Systèmes de recommandation sociaux et sémantiques

Dalia Sulieman

UNIVERSITÉ de Cergy Pontoise

École doctorale Sciences et Ingénierie

Thèse Présentée par

Dalia Sulieman

pour obtenir le grade de Docteur de l'Université de Cergy Pontoise
Spécialité :
Sciences et Technologie de l'Information et de la Communication
STIC

Towards Semantic-Social Recommender Systems

Thèse soutenue le 30 janvier 2014 devant le jury composé de :

- Madame Céline Rouveirol — Rapporteur
- Monsieur Philippe Rigaux — Rapporteur
- Madame Lynda Tamine-Lechani — Examinateur
- Monsieur Nicolas Spyratos — Examinateur
- Monsieur Dominique Laurent — Directeur de thèse
- Madame Maria Malek — Examinateur
- Monsieur Hubert Kadima — Examinateur

Abstract

Recently social networks have attracted many research efforts due to the important role they play in everyday life. Among the numerous applications of social networks, we mention recommendation systems, whose aim is to identify users likely to be interested in a given item, based on their tastes and opinions. In this thesis we propose a new approach to recommender systems, called semantic-social recommender systems. The aim of such a system is to recommend a given item to a group of users connected via a social network. For that, we propose two classes of algorithms based on depth-first search and breadth-first search strategies.
The contribution of our proposed algorithms, called semantic-social depth-first search (SSDFS) and semantic-social breadth-first search (SSBFS), is that two types of information are used, namely semantic information and social information. Semantic information refers to users preferences and items features, while social information refers to users centrality in the social network. In order to test our algorithms, we consider two real datasets, namely Amazon.com and MovieLens, and we compare our algorithms with classical recommendation algorithms known as item-based collaborative filtering algorithm and hybrid recommendation algorithm. Our experimental results show that, while the recommendations computed by our algorithms are generally of better quality (in terms of precision, recall and F-measure) than that of classical algorithms, a significant improvement in performance is achieved. This is so because our algorithms explore a small part of the graph, instead of the whole graph as done by the classical algorithms.

Résumé

Récemment les réseaux sociaux ont fait l'objet de nombres de sujets de recherche à cause du rôle important qu'ils jouent dans notre vie. Parmi les nombreuses applications des réseaux sociaux, nous mentionnons les systèmes de recommandation, dont le but est d'identifier les utilisateurs susceptibles d'être intéressés par un article donné, sur la base de leurs goûts et opinions. Dans cette thèse, nous proposons une nouvelle approche des systèmes de recommandation, appelé système de recommandation sémantique et social, dont le but est de recommander un élément à un groupe d'utilisateurs impliqués dans un réseau social. Pour ce là, nous proposons deux classes d'algorithmes basés sur la recherche en profondeur et la recherche en largeur dans un graphe. La contribution de nos algorithmes, appelés algorithme de recherche sémantique et sociale en profondeur (SSDFS) et algorithme de recherche sémantique et sociale en largeur (SSBFS), est que deux types d'information sont utilisés : information sémantique et information sociale. L'information sémantique fait référence aux préférences des utilisateurs et aux caractéristiques des éléments, alors que l'information sociale fait référence à la centralité des utilisateurs dans le réseau social.
À fin de tester nos algorithmes, nous avons considéré deux jeux de données réelles, Amazon.com and MovieLens, et nous avons comparé nos algorithmes avec les algorithmes classiques de recommandation que sont l'algorithme de filtrage collaboratif et l'algorithmes hybride. Nos expériences montrent que, alors que les résultats de nos algorithmes sont généralement de meilleure qualité (en termes de précision, rappel et F-mesure) que les algorithmes classiques, leur performance est notablement améliorée. Ce résultat s'explique par le fait que nos algorithmes explorent une petite partie du graphe, contrairement aux algorithmes classiques qui explorent le graphe en entier.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Motivation

Nowadays, social networks are becoming great resources of information, especially when they have a huge number of actors e.g. *Facebook* has 1.15 billion active users, and more than 15 million pages about business brands and organizations; and *LinkedIn* has 238 million active users, and more than 2.1 million pages about different companies.
Table 1.1 shows some important information about the most famous social networking sites, and their revenue in US dollar.
Every day a great flow of information is diffused, and enormous information is shared between actors in social networks, as shown in Table 1.1. Furthermore, actors in social networks show a wide range of variations in tastes, interests, attitudes and activities. For that, studying, analyzing and exploring the hidden and the visible sides and characteristics of the actors and their social ties in social networks are very important topics; and they are very considerable in certain domains such as: sociology and psychology [134], searching for experts in certain domains [198], searching the web [150], searching for customers for recommendation [125].

On the other hand, in the domain of recommendation, 53% of people using Twitter recommend products via their tweets; and 42% of marketers find Facebook an important environment for them. Figure 1.1 shows the effectiveness of marketing through different types

Table 1.1: Social networks statistics on 2013

Network	Active users	Launched	Data per day	Revenue
Facebook	1.15 billion	February 2004	500 terabytes	5.1 billion $
Youtube	1 billion	February 2005	100 million videos	3.7 billion $
Google plus	500 million	June 2011	24 petabytes	50 billion $
LinkedIn	259 million	May 2003	26 billion message	972 million $
Twitter	200 million	July 2006	400 million tweets	140 million $

of social networks[1]. In this Figure we find that Facebook and Twitter have significant effects on users decisions and purchases.

Indeed, the important role of social networks relies on the fact that,

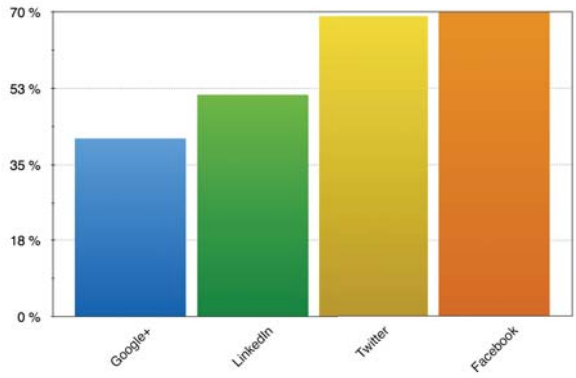

Figure 1.1: The effects of marketing through different types of social networks on users purchases

they are rich in resources from all the types of information related to actors and their different kinds of interests and interactions.

Although, all the huge amount of information, stacked in social networks, can lead to information overload. Especially when there are no suitable methods to analyze and manipulate all this information.

Information overload can appear when users receive a massive amount of information, that they are not able to recognize or to process, leading them to poor understanding and limited decision making [129]. Information overload is not only related to social net-

[1]http://social media.com

Table 1.2: Search Engines

Network	Launched	Visits per month
Google	September 1998	88 billion visits
Yahoo	March 1994	9.4 billion visits
Bing	June 2009	4.1 billion visits

works, but it happens in the whole web especially when using search engines, Table 1.2 shows how many times, search engines are visited every month. Moreover, it is important to mention that not all the visited sites are correlated to the searched subjects.

For instance, on November 2006, according to the Economist.com in their article *The phone of the future* they mentioned that: "People read around 10 MB worth of material a day, hear 400 MB a day, and see one MB of information every second"[2]. So, how can people process such an amount of information, and how can they find their suitable interests.

Since the year 2013, this information overload has been incredibly augmented, and can be easily found and widely expected specially in social networks, for example:

- In Facebook: a person having a considerable number of friends, e.g. more than $10,000$ friends, will not be able to manage all his/her friends in the list; and this person needs intelligent methods to manage and search his/her contacts according to their priority and importance.
- Searching LinkedIn: human resources will spend an enormous time and effort, on LinkedIn, in order to find experts in a certain domain.
- Searching social networks for recommendations is very costly and a complex process, using the classical searching methods.

Moreover, in the literature of recommender systems we find that[3]:

- Chris Anderson in "The Long Tail" has mentioned that: we are leaving the age of information and entering the age of recommendation.

[2]http://www.economist.com/node/8312260

[3]Recommenders Systems tutorial slides from the European Summer School of Information Retrieval by Alexandros Karatzoglou 2013

- CNN Money in "The race to create a *smart* google": the web, they say, is leaving the era of search and entering the one of discovery. What's the difference? Search is what you do when you are looking for something. Discovery is when something wonderful that you did not know existed, or did not know how to ask for, finds you.

As information searching is becoming a complex process, especially in social networks, we propose a group of special recommendation algorithms to maintain the information overload in social networks; and to mention that the age of search should have an end, by replacing the searching act with the recommendation act. Therefore, instead of wasting users time and efforts looking for interesting information, the recommender system will guide the information to their corresponding users. This is evident in some social networks e.g. Facebook has the application "People you may know", and LinkedIn has the application "People you may have worked with".

Moreover, recommender systems have shown a significant importance and success e.g. in Netflix two thirds of the rented movies have been chosen based on recommendation, 38% of Google news are generated based on recommendation and 35% of purchases in Amazon.com depend on recommendations.

Furthermore, from "Digital Intelligence Today" in the article "Word of Mouth Still Most Trusted Resource Says Nielsen; Implications for Social Commerce", it has been indicated that "the latest Global Trust in Advertising report, proved that people do not trust advertising, as much as they do trust recommendations from friends and consumer opinions expressed online[4]". Figure 1.2 shows the differences of importance between social recommendation (recommendation between friends) and the other types of recommendations. Thus, according to that recommendation in social network, where relations between users are based on common tastes and interests, can show significant importance and effectiveness.

Finally, we can say that recommender systems in social networks are important for two main reasons:

- Manage the huge amount of information in social networks.

[4]http://digitalinnovationtoday.com

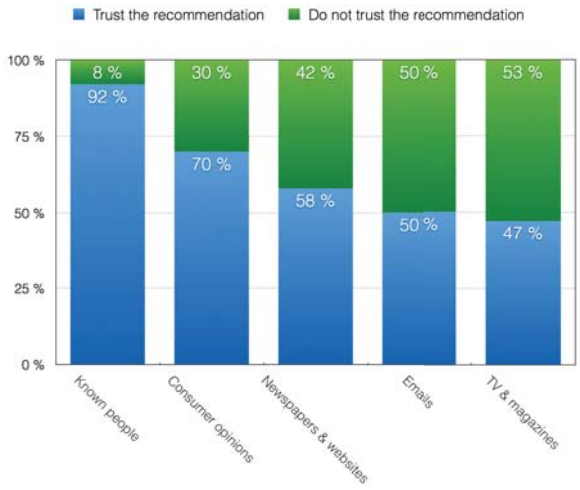

Figure 1.2: Nielsen latest global trust in advertising April 2012

- Improve the recommendation by recommending items to users based on their tastes, connections and positions in the social network.

In this thesis, we are interested in studying and analyzing social networks, and particularly the users who are connected via *co-purchases network*, in order to extract important information about the preferences and the characteristics of users. This aims to propose a system used to determine users' concerns, interests and problems, which leads to satisfying users' needs and to provide the proper recommendation for them; with the respect to high accuracy and performance. We propose to apply recommender systems methods, using social network analysis measures including user centrality and graph searching algorithms. Moreover we take into account the semantic information about users preferences and item features, that are represented in their ontological profiles.

So, our thesis outlines two topics: social networks and recommender systems, in order to use them in a model that recommends items to users connected via a social network or co-purchases network. We are mainly concerned about by improving the performance and the accuracy of the system.

1.1.1 Social Networks

Social networks are networks in which vertices represent people, called actors; and edges represent social interactions between people, called social ties or relationships.

Social networks can be modeled by a graph G in which actors are represented by a set of graph vertices $V(G)$ and the social ties are represented by a set of graph edges $E(G)$.

Graph-based searching algorithms, in social networks, are used for several purposes, such as: *Finding experts* in a given domain e.g. seeking for a java expert in a java online help-seeking community [198]. *Ranking* network actors according to their relation to a given subject e.g. ranking emails according to their correlation to users' interests [55]. *Searching engines* e.g. "google.com" [150], and *Recommender Systems* [125]. Recommender systems in social networks is the focus of our thesis.

Indeed, social networks have a vast range of applications, that require very fast and very effective graph-based searching techniques; generally, graph searching algorithms can be classified into three main categories: general computational based algorithms, network structure based algorithms and similarity based algorithms [197].

In this thesis we are interested in *Network Structure* algorithms, mainly: vertex centrality [4, 102].

1.1.2 Recommender systems

Recommender systems are very important, because they help users to find interesting items based on their tastes and preferences. In general, recommender systems have three main categories: content-based recommendation which recommend items to users according to the contents of user profile; collaborative-filtering recommendation which recommends items to users according to their historical purchases; and hybrid recommendation that combines recommendation methods from content-based and collaborative-filtering.

In this thesis we are interested in collaborative-filtering methods, applied on the graph. Graph-based recommender systems use several methods as shortest path [84, 85, 131, 6], random walk [87] and PageRank [68, 104]. In addition, we are interested in the content-based methods by considering the ontological contents of user preferences and item features.

1.1.3 User profile

User profile contains information about users, such as: name, age, friends; or it contains information about users' preferences such as items the users liked in the past. In social networks and recommender systems, user profile is a great source of information, so it is very important to be appropriately treated and operated.

According to the literature, user-profile has several types of representation [64] as follows: a bag of words, a vector of weighted keywords and ontological profiles using ontology or taxonomy.

In this thesis we use taxonomy as a special case of ontology to represent user preferences and item features.

1.2 Problem definition and proposed solution

Our approach recommends an *item* (input item) to users, these users are members of social network or co-purchases network, by submitting a recommendation query containing this *item*. Then the algorithm navigates the network and explores users' semantic profiles and users' social connections. The output of the recommendation algorithm is a *recommendation list* containing all the possible relevant users to the input item.

For that, in our approach we propose a graph-based recommender system, that includes (a) users and items represented via semantic tree profile (taxonomy profile), (b) user centrality and position in the network and (c) graph searching algorithm to explore the semantic and the social characteristics of the user in the network.

Our proposed approach supposes:

- A network connecting users, these users have various characteristics and interests; also, users have (a) centrality values according to their position in the network and (b) a weighted social ties connecting them with the other members in the network .
- A user profile, that describes users' interests and preferences. This profile has a semantic tree structure.
- An item profile that describes the item's features, by referencing it to a domain ontology. Also we suppose that, both of user profile and item profile have the same structure (the semantic tree structure).

1.2.1 Semantic-social recommendation scenario

Starting from a database, relating users with their preferred items as *Amazon.com* and *MovieLens*, the semantic-social recommender system extracts a user-item bipartite graph. Then the semantic-social recommender system builds the users one-mode projection, to finally apply the recommendation algorithm on this graph, we call the users one-mode projection graph a users co-purchases network.

Figure 1.3 shows the process of building the users co-purchases network, on which we submit the recommendation query and we apply the semantic-social recommendation algorithm. This procedure has two stages, stage (a) in which a user-item bipartite graph is built from user-item database, and a stage (b) which contains the users co-purchases network where the recommendation query is submitted as an input and a recommendation list of recommended users is given as an output.

Our approach is designed in a modular way which allows the

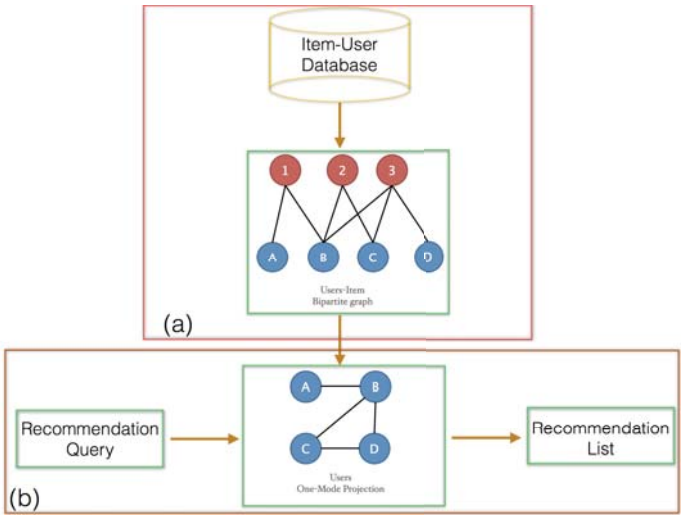

Figure 1.3: Semantic-social recommendation applied on item-user database

semantic-social recommendation algorithm to be applied on social networks as Facebook, LinkedIn and Twitter, by only considering the part (b) of our model.

1.3 Contributions

In this thesis we propose a new approach named *semantic-social recommender system*. This approach contains two different types of information *semantic information* and *social information*.

- Semantic information by using tree structure to represent user preferences and item features. This structure helps the recommender system to classify and specify the system's information.
- Social information by representing the dataset in a user-item bipartite graph to finally construct the users co-purchases network. Then we employ the structural information of the co-purchases network, as social information, in the recommendation.

The semantic-social recommendation algorithm is proposed to search the co-purchases network using a defined heuristic based on semantic information and social information. For that we propose the following measures, algorithms and experimental validation:

Proposed measures

1. User-item semantic relevancy measure, in which we propose a semantic relevancy measure to find the relevance between user preferences and item features, which is specified in the recommendation query. In fact, user-item relevancy measure is used to determine the closeness between the item features and the user taste, in order to achieve the recommendation.
2. Social heuristics measures, in which we propose to define and to compare several heuristics. These heuristics are based on (a) user centrality in the co-purchases network and (b) social ties between users in the co-purchases network.

Proposed algorithms

1. Semantic-social depth-first search SSDFS, we propose to integrate the user-item relevancy measure and the social heuristics measures with depth-first search algorithm for exploring the co-purchases network, in order to achieve the recommendation.

2. Semantic-social breadth-first search SSBFS, we propose the same method as SSDFS by integrating the user-item relevancy measure and the social heuristic measures with breadth-first search algorithm, for exploring the co-purchases network.

Experiments and validation

1. We propose to compare the semantic-social depth-first search SSDFS with the semantic-social breadth-first search SSBFS.
2. We propose to compare several user centralities integrated with the heuristics of semantic-social depth-first search SSDFS, and semantic-social breadth-first search SSBFS. These centralities are degree centrality, closeness centrality and betweenness centrality. Furthermore, we propose another heuristic that combines degree centrality with betweenness centrality.
3. Datasets, we propose to use real datasets with different sizes and types. These datasets are MovieLens dataset which contains ratings about movies given by the users; and Amazon dataset which contains information about users and their purchases.
4. Validation, we propose to test and to compare the accuracy and the performance of our proposed algorithms, with two classical recommendation algorithms, *item-based collaborative filtering* algorithm and *hybrid recommendation* algorithm. These algorithms are tested over MovieLens dataset and amazon datasets.

1.4 Thesis overview

This thesis is organized in six chapters, Figure 1.4 shows how our thesis is organized. These chapters are summarized as follows:

Chapter 1: Introduction In this chapter we briefly outline the global ideas of the semantic-social recommendation.

