responses (R_B), Simple Responses ($nC_U + re - shares$) and Approves (A). Algorithm 5 shows the algorithm for computing polarity.

For each topic-post, we calculate the polarity measurements, and transfer the transactional data into our data warehouse for further analysis. We can say, a topic-post has –

White response if (Positive comments) > (Neutral comments + Negative comments)

Black response if (Negative comments) > (Neutral comments + Positive comments)

We calculate the popularity score of each post z as

θz = (∑positive responses – ∑negative responses)×100%

θz serves as a popularity index for each post. Now for each post we have the following popularity matrix for a given topic z:

Posts	$\boldsymbol{A}$	$\boldsymbol{SR}$	$\boldsymbol{RD}$	$\boldsymbol{C}$	$\boldsymbol{\theta z}$
Post$_1$	51	231	4.91	0.11	59%
:	…	…	…	…	…
Post$_N$	…	…	…	…	…

Table 30 Popularity Matrix

Where,

Approves (A) = number of users like the post (nL)

Simple Response (SR) = number of users re-share the post + number of unique comments (nC_U)

Raising Discussion (RD) = (nC_L / nC_T) × nC_U, here nC_L = number of comments replies to other comments, nC_T = total number of comments

Controversiality (C) = $negative\ comments\ /\ positive\ comments$; we say a topic-post is controversial if $0.5 < C < 1.5$

For the polarity matrix, we conduct a binary classification such as SVM-light under its default settings (Joachims 1998) to classify most popular and less popular post with class label $A, SR, RD, C,$ and θz.

3.5.5 Running Example

Our proposed PCP-Miner algorithm has five major steps: Tokenization(), OpinionExtraction() with Apriori frequent pattern, SemanticOrientation(), and Polarity calculation. To demonstrate the working flow of PCP-Miner, we take some sample comment data from our transactional database OBIN_transaction. In this step we take comments from tblComment table where users' comments are already stored as cleaned with useful meaning. Table 31 shows a sample comment data for $w = 180356388777720$ after applying cleansing method. Then for each comment (c), the PCP-Miner algorithm performs the above mentioned process as following steps:

Post id$w$	**User idv_t**	**Time**	**Comment c**
180356388777720	100002395810151	2013-01-06T05:57:57+0000	i want
180356388777720	100003290108936	2013-01-06T10:18:16+0000	this is really cool
180356388777720	100004582655605	2013-01-06T11:35:48+0000	Cool
180356388777720	1850908608	2013-01-06T17:13:20+0000	hi sakuntla
180356388777720	100002090841333	2013-01-07T12:19:56+0000	i want to have one lyk that
180356388777720	100003365201901	2013-01-14T08:26:35+0000	o wow i crazy about it
180356388777720	3415872	2013-01-07T13:49:38+0000	Admin can upload 4.2.1 iphone 3g final official update to unjailbreakiphone

Table 31 Example of sample dataset for user comments

Step1: Apply Tokenization (c). For example, if we take the row (180356388777720,100002090841333�, $2013-01-07T12:19:56+0000$, i want to have one lyk that), the algorithm tokenizes c_5 to words according to punctuations {‘,’,’;’,’.’,’!’,’?’} and spaces {‘‘}.$TK[5]$ = {i, want, to, have, one, lyk, that}. All the tokenized comments are stored in a temporary hash table called TOK.

Step2: $NLProcessor(TK, Tag)$ is then take TOK table with a list of predefined $POS-tags$. $PO[5]$ = {i_PP, want_VBP, to_TO, have_VB, one_NN, lyk_UH_IN, that_PP}. All the POS-tagged comments are stored in a temporary hash table called POS.

Step3: A list of adjectives, verbs, adverbs, and nouns are extracted from step 2 for all the user comments. For example, from c_5, a set of features are $PO[1]$ = {want_VBP, have_VB, that_PP}. Here the feature $\{that\}$ is infrequent feature. To identify the corresponding feature for infrequent feature $\{that\}$, we apply $AssociationRule()$ algorithm and found feature $FFT[5]$ = {iphone} i.e., the post $Term$ itself. All the frequent and infrequent features are stored in a temporary hash table called $FeqFT$.

Step4: OpinionExtractor() algorithm then extract opinion words from the POS table. Opinion words are the adjectives, verb, adverb across the extracted features. Extracted opinion words are stored in a temporary hash table called OE.

Step5: To compute the polarity measure of a comment, we need to identify the semantic orientation and polarity of the opinion words stored in the table OE. For each opinion word OW_i in the list OE, we search its synonyms or antonyms in WordNet and collect its orientation. For example, $OE[1]$ = {want, positive}, $OE[2]$ = {cool, positive}, $OE[5]$ = {want, positive}. If a negative word comes infront of an opinion word, we consider the semantic orientation of the opinion word is its opposite orientation.

Step6: according to table OE, we have all the orientation i.e., the polarity of individual comment. To compute the popularity score θ_z of a post, we calculate the differences between all positive oriented comments and negative oriented comments. For example, Table 32 and Table 33 show the resultant popularity matrix for

$w = 468646856506286$where $\theta_z = (5 - 0) = 5$, and (Positive) > (Neutral + Negative) i.e., $5 > (2 + 0)$.

Post id$w$	**User idv_t**	**Polarity**	**Time**	**Comment c**
180356388777720	100002395810151	positive	2013-01-06T05:57:57+0000	i want
180356388777720	100003290108936	positive	2013-01-06T10:18:16+0000	this is really cool
180356388777720	100004582655605	positive	2013-01-06T11:35:48+0000	Cool
180356388777720	1850908608	NULL	2013-01-06T17:13:20+0000	hi sakuntla
180356388777720	100002090841333	positive	2013-01-07T12:19:56+0000	i want to have one lyk that
180356388777720	100003365201901	positive	2013-01-14T08:26:35+0000	o wow i crazy about it
180356388777720	3415872	NULL	2013-01-07T13:49:38+0000	Admin can upload 4.2.1 iphone 3g final official update to unjailbreakiphone

Table 32 Example data in tblComment table

Post id$w$	**A**	**SR**	**θ_z**	**Term**
180356388777720	20147	1880	51	Amazing iPhone!

Table 33 Example data in tblPost table

All the data of relevant nodes v_s, nodes who commented v_t, posts w, and comments c basedon the popularityscore θ_z, Approve A, and Simple Response SR, are then transferred to data warehouse OBIN_dwh.

3.6 SOCIAL INFLUENCE GRAPH AND COMMUNITY PREFERENCE

After the processing of PCP-Miner, we obtain a ranked list of nodes and their posts for topic z. The goal of the current step is to find a sub-network that closely connects to top $k - influential\ nodes$ so that we can find a community based on their popularity. For a topic z on a node v, we have all the nodes u influenced by v. We calculate the influence score as follows:

Step1: We calculate the number of times a node u has respond to all the topic-posts posted by node v. We denote v_S as the node who posts the topic-post, and v_t as the node who responses the posts.

Influence score μ_{st}^{z} = number of responses by v_t to v_S

In our proposed thesis, to generate a social influence graph on topic z, we first filter out irrelevant nodes, i.e., nodes that have a lower influence score than a predefined threshold. An alternative way is to keep only a fixed number of (e.g., 100) of high scored nodes. Then we will get a matrix called influence matrix as Table 34.

Nodes	$\boldsymbol{\mu_{st}^{z}}$
Node$_1$	120
Node$_2$	118
:	…
Node$_N$	96

Table 34 Influence Matrix

Step2: For each node in the influence matrix, we then find if the node is also a friend of any other nodes in the list. For each pair of nodes $\{v_{ti}, v_{tj}\}$, we create an edge between them if they are connected with each other, and we denote the relationship as co-like relationship.

Algorithm: Popularity Graph Generator – PoPGen(E, V) called by OBIN

Input:

1. Node V_S // node who has posted topic z
2. List of nodesV_t who response to V_S
3. List of posts W_i posted by V_S
4. Popularity score Q_Z // from popularity matrix in PCP-Miner

Output:

1. Influence score M
2. Popularity/Influence Graph G_z (E, V)
3. Influence matrix IMAT

```
BEGIN
1. FOR each post $W_i$ by node $V_S$
    1.1. IF$V_t$ respond in $W_i$
        1.1.1. M[$V_t$] = M[$V_t$] + 1
        1.1.2. $V = V + \{V_t\}$ // add the vertex $V_t$ to the graph
    1.2. END IF
2. END FOR
3. FOR each node $V_i$ in M
    3.1. FOR each node $V_j$ in M where $j = i + 1$
        3.1.1. IF$V_i$ is connected with $V_j$
            3.1.1.1. $E = E + \{(V_i, V_j)\}$ // add an undirected edge between
                  $V_i$ and $V_j$
        3.1.2. END IF
    3.2. END FOR
4. IMAT $(W, V_t)$ = $[W_i][Q_Z]$ // add popularity of each post posted by each
   node to generate influence matrix
5. END FOR
6. $G_z$ = $\{z, (E, V)\}$ // Popularity/Influence graph for topic $z$
END
```

Algorithm 9 Popularity Graph Generator

Now we have the influence graph for each node v_S on topic z. So we have all the top fixed number of nodes (v_t) who have given responses to the topic-post posted by all the nodes (v_S) on same topic z.

We then find if those nodes (v_t) are internally/externally connected or not, and create an edge between them. This way we can find an overall influence graph G_Z on topic z. The influence graph G_Z gives us the community preference for the topic z. This influence graph G_Z can be used further for influence maximization which is the problem of detecting a small subset of social network graph that could maximize the spread of influence (Kempe et al. 2003).

3.6.1 Running Example

Form our data warehouse $OBIN\_dwh$, we have a ranked list of relevant nodes v_s, their posts w, and comments c on w, and the set of nodes v_t who commented on the posts w. From the set of nodes v_t, we compute the influence score μ_s and index them. For example, Table 35 shows a list of influenced nodes v_t who responded on the topic z, here for simplicity, we put a small number in bracket beside the original node id, for instance (1) means the node 1033467. The algorithm $PoPGen$ (popularity/influence graph generator) generates a social network influence graph $G_z = (V, E)$ on topic z using the influencematrix $IMAT$. $PoPGen$ add a node v_t to the vertex list according to predefined threshold. For all vertices v_t, $PoPGen$ find if v_{ti} has a relation with v_{tj} where v_{ti}, $v_{tj} \epsilon v_t$. For example, table 36 shows relationship between the nodes, and table 37 shows an influence matrix for all the nodes v_t. If there is a relationship exist based on response to posts (Table 35) or external friendship (Table 36), corresponding field value in Table 37 will be 1 and 0 otherwise. $PoPGen$ then add an edge between the vertices $v_{ti}$and v_{tj} if the field value is 1.