Chapter 2: Social Network Analysis: a survey In this chapter we introduce an overview about social networks and social network analysis methods. Starting from the fundamental definitions in both

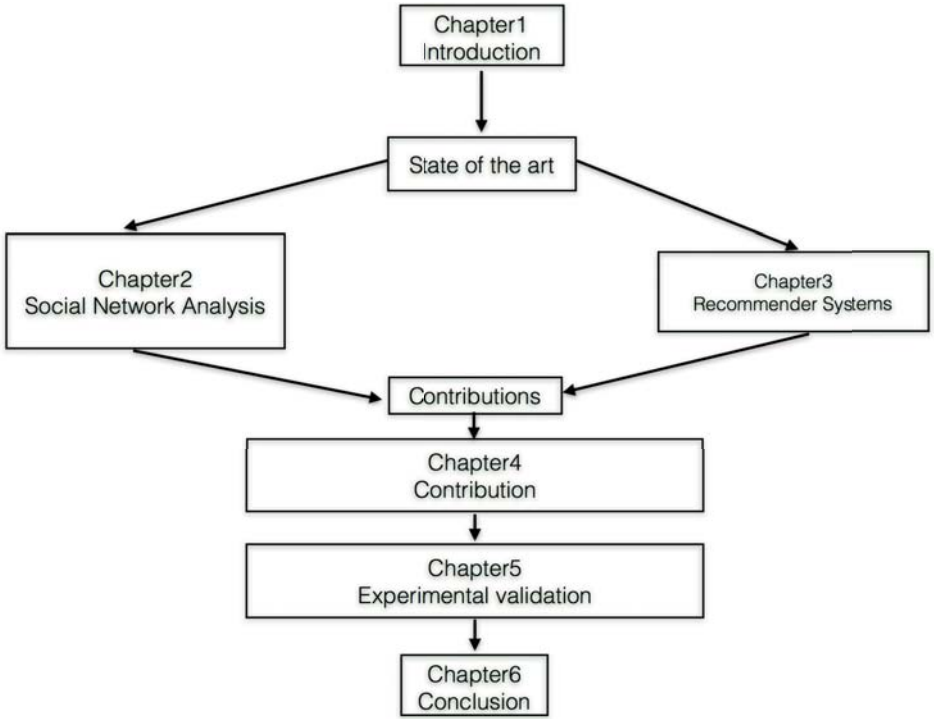

Figure 1.4: Thesis Structure

graph theory and social network. Passing by the characteristics and the models of social networks. Then we introduce some definitions of the most important social network analysis measures. Also, we present some of the well known algorithms used for community detection and for searching social networks. Then we present examples of some of the well known social network, and finally we present some of the softwares used for social network analysis.

Chapter 3: Recommender Systems: a survey In this chapter we introduce a brief survey about recommender systems. Recommender systems have three main types: Content-based, Collaborative filtering and hybrid recommendation. In content-based recommendation the content of user interests and item features are involved in the recommendation process. In collaborative filtering recommender systems, recommendations are based on the ratings of like

minded users. Hybrid recommender systems combine content-based methods with collaborative filtering methods, in order to bypass the limitations of content-based and collaborative filtering systems. Moreover, we detail some of the well known recommendation algorithms that are categorized into two main categories: Model-based and Memory-based. Then, we discuss some of Graph-based recommender systems. Finally, we introduce some of the most famous methods that are used to evaluate recommender systems.

Chapter 4: Semantic-Social Recommender System In this chapter we introduce our approach the *semantic-social recommender system*. The semantic-social approach combines two types of information: semantic information with social information.

Chapter 5: Experimental results We apply our proposed algorithms on MovieLens and Amazon datasets; and we compare our proposed algorithms with the classical recommendation algorithms item-based and hybrid recommendation algorithms, and we use precision, recall and F-measure as accuracy measure to assess the accuracy of our algorithms. Moreover, we evaluate the performance of the algorithms by comparing the data coverage and the time of execution between the different algorithms. Our experiments show better results than collaborative filtering and hybrid recommendation algorithms, by significantly reducing the data coverage and the execution time; while ensuring a slightly better accuracy.

Chapter 6: Conclusion this chapter outlines a brief summary of the thesis, and our perspectives for future work.

1.5 Published work

National conferences

1. Combining social and semantic information for recommandation : comparative study Dalia Sulieman, Maria Malek, Hubert Kadima, Dominique Laurent conférence MARAMI 2013.
2. Semantic social breadth-first search and depth-first search recommendation algorithms Dalia Sulieman, Maria Malek, Hubert Kadima, Dominique Laurent conférence MARAMI 2012.

3. Recommendation algorithm as application in Semantic Social Network. Dalia Sulieman, Maria Malek, Hubert Kadima and Dominique Laurent. EGC, Bordeaux, France 2012.

4. Towards Semantic Social Recommendation Algorithm. Dalia Sulieman, Maria Malek, Hubert Kadima and Dominique Laurent conférence MARAMI 2011.

5. Vers un algorithme de recherche guidé pour la recommendation d'un expert dans un réseau professionnel. Maria Malek, Dalia Sulieman Journée doctorales: fouille de grands graphes, Toulouse 13 Octobre 2010.

International conferences

1. Exploiting Semantic and Social Information in Recommendation Algorithms Dalia Sulieman, Maria Malek, Hubert Kadima, Dominique Laurent, volume 0146 of the Communications in Computer and Information Science series ISIP2012 pp.91 − 100 Springer.

2. Graph Searching Algorithms for Semantic-Social Recommendation. Dalia Sulieman, Maria Malek, Hubert Kadima, Dominique Laurent. ASONAM 2012 pp.773 − 738 IEEE [184].

3. Exhaustive and guided Algorithm for recommendation in a professional social network. Maria Malek, Dalia Sulieman. 7th conference on application of social network analysis Zürich 15-17 September 2010

4. A graph based algorithm for recommendation in a professional social network. Maria Malek, Dalia Sulieman. Social networking in Cyberspace conference, Wolverhampton, England April 2010 (Poster)

International journals

1. Integration of semantic user profile within a social recommendation system semantic relevance measures characterization. Maria Malek, Dalia Suliemant, Hubert Kadima. IJCSS International Journal of Complex Systems in Science October 2011

Chapter 2

Social Network Analysis a Background

2.1 Introduction

Networks are defined as a collection of points connected to each others by lines. In general, networks have several kinds such as: topological networks, social networks, information networks and biological networks [145].

On the other hand, social networks are networks in which points represent people, called actors, and edges represent social interactions between people, and they are called social ties or relationships. Social networks can be modeled by a graph G in which actors are represented by the set of graph vertices $V(G)$ and the social ties are represented by the set of graph edges $E(G)$. So, social networks represent people and the interactions between them.
In our real life, social networks have a vast range of applications, in several domains such as: marketing, finding users with similar tastes, recommendation, searching for expertise [197, 198] and e-commerce. These applications are evident in the recent social networking sites as Facebook, LinkedIn and google+. For that social networks have attracted the attention of scientists from several domains such as: psychology, social science, physics and mathematics; and they are used in several studies about different social issues like social relationships (friendship), professional relationships (collaboration networks), exchange of goods or money, communications, romantic relationships and many other types of connections.

Furthermore, social network analysis (SNA) is used to study social networks. The origins of studying and analyzing social networks is referred to the Romanian psychiatrist *Jacob Moreno*, in his study of friendship relations between boys and girls in a class of schoolchildren in 1930s [134]. Thanks to this study, Moreno is considered as the founder of social network analysis [136].
Social Network Analysis is a branch of computational social science, and it is involved in studying and evaluating the structure of social networks. Social network analysis is very important subject because it helps to investigate the characteristics of different social networks and to study the connectivity between users in these networks.

The study of social networks analysis has three main directions: vertex based, structure based and community detection based. In this thesis we are concerned by vertex-based analysis. Using vertex-based we study vertices centrality like degree and betweenness centrality. Moreover, we are interested in graph searching algorithms to explore and search the social networks.

In this chapter we introduce the concept of social networks and social network analysis methods starting from basic definitions and algorithms in graph theory as in section 2. In section 3 we define social networks and social network analysis and we detail some of social network analysis measures, mainly vertex centrality. Section 4 reviews some of social network searching algorithms. Section 5 and Section 6 shows examples about some of the famous social networks and the well known tools used for analyzing these networks. Finally we conclude this chapter in Section 7.

2.2 Definitions and Algorithms From Graph Theory

Graph theory is one of the fundamental concepts in mathematics and computer science. It has a huge number of applications on several domains such as, telecommunications and transportations. According to the mathematical structure of graph, data is organized in two sets: a set of nodes or vertices and a set of edges connecting these vertices.
Graph theory has been recognized first in Königsberg bridges prob-

lem, Königsberg is a city that has seven bridges, in this problem we seek for a path in which the seven bridges can be traversed in a single trip without passing from the same bridge more than once. In 1736 the swiss mathematician and physician *Leonhard Euler*, has studied the Königsberg problem and he suggested to represent the seven bridges and their connecting points as edges and vertices in a graph. In this study, Euler proved that the Königsberg bridges problem has no solution.
Euler's solution of the Königsberg bridges problem is considered to be the first brick in graph theory, and to be the base of several algorithms especially in combinatorial optimization and networks.

Figure 2.1 shows the seven bridges of Königsberg city and their representation in a graph of four vertices and seven edges.

In this section we introduce some of the fundamental definitions and algorithms of graph theory. These definitions and algorithms correspond to our contributions.

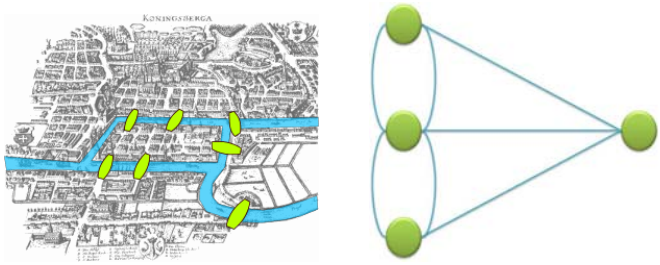

Figure 2.1: Königsberg bridge graph

2.2.1 Definitions

The following definitions are basic definitions in graph theory and strongly related to social network analysis and to our contributions. The majority of these definitions and algorithms are cited from [49].

2.2.1.1 Basic Definitions

Graphs have several types, as: weighted, complete and connected. Moreover, graphs can be represented via adjacency matrix and they

have characteristics, such as: density, clustering coefficient and transitivity. These types of graphs and their characteristics are widely used in social network analysis and they are described as follows:

Definition 1 (Graph)**.** *A graph G is defined by two finite sets: a non empty set of elements called vertices, denoted by $V(G)$, and a set of elements called edges, denoted by $E(G)$; $E(G)$ is a set of two elements of the form (v, v'), where v and v' are elements from $V(G)$.*

In a graph G each edge of $E(G)$ is denoted by $e(v, v')$, where v and v' are called the adjacent vertices. Moreover, in *undirected graphs* the set of pairs of elements of $E(G)$ are symmetric, that means $e(v, v')$ is as same as $e(v', v)$ and v and v' are named the endpoints of this edge.

Figure 2.1 is an example of a graph G with two sets: a set of four vertices $V(G)$ and a set of seven edges $E(G)$. Each edge connects two vertices called adjacent vertices or neighbors.

Definition 2 (Adjacency Matrix)**.** *Let G be a graph with a set of vertices $V(G) = \{v_1, ..., v_n\}$ and a set of edges $E(G) = \{e_1, ..., e_m\}$. The adjacency matrix of G, denoted by $A(G)$, is the $n \times n$ matrix where: rows and columns are indexed by $V(G)$. In $A(G)$, if v and v' are adjacent, or $e(v, v')$ exists in $E(G)$, then the matrix entry (v, v') equals to 1, and it equals to 0 else other.*

In undirected graphs the adjacency matrix is a symmetric matrix, while in directed graphs the adjacency matrix is a non symmetric matrix.

Definition 3 (Weighted Graph)**.** *Weighted graph G is a graph in which each edge has a corresponding value (integer or real). Thus, weighted graph is a graph G with the edge function $F : E \longrightarrow R$.*

In this thesis we are concerned by undirected weighted graphs. Furthermore, in social network analysis graph density and clustering coefficient are very important measures to characterize the social network, for that we use them to characterize the social networks which we use in our contributions. Graph density and clustering coefficient are defined as follows:

2.2.1.2 Graph density

Graph density for a graph G is used to study the graph sparsity and connectedness and it is given by Formula 2.1:

$$D = \frac{2|E(G)|}{|V(G)|(|V(G)|-1)} \tag{2.1}$$

where $|E(G)|$ is the cardinality of $E(G)$, $|V(G)|$ is the cardinality of $V(G)$ and $\frac{1}{2}|V(G)|(|V(G)|-1)$ is the maximum possible number of graph edges, in the case of complete graphs.
In social network analysis, network density is used to find the ratio between the sum of the values of all the connections between actors and the number of all the possible connections in the network. Furthermore, density is considered to be an important characteristic of the social network.
Moreover, in order to have an inner understanding of graph density, we have to define complete graphs and connected graphs as follows:

Definition 4 (Complete Graph)**.** *A graph G is said to be a complete graph if each vertex v in $V(G)$ is adjacent to all the other vertices in the graph.*

In any complete graph, the number of edges equals to $\frac{1}{2}|V(G)|(|V(G)|-1)$, where $|V(G)|$ is the cardinality of the set of the graph vertices $V(G)$.

Definition 5 (Connected Graphs)**.** *the graph G is said to be a connected graph, if all of its pair of vertices are connected via at least one edge.*

Conversely, a graph G is said to be disconnected, if there is no edges between, at least one pair of its vertices. Moreover, any subgroup of a disconnected graph is called *component*.

2.2.1.3 Clustering Coefficient

Clustering coefficient plays an important role in social network analysis. Indeed, in a given social network and for any three vertices x, y and z: if x has a connection with y and if x has another connection with z, then it is very possible that y and z to have the same kind of connection as the ones to x [191]. For that reason, *clustering coefficient* is used to study the possibility of having such kinds of

ties between vertices.
In social networks, clustering coefficient equals to the number of triangles divided by the number of triples in the network. The following definitions explain how to compute the number of triangles and triples of a social network, and how we can use them to find out the clustering coefficient[174].

Definition 6 (A Triangle Δ)**.** *Let G be an undirected graph with a set of vertices $V(G)$ and a set of edges $E(G)$, any complete subgraph of three vertices is a triangle Δ: if for a given vertices $\{u, v, w\} \in V(G)$ there is $\{e(u, v), e(v, w), e(w, v)\} \in E(G)$. The number of triangles $\delta(G)$ in any graph G is given by Formula 2.2:*

$$\delta(G) = \frac{1}{3} \sum_{v \in V} \Delta(v) \tag{2.2}$$

where $\Delta(v)$ is the number of triangles passing from the vertex v.

Definition 7 (A Triple Υ)**.** *In any graph G a triple Υ at a vertex $v \in V(G)$ is a path of a length equals to two, in which v is the center vertex. Thus, the summing of the triples of all vertices in the graph G is denoted by Formula 2.3:*

$$\tau(G) = \sum_{v \in V} \Upsilon(v) \tag{2.3}$$

a path is defined as a sequence of vertices in the graph, and the path length is defined as the number of edges connecting these vertices.

Definition 8 (Clustering Coefficient)**.** *The clustering coefficient for a vertex v with a number of neighbors $d(v) \geq 2$, is denoted by Formula 2.10:*

$$C(v) = \frac{\Delta(v)}{\Upsilon(v)} \tag{2.4}$$

and, the clustering coefficient $C(G)$ of a graph G is the average over the clustering coefficients of its vertices and it is given by Formula 2.5:

$$C(G) = \frac{1}{|V'|} \sum_{v \in V'} C(v) \tag{2.5}$$

where V' is the set of vertices v where $d(v) \geq 2$.

2.2.1.4 Bipartite Graphs

Bipartite graphs are graphs in which vertices are grouped into two disjoint sets, and edges connect only vertices from different sets. Bipartite graphs are used to model social networks [137, 199] and they are defined as follows:

Definition 9 (Bipartite Graph)**.** *Let G be a graph, with a set of vertices $V(G)$ and a set of edges $E(G)$. Let $X \in V(G)$ and $Y \in V(G)$ be two subsets of $V(G)$. The graph G is said to be a bipartite graph, with two partitions $\{X, Y\} \in V(G)$, if $X \cap Y = \emptyset$ and $X \cup Y = V(G)$, and for all the graph edges $e(x, y) \in E(G)$: $x \in X$ and $y \in Y$.*

Figure 2.2 is an example of a bipartite graph G with two sets of vertices X and Y from $V(G)$. From this Figure, we can notice that $X \cap Y = \emptyset$ and $X \cup Y = V(G)$, edges only connect vertices from different sets.

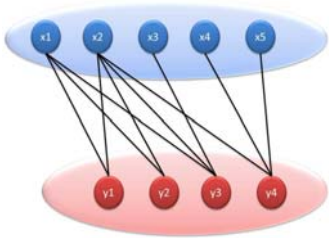

Figure 2.2: A Bipartite Graph

Bipartite graphs give a complete representation of a particular graph, but sometimes it is necessary to work with vertices of just one type. For that, one-mode projection is proposed as another type of graphs extracted from the original bipartite graph. One-mode projection is defined as follows:

Definition 10 (One-Mode Projection)**.** *Let G be a bipartite graph with two partitions $\{X, Y\} \in V(G)$. One-mode projection of X is defined a graph G_x in which the set of vertices is $V(G_x) = X$, and the set of edges is $E(G_x)$. In the set of the edges of G_x, an edge $e(x_1, x_2) \in E(G_x)$ if and only if $\exists y \in Y$ where (x_1, y) and $(x_2, y) \in E(G)$.*

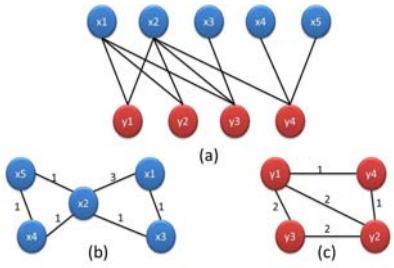

Figure 2.3: One Mode projection graphs

The same thing can be defined in the case of one projection of the set $Y \in V(G)$, for the G_y one-mode projection graph.

Adding weights to edges of one-mode projection is important, because it helps to know how many vertices from the bipartite graph are in common between the connected vertices in the one-mode projection graph. But one-mode projection graphs have problem of identifying the vertices of the other partition of the graph. Thus, in one-mode projection we can know how many vertices are shared in common but we lose the information about the identity of these vertices [145, 199].

One-mode projection is widely used in social network analysis, where these kinds of projections are known as collaboration networks [141, 142, 41, 74, 156]. In [141, 142] Newman studies the scientific collaboration between scientists (in the fields of physics, biomedical and computer science), by extracting authors one-mode projection from authors-citations bipartite graph. Also, the relation user-music is used to build graph of connected music artists [41]. In [74] authors build movies collaboration network extracted from Internet Movie Database (IMD), vertices represent movies and edges represent common users who have voted for these movies. In [156] authors study and compare three collaboration social networks: the network of movie actors obtained from Internet Movie Database (IMD), the network of scientific collaboration, and the network of company directors, in which two directors are linked if they sit on the same board of directors.