Node id v_s	**Post id$w$**	**Node id v_t**
1033467 (1)	49823667	33889 (4)
1033467 (1)	49823667	458089 (5)
1033467 (1)	49823667	221458 (6)

1033467 (1)	55090883	1120347 (7)
9980345 (2)	11250901	221458 (6)
9980345 (2)	11250901	114509 (8)
9980345 (2)	22370903	447880 (9)
11567090 (3)	2348095	1033467 (1)
11567090 (3)	2348095	458089 (5)
11567090 (3)	2348095	114509 (8)

Table 35 Example data for post - user relationship

Node id V_{t1}	**Node id V_{t2}**
33889 (4)	221458 (6)
221458 (6)	114509 (8)
114509 (6)	447880 (9)

Table 36 Example data for user - user relationship

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	0	0	1	1	1	1	1	0	0
2	0	0	0	0	0	1	0	1	1
3	1	0	0	0	1	0	0	1	0
4	1	0	0	0	0	1	0	0	0
5	1	0	1	0	0	0	0	0	0
6	1	1	0	1	0	0	0	1	1
7	1	0	0	0	0	0	0	0	0
8	0	1	1	0	0	1	0	0	0
9	0	1	0	0	0	1	0	0	0

Table 37 Example data for Influence Matrix (IMAT)

For example, the first row in table 35 shows that the node ‘1033467’ posted a post that has id ‘49823667’ and another node ‘33889’ gave his opinion on it. So node = ‘1033467’ has an influence on node = ‘33889’. In table 36, the first row shows that node = ‘33889’ also has a friendship connection with node = ‘221458’. So if we select the node =

‘1033467’, the influence spreads through node = ‘33889’ and node = ‘221458’. In table 37, the first row shows that, node = 1 has relation with node 3,4,5,6, and 7.

Figure 14 shows an influence graph generated from the influence matrix $IMAT$. The generated influence graph G_Z represents the community preference for a product z.

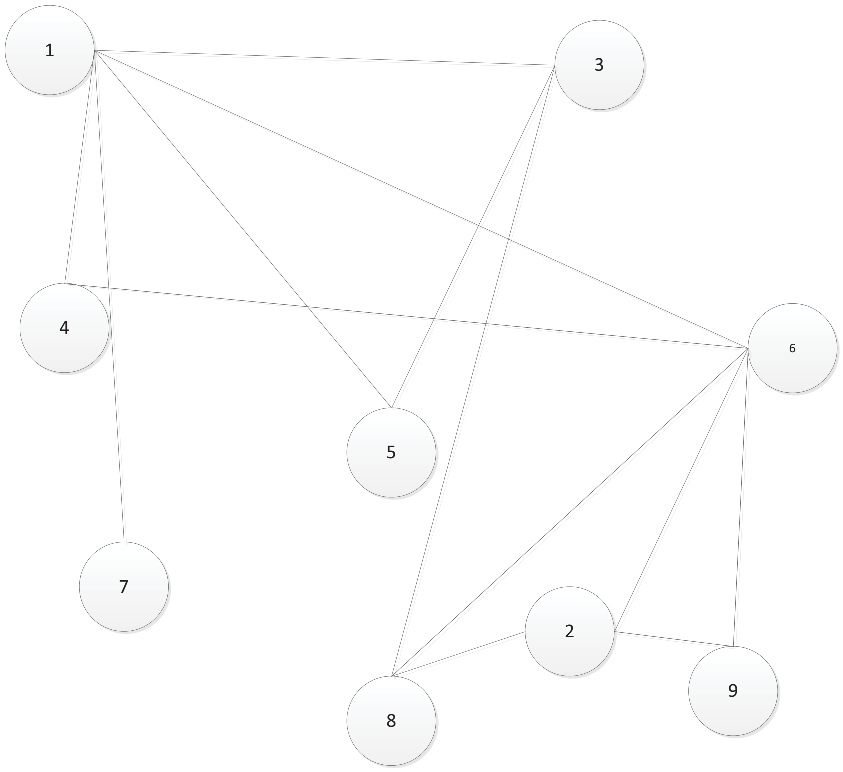


Figure 14 Social Network Influence graph$\boldsymbol{G_z}$ modelled from IMAT

3.7 COMPLEXITY ANALYSIS

We analyze the complexity of OBIN by determining two major processes, computing popularity score based on opinion mining of discovered relevant users and the run time required to compute the user-user relationships and their influence score to generate influence network. Popularity score computation based on opinion mining has run time complexity $O(A^*N)$, where A is the number of posts/comments and N is the number of users. This is because, for each user, the algorithm has to compute the popularity score for all posts. In this case, the algorithm could run longer time if number of posts/comments increase along with number of users. To tackle this, we restricted the number of posts/comments such as latest 100 posts of each user and latest 300 comments of each post, since we are interested to mine influential users for a given timestamp (e.g., who are the influential users in 2013?). Hence A become constant, and run time complexity for computing popularity score is $O(N)$.