Example 1. Figure 2.3(a) represents the bipartite graph G, where $V(G) = X \cup Y$ and every vertex x from X is connected to a vertex y

from Y. Figure 2.3(b) represents the one-mode projection graph G_x in which all the vertices set is $V(G_x)$, and all the edges is $E(G_X)$. Figure 2.3(c) represents the one-mode projection of the graph G_y in which the set of vertex is $V(G_y)$ and the set of edges is $E(G_y)$, in the both graph G_x and G_y are weighted graphs.

Definition 11 (Tripartite Graph)**.** *Let G be a graph, with a set of vertices $V(G)$ and a set of edges $E(G)$. Let X, Y and Z be subsets of $V(G)$. The graph G is said to be tripartite graph, with three partitions $X \subseteq V(G)$, $Y \subseteq V(G)$ and$Z \subseteq V(G)$, if these sets are disjoint sets, and for every edge $e(v, v')$ from E(G): if one of the edge's endpoints is one of the disjoint sets, then the other must be one of the other two sets.*

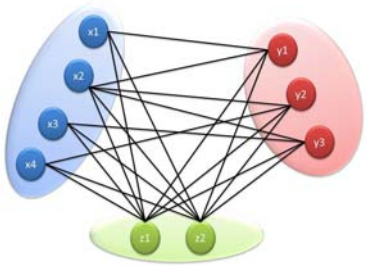

Figure 2.4: A Tripartite graph

Figure 2.4 is an example of a tripartite graph G, from this graph vertices are grouped in three disjoint sets X, Y and Z; and edges connect only vertices that are members of different sets.
Tripartite graphs have several applications especially in the context of semantic web and Folxonomy [127, 83]. In [127] authors represent semantic-social networks in the form of a tripartite graph of person, concept and instance associations, proposing to extend the concept of ontologies with the social dimension. In [83] authors transformed the Folksonomy $F := (U, T, R, Y)$ into a tripartite graph G. Graph vertices $V(G)$ is the set of vertices $V(G) = U \cup T \cup R$, U represents users, T represents tags and R represents resources. While graph edges $E(G)$ are weighted edges and they connect tags and users, users and resources and tags and resources.

2.2.2 Paths and walks in the graph

In [31] authors mentioned several types of flow in networks: geodesics, paths, trails and walks. A path in a graph is a sequence of edges which connect a sequence of vertices, paths in the graph have several types such as: cycle, simple path, simple cycle, Hamiltonian path and else more [145, 49]. On the other hand, a walk in a graph G consists of alternating vertices and edges of G [145, 31]. In this subsection we study the shortest path and the random walk.

2.2.2.1 Shortest Path (Geodesic Path)

In a given graph G a *path* $p(v, v')$ between two vertices v and v' from $V(G)$ is defined as the sequence of vertices and edges connecting these vertices, and the *length* of a path is defined as the number of traversed edges between v and v'.
Shortest path between two vertices is defined as a path which has the minimum length in the graph [49]:

Definition 12 (Shortest path)**.** *Let* $P(v, v') = \{p_1(v, v'), p_2(v, v'), \ldots, p_n(v, v')\}$ *be a the set of all possible paths between the vertices* v, v'*. The path* $p_j(v, v')$ *from the set of paths* $P(v, v')$*, is said to be shortest path if it has the minimum length compared with the other paths in* $P(v, v')$*.*

The length of shortest path is often called the geodesic distance or the shortest distance, and the shortest path whose shortest distance is the longest in the graph is called *graph diameter*. Moreover, if the vertices are disconnected then no shortest path exists between them. Furthermore, it is very possible to see several shortest paths between the pairs of vertices [143]. In social network analysis shortest path is used in several algorithms as: the algorithms used for computing Betweenness centrality [143, 33] and closeness centrality [61], also shortest paths are used in some algorithms for investigating the community structure of networks [140].

2.2.2.2 Random Walk

In a given graph G and starting from an initial vertex v from $V(G)$, *random walk* algorithm, selects randomly a neighbor of v, and moves to it, and repeats this operation [145]. The random sequence of vertices and edges selected in this way creates a random walk in the

graph [116]. So, a random walk is a path across the graph created by repeating random steps, and it is allowed to go along edges and visit vertices more than once.

In social network analysis random walk is used in several algorithms such as: random walk betweenness [144] and link prediction [148, 12].
In an undirected graph G we define $p_i(t)$ as the probability that the walk is at the vertex i at the time t. If the walk is at the vertex j at time $t-1$, then the probability of taking one step from j via one of its connected edges is $1/d_j$, as denoted in [145], and is defined by Formula 2.6:

$$p_i(t) = \sum_j \frac{A_{ij}}{d_j} p_j(t-1) \tag{2.6}$$

where d_j is the degree of the vertex j and A_{ij} is the value of the entry (i, j) in the adjacency matrix of the graph G.

2.2.3 Algorithms

In this subsection we detail and discuss three fundamental and important algorithms, which are used for searching and exploring the graph. These algorithms are: depth-first search DFS, breadth-first search BFS and A*. Depth-first search and breadth-first search explore all the graph, while A* algorithm explores a part of the graph according to some predefined heuristics. In our approach we propose to use depth-first search DFS algorithm and breadth-first search BFS algorithm with heuristics; for that reason, in this subsection, we discuss DFS, BFS and A* referencing them from [49, 167].

2.2.3.1 Depth-First Search DFS Algorithm

Since the nineteenth century, Depth-First Search algorithm (DFS) has been one of the fundamental graph searching algorithms [186, 98]. DFS is used in artificial Intelligence, in order to solve several problems such as: navigating systems, mapping, decision making and combinatorial problems.
DFS algorithm searches deeper in the graph whenever it is possible. It explores the graph's edge out of the most recently discovered vertex v that still has unexplored edges leaving it. Once all of the

v edges have been explored, the search backtracks to explore edges leaving the vertex from which v was discovered. This process continues until discovering all the vertices that are reachable from the original source vertex. If any undiscovered vertices remain, then DFS selects one of these remained vertices as a new source, and it repeats the search from it. The algorithm repeats this entire process until discovering every vertex in the graph [49]. DFS is detailed b Algorithm 1 and by the recursive Procedure 2. DFS search has advantages related to memory requirement and time complexity, but the main drawbacks of DFS are that DFS is not guaranteed to find solution.

Algorithm 1: Depth-First Search Algorithm

Require: (i) A graph G
1: **for all** $v \in V(G)$ **do**
2: $v.label$ = unexplored
3: **end for**
4: **for all** $e \in E(G)$ **do**
5: $e.label$ = unexplored
6: **end for**
7: **for all** $v \in V(G)$ **do**
8: **if** $v.label$ = unexplored **then**
9: $DFS - Visit(G, v)$
10: **end if**
11: **end for**

Algorithm 2: DFS-Visit(G,V)

1: $v.label$ = explored
2: **for all** $e \in G.incidentEdges(v)$ **do**
3: **if** $e.label$ = unexplored **then**
4: $w \leftarrow opposite(v, e)$
5: **if** $e.label$ = unexplored **then**
6: $e.label$ = explored
7: $DFS - Visit(G, w)$
8: **else**
9: $e.label$ = explored
10: **end if**
11: **end if**
12: **end for**

2.2.3.2 Breadth-First Search BFS Algorithm

Breadth-First Search Algorithm BFS is one of the simplest graph searching algorithms. BFS expands all the vertices one level away from the initial vertex, then it expands all vertices two levels away from the initial vertex, then three levels, until a goal state is reached. Since BFS expands all vertices at a given depth before expanding any vertices at a greater depth, the first solution path found by breadth-first search will be the one of the shortest length [99]. So, for a given graph G and a distinguished source vertex v, BFS explores the edges, connected to v, in order to reach every vertex that is connected to v. Thus, BFS computes the distance from the vertex v to every reachable vertex from v. It also produces a breadth-first tree starting from the root v, this tree contains all the reachable vertices from v [49].
BFS algorithm works on both directed and undirected graphs, and it has advantages such that BFS never enter an infinite cycle, and it definitely finds all the possible solutions; but yet, BFS has some limits regarding to time and space complexity. BFS is described by Algorithm 3.

Algorithm 3: Breadth-First Search Algorithm

Require: (i) A graph G, (ii) Source vertex
1: **for all** vertices v in $V(G)$ **do**
2: $v.label = unvisited$
3: **end for**
4: $user\_list$ = empty list
5: Q = empty queue
6: **while** $Q \neq \emptyset$ **do**
7: $v = dequeque(Q)$
8: $v.label = visited$
9: **for all** $e(v, v')$ and $e(v', v)$ in $E(G)$ **do**
10: **if** $v'.label = unvisited$ **then**
11: $v'.label = visited$
12: $enqueue(Q, v')$
13: **end if**
14: **end for**
15: **end while**

2.2.3.3 A* Algorithm

A* algorithm was firstly proposed in 1968 [77] as an extension of Dijkstra's algorithm [167]. A* algorithm has several applications such as message routing in large networks, resource allocation and vehicle navigation systems [10]. Moreover, this algorithm is used to find the minimum path cost between two points in a two dimensional rectangular grid with obstacles.

As classical graph searching algorithms A*, finds the minimum cost path between the start vertex and the goal vertex, but classical algorithms search all the possible paths and in the all directions around the starting vertex in the graph, while A* algorithm concentrates the search on a very limited number of paths and directions around the the starting vertex. For that, A* algorithm uses a heuristic function $f(n)$.
According to the literature, A* heuristic function $f(n)$ is combined of two functions $h(n)$ and $g(n)$. The first function $h(n)$ is called the admissible heuristic, which minimizes the estimated cost of the cheapest path from the current vertex to the goal vertex; while the second function $g(n)$ gives the path cost from the start vertex to the current vertex [167]. By combing the two functions $h(n)$ and $g(n)$ we can have the score path function $f(n) = h(n) + g(n)$ which is defined as the estimated cost of the cheapest solution passing through the vertex n. Thus, according to that heuristic function, A* algorithm is guaranteed to find the optimal solution (the path with the minimum cost) in the graph. Algorithm 4 shows the details of A* algorithm.

2.3 Social Network Analysis

Social Network is a graph, in which the set of vertices represents actors, and the set of edges represents the social ties between the actors. *Social Network Analysis SNA* is a branch of computational social science, that is involved in studying and evaluating the structure of social network. Social network analysis is very important issue because it helps to investigate the characteristics of different social networks and to study the connectivity between the users in the social networks. The study of social networks analysis has three main axes: vertex based, structure based and community detection

Algorithm 4: A* Algorithm

Require: (i) Start vertex s, (ii) Goal vertex g

$OpenSet \leftarrow s$
while $OpenSet \neq \emptyset$ **do**
  for all $n \in OpenSet$ **do**
    $current \leftarrow n$ with minimum $f(n)$
  end for
  Remove $current$ from $OpenSet$
  $ClosedSet \leftarrow current$
  if $current = g$ **then**
    Return the path from $current$ to s
  else
    $currentNeighbor \leftarrow current.neighbor()$
    for all $cn \in currentNeighbor$ **do**
      if $cn \notin OpenSet$ and $cn \notin ClosedSet$ **then**
        $g(cn) = g(n) + path(n, cn)$
        $f(cn) = g(cn) + h(cn)$
        if $cn \in OpenSet$ and its value is lower **then**
          update $g(cn)$
        end if
        if $cn \notin ClosedSet$ and $cn \notin OpenSet$ **then**
          $OpenSet \leftarrow cn$
        end if
      end if
    end for
  end if
end while

based.

- *Vertex based* analysis methods study the position and the role of the vertices in the social network. This study is mainly based on vertex centrality using centrality measures [145] and social network analysis algorithms as, PageRank [35] and HITS [93].
- *Structure based* analysis depends on link prediction [19, 66] and network evolutionary [109, 101]. Link prediction algorithms study and suggest which vertices are more likely to be connected with other vertices, while network evolutionary studies the structure of network and its dynamics over the time.
- *Community detection* methods propose to group graph vertices into clusters using community detection algorithms, such as modularity based ones [138, 27] and graph partitioning [180, 2] algorithms.

In this thesis we are interested in two axes of social network analysis, these axes are vertex centrality and community detection. Therefore, in this section, we discuss in details the most common social network centrality measures and some of the notable community detection algorithms, but before that we firstly introduce an overview some of the well known graph models and their characteristics.

2.3.1 Introduction

Studying the structure and the functional characteristics of social networks is an important issue in social network analysis. But, studying and analyzing social networks require realistic models and standard characteristics of these networks. For that, we present here some of the famous network models and their characteristics, as found in the literature of social networks.

2.3.1.1 Graph models

The first study of a realistic model of social networks was achieved in 1959 by Erdös and Rényi [58], when they proposed a model based on "random graphs", in which n vertices are connected randomly to m edges according to a probability p. Each pair of vertices is connected with equal probability p independently of the other pairs. Figure

2.5 is an example of Erdös-Rényi network, in this network $n = 50$, $m = 123$ and $p = 0.1$.

Figure 2.5: Erdös and Rényi network

Later in 1969 Milgram and Travers [188] proposed the "Small World" model, in their attempt to find the average path length connecting any pair of vertices in the network, and they concluded that vertices are connected via short paths with maximum average length of 6. This was later known as the "six degrees of separation" in social networks.

In 1977 Zachary [196] studied the "community structure" of a small social network connecting members of a Karate club.

Watts and Strogatz [191] proposed a model starting from a ring lattice with n vertices and k edges per vertex, where each edge is rewired at random with a probability p. This construction allows to tune the graph between regular finite ring lattice when $p = 0$ and disordered random graph when $p = 1$. Figure 2.6 is an example of Watts-Strogatz's model, this network is generated using the values $n = 50$, $k = 8$ and $p = 0.2$.

Barabási and Albert [9, 161] have proposed a network model, in which a preferential attachment is defined. Preferential attachement means that the probability of having connection to a vertex depends on the vertex's degree. Thus, in this model vertices degree follow the power-law distribution.
This type of networks are used for friendship networks, the World Wide Web, business and commerce networks. Figure 2.7 is an example of Barabási and Albert scale-free network, this network is generated using the values $n = 50$ and an average edge degree equals to 2.

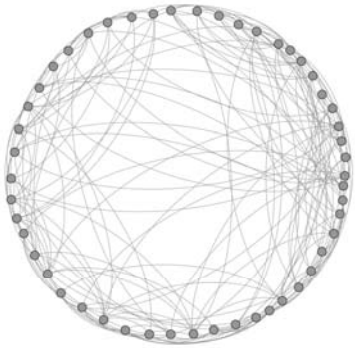

Figure 2.6: Watts-Strogatz model

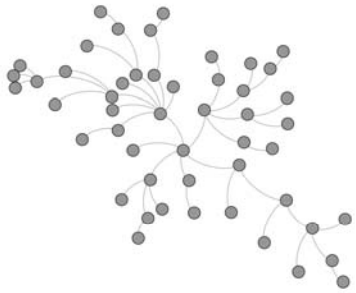

Figure 2.7: Barabási and Albert scale-free network

Furthermore, other models based on bipartite graphs [142, 137] are proposed as another class of network models. In these networks both of the individuals and groups are represented by two types of vertices; and edges only connect individuals with their corresponding groups representing the group membership. Although, one-mode projection can be applied on bipartite graph in order to extract the collaboration networks.

2.3.1.2 Social networks features

Social networks have three main features: the small world effect, the power-law degree distribution and community structure [142, 137].

Table 2.1: Social Networks models and characteristics

Network	Diameter	Clustering coefficient	Power Law
Erdös and Rényi [58]	√	X	X
Barabàsi and Albert [161]	√	X	√
Watts and Strogatz [191]	√	√	X
Bipartite model [142]	√	√	√

- The *small world effect* is identified when the social network has low graph diameter and high clustering coefficient.
- The *power-law degree distribution* means that network vertices should follow the power-law distribution, which implies that a few vertices have high degree and the majority of graph vertices have small degree.
- *Community structure* means that vertices are divided into groups, in which vertices of the same group has strong and dense connections while vertices from different groups have weak connections.

According to Erdös and Rényi model the graph diameter is small, the degree distribution follows poisson distribution and the clustering coefficient is small. In Watts and Strogatz model the generated network has small diameter, high clustering coefficient, but the vertices of this network follow the poisson distribution. In Barabàsi and Albert model, the network has a small network diameter, small clustering coefficient and its vertices follow the power-law distribution. Furthermore, bipartite graphs show a small network diameter, high clustering coefficient, and vertices follow the power-law distribution. Table 2.1 synthesizes the previously discussed models with their characteristics [1].

2.3.2 Social network analysis: centrality measures

Centrality measures are used to capture the features and the characteristics of social networks, and to understand the topology and the role of its connected actors. In this subsection, we present some of these centrality measures. Table 2.2 synthesizes these measures.

[1] Introduction à l'analyse des réseaux sociaux, Rushed Kanawati université de Paris13

Table 2.2: Social networks centrality measures

Centrality measure	Characteristics
Degree	Number of direct connections with other vertices
Closeness	Sum of length of the shortest paths passing from v
Betweenness	Frequency of the shortest paths passing from v
Eigenvector and Katz	Importance of the vertex's neighbors

2.3.2.1 Degree Centrality

Degree centrality is described as a local and structural measure, and it is computed in two simple ways: by counting the number of direct connections of each vertex in the graph, or by finding the sum of each row in the adjacency matrix of the graph [147, 61, 145].
Degree centrality $C_D(v_k)$ of any vertex v_k in the graph G is given by Formula 2.7:

$$C_D(v_k) = \sum_{i=1}^{n} e(v_i, v_k) \tag{2.7}$$

where $e(v_i, v_k) = 1$ if v_i and v_k are connected by an edge (adjacent), and $e(v_i, v_k) = 0$ otherwise.
According to Formula 2.7, large value of $C_D(v_k)$ means that v_k is adjacent to a large number of vertices, and small value of $C_D(v_k)$ means that v_k is adjacent to a small number of vertices. Thus, in any graph G, the maximum degree centrality is equal to $n-1$, n is the number of graph vertices, and the minimum value of degree centrality is equal to 0. Vertices with degree centrality that is equal to $n-1$ are said to be *dominating* vertices, and vertices with degree centrality that is equal to 0 are said to be *isolated* vertices and out of contacts.

Degree centrality of any vertex could be proportional to $n-1$ the maximum degree in the network [61], in this case degree centrality is called relative-degree centrality $C'_D(v_k)$ and it is given by Formula 2.8:

$$C'_D(v_k) = \frac{\sum_{i=1}^{n} e(v_i, v_k)}{n-1} \tag{2.8}$$

In the literature, there are some measures that are considered to be variants of degree centrality as k-path centrality which counts all the paths crossing the vertex [168], having a length of k or less; and vertex disjoint k-path centrality which counts all the vertex-disjoint

paths, of length k or less [8]. Vertex-disjoint paths are paths that have no vertices in common, so every vertex from these paths belongs to only one path.

Moreover, directed graphs have two types of degree centrality in-degree which indicates the vertex importance in the network, and out-degree which indicates the vertex powerfulness and ability of information transmission in the network.

Example 2. From the social network illustrated in Figure 2.8 we can find that: bigger vertices have higher degree centrality, and smaller vertices have smaller degree centrality. Moreover, the maximum degree value in this social network is 9, represented by the green vertex; and the minimum value of degree centrality is 0, represented by the red vertex.

If any vertex in the social network has a large value of degree centrality, that means it has a significant position and a very influential role in this network, as a result of having large number of direct connections with other vertices (the green vertex).

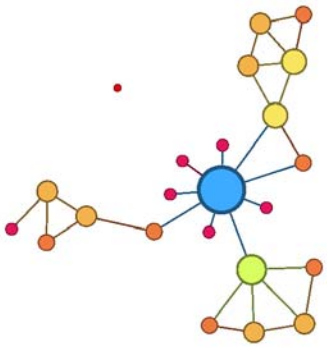

Figure 2.8: Degree Centrality

2.3.2.2 Closeness Centrality

Closeness centrality is based on the computation of the distances between the vertices in the network [16]. Closeness centrality of a

vertex v_i is defined as the sum of the graph's *shortest paths* (geodesic paths) linking this vertex with all the other vertices [31]. Closeness is classified as a radial measure and is given by Formula 2.9:

$$C(v_i) = \sum_{j=1}^{n} d(i,j) \tag{2.9}$$

$d(i,j)$ is the number of edges in the shortest paths connecting v_i and v_j.
From Formula 2.9, we can notice that important vertices have small closeness values and non important vertices have large closeness values. Thus, closeness centrality is defined as the inverse of $C(v_i)$ and it is described as follows [61, 31, 145]:

$$C_C(v_i) = \frac{1}{\sum_{j=1}^{n} d(i,j)} \tag{2.10}$$

As the maximum distance between any pair of vertices in the network is equal to $n-1$ (n is the number of vertices), then the vertex relative centrality is denoted by 2.11:

$$C_C^{'}(v_i) = \frac{n-1}{\sum_{j=1}^{n} d(i,j)} \tag{2.11}$$

Closeness centrality has two main drawbacks [145]: firstly, closeness has small range of values among the largest and the smallest closeness centrality, that means, it is difficult to distinguish the more or the less central vertices using this measure; secondly, it only works on connected graphs, as the fact that shortest paths have infinite values if vertices fall into two different components. If v_i and v_j fall in two different components of the network then the geodesic path between them is infinite because:

$$\sum_{j=1}^{n} d(i,j) = \infty \Rightarrow C_C(v_i) = 0$$

In the literature, several variant of closeness centrality have been proposed such as, immediate effects centrality [63] which counts the distance between any vertices in a graph as the average length of all paths between them; information centrality IC [182] which measures the reciprocal of the topological distance between the corresponding

vertices using the harmonic average; centroid centrality [76] which specifies one or more vertices as a graph centroid then, and in order to find the centrality of the vertex, it computes the distance between that vertex and the centroid.

Example 3. Figure 2.9 shows a social network in which the bigger vertices are the ones having high closeness centrality, and the smaller vertices are the ones having smaller closeness centrality. According to this example, the maximum closeness value is equal to 0.538 (the blue colored vertex) and the minimum value is equal to 0 (the red colored vertex). We notice that, the range between the smallest value and the biggest value is limited.
Obviously, vertices with high closeness centrality have short distances connecting them with other vertices in the social network, which means these vertices are easy to be reached by others, so they can spread their messages and opinions easily in the social network. Moreover, sending messages from vertex with high closeness centrality (the blue colored vertex in our example) takes short time and low costs, and in this case the vertex is said to have a good position enabling it to receive and transmit information rapidly and easily.

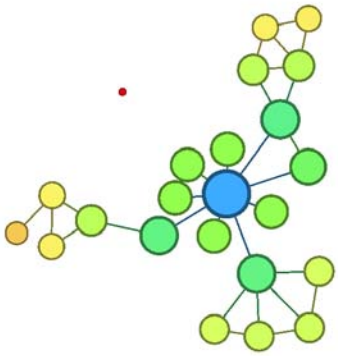

Figure 2.9: Closeness Centrality

2.3.2.3 Betweenness Centrality

Betweenness centrality of the vertex v_k in the graph G is defined as the number of times that any vertex $v_i \in V(G)$ needs the vertex v_k in order to reach other vertices in the graph via a shortest path [31]. That means, betweenness centrality indicates the frequency that a vertex falls on a shortest path between any two vertices in the graph, and it is computed by counting the number of the shortest paths passing through this vertex [61].

Computing betweenness centrality is simple and easy, if there exists only one shortest path connecting the pairs of vertices in the graph, but sometimes this is not the case. Often and for the majority of graphs, there exist several shortest paths between vertices, this fact makes betweenness centrality more complicated to compute; and a partial betweenness can be assigned in terms of probabilities [145].

According to Newman [145], the probability $p(v_i, v_j)$ of using one of several different shortest paths connecting two vertices v_i and v_j is denoted by 2.12:

$$p(v_i, v_j) = \frac{1}{g_{ij}} \tag{2.12}$$

where g_{ij} is equal to the number of the shortest paths between v_i and v_j.
Furthermore, the potential of the vertex v_k controlling the information flow between v_i and v_j is defined as the probability that v_k falls on a randomly selected shortest path linking v_i and v_j.
So, if $g_{ij}(v_k)$ refers to the number of the shortest paths between v_i and v_j then, the probability that v_k falls between these vertices is given as follows 2.13:

$$b_{ij}(v_k) = \frac{1}{g_{ij}} \times g_{ij}(v_k) = \frac{g_{ij}(v_k)}{g_{ij}} \tag{2.13}$$

Moreover, based on Formula 2.12 and Formula 2.13 and in order to determine the overall betweenness centrality of the vertex v_k in the graph G, where n is the number graph vertices, a sum is applied on all the v_k partial betweenness values over all the unordered pairs of vertices, in the graph, where $i \neq j \neq k$ as shown in 2.14:

$$C_B(v_k) = \sum_{i=1, j=1, i \neq j \neq k}^{n} b_{ij}(v_k) \tag{2.14}$$

Moreover, in a graph G the maximum betweenness value $C_B(v_k)$ of a vertex v_k is as follows:

$$C_{B_{max}}(v_k) = \frac{n^2 - 3n + 2}{2} \tag{2.15}$$

As consequence, the relative betweenness centrality of any vertex in the graph is given by 2.16:

$$C'_B(v_k) = \frac{2C_B(v_k)}{n^2 - 3n + 2} \tag{2.16}$$

The factor between the largest and the smallest betweenness centralities in the social network, is almost equal to $\frac{1}{2n}$.

Betweenness centrality has several variants, such as k-betweenness centrality [69, 63] which bounds the path by the length of k, where k represents the maximum number of edges in the explored path. Freeman [62] proposed another betweenness centrality by replacing the shortest paths by the edge disjoint paths. Moreover, Newman proposed betweenness measure based on random walk [144], by replacing the number of shortest paths by the number random walks.

In contrary to *degree* and *closeness* centralities, *betweenness* centrality does not measure the vertex ability to control information spread and transmission, but it measures the ability to facilitate the information flow passing from a vertex to another in the social network [31]. Thus, Betweenness centrality is not a measure of how a vertex is well connected, but it is a measure of how much a vertex falls in the communication channels between vertices [145, 61]. Therefore, a vertex could have a low degree and a long distance far from others in the social network, but still have high betweenness. Furthermore, removing a vertex with high betweenness will cut off the communications in the social network, because it has a position that lies on the largest number of shortest paths, that are used to spread information and messages in the network.

Example 4. Figure 2.10 is an example of a social network in which bigger vertices have high betweenness centrality, and smaller vertices have small betweenness centrality. According to this network the maximum betweenness value is equal to 0.758 (the blue colored vertex) and the minimum value is equal to 0 (the red colored vertices). In this Figure, according to betweenness centrality the blue vertex is considered to be an important mediator, because it participates in the majority of the shortest paths in the social network.

By removing this vertex from the social network, this network will be disrupted and the information will be hold. In the other case, removing one of the red vertices in the network will never influence the communication.

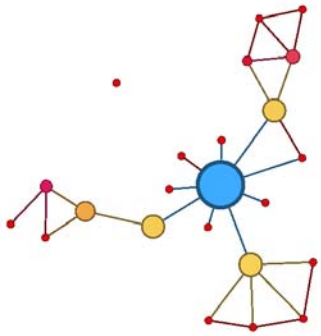

Figure 2.10: Betweeness Centrality

2.3.2.4 Eigenvector Centrality

Eigenvector centrality is considered as an extension of degree centrality, because it is concerned by the number of the vertex's neighbors (as degree centrality) and the importance of these neighbors. Thus, according to this centrality vertex's prestige in the graph is increased when it is connected to other vertices that are themselves important [30, 29].
For that, *eigenvector centrality* gives each vertex a score proportional to the sum of the scores of its neighbors.
Generally, eigenvector centrality of a vertex v_i in the social networks is given by 2.17:

$$\lambda x = Ax \tag{2.17}$$

where A is the adjacency matrix of the graph, λ is a constant of the eigenvalue, and v is the eigenvector. Thus, eigenvector of the vertex

v_i is given by 2.18:

$$x_{v_i} = \frac{1}{\lambda} \sum_{j=1}^{n} A_{ij} x_{v_j} \tag{2.18}$$

where λ is a constant that must be the largest eigenvalue of the adjacency matrix and x is the corresponding eigenvector.

Example 5. Figure 2.11 is an example of a social network in which bigger vertices have high eigenvector centrality (the blue colored vertex), and small vertices have low centrality (the red colored vertex). According to this network, the maximum eigenvector value is equal to 1, for the blue vertex, and the minimum value is equal to 0, for the red vertex. Moreover, the blue vertex has high influence in the social network because it is connected to important vertices, the green vertex, the yellow and the orange ones. So, if the blue vertex has a direct influence on its neighbors, then these neighbors will transmit this influence to the maximum possible number of vertices according to their degree centrality.

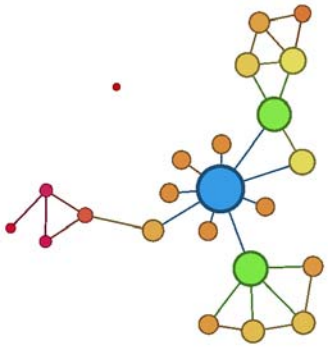

Figure 2.11: Eigenvector Centrality

2.3.2.5 Katz Centrality

Katz centrality adds some weights to the vertex degree. These weights are equal to the number of the paths between the vertex

and all the other vertices in the network [90] (the overall interconnectedness). In the graph G with n vertices, if A is the adjacency matrix, α is the weight factor, and A^k is the matrix representing the number of the paths of length k between v all the other vertices in the network, then Katz centrality is given by 2.19:

$$C_{Katz}(v_i) = \sum_{k=0}^{\infty} \sum_{j=0}^{n} \alpha^k (A^k)_{j,i} \tag{2.19}$$

Formula 2.19 shows that if a vertex with high centrality is connected to other vertices then these other vertices will also obtain a high centrality, e.g. if a vertex with a very high centrality is connected to one million other vertices this high centrality will be transferred to the one million vertices.

2.3.3 Social network analysis: Community detection

The study of community structure in social networks gives a detailed information about the topology and the role of the vertices in these networks. Moreover, almost all the social networks have a topology of connected or disconnected clusters, where vertices are grouped into clusters.

Ties and interactions between vertices belonging to the same cluster are very strong and intense, while ties and interactions between vertices belonging to different clusters are very rare and weak.

In social networks, the clusters of vertices are named *communities*, and the study of these communities is named *community detection*.

The origins of community detection comes from a study of political communities of people according to their votes [165]; and a study of the structure of a complex government and the relationship between its workers in their different agencies [192].

For example in Zachary's karate club, as shown in Figure 2.14, the social network has two main communities the blue one and the yellow one. The vertex with the highest degree in the blue community represents the first instructor in the club, connected to the members that are training with him; and the vertex of the highest degree in the yellow community represents the second instructor in the club, connected to the members that are training with him. Obviously,

club members who are training with the instructor have intense connections between them and moderate connections between the other ones who are training with the other instructor. Moreover, the two instructors have no direct connection between them because they have some conflicts considering to some administrative problems, so they have no social interactions between them.

Generally in social networks, vertices having high degree and closeness appear to be in the center of the community, with intense connections with vertices from the same community; while vertices having high betweenness appear to be at the boundaries of the community, and they are connected to vertices from different communities.

In the literature several studies have been achieved in the context of community detection and their applications in the different networks, such as the world wide web, the citation networks and the co-authorship networks; and several network clustering algorithms have been proposed. In this subsection we introduce some of these algorithms.

2.3.3.1 Traditional clustering algorithms

Traditional graph clustering algorithms can be used for community detection in social networks, these algorithms are grouped in several categories [59] as, graph partitioning, hierarchical clustering, partitional clustering and spectral clustering. Table 2.3 shows a synthesis of these algorithms, and these algorithms are described as follows:

Graph Partitioning is an important issue in computer science, and it is known as an NP-hard problem. According to graph partitioning methods, graph vertices are grouped in several partitions, where the number of edges connecting these partitions is minimum (the number of edges connecting different partitions is called the cut size). In the literature, several graph partitioning algorithms have been proposed such as, Kernighan-Lin algorithm [91] which proposes to divide the graph into bisections iteratively; maximum flows algorithms [2] which are based on the max-flow min-cut theorem by Ford and Fulkerson; and algorithms based on minimizing measures related to the cut size [180]. In such algorithms, the number of clusters should be predefined and in some cases the cluster size should

be also predefined.

Hierarchical Clustering in some cases it is difficult to predefine the number of clusters or the size of clusters for graph partitioning, for that the hierarchical clustering methods have been proposed. The hierarchy in these methods is based on a predefined similarity measure between vertices, and they suppose that graph vertices are grouped in several levels where smaller clusters are included in larger clusters. Hierarchical clustering algorithms have two main categories (a) agglomerative clustering with a bottom-up structure that starts from a single vertex as a separated clusters to end up by the whole graph vertices as one cluster [78]; (b) divisive clustering with a top-down structure that splits the graph into clusters by removing edges connecting vertices with low similarity [141]. Such algorithms depend on predefined functions between vertices as modularity, and sometimes vertices are not classified correctly. It also in many cases that some vertices are missed even if they are central or important, moreover agglomerative algorithms have a drawback that they do not scale well [59].

Partitional clustering methods represent vertices as points in the metric space, then they define the distances between these vertices, according to a cost function, in order to group the vertices into k clusters with the respect to minimizing or maximizing the cost function. Partitional clustering can be achieved using several algorithms such as k-means clustering [118, 158] and fuzzy k-means clustering [56]. In such algorithms the number of clusters should be defined at the beginning of the algorithm.

Spectral Clustering clusters the graph using the eigenvector values of the vertices similarity matrices. This could be achieved via the laplacian matrix [22], using normalized spectral clustering [179] or unnormalized spectral clustering [117].

2.3.3.2 Optimization based community detecting algorithms

Hierarchical Clustering is considered as an optimization based approach for community detection. Here we discuss three of the most

Table 2.3: Graph Clustering methods

Methods	Algorithms	Characteristics
Graph partitioning	bisection-partition [91]	Clustering vertices, with minimum cut size, clusters number or size should be predefined.
	max-flow min-cut [2]	
	cut size [180]	
Hierarchical Clustering (optimization based)	Agglomerative algorithms [78]	Bottom-up structure, never scale well
	Divisive algorithms [141]	Top-down structure, sometimes vertices are not well classified.
Partitional clustering	k-means [118, 158]	Clustering depends a predefined cost function predefining clusters number.
	Fuzzy k-means [56]	
Spectral clustering	Laplacian matrix [22]	Using eigenvector values of vertices similarity matrix
	Normalized spectral clustering [179]	
	Unnormalized spectral clustering [117]	

Table 2.4: Hierarchical clustering algorithms

Hierarchal clustering	Algorithms	Characteristics
Divisive hierarchy	Grivan-Newman algorithm [65]	greedy method, splits the graph, by deleting edges with high betweenness.
Agglomerative hierarchy	Newman algorithm [141]	greedy method, merges clusters according to the modularity value.
	Louvain algorithm [27]	heuristic method, maximizes the modularity, network of communities (communities of communities structure).

famous community detection algorithms, these algorithms are Grivan-Newman algorithm, Newman's greedy algorithm and Louvain algorithm. Table 2.4 synthesizes the characteristics of these algorithms.

Girvan-Newman algorithm This algorithm has been introduced as a very distinguished approach in the field of community detection. It is categorized as a divisive hierarchal clustering algorithm, in which clusters are iteratively splitted by removing some edges according to a predefined criteria. Since inter-community edges (edges

connecting different communities) are most likely to have a high betweenness, Girvan-Newman algorithm has used edge betweenness as a criterion to identify the network communities.

As vertex betweenness, *edge betweenness* is proposed as a measure of edge importance and influence in the social network [65], and it is equal to the number of all the shortest paths passing through this edge.

The main idea of Grivan-Newman algorithm is to find the inter-community edges, edges with high betweenness centrality, and delete them progressively from the graph. This method, leads to have separated groups of vertices, representing the possible graph communities.
Girvan-Newman algorithm is described as follows [65]:

1. Compute the betweenness centrality of all the edges in the social network.
2. Remove edges with the largest betweenness centrality.
3. Recalculate the new values of edges betweenness centralities.
4. Repeat the steps 2 and 3 until no edges remain.

Each iteration of Girvan-Newman algorithm could lead to a new group of partitions ending up by a hierarchy of partitions [65]. Figure 2.12, illustrates the top-down clustering nature of Grivan-Newman algorithm [140].
This method imposes the following question: "how to choose the best partitions?". The answer on this question is proposed in [140], where Newman and Girvan proposed a quality function to assess the quality of the discovered graph partitions, this function uses the *modularity*.

Modularity based methods Modularity is one of the most popular graph partitioning quality functions, that assigns a score to each partition in the graph. This score ranks the graph partitions according to their quality. Thus, partitions with higher scores have higher quality [59]. Modularity is given by formula 2.20:

$$Q = \frac{1}{2m} \sum (A_{ij} - \frac{k_i k_j}{2m}) \delta(C_i, C_j) \tag{2.20}$$

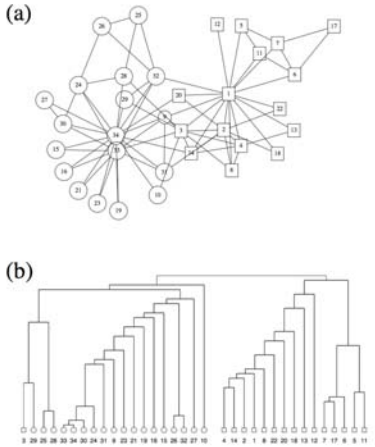

Figure 2.12: Girvan-Newman algorithm top-down clustering [140]

where $\delta(C_i, C_j) = 1$ if the vertices i and j are in the same community and $\delta(C_i, C_j) = 0$ otherwise, m is the total number of edges in the graph, and $\frac{k_i k_j}{2m}$ is the expected number of edges between vertices i and j in the null model graph. The null graph model is a copy of the original graph keeping all its structural properties except the community structure.