The OBIN algorithm will also execute N^2 times to compute the users–user relationships and their influence score to generate the influence network, i.e., for each relevant user, process influence graph generation if the user has a relation with any other user in the discovered relevant users. So influence graph generator has $O(N^2)$ run time complexity. Hence, if there are N number of users in the network, the run time complexity of OBIN is $O(N^2)$ in worst case.

CHAPTER 4

EXPERIMENTS AND ANALYSIS

In this section, we present various experiments to evaluate the efficiency and effectiveness of the proposed OBIN approach.

4.1 EXPERIMENTAL SETUP

A system, called OBIN (Opinion-Based Influence Network), based on the proposed techniques has been implemented in PHP, Javascript, JQuery, MatLab and supported by Apache and MySQL.

4.1.1 Dataset

We conducted our experiments using the users' posts and opinions of Facebook as a friendship network since it is currently the most popular social media website. However, the proposed approach can be easily applied to other friendship networks such as GooglePlus, Twitter.

In this thesis, we perform our experiments on Facebook real-world data set. We extracted data for two Apple products: iPhone and iPad, and one Samsung product: Samsung Galaxy. Those products also have sub-categories such as iPhone 4, iphone 4s, iphone 5, Galaxy III, Galaxy Ace, Galaxy S4 and many more. Our proposed TPD (Topic-Post Distribution) method automatically extracts the relevant data through Graph API and FQL, and stores the data into data warehouse OBIN_dwh.

The first data set consists of user-user relationships that have fields as listed in Table 38.

Field Name	**Description**
USER_ID	Stores ID of the influential users who is posting about the product
CM_USER_ID	Stores ID of the influenced users who is expressing opinions on the post of the product
INFLUENCE_SCORE	Number of responses made by the influenced users, μ_{st}^{z}

Table 38 User-User relationships dataset

Each row in this data table represents a link between influential user (V_{t1}) and influenced user (V_{t2}). The INFLUENCE_SCORE field either a positive numeric value, meaningV_{t1}

has a relation to V_{t2} or 0, meaning no relation. Since our main goal is to show the quality of nodes selected by OBIN is better than that of CELF (Leskovec et al., 2007) and T-IM (Ahmed and Ezeife, 2013), and also any network with nodes more than 10,000 may run for days, we selected small snap shot from the dataset based on Approve (A) and Simple Response (SR).

Approve (A) – we considered as approved those nodes having more than 1000 nodes connected (i.e., $A \geq 1000$). With this characteristic, we had 1178 nodes with 42,664 relevant and irrelevant posts.

Simple Response (SR) – we considered as SR for the posts that have total number of re-shares and unique comments more than 20 (i.e., $SR \geq 20$) and also consider $A \geq 10$ for posts, which gives 3793 relevant posts.

The second dataset consists of opinion information (Table 39). This information is extracted for nodes corresponding to Table 38 and based on the polarity score θ_z.

Field Name	**Description**
USER_ID	Stores ID of the relevant users who is posting about the product
POST_ID	Stores ID of the post on the product
APPROVE	Stores the number of likes on the post
SIMPLE_RESPONSE	Stores the number of unique comments and re-shares of the posts
SCORE	Stores the computed polarity score by opinion mining
TIME_POSTED	Stores the time when the post has been published

Table 39 Opinion information dataset

After applying TPD and PCP-Miner, we obtained 343 influential nodes and 45,126 influenced nodes with 47,298 relationship edges, according to the computed influence score (i.e., the number of responses/actions performed by influenced nodes) and the computed popularity score of influential nodes.

As noted, CELF and T-IM are not product specific, we extracted data set for CELF and T-IM by randomly choosing popular nodes from the network based on the actions performed as action log and assigned probability as described by CELF and T-IM.

4.1.2 Evaluation Measure

We evaluate our proposed OBIN from four performance matrices:

1. **CPU time** – it is the execution elapsed time of the computation. This determines how efficient our method is.
2. **Recall and Precision** –recall is the ratio of the number of relevant nodes retrieved to the total number of relevant nodes in the social network.

$R = \frac{A}{A+B} \times 100\%$ where, A = number of relevant nodes retrieved, B = number of relevant nodes not retrieved

Precision is the ratio of the number of relevant nodes retrieved to the total number of relevant and irrelevant nodes retrieved.

$P = \frac{A}{A+C} \times 100\%$ where C = number of irrelevant nodes retrieved

F-score = $2.\frac{P \times R}{P+R}$

This performance measure determines the accuracy of our proposed approach.

3. **Statistical Analysis** – it shows the statistical significance of our results. For each set of experiments, we calculate the 95% confidence interval (C.I) to measure the reliability of our system (Levine, 2010). We have specified the interval by an upper bound (U) and a lower bound (L), and we are confident that in the 95% of the cases, the mean of our sample data will be within the confidence limits L and U.

95% $C.I = \bar{x} \pm 1.96 \times \frac{s}{\sqrt{n}}$ where, $\bar{x} =$ mean of the samples, $n =$ size of the samples, and $s =$ Standard Deviation = $\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$

4. **Application improvement** – it shows how the influence spread achieved by our OBIN algorithm improves the influence spreads that can be achieved by standard IM approaches like CELF of Leskovec et al. (2007) and T-IM of Ahmed et al. (2013).