It is remarkably that the modularity is based on the idea in which random graphs are not expected to have a community structure, so according to modularity the possible existence of clusters is revealed by comparing the edges density in the actual graph and the expected edges density in the null model graph, regardless to the community structure.

Newman's Greedy algorithm Newman proposed a greedy algorithm based on maximizing the modularity of the discovered communities in the network [138]. This algorithm is categorized as an agglomerative hierarchical clustering algorithm, in which clusters are iteratively merged according to a predefined criterion, the *modularity* criterion.
The algorithm starts from n clusters (n is the number of all the graph vertices), at the beginning these clusters are disconnected, having no edges between them; then edges between vertices are added, if

and only if they maximize the modularity of their corresponding clusters.
However, in order to decide whether to merge the two clusters or not, the algorithm computes the variation of modularity ΔQ between every two different clusters at each iteration step. The algorithm stops when $\Delta Q \leqslant 0$ for any two clusters in the network.

Louvain algorithm is a heuristic-based and agglomerative hierarchical clustering algorithm [27]. Louvain algorithm includes two steps, in the first step the algorithm divides the network into n disconnected communities (n is the number of all the vertices in the graph); then for each vertex v the modularity gain is evaluated, by placing v in each community of each one of its neighbor, to finally be added to the community in which the gain of modularity is positive and has the maximum value. If the modularity gain is not positive then the vertex v stays in its own community. This process is applied repeatedly and sequentially for all the vertices until no further improvement can be achieved and in this case the first step of the algorithm is completed.
The second step of the algorithm constructs a new weighted network, a meta-communities network (a network of communities), in which each vertex represents one of the discovered communities given by the first step. In this network edges have two types of weights (a) weights of edges from the same community, which is equal to the sum of the edges connecting vertices of the same community; and (b) weights of edges connecting different communities, which is equal to the sum of the edges connecting vertices from the different connected communities. Once the second step is accomplished, it is then possible to reapply the first step of the algorithm on the new weighted network and to iterate until no new changes, in modularity, occur on the obtained network. Figure 2.13 shows that each pass is made of two phases: one where modularity is optimized by allowing only local changes of communities; one where the found communities are aggregated in order to build a new network of communities. The passes are repeated iteratively until no increase of modularity is possible [27].

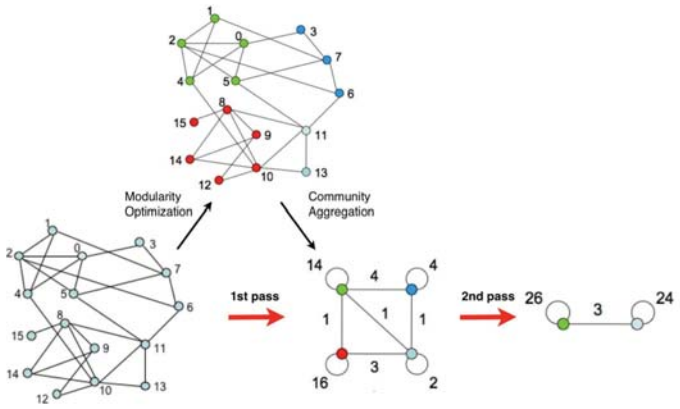

Figure 2.13: Louvain algorithm [27]

2.4 Searching Social Networks: Algorithms

Social networks become a great source of information, specially if these networks have a huge number of actors e.g. *Facebook* that has 1.15 billion active users and *LinkedIn* that has 238 million active users. In such cases, social networks play an important role, because they contain huge amount of information about users, experts, locations, products and more else. According to that, social networks have a vast range of applications, that require very fast and very effective graph-based searching techniques.

Graph-based searching algorithms, in social networks, are used for several purposes, such as: *Finding experts* in a given domain e.g. seeking for a java expert in a java online help-seeking community [198]. *Ranking* network actors according to their relation to a given subject e.g. ranking mails according to their correlation to users' interests [55]. *Searching engines* e.g. "google.com" [150], and *Recommender Systems* [125].

In [197] authors classified graph searching algorithms into three main categories: General computational based algorithms, Network structure based and Similarity based.*General computational* algorithms include breadth first search and random walk search. *Network structure algorithms* include vertex degree based algorithms algorithm using: vertex degree [102], PageRank algorithm or HITS

Table 2.5: Searching network structure algorithms

	Algorithms	Methods
Vertex-based searching algorithms	Degree-based	in-degree [121, 32]
		high-degree seeking [4]
		h-distance & target physical location [102]
	PageRank	Personalized PageRank [89, 123]
		flow values, traffic rank PageRank [187]
	HITS	Weighted vertices and edges [47]
		Randomized HITS [146]

algorithm [198], weak tie and strong tie search [70]. *Similarity-based* includes using referral systems associated with the social network of its users. Such systems attach agent to each user, this agent learns user's preferences and interests. So agents order incoming quires by issuing referrals where others might be more suitable to field a given query [194].

In our thesis we are interested in *Network Structure* algorithms, mainly: vertex degree [4, 102], PageRank algorithm [150, 21, 32] and HITS algorithm [93, 94, 198].
Generally, vertex degree-based algorithms suppose that all the vertices with high degree are equally important. Moreover, PageRank algorithm supposes that vertices with high degree are not necessarily important and useful information resources. While HITS algorithm groups graph vertices into two special categories, as follows: hubs and authorities.

The following subsections give essential details about network structure algorithms, and Table 2.5 summarize the previously discussed algorithms.

2.4.1 Degree-based searching algorithms

Degree-based algorithms rank graph vertices according to their degree centralities. In [121, 32] authors proposed ranking vertices according to the in-degree, supposing that the more in-degree value the more is the vertex importance. This algorithm is not efficient because not all the vertices having high in-degree are important.
In [4] authors proposed a high-degree seeking strategy, this strategy

performs a sequential search which visits vertices with highest degree in sequence. They proved that: high degree seeking strategies are effective in networks with a power-law degree distribution with an exponent γ close to 2.
In[102] authors proposed two other strategies beside to the high-degree seeking strategy, as follows: the first strategy is based on exploring the organizational hierarchy of the social network, in order to find the closest possible people to the target. This is achieved by labeling the vertices by their hierarchical distance (h-distance) from the target, to finally pass the information to the closest vertex to the target in the social network. Their experiments on an email network showed that individuals closer to each other in the organizational hierarchy are more likely to exchange emails between them. The second strategy is based on target physical location, such as: city, street, building and floor number. The experiments on email exchanging network showed that: The general tendency of individuals in close physical proximity to correspond holds: over 87% percent of the 4000 email links are between individuals on the same floor.

2.4.2 PageRank

PageRank [150, 35] is one of the most famous searching algorithms, whose name has been attached to *Google*, the popular and successful web searching engine, since 1998. PageRank is defined as a variant of Katz centrality, and it supposes that the PageRank of a given vertex is proportional to its centrality divided by its out-degree.
PageRank algorithm answers text quires by generating a list of relevant web pages. The relevancy between the suggested web pages and the query depends on two ranking scores: the first one is related to the matching value between the query's text and the web pages' weighted index, web weighted index could be: title, anchor, URL, plain text large font, plain text small font; and the second ranking score is the PageRank score of the web pages.
Google searching engine shows a high ability to find relevant web pages, because it orders its results according to their scores, where results on the top of the ranked list are more relevant than results that are in the down of the ranked list.

The PageRank score of a given vertex v_i is described as in 2.21:

$$v_i = \alpha \sum_j A_{ij} \frac{v_j}{d_j^{out}} + \beta_i \tag{2.21}$$

where $\alpha = 0.85$ as defined by google search engine, d_j^{out} is the number of the out degree of the vertex v_j and β_i gives a value based on a predefined criteria, as the textual relevancy to the search query [145].

PageRank algorithm has several variant which are proposed in the literature, as: personalized PageRank [89, 123], and flow values, traffic rank PageRank [187].

2.4.3 Hyperlink-Induced Topic Search HITS algorithm

HITS algorithm was firstly proposed by Kleinberg in 1999 [93, 94]. HITS algorithm gives each vertex v_i in the network two centralities: authority centrality a_i and hub centrality h_i. The vertex having high authority a_i, is defined as a vertex that has been pointed to by several hubs; and the vertex having high hubs centrality h_i is defined as a vertex that points to several vertices with high authorities. However, citation networks are good examples of hubs and authorities, where articles that have been cited by high number of other articles, e.g. more than one hundred articles, are supposed to be a very significant articles. While articles that have cited several important article are not necessarily supposed to be important articles. Furthermore, we can define authorities as vertices that contain important information and hubs as vertices that are guiding to important vertices.
According to Kleinberg authority centrality a_i of a given vertex v_i is given by Formula 2.22:

$$a_i = \alpha \sum_j A_{ij} h_j \tag{2.22}$$

where α is constant.
Hubs centrality h_i of a given vertex v_i is given by Formula 2.23:

$$h_i = \beta \sum_j A_{ji} a_j \tag{2.23}$$

where β is a constant. HITS algorithm works on directed networks, and it was used in some search engines as: *ask.com*.

HITS algorithm has some variations as algorithms that weigh vertices and edges with textual information [47], and randomized HITS [146].

2.5 Social networks: examples

In this section we present some of the well known social networks, that are used as benchmarks to test several algorithms proposed for social network analysis. These networks are: Zachry's karate club network [2], as shown in Figure 2.14, Newman's co-authorship collaboration network [3], as shown in Figure 2.15, les misérables social network [4], as shown in Figure 2.16, and protein interaction network, as shown in Figure 2.17.
Table 2.6 provides some statistics about these networks, mainly the number of vertices $|v(G)|$, the number of edges $|E(G)|$, the graph diameter Dia, the graph density D and the clustering coefficient CC.

- Zachary's karate club network: is a social network proposed by Wayane Zachary in 1970s [196], to study the social interactions between members of karate club at one of the universities in USA. Vertices of this network represent club members, and edges represent the social interactions between these members, inside and outside the club. This network has been studied by Newman [65, 140] in the context of analyzing the network structure for community detection. Zachary's karate club network is shown in Figure 2.14.

- Newman co-authorship network: is an undirected network, in which vertices represent scientists from three domains biology, physics, and mathematics; and edges represent the ties between these scientists. Newman's co-authorship network is a collaboration network, where two scientists (vertices) are connected if they have, at least, one written paper in common; edges are weighted and their weights is equal to the number of common papers written by the connected scientists. Co-authorship collaboration networks is used to study the structure of commu-

[2] http://moreno.ss.uci.edu/data.html
[3] http://wiki.gephi.org/index.php/Datasets
[4] http://www-personal.umich.edu/ mejn/netdata/

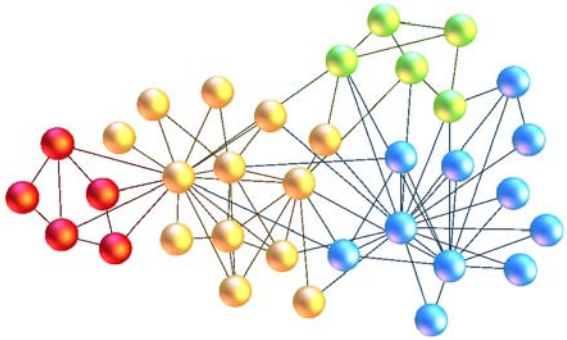

Figure 2.14: Zachary's Karate Club Social Network

nities in social networks [139], and it is shown in Figure 2.15.

- Les Misérables: is a weighted and undirected network, representing the characters of Victor Hugo's novel *Les Misérables*. This network is a collaboration network, where vertices represent the characters of the novel, and edges represent how many times the characters have appeared together in the same chapter [95]. This network has been used to study and analyze the community structure in social networks [140], and it is shown in Figure 2.16.
- Protein interaction network: is a network of physical interactions between nuclear proteins, vertices represent proteins and edges represent interactions between these proteins [37, 122]. Protein interaction network is shown in Figure 2.17.

Beside to the previously discussed networks, several datasets have been studied and used for modeling and analyzing social networks such as: ego-Facebook networks [5], which are extracted from Facebook [124]; wiki-vote networks [6], which are extracted from users's

[5]http://snap.stanford.edu/data/egonets-Facebook.html
[6]http://snap.stanford.edu/data/wiki-Vote.html

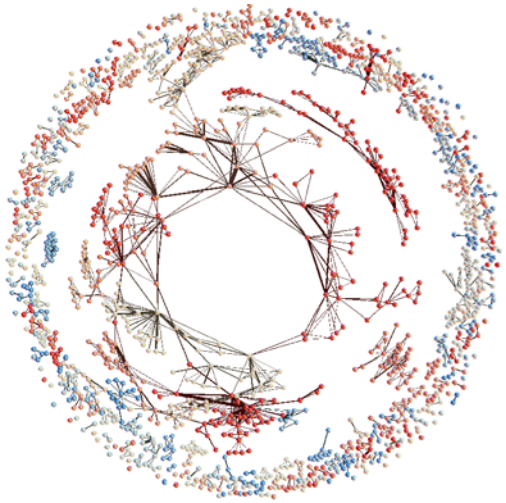

Figure 2.15: Newman's Co-authorship collaboration Network

Table 2.6: Social Networks Examples

Social Network	$\|V(G)\|$	$\|E(G)\|$	Dia	D	CC
Karate Club	34	78	5	0.139	0.588
Co-authorship	1589	2742	17	0.002	0.878
Les misérables	77	254	5	0.087	0.736
Protein-Protein	2361	6646	11	0.002	0.2

votes on the contexts of Wikipedia [108]; and DBLP collaboration network [193].

2.6 Social Network Analysis Tools

In this section we present four of the most common tools used for social network analysis and visualization, these tools are: JUNG, NetworkX, Gephi and Pajek. Moreover, we compare these tools[7], as described in Table 2.7.

- JUNG (Java Universal Network Graph) Framework: is a free and open-source library, that is written in java and used for

[7]Etude comparative d'outiles de fouille et d'analyse de réseaux social. David Combe, Christine Largeron, Elöd Egyed-Zsigmond et Mathias Géry. 2ieme jour née thématique: Fouille de grands graphes -20 Octobre 2011

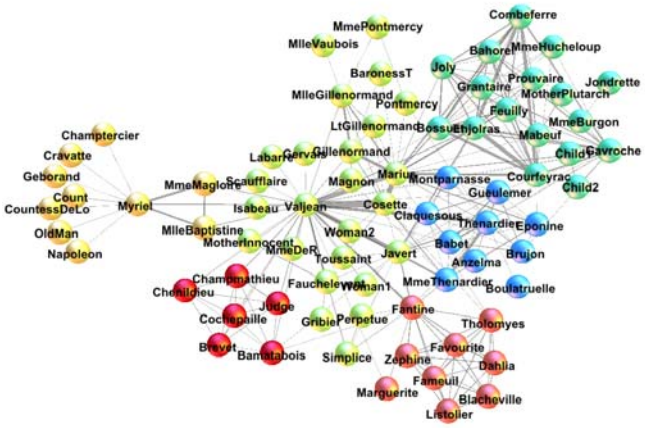

Figure 2.16: Les Misérables Social Network

modeling, analyzing, and visualizing graphs and networks[8]. The first version of JUNG was released in 2003 and the last version of JUNG was released in 2010. JUNG supports GraphML and Pajek input/output files, although users have the ability to create exporters and parsers from any kinds of formats. JUNG supports several classes to manipulate several kinds of graphs, such as: hyper graphs, k-partite graphs and multi graphs. Moreover, JUNG supports several classes for social network analysis: such as centrality measures, PageRank, HITS and shortest paths algorithms. The presence of community detection algorithms and visualization algorithms is limited in JUNG.

- NetworkX: is a free and open-source library for social network analysis, that is written in python[9]. The first version of NetworkX was released in 2005 and the last version of NetworkX was released in 2012. NetworkX supports GML, GraphML, NetworkX and Pajek input/output files. NetworkX supports several types of graph such as bi-partite graphs, and it contains several social network analysis algorithms; but it has some limi-

[8]http://jung.sourceforge.net
[9]http://networkx.github.io

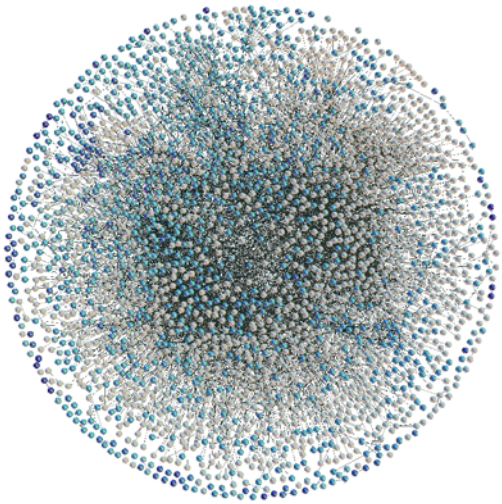

Figure 2.17: Protein Interaction Network

tations considering the lack of community detection algorithms and the social networks visualization methods.

- Gephi: is an open-source software, that is written in java and it is used to visualize and analyze social networks[10]. The first version of Gephi was released in 2010, and the latest version of Gephi was released in 2013. Gephi supports input files in several formats such as: GraphML, GML, Pajek, Tulip, UCINET, and it supports output files in format of Gephi, and png. Gephi provides various social network analysis algorithms such as: centrality measures, PageRank, HITS and modularity. Also, Gephi supports several algorithms for graph visualization and 3D mode network visualization.

- Pajek: is a program for windows, used to analyze and visualize large scale networks[11]; Pajek is a free software for non commercial use, and it is written in Delphi (Pascal) programming language. Pajek first version was released in 1996, and Pajek last version was released in 2013. Pajek supports input/output files in several formats such as: Pajek, Ucient and

[10]https://gephi.org
[11]http://pajek.imfm.si

Table 2.7: Social Network Analysis Tools

Tools	JUNG	NetworkX	Gephi	Pajek
First Version	JUNG 1.0.0 (2003)	NetworkX (2005)	Gephi 0.7-alpha (2010)	pajekXXL 0.0.1 (1996)
Last Version	JUNG 2.0.1 (2010)	NetworkX 1.8, 1 (2013)	Gephi 0.8.2-beta (2013)	pajekXXL 3.12 (2013)
Platform	java library	python library	Software	Software
Social Networks Algorithms	++	++	++	+
Community Algorithms	−	−	+	+
Visualization	−	−	++	+
Graph Types	++	+	−	+

GraphML. Furthermore, Pajek provides many tools for studying and analyzing social network structure. For that it supports several types of graphs as: bipartite graphs, multiple relations networks and temporal networks; also, it provides several clustering and visualization algorithms. Moreover, Pajek supports networks visualization in 3D mode.

2.7 Conclusion

In this chapter we introduced an overview about social networks and social network analysis methods. Starting from fundamental definitions in both graph theory and social network. Passing by the characteristics and models of social networks. Then we introduced some definitions of the most important social network analysis measures. After that, we presented some of the well known algorithms in community detection domain and in searching social networks domain. Then we presented examples of some of the well known social network examples. And finally we presented some of softwares used for social network analysis.

Chapter 3

Recommender Systems Background

3.1 Introduction

Recommender systems [162] have attracted the attention of researchers in the last two decades, and nowadays they have a strong impact on our modern life [164]. Recommender systems are used to help users in making choices about interesting items based on their tastes and opinions. So, recommender systems provide personalized recommendation based on the explicit and implicit information about the users. Tapestry was the first non-personalized recommender system which was proposed in 1992 [67]. Later in 1994 GroupLens proposed UseNet recommender system as a personalized recommender system for news recommendation [1, 96]. Moreover, recommender systems have a wide range of implementations in the domain of e-commerce [171]. Also, recommender systems are used in Aamzon.com, eBay and google searching engine.

Recommender systems have three main categories as follows: content-based recommendation which achieves the recommendation based on the relevance between item features and user profile; collaborative filtering systems which propose to exchange the personal experiences of the users, on their different choices, with other people who could share or could not share their same tastes or interests; and hybrid recommendation which combines content-based recommendation methods with collaborative filtering methods. Figure

3.1, shows the three main classes of recommendation, each of them can use model-based methods or memory-based methods according to the recommendation process, as we will see later in this chapter.

In this chapter we provide a brief explanation about the common types of recommender systems, the famous algorithms used for recommender systems, the drawbacks of recommender systems, and finally we present some methods for evaluating the accuracy and efficiency of recommender systems.

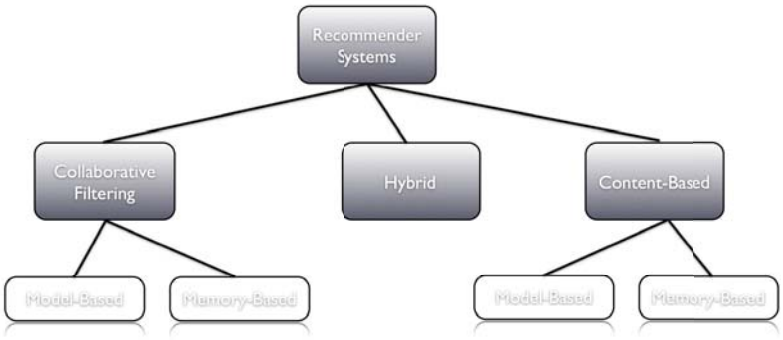

Figure 3.1: Recommender Systems Types

3.2 Content-Based Recommender Systems

Content-based recommendation has its origins from information filtering and information retrieval [5].

Content-based recommender systems are content oriented, which means the content of users' interests and the content of the features of items play an essential role in the recommendation process. Actually, the recommender system compares the content of user interests with the content of item features, using one of the well known methods such as Bayesian networks [24, 135], Neural networks [153, 151], Decision trees [100] and TF-IDF [25]. However, Bayesian networks, Neural networks and Decision trees are classified as model-based methods, while TF-IDF is classified as memory-based methods, as described in Table 3.1.

Table 3.1: Content-based recommender systems

CB Method	Used technique	Example
Model-based	Bayesian Classifier	NewsDude [24], Libra[135]
	Decision trees	InfoFinder [100]
	Neural Networks	Syskill & webert [153, 151]
Memory-based	TF-IDF	DailyLearner [25]
	Semantic relevancy	QuickStep [126]

Content-based recommender systems impose a very important issue related to the representation of the content of user interests and the content of item features. For that, we need to have a structured form of users' interests and a structured form of items' features. The structured form of users and items is very important in the recommendation process, because it facilitates finding the relevancy between items and users in the system. In the literature, the structured form of user's interests and item's interests is represented via user profile and item profile [115].

Generally, content-based recommender systems have two inputs first a target user, defined by a certain profile containing the user's preferences and second a group of items defined by certain profiles defining their features. The output of content-based recommender systems is a recommendation list containing the most relevant items to the target user (based of the contents of user profile and item profile). For that, the system compares item profile with user profile using one of the well known model-based or memory-based algorithms.

In this section we introduce the content-based recommender system, then we describe some of the well known content-based recommendation algorithms and finally we mention some of the limitations related to content-based recommender systems.

3.2.1 Background

Content-based recommender systems, recommend an item i to a target user u, if the content of i is similar to the content of other items that have been liked by u.

In such systems, user profile becomes important because it contains the contents of all the user's preferred items.

In content based recommender systems, the utility function determines the importance of an item i to a user u. So, the utility function $f(u, i)$ of an item i for a user u is estimated based on the utilities $f(u, i_n)$ assigned by the user u to items $i_n \in I$ that are similar to item i. Thus, content-based recommenders suggest items similar to the ones the user has liked in the past. Where, the similarity between items depends on the content of these items.

Example, in a "book" content-based recommender system, the recommender system determines the books that are highly rated by the user u, then it characterizes the user's preferences by analyzing the content of the user's highly rated books e.g. title, author, editor, year, keywords, etc. Then it explores the common preferences between the book, and the user's preferred items (books). If the book has a high similarity to user's preferences then this book will be recommended to the user u.

In general, content-based recommender systems depend on three main substances as follows:

- User-profile: which contains the users' preferences, such as: items the users rated and liked in the past. Also, user-profile can contain extra information about users such as: name, age, city, friends, etc.
- Item-profile: which contains all the features and the characteristics describing the item.
- User-item utility function: which is used to find the relevancy between user-profile and item-profile. If the relevancy is high, then the item can be recommended to the user.

Definition 13. *Let $u_c(u_{c_1}, u_{c_2}, \ldots, u_{c_n})$ be the profile of the user u where $\forall x \in [1, n] : u_{c_x}$ is the content of the preferred items of u, and let $i_c(i_{c_1}, i_{c_2}, \ldots, i_{c_m})$ be the contents of the item i where $\forall y \in [1, m] : i_{c_y}$ is the content of i. The utility function $f(u, i)$ between the user u, and the item i is described as:*

$$f(u, i) = score(u_c, i_c)$$

Both of user-profile and item-profile should have a well defined structure, constructed in the same way, in order to be processed easily by the user-item utility function (relevancy measure). For instance, if user-profile and item-profile have a textual description, then user-item relevancy measure will require a very complex process. On the other hand, if user-profile and item-profile have a keyword profile then the user-item relevancy will become a very easy process, achieved by one of the well known similarity measures, such as cosine similarity.

Figure 3.2 is an example of a very simple Content-based recommender system. According to this example, the recommender system is composed of three main components: user profile, item profile and the recommender system (user-item utility function). User profile contains all the important information about user's interests in forms of groups of terms, and this is the same for item profile. The recommender system compares the terms of user profile with the terms of the items in the system, to finally decide which item is the most appropriate for the user. In this example, the recommender system compares all the terms of user profile with all the terms of item profile of all the items in the system, by applying the intersection between all the terms of user profile with all the terms of all the item profiles. If the intersection value is high then the item will be added to the recommendation list; and the score function of this example is described as follows:

$$f(u,i) = u_{profile} \cap i_{profile}$$

3.2.1.1 User profile

In content based recommender systems user profiles and item profile are very important, because they are used to hold information related to the characteristics of items and preferences of users.

However, item characteristics and user preferences are a very fundamental aspects in content based recommender systems. Furthermore, the accuracy of recommendation is highly related to the information stacked in user profile and item profile, and also it is related to the type of this information. The following paragraph describes

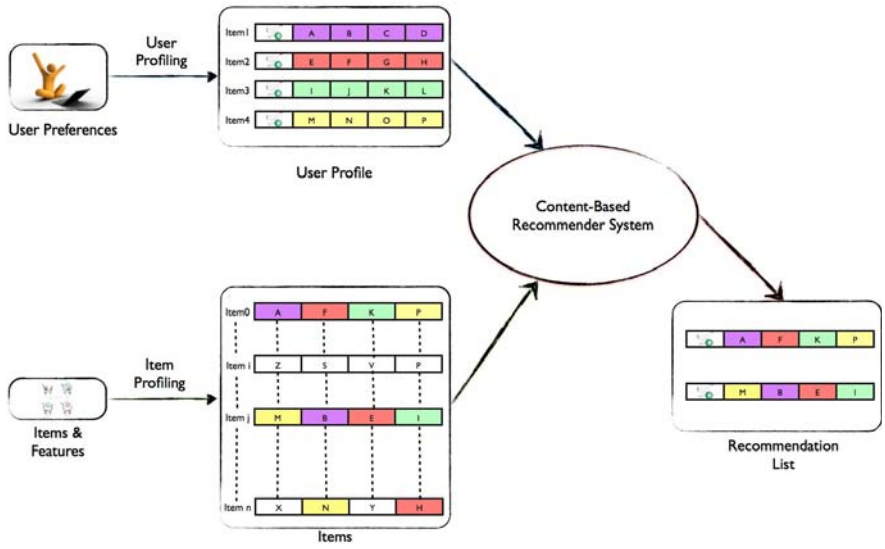

Figure 3.2: Content-Based Recommender System example

user profile and its representation methods. However user profile description and methods could also be applied for item profile.

User-profile In Content-based recommender systems, the content of user's preferences is a very important subject, because it plays an important role in the recommendation process.

Indeed, user-profile describes in details all the possible information that are related to the user, such as name, age, job, preferences, etc. However, this information could be presented in several forms.

According to the literature, user-profile has several types of representation [64] as follows: a bag of words; a vector of weighted keywords, using weighted or non-weighted vectors of keywords; semantic network profiles, using ontology; and conceptual profiles, using taxonomy (as special case of semantic networks) [115]. Table 3.2 summarizes the different types of user profile.

Keyword user-profile Keyword based profiles are represented via vectors of one or more dimensions; where, each user is associated with a vector of weighted terms. Formally, a user $u : u \in U$, where

Table 3.2: User Profile

User Profile	Definition	Example
Keyword user profile	Vectors of n dimensions of weighted or non-weighted keywords	NewsTailor [178] & YourNews [7]
Semantic user profile	Weighted semantic network using Ontology	QuickStep [126], News@hand [42]
Conceptual user profile	Abstract topics of user interests, using Taxonomy	SEWeP [57]

U is the set of all the users in the system, is represented as follows: $u = [w_{t_1}, w_{t_2}, \ldots, w_{t_n}]$, where w_{t_1} is the weight of the term $t_1 \in T$, and T is the set of all the terms in the system. Furthermore, in the case of non-weighted terms u is represented as $u = [t_1, t_2, \ldots, t_n]$, where $t_1, \ldots, t_n \in T$, T is the set of all the terms in the system.
In the literature several recommender systems use keyword profiles in order to describe users and items such as: Letizia [110] and Personal Web Watcher [132] which are used for web recommendation. News Tailor [178], YourNews [7] and NewsDude [24] which are proposed for news recommendation. Libra for book recommendation [135]. Citeseer for scientific articles recommendation [28]. Intimate for movie recommendation [120]. Last.fm [1], Pandora [2] and FOAFing the music [46, 45] for music recommendation.

Semantic network user profile Profiles are represented by weighted semantic network or ontology, in which each vertex represents a specific concept and each edge represents relations between concepts. In such profiles, concepts are very specific and they use external knowledge bases, e.g. domain ontology, as references. Actually, using semantic networks to represent user profile increases the accuracy of the recommendation because it helps in interpreting and analyzing the content of these profiles, taking in advantage the reasoning characteristics associated with the ontology.

In the literature, several models have used semantic user profiles. SiteIF [119] is a sense-based recommender system for web news recommendation. In this system, users' interests have been described based on the synonyms of the word. However, word synonyms are in-

[1] http://www.last.fm
[2] http://www.pandora.com

herited from MultiWordNet, which considered as reference ontology, and user profile is represented as a semantic network, in which vertices represent the synonyms and edges represent the co-occurrences of these synonyms in the user's document of interests. QuickStep [126] is proposed to recommend research papers. Profiles are extracted from web browsing history and user's feedbacks, and they are represented as topic ontology. Moreover, research papers are classified using ontological classes and a topic ontology extracted from the current topics of interests of similar people who have seen this paper. Recommendation is achieved by finding the relevancy between user interest profiles and classified paper topics. Informed recommender [3] uses ontology to interpret users' opinions. News@hand [42] is another recommender system, for news recommendation, in which users and items are represented using ontological profile, and profile concepts are referenced to a group of domain ontologies. In [26] Ontology Web Language OWL is used as a formalism to represent user profile for an Interactive Digital TV programs recommender system.

Conceptual user profile Concept-based profiles could be considered a special case of semantic network profiles, in which the nodes represent abstract topics describing user preferences, rather than specific words (in the case of semantic profiles) or sets of related words (in the case of keyword profile). In such profiles, taxonomies are remarkably used. SEWeP (Semantic Enhancement for Web Personalization) [57] is a web personalization system, it uses a domain specific taxonomy in order to semantically annotate the Web pages.

3.2.2 Content-based Recommender Systems: Methods and Algorithms

Content-based recommendation algorithms are grouped into two main groups *Model-based* and *Memory-based*. In model-based methods the algorithms adapt a model and learn it in order to achieve the recommendations [15, 151, 159]; while in memory-based or heuristic-based methods recommendations are made by analyzing the entire collection of data [135].

3.2.2.1 Model-Based

In model-based systems, the recommendation is achieved using a model learned from the data based on statistical learning and machine learning techniques as: bayesian classifier [159], clustering, decision trees and neural networks [151]. Here we choose to give some details about the most used method, in literature, the Naïve Bayesian.

Naïve Bayesian Naïve Bayesian is a probabilistic inductive learning approach. It generates a probabilistic model based on previous data, and it estimates the probability that a document d belongs to a class c.
In these models, items are represented via a group of terms instead of a vector of terms. And user profile is learned using probability based on what the user has previously liked. The probability $P(c \mid i)$ is computed using the Bayes theorem as follows 3.2:

$$P(c \mid i) = \frac{P(c)}{P(i)} \times P(i \mid c) \tag{3.1}$$

where $P(c)$ is the probability of the class c, $P(i)$ is the probability of that item i occurring and $P(i \mid c)$ is the probability that item i appears in the class c. To recommend an item i, the system needs to calculate $P(c \mid i)$ for each of the class in C, and find the largest probability. As each of these calculations involves unknown but fixed probability $P(i)$, it can be ignored, which simplifies the model to:

$$P(c \mid i) = P(c) \times P(i \mid c) \tag{3.2}$$

Under the assumption that each term appears independently from each others, the probability $P(i \mid c)$ can be decomposed as follows:

$$P(i \mid c) = \prod_{l=1}^{\vec{content}(i)} P(t_{l,i} \mid c) = P(t_{1,i} \mid c) \times P(t_{2,i} \mid c) \times \cdots \times P(t_{n,i} \mid c) \tag{3.3}$$

where each term is supposed to be independent for the others, $P(t_{l,i} \mid c)$ is the probability of having the l^{th} term when we have the class c and it indicates to the number of times the l^{th} term appears in the class c divided by the total number of terms in c. P(c)

is defined as the total number of terms in class c divided by the total number of terms in set of classes C.
The item is recommended by computing all the conditional probabilities $P(c \mid i)$ for each class c and item $i \in I$. As shown in Equation 3.4:

$$P(c\backslash i) = P(c) \times \prod_{l=1}^{\vec{content}(i)} P(t_{l,i} \mid c) = P(c) \times P(t_{1,i} \mid c) \times P(t_{2,i} \mid c) \times \cdots \times P(t_{n,i} \mid c) \tag{3.4}$$

Bayesian classifiers give high classification accuracy but they have limitations concerning that terms are independent from each others and it takes a lot of data training in order to estimate the various probabilities.

3.2.2.2 Memory-Based

Memory-based algorithms are used in content based recommender system. These algorithms continuously analyze users and items data, in order to achieve the recommendation. Term Frequency and Inverse Document Frequency TF-IDF is one of the most used methods in this context.

Term Frequency and Inverse Document Frequency (TF-IDF) TF-IDF is one of the most popular measures used for weighting keywords in information retrieval [13]. TF-IDF measures the relevancy of a term keyword to a document, with the respect to the fact that terms that occur frequently in one document but rarely in the other ones are more likely to be relevant to that document.
Term frequency TF counts how many times the term t_i occurs in the document d_j. So according to TF, the relevancy of the term t_i to the document d_j is determined by how many times it appears in the document, and the higher TF of t_i means the more relevancy to d_j. The term frequency is given by Formula 3.5:

$$TF(t_i, d_j) = \frac{f(t_i, d_j)}{maxf(d_j)} \tag{3.5}$$

where $maxf(d_j)$ is the maximum frequency of all the terms that appear in the document d_j.

Inverse document frequency IDF is obtained by calculating the ratio between the total number of documents in the system, and the number of documents containing this term. IDF means that terms with high IDF are rarely to appear in the system's documents, and it is described by Formula 3.6

$$IDF(t_i) = log\frac{N}{n_{t_i}} \tag{3.6}$$

where N is the number of documents in the system, and n_{t_i} is the number of documents containing t_i.

Based on Formula 3.5 and 3.6, the TF-IDF weight for term t_i in document d_j is defined by Formula 3.7:

$$w_{t_i,d_j} = TF(t_i, d_j) \times IDF(t_i) \tag{3.7}$$

In content based recommendation the contents of user profile and the contents of items are presented as TF-IDF vectors, and the utility function $f(u, i)$ can be computed using some scoring functions between the user's terms vector w_u and the item's terms vector w_i, such as cosine similarity measure [170] as shown in Formula 3.8:

$$f(u,i) = cos(w_u, w_i) = \frac{\sum_{k=1}^{m} w_{k,u} w_{k,i}}{\sqrt{\sum_{k=1}^{m} w_{k,u}^2}\sqrt{\sum_{i=1}^{k} w_{k,i}^2}} \tag{3.8}$$

where m is the number of total keywords in the system, $w_{k,u}$ is the weight of the keyword k given by the user u and $w_{k,i}$ is the weight of the keyword k given by the user i.

3.2.3 Limitations of Content-Based Recommender Systems

Content-based recommender systems have some limitations. These limitations have been mentioned and well defined in the literature [5, 115], and we present them bellow:

3.2.3.1 Overspecialization

Overspecialization problem also named serendipity problem occurs because content-based recommender systems are employed to find expected recommendations based on predefined user's preferences.

So, the ability to receive recommendation of new items with new features, out of user profile, is very limited. For example, a user who has rated books only about science fiction will receive recommendations considering this type of books ignoring all the other possibilities. However, a content-based recommender system would rarely find new and useful recommendations, and as a consequence there will be no chance to receive unexpected recommendations.