4.2 PERFORMANCE ANALYSIS

Table 40 shows the accuracy measure of CELF and T-IM and proposed OBIN. We can see that the recall value of OBIN is 93.7%, this is because 90 relevant nodes (out of 2407 relevant and irrelevant nodes) were not extracted by OBIN, and also 26 more nodes might be relevant but could not extracted due to information in language other than

English. Precision is 98.24%, this is because 31 irrelevant nodes are extracted (out of 2407 nodes) by OBIN. With the same dataset, we applied CELF and T-IM, and observed that OBIN is dramatically better in precision and F-score with slight loss in recall.

	Precision	**Recall**	**F – score**
CELF	80.02%	92.7%	85.4%
T-IM	81.36%	96.09%	88.1%
OBIN	98.24%	93.71%	95.3%

Table 40 Comparison of discovering influential nodes by CELF, T-IM and OBIN

Table 41 shows the values of 95% Confidence Interval for improvement in nodes used for our proposed OBIN with CELF. For example, in table 41, for a 100 influential nodes, the 95% C.I based on number of likes for each node is between 271.42 and 612.09, and based on computed influence score for each node is between 62.74 and 68.47. This means that, the mean of percentage improvement will not be less than 62.74 and will not be more than 68.47 in terms of influence score, 95% of the time.

Top Nodes	**Opinions**	**OBIN**			**CELF**		
		Lower Bound	Average Bound	Upper Bound	Lower Bound	Average Bound	Upper Bound
100	Approve	271.42	441.75	612.09	117.67	465.39	813.12
	Influence score	62.74	65.61	68.47	39.2	55.27	71.33
200	Approve	154.66	243.95	333.25	102.5	181.2	259.9
	Influence score	56.27	58.87	61.48	29.12	33.39	37.67
300	Approve	108.49	169.17	229.85	37.29	52.27	67.24
	Influence score	51.06	53.48	55.9	11.05	17.97	24.89

Table 41 Comparison of 95% CI for different number of discovered influential nodes

As we increase the discovered number of influential nodes, we can see in table 41 that OBIN improves CELF, this is because CELF discovered more irrelevant nodes along with relevant nodes hence the computed influence score reduces.

Figure 15 shows the influence spread over network by different algorithms. We measure the influence spread by measuring the number of nodes activated (influenced) by the influential nodes extracted. As we see, with small number of nodes, CELF and T-IM give better performance in influence spread, but as we increase the number of nodes, OBIN performs better in influence spread. This is because, for a specific product, CELF and T-IM discovers relevant nodes along with more irrelevant nodes which reduce the performance.

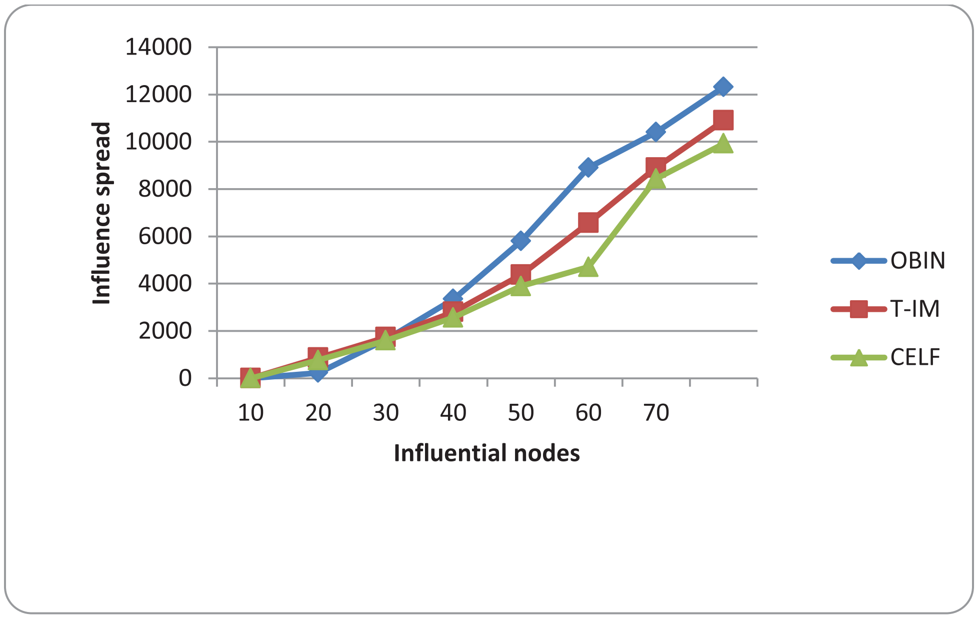


Figure 15 Comparison of influence spread by different number of influential nodes

To compare runtime of OBIN with CELF and T-IM, we recorded time required to select influential nodes of different size. Figure 16 reports the runtime comparison on Facebook by extracting nodes for OBIN by executing product specific SQL query, and by extracting nodes for CELF and T-IM through randomly choosing popular nodes.

As shown in figure 16, OBIN takes longer than CELF as the size of the required set of influential nodes increases. This was expected as OBIN performs additional operations such as opinion mining and computation of influence score. For example, to measure the popularity score and influence score, OBIN requires to extract comments, apply NLProcessor and Apriori frequent pattern to determine the polarity, aggregate all kinds of opinions such as likes, re-shares, positive/negative comments. As shown in figure 16, OBIN takes slight longer than T-IM which is not that much significant. This is because, T-IM requires crawling the whole network to extract actions performed by users, and for each user crawl the network again to process all other users who perform the action after any user and for each user process all friends of that user who did not perform the action after any user. Moreover, T-IM also performs additional operations such as delete and swap which is computationally expensive as it requires to remove/swap each element in node set with every element not in node set but in network.

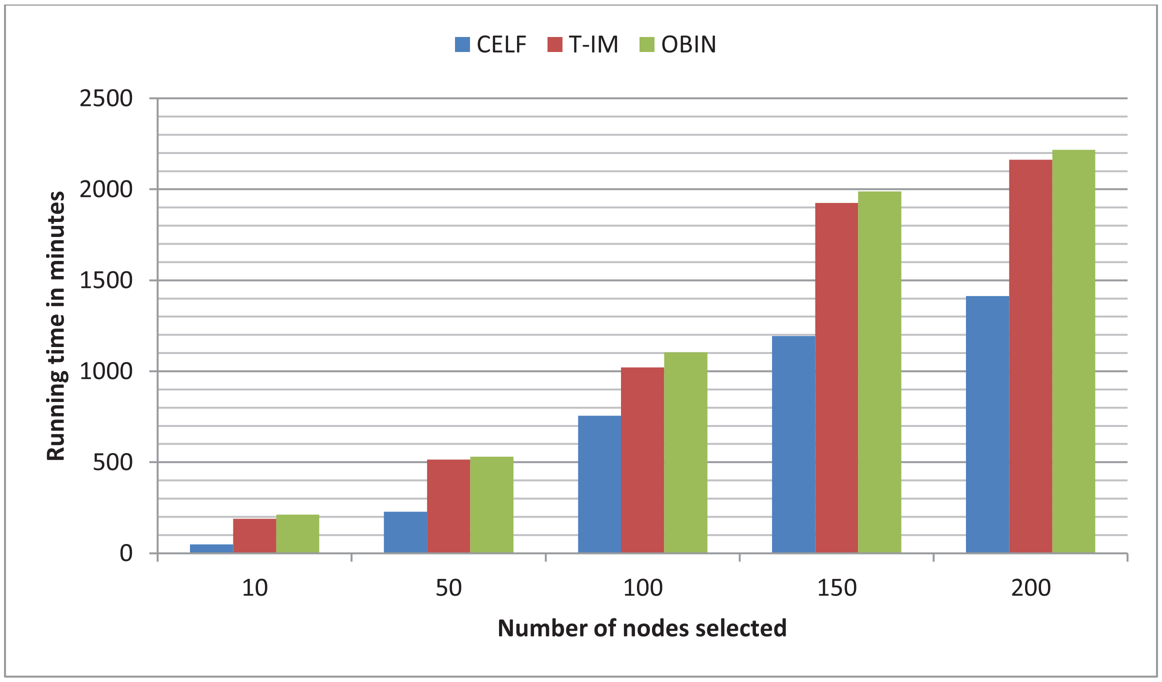


Figure 16 Running time of different algorithms

CHAPTER 5
CONCLUSION AND FUTURE WORKS

In this thesis, we proposed an effective method for discovering relevant influential nodes from friendship network which enables more focused target marketing than existing IM algorithms. However, previous research consider opinion mining only in user-service network of single product page, where OBIN mines opinions from complex user-user relationship network of multiple posts, multiple products, considering both implicit and explicit opinions. Experimental results show that the proposed technique performs markedly better than the existing general IM methods. Moreover, the information extracted and computed from friendship network further can be applied to provide recommendation systems to improve business opportunity. The resultant data stored in the data warehouse can also answer some crucial business queries such as "which relevant post is most popular?", "who are the most influential and influenced users on the post?", "who like the product and who do not", "how do the users connected to each other?".

The IN generation process in social network has a similar view as techniques used by Google to search important web pages. Google uses Page Ranking and number of hits techniques, a web page is crawled by the Google crawler, moving from link to link and building an index page that has certain keywords matched with the search query, and provide that page to the query generator. The differences between PageRank system and our proposed approach are that, PageRank algorithm provides relevant webpages based on the keywords explicitly described in the "keyword" tag of HTML webpage, or in advance, the keywords are mentioned several times in the web document. In our proposed approach, we also have to crawl the network, but in addition we need to find out the user – user relationships based on the opinions expressed implicitly or explicitly on published posts in a timely manner that we need to mine to extract the sentiment of the opinions. However, in future we would like to further apply techniques learned in influence network generation with Google page ranking algorithm, that could result in new insights into the influence maximization problems.

However, as the network grows dramatically, our system goes slow down due to execution time in large-scale network. As shown in our experiment, OBIN has longer execution time than CELF and slight longer than T-IM, this difference is not significant compared to the discovered influential users and influence spread over the network. However, in future, we want to improve this run time due to network evolution and network dynamics.

REFERENCES

1. Agrawal, R., Rajagopalan, S., Srikant, R., & Xu, Y. (2003, September). Mining newsgroups using networks arising from social behavior. *In proceedings of the 12th international conference on World Wide Web* (pp. 529-535). New York, USA: ACM.