3.2.3.2 Limited Content Analysis

Content-based recommender systems have a natural limit in the number and type of features that are associated, whether automatically or manually, with the objects they recommend. Domain knowledge is often needed, e.g. for movie recommendations the system needs to know the actors and directors, and sometimes, domain ontologies are also needed. No content-based recommendation system can provide suitable suggestions if the analyzed content does not contain enough information to distinguish items the user likes from items the user does not like. Some representations capture only certain aspects of the content, but there are many others that would influence a user's experience. For instance, often there is not enough information in the word frequency to model the user interests in jokes or poems, while techniques for affective computing would be most appropriate. Again, for Web pages, feature extraction techniques from text completely ignore aesthetic qualities and additional multimedia information. To sum up, both automatic and manual assignment of features to items could not be sufficient to define distinguishing aspects of items that turn out to be necessary for the elicitation of user interests.

3.2.3.3 Cold Start: New User Problem

Content-based recommender system recommends items only if the items have a high degree of similarity between the user's preferences. Furthermore, if the user is new to the system and has no preferences history, the recommender system will be unable to recommend items to this user until this user rates a sufficient number of items to determine the user's preferences.

3.3 Collaborative Filtering Recommender Systems

Collaborative filtering is one of the most popular approaches used for recommendation. The early beginnings of collaborative filtering recommender systems were introduced in 1994 by GroupLens, when they proposed a NewsNet system for news recommendation [1, 96]. In such systems the only information related to users and items are their ratings. Furthermore, users rated items are correlated to other similar users called neighbors, and the recommender system explores the correlation between similar users and items in order to achieve the recommendation. *Collaborative-Filtering* recommender systems are also named *Social Filtering* recommender systems [54].

Collaborative-filtering recommender system is used to predict the utility of items for a particular user based on the items previously rated by other users.

The formal definition of the utility function of a collaborative-filtering recommender system is given as follows [5]:

Definition 14. *Let U be a set of m users $U = \{u_1, u_2, \ldots, u_m\}$ and I be a set of n items $I = \{i_1, i_2, \ldots, i_n\}$. The utility function $f(u, i)$ of item i for user u is estimated based on the utility functions $f(u_j, i)$ assigned to item i by those users $u_j \in U$ who are similar to the user u, like minded users.*

In Collaborative-Filtering, users explicitly rate items according to their preferences, then these rates are stored in a user-item matrix R. This matrix contains all the users's profiles, where rows represent the users $U = \{u_1, u_2, \ldots, u_m\}$ and columns represent the set of items $I = \{i_1, i_2, \ldots, i_n\}$ and $R_{u,i}$ is the rating assigned to item i by the user u. So, the key idea is that the rating of u for a new item i is likely to be similar to that of another user u', if u and u' have rated other items in a similar way.

Algorithms for collaborative-filtering recommender systems have been classified into two main categories: *Model-based* and *Memory-based*. As shown in Figure 3.3.

In this section, we will give an overview about model-based methods and memory-based methods, and the algorithms and techniques that are used by these methods.

But, the focus of this thesis is on the memory-based collaborative-filtering using graph techniques.

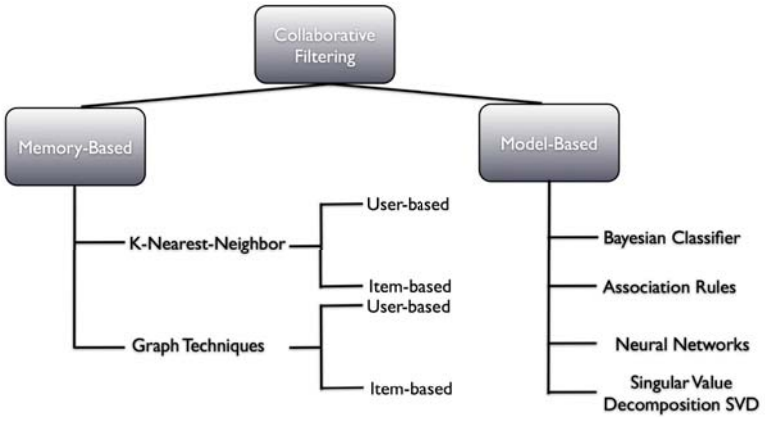

Figure 3.3: Collaborative Filtering Methods and Algorithms

3.3.1 Model-based collaborative-filtering

Model-based collaborative filtering learns the interactions between items and users, in the system, in order to extract a predictive model that is used to predict the unknown users's rantings.

Thus, the main concept of model-based algorithms is to determine an off-line model of the data, in order to rapidly obtain an on-line predicted ratings [40].
Moreover, from a probabilistic point of view, model-based collaborative filtering could be considered as the expected rating of the user u on the item i, as described by Formula 3.9:

$$E(r_{u,i}) = \sum_{k=0}^{m} Pr(r_{u,i} = k / r_{u,i'}, i' \in I_u)k \tag{3.9}$$

where I_u is the set of the items that have been previously rated by the user u, $k \in [0, m]$ is an integer representing the rating value, and the interval $[0, m]$ denotes the rating range in the system.

In this context, two methods have been proposed in order to estimate users' ratings, the clustering methods and bayesian network

methods [34]. In clustering methods, users are grouped in classes according to their preferences; while in bayesian networks methods items are represented as nodes in the bayesian network and the states of each node correspond to the possible rating values.

Furthermore, other machine learning approaches have been proposed for model-based collaborative-filtering such as, association rules [133, 112, 52], and artificial neural networks. In fact, association rules have some limitation related to performance issues, while artificial neural networks have limitations related to the complexity of learning time to finally reach the convergence [23].

Besides to machine learning methods other methods have been proposed, such as: singular value decomposition (SVD) for matrix factorization [97, 17, 185], latent semantic analysis [82], maximum Entropy [201], Boltzmann machines [169] and Support Vector Machines [71].

3.3.2 Memory-based collaborative-filtering

Memory-based algorithms, which are also named neighborhood-based, are heuristics that make rating predictions based on the entire collection of previously rated items by the users [5]. In such algorithms, aggregation functions are commonly used in order to predict the unknown users ratings. So, the prediction of the unknown rating $r_{u,i}$ for a user u on the item i is usually computed as the aggregation of the ratings of other users for the same item i as follows 3.10:

$$r_{u,i} = aggr_{u' \in U}(r_{u',i}) \tag{3.10}$$

The aggregation function given by 3.10 could be an average rating or weighted sum function. Furthermore, aggregation function could be replaced by a normalized user-user similarity $sim(u, u')$ in the user-based collaborative filtering, or could be replaced by a normalized item-item similarity $sim(i, i')$ in item-based collaborative filtering (as we will see later).

Memory-based collaborative-filtering has two principal types user-based and item-based. Moreover, Memory-based algorithms can be grouped into two main categories: K-nearest neighbors algorithms and graph-based algorithms. Here, we will discuss memory-based types and algorithms.

3.3.2.1 Memory-based collaborative-filtering types

Memory-based collaborative-filtering has two types user-based and item-based.

User-based collaborative filtering computes the missing rating $r_{u,i}$, of an item i for a target user u, based on the opinions of the other like-minded users (users having similar tastes with u). Like-minded users are also called nearest-neighbors.

User-based collaborative-filtering compares users preferences and considers that a user who has liked and rated an item i, will like and rate the item i', if other users who have liked and rated the item i also have liked and rated the item i'.

For a target user u, user-based collaborative filtering algorithms, achieves the following tasks:

- At first, the algorithm constructs the user-item rating matrix R.
- Then, the algorithm finds the like-minded users (or the nearest neighbors), based on the previous ratings of the target user u.
- Finally, the algorithm computes the value of the predicted rating.

The predicted rating $r(u,i)$ for a target user u on the item i, depends on the similarity between the previous rating values of this user u, and the previous rating values of her/his nearest-neighbors as described by Formula 3.11:

$$r(u,i) = c \sum_{u' \in N_k(u,i)} sim(u,u') \times r(u',i) \tag{3.11}$$

where c is a normalization factor, $N_k(u,i)$ is the k nearest-neighborhood of u, $sim(u,u')$ is the similarity between the target user u and $u' \in N_k$ and $r(u',i)$ is the rating of the user u' on the item i.
And also the predicted rating $r(u,i)$ for a target user u on the item i could be described by Formula 3.12

$$r(u,i) = \bar{r}_u + c \sum_{u' \in N_k(u,i)} sim(u,u') \times (r(u',i) - \bar{r}_{u'}) \tag{3.12}$$

Table 3.3: User-Based Collaborative Filtering Example

	i_1	i_2	i_3	i_4	i_5
u_1	1		1	0	0
u_2	1	1	0	1	1
u_3	1	1	?	1	1
u_4	0	0	1		0
u_5		0	1	1	0

where $\bar{r}_u$ and $\bar{r}_{u'}$ are the average ratings of the both users u and u'.

In the literature, several user-based recommender systems have been proposed such as: Usenet recommender system [1, 96] proposed by GroupLens for news recommendation, MovieLens recommender systems [128][3] for movie recommendation and Ringo recommender system [189] for music recommendation.

Example 6. Table 3.3 represents user-item rating matrix R. Ratings have two values: 1 if the user likes the movie and 0 if the user does not like the movie. From this table we can find that: u_2 and u_3 seem to be like-minded users (because they rate same movies with same values). So, in order to predict the missing rating of the user u_3 on the item i_3: $r(u_3, i_3)$ we should look to the rating of users who have similar tastes to u_3: which is u_2 in our example. As a consequence, if the user u_2 does not like the item i_3, then the user u_3 will not like the item i_3. Thus, the predicted rating of user u_3: $r(u_3, i_3) = r(u_2, i_3) = 0$.

Item-based collaborative filtering explores the similarity between items instead of exploring the similarity between users. In general, item-based collaborative filtering deals with the target user's u preferred items. An item-based collaborative filtering system suggests that a user u who likes item i should be recommended by item i' if the item i' is found to be similar to item i regarding to the list of user's u preferred items. So the key idea of item based collaborative filtering is to find items similar to other items that the user has liked in the past.

Figure 3.4 is an example of Amazon.com item-based collaborative filtering, which it recommends items that are similar to the item i

[3] http://movielens.umn.edu/

with the respect to the opinions of other users.

For an item i, item-based collaborative-filtering algorithms, the recommendation is achieved as follows:

- At first, the algorithm constructs the user-item rating matrix R.
- Then, the algorithm finds items that are similar to i, according to the users ratings.
- Finally, the algorithm computes the value of the predicted rating $r(u, i)$.

The predicted rating $r(u, i)$ according to item-based by Formula 3.13:

Figure 3.4: Item-Based recommendation

$$r(u, i) = c \sum_{j \in I_u} sim(i, j) \times r(u, j) \tag{3.13}$$

where c is a normalization factor, I_u is the set of items rated by the user u, $sim(i, j)$ is the similarity between item i and item j rated by the user u and $r(u, j)$ is the rating of user u and the item $j \in I_u$.

In the literature, item-based collaborative filtering algorithms are widely used as Ringo the music recommender system [189], Amazon item-based recommender system [114] and Google news recommender system [51].

Moreover, in [172] item-based collaborative filtering has shown better performance than user-based collaborative filtering.

Example 7. Table 3.4 represents user-item rating matrix R. Ratings have two values: 1 if the user likes the movie and 0 if the user does not like the movie. From this table we can notice that: users who liked i_2 also liked i_4, and some of people who liked i_1 also liked i_3.

Table 3.4: Item-Based Collaborative Filtering Example

	i_1	i_2	i_3	i_4	i_5
u_1	1	?	1	0	
u_2	1	1	0	1	1
u_3	0	1		1	1
u_4	1	0	1	0	
u_5		0	1	0	1

So in order to predict a rating value for user u_1 on item i_2, the item-based collaborative filtering will look for the items having similar rates as i_2, which is i_4 in our example, and the predicted rating value of u_1 on i_2 will be similar to the rating of u_1 on item i_4.

Comparasion between User-based and Item-based Here we discuss the effects of using user-based and item-based collaborative filtering on the system from three points of view [54]: accuracy, efficiency and stability, as follows:

- Accuracy: user-based collaborative filtering systems are user oriented systems, that means the recommendation algorithm depends on the user's neighborhood. On the other hand, item-based collaborative filtering algorithms are item oriented systems, that means recommendation algorithm depends on the item's neighborhood. Moreover, the number of users in the majority of recommender systems is greater than the number of items. So, in collaborative filtering system, in the case of user-based collaborative filtering having large number of users, each user can have a large number of neighbors, that could negatively affect the recommendation accuracy; while in the case of item-based collaborative filtering having small number of items each item can have a small number of neighborhood, this case could positively affect the recommendation accuracy [189, 172].
- Efficiency: of recommender systems is related to the number of users and items in the system. In the case of user-based collaborative filtering the larger number of users means the more memory and time to calculate $sim(u, u')$; and in the case of item-based collaborative filtering, the smaller number of items means the less memory and time to calculate $sim(i, i')$.

- Stability: in the majority of collaborative filtering systems similarity relations between users may change over the time, as a result users similarity weights must be updated frequently over the time. While relations between items are static and they are very rare to be changed over the time, in this case the frequency of updating the items similarity is very low.

Item-based methods are simple to compute and easy to be justified; but they have a drawback related to the fact that item-based recommender systems recommend similar items, so there is no variety in the recommendation e.g. a user who previously rated jazz music will be constantly recommended by jazz music, ignoring the other types of music.
Moreover, some systems combine item-based collaborative filtering with user-based collaborative filtering in one model. In such systems, the prediction is achieved by exploring the similarity between users and the similarity between items in the same time; actually these systems showed better performance than using only item-based collaborative-filtering or only user-based collaborative-filtering [190].

3.3.2.2 Memory-based collaborative-filtering methods

In this thesis, we propose to classify memory-based algorithms into two main categories: K-nearest neighbors algorithms and graph-based algorithms.

K-Nearest Neighbor Methods In K-Nearest Neighbor algorithms recommendations, similarity between users or items is achieved using some of the well known similarity measures, such as: cosine similarity, Pearson correlation, Spearman correlation, as follows:

Cosine Similarity this measure maintains the similarity between two users by computing the cosine of the angle formed by their corresponding rating vectors [34, 53]. User's rating vector contains the users' ratings, and cosine similarity is denoted by Formula 3.14.

$$Cosine(u_i, u_j) = \frac{r_{u_{i,k}}.r_{u_{j,k}}}{\|r_{u_{i,k}}\| \times \|r_{u_{j,k}}\|} = \frac{\sum_{k=1}^{n} r_{u_{i,k}}.r_{u_{j,k}}}{\sqrt{\sum_{k=1}^{n} r_{u_{i,k}}^2 . \sum_{k=1}^{n} r_{u_{j,k}}^2}} \tag{3.14}$$

where $r_{u_{i,k}}$ is the rating of user u_i on the item i_k, $r_{u_{j,k}}$ is the rating of user u_j on the same item i_k and n is the number of items in the system.

Adjusted Cosine Correlation Cosine similarity is not capable to detect the differences in rating scales between users [172]. For that, adjusted cosine Correlation is proposed. In fact, adjusted cosine correlation is used to measure the similarity between two items in an item-based collaborative filtering by taking into consideration the user average ratings from each pair of items [172, 183]. Adjusted cosine similarity between two items x and y is denoted by Formula 3.15:

$$AC(x,y) = \frac{\sum_{u \in U}(r_{u,x} - \bar{r}_u).(r_{u,y} - \bar{r}_u)}{\sqrt{\sum_{u \in U}(r_{u,x} - \bar{r}_u)^2 . \sum_{u \in U}(r_{u,y} - \bar{r_u})^2}} \tag{3.15}$$

$r_{u,x}$, $r_{u,y}$ are the rating values of the user u on the items x, y and $\bar{r}_u$ is the average of the user u ratings.

Pearson Correlation Coefficient computes the similarity between two users u and v using Pearson correlation coefficient [1, 177, 80, 189], where $r_{u,i}$ is the rating of the user u on the item i, and $\bar{r}_u$ is the average of all the ratings of the user u. Pearson correlation is given by Equation 3.16:

$$Pearson(u,v) = \frac{\sum_{i \in I}(r_{u,i} - \bar{r}_u).(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I}(r_{u,i} - \bar{r}_u)^2 . \sum_{i \in I}(r_{v,i} - \bar{r}_v)^2}} \tag{3.16}$$

Equation 3.16 is used for user-user collaborative filtering. Furthermore, Pearson correlation coefficient for item-item collaborative filtering is given by Equation 3.17, where i, j are the items, $r_{u,i}$ is the rating of user u on item i and $\bar{r}_i$ is the average rating of the ith item by those users [183, 172].

$$Pearson(i,j) = \frac{\sum_{u \in U}(r_{u,i} - \bar{r}_i).(r_{u,j} - \bar{r}_j)}{\sqrt{\sum_{u \in U}(r_{u,i} - \bar{r}_i)^2 . \sum_{u \in U}(r_{u,j} - \bar{r}_j)^2}} \tag{3.17}$$

Constrained Pearson correlation is proposed as a variant of Pearson correlation [189, 79]. This correlation can have positive and negative values, because its computation is based on a fixed midpoint

on the system's rating scale R. So, the recommender system selects neighbors whose correlation is greater than a predefined threshold with the respect to the fixed rating value R [177]. See Equation 3.18:

$$CPearson(u,v) = \frac{\sum_{i \in I}(r_{u,i} - R).(r_{v,i} - R)}{\sqrt{\sum_{i \in I}(r_{u,i} - R)^2 . \sum_{i \in I}(r_{u,j} - R)^2}} \quad (3.18)$$

where $r_{u,i}$ and $r_{v,i}$ are the ratings of the users u, v on the item i.

Spearman Correlation Coefficient Spearman rank correlation coefficient is similar to Pearson, but it computes the correlation between ratings' ranks instead of the correlation between ratings [79]. So, users' ratings are converted into ranks and Pearson correlation is applied on ranks instead of ratings. Spearman correlation between two users u and v is given bye Equation 3.19:

$$Spearman(u,v) = \frac{\sum_{i \in I}(k_{u,i} - \bar{k}_u).(k_{v,i} - \bar{k}_v)}{\sqrt{\sum_{i \in I}(k_{u,i} - \bar{k}_u)^2 . \sum_{i \in I}(k_{v,i} - \bar{k}_v)^2}} \quad (3.19)$$

where $k_{u,i}$ is the ranking of the rating of user u on item i, and $\bar{k}_u$ is the average ranking of user u ratings.

Jaccard Similarity Coefficeint between two users u_i and u_j equals to the number of items rated in common between u_i and u_j, divided by the number of u_i rated items plus to the number of u_j rated items [11]. Jaccard similarity coefficient is given by Equation 3.20 where I_{u_i} is the set of the user u_i rated items and I_{u_j} is the set of the user u_j rated items.

$$Jaccard(u_i, u_j) = \frac{\|I_{u_i} \cap I_{u_j}\|}{\|I_{u_i} \cup I_{u_j}\|} \quad (3.20)$$

Jaccard similarity coefficient is a simple measure, it deals with the number of rated items, discarding the information related to their rating values. That may reduce the accuracy of the recommender system.