2. Agrawal, R., Srikant, R., & others. (1994, June). Fast algorithms for mining association rules. *In proceedings of the 20th Int. Conf. Very Large Data Bases, VLDB*, (pp. 487-499). San Jose, CA.

3. Ahmed, S., & Ezeife, C. (2013, March). Discovering Influential Nodes from Trust Network. *In proceedings of the 28th ACM Symposium on Applied Computing (ACMSAC) – Data Mining Track* (pp. 121-128). Coimbra, Portugal: ACM.

4. Bird, C., Gourley, A., Devanbu, P., Gertz, M., & Swaminathan, A. (2006, May). Mining email social networks. *In proceedings of 2006 international workshop on Mining software repositories* (pp. 137-143). Shanghai, China: ACM.

5. Bonchi, F., Castillo, C., Gionis, A., & Jaimes, A. (2011, April). Social network analysis and mining for business applications. *ACM Transactions on Intelligent Systems and Technology (TIST), 2*(3), 22.

6. Brill, E. (1994). A simple rule-based part of speech tagger. *In proceedings of the workshop on Speech and Natural Language* (pp. 112-116). Trento, Italy: Association for Computational Linguistics.

7. Cooley, R., Mobasher, B., & Srivastava, J. (1997, November). Web mining: Information and pattern discovery on the world wide web. *In proceedings of the ninth IEEE International Conference on Tools with Artificial Intelligence, 1997. Proceedings.* (pp. 558-567). IEEE.

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

9. Cover, T., & Hart, P. (1967, January). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory, 13*(1), 21-27.

10. Craswell, N., Hawking, D., Vercoustre, A., & Wilkins, P. (2001, April). P@ noptic expert: Searching for experts not just for documents. *In Ausweb 2001.* Queensland, Australia. Retrieved from http://es.csiro.au/pubs/craswell.ausweb01.pdf

11. Dave, K., Lawrence, S., & Pennock, D. (2003, May). Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. *In proceedings of*

the 12th international conference on World Wide Web (pp. 519-528). Budapest, Hungary: ACM.

12. Ding, X., Liu, B., & Yu, P. (2008, May). A holistic lexicon-based approach to opinion mining. *In proceedings of the international conference on Web search and web data mining* (pp. 231-240). Budapest, Hungary: ACM.

13. DiNucci, D. (1999). Fragmented future. *Print, 53*(4), 32.

14. Du, N., Wu, B., Pei, X., Wang, B., & Xu, L. (2007, August). Community detection in large-scale social networks. *In proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis* (pp. 16-25). San Jose, California: ACM.

15. Ezeife, C., & Mutsuddy, T. (2012, October-December). Towards Comparative Mining of Web Document Objects with NFA: WebOMiner System. *International Journal of Data Warehousing and Mining (IJDWM), 8*(4), 1-21.

16. Ezeife, C., Lu, Y., & Liu, Y. (2005, August). PLWAP sequential mining: open source code. *In proceedings of the 1st international workshop on open source data mining: frequent pattern mining implementations* (pp. 26-35). Chicago, USA: ACM.

17. Facca, F., & Lanzi, P. (2005). Mining interesting knowledge from weblogs: a survey. *Data & Knowledge Engineering, 53*(3), 225-241.

18. Flake, G., Lawrence, S., & Giles, C. (2000, August). Efficient identification of web communities. *In proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 150-160). Boston: ACM.

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

20. George, K. (1949). Zipf. Human behavior and the principle of least effort. *An Introduction to Human Ecology*. Boston: Houghton-Mifflin.

21. Girvan, M., & Newman, M. (2002, June). Community structure in social and biological networks. *National Academy of Sciences, 99*(12), 7821.

22. Gomez, V., Kaltenbrunner, A., & Lopez, V. (2008, April). Statistical analysis of the social network and discussion threads in slashdot. *In proceedings of the 17th international conference on World Wide Web* (pp. 645-654). Beijing, China: ACM.

23. Han, J., & Kamber, M. (2006). *Data mining: concepts and techniques.* Morgan Kaufmann.

24. Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The Elements of Statistical Learning: Prediction, Inference and Data Mining*. New York, USA: Springer.

25. Hu, M., & Liu, B. (2004, August). Mining and summarizing customer reviews. *In proceedings of the 10th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 168-177). Seattle, Washington, USA: ACM.

26. Jin, W., Ho, H., & Srihari, R. (2009, June). OpinionMiner: a novel machine learning system for web opinion mining and extraction. *In proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 1195-1204). Paris, France: ACM.

27. Kadri, O., & Ezeife, C. (2011, March). Mining Uncertain Web Log Sequences with Access History Probabilities. *In proceedings of the 26th ACM Symposium on Applied Computing, DTTA - Database Theory, Technology, and Application* (pp. 1064-1065). TaiChung, Taiwan: ACM.

28. Kempe, D., Kleinberg, J., & Tardos, E. (2003, August). Maximizing the spread of influence through a social network. *In proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 137-146). Washington, USA: ACM.

29. Kosala, R., & Blockeel, H. (2000, July). Web mining research: A survey. *ACM Sigkdd Explorations Newsletter, 2*(1), 1-15.

30. Krebs, V. (2002, April). Uncloaking terrorist networks. *First Monday, 7*(4-1). Retrieved from http://firstmonday.org/ojs/index.php/fm/article/view/941/863

31. Kriegel, H., Kroger, P., Sander, J., & Zimek, A. (2011, June). Density-based clustering. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 1*(3), 231-240.

32. Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., & Glance, N. (2007, August). Cost-effective outbreak detection in networks. *In proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 420–429). New York, USA: ACM.

33. Leskovec, J., Lang, K., & Mahoney, M. (2010, April). Empirical comparison of algorithms for network community detection. *In proceedings of the 19th international conference on World wide web* (pp. 631-640). North Carolina: ACM.

34. Levine, D. M., Krehbiel, T. C., & Berenson, M. L. (2010). *Business Statistics.* Prentice Hall.

35. Li, X., Guo, L., & Zhao, Y. (2008, April). Tag-based social interest discovery. *In proceedings of the 17th international conference on World Wide Web* (pp. 675-684). Beijing, China: ACM.

36. Mabroukeh, N., & Ezeife, C. (2010, November). A taxonomy of sequential pattern mining algorithms. *ACM Computing Surveys (CSUR), 43*(1), 3:1-3:41.

37. MacQueen, J., & others. (1967, June). Some methods for classification and analysis of multivariate observations. *In proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, *1*, p. 14. California, USA.

38. Manning, C., & Schutze, H. (1999). *Foundations of statistical natural language processing.* Cambridge, Massachusetts: MIT press.

39. McCallum, A., Nigam, K., & others. (1998, July). A comparison of event models for naive bayes text classification. *In proceddings of the AAAI-98 workshop on learning for text categorization. 752*, pp. 41-48. Citeseer.

40. Miller, G., Beckwith, R., Fellbaum, C., Gross, D., & Miller, K. (1990). Introduction to wordnet: An on-line lexical database. *International journal of lexicography, 3*(4), 235-244.

41. Mishne, G., & Glance, N. (2006, May). Leave a reply: An analysis of weblog comments. *In proceedings of the third annual workshop on the WWW 2006 Weblogging ecosystem: Aggregation, Analysis and Dynamics* (pp. 22-26). Edinburgh, UK: WWW06.

42. Nigam, K., & Hurst, M. (2006). Towards a robust metric of polarity. *Computing Attitude and Affect in Text: Theory and Applications*, 265-279.

43. *NLProcessor - Text Analysis Toolkit*. (n.d.). Retrieved from infogistics: http://www.infogistics.com/textanalysis.html

44. Pang, B., Lee, L., & Vaithyanathan, S. (2002, July). Thumbs up?: sentiment classification using machine learning techniques. *In proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10. 10*, pp. 79-86. Philadelphia: Association for Computational Linguistics.

45. Park, N., & Lee, W. (2004, March). Statistical grid-based clustering over data streams. *ACM SIGMOD Record, 33*(1), 32-37.

46. Quinlan, J. (1986). Induction of decision trees. *Machine learning, 1*(1), 81-106.

47. Richardson, M., & Domingos, P. (2002, July). Mining knowledge-sharing sites for viral marketing. *In proceedings of the eighth ACM SIGKDD international conference*

on Knowledge discovery and data mining (pp. 61-70). Edmonton, Alberta, Canada: ACM.

48. Salton, G., Wong, A., & Yang, C. (1975, November). A vector space model for automatic indexing. *Communications of the ACM, 18*(11), 613-620.

49. Santorini, B. (1990, July). *Part-of-speech tagging guidelines for the Penn Treebank Project (3rd revision).* University of Pennsylvania, Department of Computer & Information. ScholarlyCommons.

50. Sun, J., Faloutsos, C., Papadimitriou, S., & Yu, P. (2007, August). GraphScope: Parameter-free Mining of Large. *In proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 687-696). San Jose, California: ACM.

51. Tang, J., Sun, J., Wang, C., & Yang, Z. (2009, June). Social influence analysis in large-scale networks. *In proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 807-816). Paris, France: ACM.

52. Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., & Su, Z. (2008, August). ArnetMiner: extraction and mining of academic social networks. *In proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 990-998). Las Vegas, Nevada, USA: ACM.

53. Turney, P. (2002, July). Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. *In proceedings of the 40th Annual Meeting on Association for Computational Linguistics* (pp. 417-424). Philadelphia: Association for Computational Linguistics.

54. Watts, D., & Strogatz, S. (1998, June). Collective dynamics of 'small-world'networks. *nature, 393*(6684), 440-442.

55. Willett, P. (2006). The Porter stemming algorithm: then and now. *Program: electronic library and information systems, 40*(3), 219-223.

56. Zhongbao, K., & Changshui, Z. (2003, March). Reply networks on a bulletin board system. *Physical Review E, 67*(3), 036117.

57. Ziegler, C., & Lausen, G. (2005). Propagation models for trust and distrust in social networks. *Information Systems Frontiers, 7*(4-5), 337-358.

VITA AUCTORIS

Tamanna Mumu was born in 1982 in Sylhet, Bangladesh. She received her Bachelor's degree in Computer Science and Engineering from Shah Jalal University of Science and Technology, Sylhet, Bangladesh in 2007. She completed her M.Sc in Computer Science from the University of Windsor, Ontario, Canada from September 2011 to April 2013. Her research interests include data mining, social network analysis and machine learning.