Graph-Based Methods Graph theory methods could be used for computing the *nearest neighbors* for a given user in the collaborative

filtering system [5, 54]. These methods showed a great capability of solving the sparsity problem in collaborative filtering recommender systems [84].

In classical collaborative-filtering systems, where no graph representation is used, the path between two users u_1, u_2 sharing the same item i_1, is described as the sequence: $u_1 \rightarrow i_1 \rightarrow u_2$ and it has a path length equals to 3. All the paths in classical collaborative filtering systems have the same structure and the same length. While, in graph-based collaborative filtering: user-item path sequence and path length are not limited and they depend on graph structure, e.g. $u_1 \rightarrow i_1 \rightarrow u_5 \rightarrow i_4 \rightarrow u_1$. Furthermore, transitive relations could be widely exploited in recommender system using graph-based representation. Such kind of representations using bipartite graphs invoke the transitive association between users which has the important role in the recommendation process.

Figure 3.5 is an example of graph representation of the rating matrix defined by Table 3.3 and Table 3.4. The graph is a bipartite graph in which red vertices represent items and blue vertices represent users, although an item-user edge is created if the user rated the item.

Table 3.5 synthesizes the different methods used in graph-based collaborative filtering algorithms. These algorithms are explained as follows:

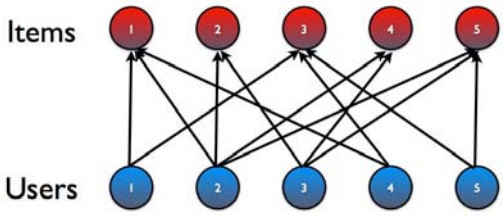

Figure 3.5: Bipartite graph extracted from item-user rating matrix

Shortest-Path based methods in [6] authors propose a graph-based algorithm called IRA (the Intelligent Recommendation Algorithm). This algorithm works on directed graphs, where vertices

Table 3.5: Graph-Based Collaborative Filtering

Graph Method	Definition	Example
Shortest path	IRA System [6]	User directed graphs Horting & Predictability edges Shortest-path algorithm
	Jumping connection [131]	User-Item bipartite graph User graph using jumping function Recommender graph equals to: User-item graph ∪ user graph Shortest path on recommender graph
	Two-layers graph RS [85]	Graph of two layers Item-Item layer & User-User layer If the user have purchased the item: Item-user connection between layers Exploring the path between layers
	Paths number [84]	User-Item bipartite graph Transitive association between users based on graph structure & paths length
Random Walk	CT, PCA CT & PI of LM [60]	User-item bipartite graph Random walk algorithm
	TrustWalker [87]	User-based trust network Random walk is applied
PageRank	ItemRank [68]	User-user graph Recommends top ranked items Using PageRank
	PathRank [104]	Graphs are heterogeneous PageRank is applied on Edges and vertices from different types

represent users and edges represent two types of user-user relation called *Horting* and *Predictability*.
Horting is a non symmetric and non transitive relation between two users u_i and u_j, in which u_i is said to hort u_j if they have rated items in common: u_i horts u_j if $(I_{u_i} \cap I_{u_j})/I_{u_i} \geq \alpha$ where I_{u_i} and I_{u_j} are the sets of u_i and u_j rated items and α is a predetermined threshold.
On the other hand, Predictability is defined as another relation between u_i and u_j in which the value of common ratings is considered. Thus, u_j is said to predict u_i: if u_i horts u_j and there exists a linear transformation $F : S \to S$, which maps u_j ratings to u_i, as given by

Equation 3.21:

$$\frac{1}{|I_{u_i} \cap I_{u_j}|} \sum_{x \in I_{u_i} \cap I_{u_j}} |r_{u_{i_x}} - f(r_{u_{j_x}})| \leq \beta \tag{3.21}$$

where β is a predefined threshold.
After applying the recommendation query for a given user u_i: users who apply the horting and predictability commonality constraints join the set P, and a shortest path in the directed graph from any user $u \in P$ to a user who have rated the item i_x is computed, if the shortest path is found then the prediction is computed considering the path weight and if not the algorithm stops with failure.

In [131] Jumping connections recommender system: recommender system dataset is modeled in a bipartite graph $G = (V, E)$ where $V = \{U \cup I\}$, U is the set of users (people), I is the set of items (movies), and the edges in E connect a user with an item if the user rates this item.
After modeling the dataset in a bipartite graph, a *Jump* function is used. The *Jump* function is given by: $J : R \rightarrow S$, where R is the recommender dataset and $S \subseteq U \times U$ is the output of the function containing pairs of users (u_i, u_j). According to the Jump function: pairs of users $(u_i, u_j) \in U \times U$ are created if and only if u_i can reach u_j via one single jump. In fact, the jump function is used to create the social network graph.
The *social network graph* $G(U, E_s)$ is the users one-mode projection graph extracted from the user-item bipartite graph. In this social network vertices U represent users and edges are created according to the *Jump function*: if the pair $u_i, u_j \in S$, then an edge e will connect the vertices u_i and u_j in the social network graph. As the Jump function ignores edges weights w authors proposed to use *Hammock jump* instead of jump function where edges are weighted by the number of common rated items between the pair of users. The weight of hammock jump w equals to the number of common items in common between users and it is called the Hammock width.
The *recommender graph* is constructed from the union between two graphs : the bipartite graph $G(V, E)$ and the social network graph $G(U, E_s)$.
The recommendation algorithm explores the shortest paths in the recommender graph, in order to achieve the recommendation. Figure 3.6 shows the user-item bipartite graph, the social network

graph and the recommender graph. In [85] authors propose rec-

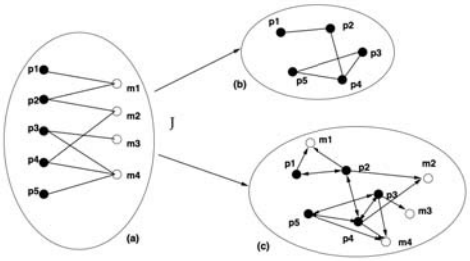

Figure 3.6: Jumping connections recommender system (a) bipartite graph, (b) social network graph, (c) recommender graph [131].

ommender system that combines content-based and collaborative filtering methods. According to that: users have feature vector representing their demographic information, and items have feature vector containing attributes and textual information describing them.
This algorithm is a graph-based recommendation algorithm in which three types of correlations are presented, as follows: user-user correlation, item-item correlation and user-item correlation, see Figure 3.7. Apparently, the recommendation graph is composed of two layers: users layer and items layer. In the users layer: each vertex represents user and each edge represents the demographic similarity between users. In items layer: each vertex represents item and each edge represents the content similarity between items. Moreover, besides the inner-layer links, inter-layer links between vertices from different layers are used to connect users in the users layer with items in the items layer. In fact, user-item connection is created if and only if the user has purchased the item.
According to this model, the recommendation process needs a graph searching algorithm in order to explore the best paths between users and items from the different layers. Thus, to achieve the recommendation two algorithms have been proposed: the first one is a simple graph search that uses only low-degree associations between users and items. While, the second algorithm is a neural network approach using Hopfield algorithm to achieve spreading activation and exploit higher degree associations between users and items. In [84] the recommendation algorithm is applied on the user-item bipartite

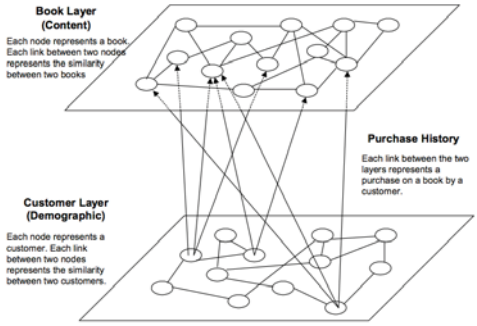

Figure 3.7: Two-layer graph model of books, customers and purchases [85].

graph $G(V, E)$, $V = \{U \cup I\}$, U is the set of vertices representing the users in the recommender system, I is the set representing the items in the recommender system, and E is the graph edges. Graph edges connect a user $u \in U$ with an item $i \in I$ if and only if the user u has bought the item i.
The algorithm recommends items to users by extracting the transitive association between users based on: the structure of the user-item bipartite graph, and the paths lengths connecting these users and items.
After building the user-item bipartite graph, user-item transitive association is computed. User-item transitive association $a(u_x, i_y)$ is defined as: the sum of all the weights of all the distinctive paths connecting u_x and i_y. In this computation, only paths whose length is less than or equal to a predefined maximum length M are considered.
Given the user-item interaction matrix A, a predefined path weight parameter α and the maximum path length M: the transitive associations between users and items are given by the matrix A_α^M as defined in Equation 3.22

$$A_\alpha^M = \begin{cases} \alpha A & \text{if } M = 1 \\ \alpha^2 A.A^T.A_\alpha^{M-2} & \text{if } M = 3, 5, 7, \ldots \end{cases} \tag{3.22}$$

As computing A_α^M requires extensive computing resources, authors proposed to use spreading activation algorithms [50] to efficiently explore transitive associations between users and items in the system. This approach works on sparse data but when data becomes

denser, spreading activation algorithms show the over-activation effect. Actually, over-activation effect is the main drawback of this approach and it appears when the user-item matrix becomes denser, leading to a worse performance compared to user-based and item-based collaborative filtering.

Random walk based methods these methods recommend items to users based on the random walk in the graph.
In [60] authors propose three graph-based recommendation algorithms, applied on a bipartite graph. These algorithms are: Average Commute Time (CT), Principal Component Analysis based on Euclidean Commute Time Distance (PCA CT), Pseudo-inverse of the Laplacian Matrix.
In Average Commute Time (CT) algorithm: distance between two vertices u_i and i_j equals to the average number of steps that a random walker takes starting from u_i, passing through i_j and ending by going back to u_i.
In Principal Component Analysis based on Euclidean Commute Time Distance (PCA CT): the eigenvector decomposition of the pseudo-inverse of the graph's Laplacian matrix is used in order to map vertices into a new Euclidean space that preserves the Euclidean Commute Time Distance. Then distances between vertices, in the reduced space, is computed to be used for items ranking.
In Pseudo-inverse of the Laplacian Matrix: is the matrix containing the inner products of the vertices vectors in the Euclidean space. So, it can be considered as a similarity measure between graph vertices. The main drawback of these methods that they do not scale very well in the case of large database.

Trust walker [87]: Authors propose another type of recommendation graph: a trust network between users.
The trust network is defined as a graph $G(U, T)$ where U is the set of users and T is the set of edges. An edge $t \in T$ connects two users $u \in U$ and $u' \in U$ if they trust each others. Then the neighborhood will be defined using the information related to the trust network instead of the information related to the similar ratings between users.
TrustWalker model is based on two leading ideas: the first idea considers searching the trust network using random walk. While the

second idea considers the probabilistic item selection, which is used for computing the similarities between items. To recommend a rating for a source user u_0 on target item i, we perform random walks on the trust network, each starting at u_0 to find a user having expressed rating for i or items similar to i. Accordingly, the random walk starts from the source user u_0, and at each step of the random walk a node u is selected. For each node u:

- If u has already rated the item i, the random walk will stop and the rating value $r_{u,i}$ will be returned.
- If not, we will face two different cases:
  - Stay at the vertex u and randomly select one of the items similar to i rated by u and return their rates $r_{u,i}$.
  - Continue the random walk, following one of the user u's trusted neighbors.

Based on the previous points, the algorithm achieves several random walks and the final predicted rating is given by Equation 3.23 as follows:

$$\hat{r}_{u_0,i} = \sum P(XY_{u_0,i} = (v,j))r_{v,j} \tag{3.23}$$

Where $XY_{u_0,i}$ is the random variable for stopping the random walk at vertex v and selecting item j rated by v, while starting the random walk fro, the source user u_0 looking for the item i.
In this algorithm there is a risk of entering a non-stop random walk, for that authors proposed to terminate the random walk after going farther than the maximum depth of the trust network.

PageRank [36] based methods ItemRank [68] is a personalized PageRank scoring recommendation algorithm which recommends top ranked items to the interested user. In this algorithm graph vertices represent items and graph edges connect items if they have rated users in common. In this algorithm, ItemRank equation is similar to PageRank equation and it is denoted by Equation 3.24:

$$IR_{u_i}(t+1) = \alpha.C.IR_{u_i}(t) + (1-\alpha).d_{u_i} \tag{3.24}$$

where:

- IR_{u_i} is the score vector for the user u_i

- $IR_{u_i}(0)$ is the initial status of the algorithm and it follows the uniform distribution: $IR_{u_i}(0) = \frac{1}{|M|}.1_{|M|}$
- M is the number of graph items
- c is the items correlation matrix
- d_{u_i} is the user u_i preferences vector: $d_{u_{i_j}} = r_{i,j}$ (the rating of user u_i for the item j) if the user u_i rates the item j and $d_{u_{i_j}} = r_{i,j} = 0$ in the other case.

The algorithm is updated iteratively until the convergence. Once $IR_{u_i}(\infty)$ has been computed, the system recommends to u_i item i_j for which IR_{u_i} is the highest.

PathRank: a Graph-based multi dimensional recommender system [104] using personalized PageRank. Recommendation is achieved on heterogeneous graphs which consist of different types of vertices and edges. So the heterogeneous graph G is represented by a set of weighted adjacent matrices $A = \{A_{t_1}, \ldots, A_{t_n}\}$ where each $A_{t_i} \in A$ represent the graph's edges whose type is t_i. Such type of graphs could contain various paths depending on the types of graph vertices and edges. Vertices could present users, items, places and more. Edges could present relations such as: friend with, lives in, likes and more. So a path is composed of a consequence of vertices from different types connected by edges from several types. e.g user u_1 lives in city (Paris) friends with user u_2 likes the movie (Titanic). The main advantage of such representations is the ability to have several recommendations according to the selected path. Thus, the recommendation query is given by: $Q = \{O, P, D_t\}$ where O is a set of entities e.g. $O = \{Toto \in users, Paris \in city\}$, D_T is the recommendation target e.g. movie, and P is the selected path extracted from the adjacency matrix A e.g. p_i with weight w_i extracted from the adjacency matrix $A_i \in A$. Then a personalized PageRank algorithm "PathRank" is used to achieve the recommendation. Such types of recommender systems are used to achieve several kinds of recommendation request depending on the type of the selected path e.g. recommend a user who lives in Paris a proper movie. In [105, 106] authors propose two algorithms using the personalized PageRank algorithm applied on the homogeneous graph (graph with one type of edges) instead of heterogeneous graph.

3.3.3 Collaborative Filtering: Limits

Collaborative filtering methods have some drawbacks [5], as follows:

3.3.3.1 Cold Start New User Problem

Collaborative filtering algorithms can predict users' preferences only by learning their precedent ratings. So, users with no rating history (mainly new users) will not receive accurate recommendations until they rate a proper number of items. To solve this problem two approaches have been proposed: the first one uses hybrid recommendation which combines content-based with collaborative filtering [38], and the second one determines the best items for new users based on item popularity, item entropy, user personalization [157, 195].

3.3.3.2 Cold Start New Item Problem (Latency)

Collaborative filtering systems depend on users' ratings on items to achieve the recommendation. Thus, if any item has no ratings, which is the case of new item, the collaborative filtering will not be able to recommend it. To address this problem hybrid recommender systems are used [175].

3.3.3.3 Sparsity

Sparsity is a fundamental problem in collaborative filtering recommender systems. Sparsity appears when the difference between the number of rated items given by users is very small compared to the number of items in the whole system. As consequence, user-item matrix R becomes very sparse and empty, which could lead to have some restrictions concerning the construction of the user neighborhood.
E.g. Amazon has millions of items, the majority of users can rate or purchase a very small number of items compared to the number of the all items in Amazon, this fact leads to have a very small ratings density and an almost empty user-item matrix.
To overcome sparsity problem, some authors proposed to add additional information (content-based) to the collaborative filtering (hybrid recommendation). In [149] authors used data mining techniques to support the ratings-based profile with additional knowledge about item similarity. Moreover, additional information such

(age, job, education, demographic segment) could be added to the user profile [152]. Moreover, singular value decomposition (SVD) can also be used to reduce the dimensionality of user-item matrix [170]. In [200] authors proposed to use taxonomy to estimate user preferences.

3.3.3.4 Scalability

As the number of users and items in the system grows, the complexity of the system increases. As consequence, the computation of neighborhood becomes more complicated. Several methods have been proposed to overcome the scalability problem such as clustering algorithms [48, 44]. But as clustering algorithms defeat the scalability effects in the system, on the other hand, they decrease the recommendation accuracy.

3.3.3.5 Grey Sheep problem

Sometimes recommender systems contain users with particular tastes. These users can have a very uncommon preferences compared to others, so it is difficult for them to find similar users and relevant neighborhoods. This problem is named Grey Sheep Problem: users who fall on the border of two clusters of users. Generally, collaborative filtering works best for users who have usual preferences, because having ordinary tastes helps users to integrate with many neighbors of similar tastes.
Finally, to overcome this problem hybrid recommender systems are used because they combine the contents of user profile (containing user preferences) with the collaborative filtering.

3.3.3.6 Non Diversity Problem

Collaborative filtering systems mainly depend on the historical ratings of users. This could lead to frequent recommendations of the most popular items, avoiding the recommendation of diverse interesting items. However, using hybrid recommender system can help to overcome this problem.

3.3.3.7 Privacy

Since the users' ratings are located in an external server, the privacy of users could be threatened by a third party which has the possibility to access the data and attack the user's privacy. Some authors propose to add random noise to user's ratings in order to overcome this problem [155].

3.4 Hybrid Recommender Systems

Hybrid recommender systems combine *Content-based* and *Collaborative Filtering* methods in one model, in order to bypass their limitations. In [38, 39] hybrid recommender systems have been classified as shown in Table 3.6:

- *Weighted Hybrid Recommenders:* use models of both content-based and collaborative filtering, in the same time, by applying the recommendation query individually on each model. Then scores are aggregated by combining the results of each model. The results are usually merged by linear combinations or vote consensus schemes. Weighted hybrid approach is very used in the literature and it is very effective in the terms of accuracy [18].
- *Switched Hybrid Recommenders:* these systems use some predefined criteria in order to switch between recommendation systems. For example, in the DailyLearner system [25] two systems have been used, the first one uses k-nearest-neighbor algorithm to model the user's short-term interests, while the second system uses a naive bayesian to learn user's the long-term interests. However, switched hybrid recommender systems show some limitations, considering the system complexity, because the switching criterions must be determined with an additional level of parameterization and this can impose more complexity to the recommendation.
- *Mixed Hybrid Recommenders:* These systems mix the results given by the different recommendation techniques and present them together. In [181] a content-based algorithm is used to recommend TV shows to users, this algorithm is based on the textual description of TV shows, and a collaborative filtering

algorithm is used for recommendation based on users' preferences. The final recommendation of this system combines the results of the both algorithms.

- *Feature combination:* In this methods of hybridization, one recommendation algorithm is applied as a main recommendation algorithm, and another recommendation algorithm is inserted, as an additional feature associated to the main algorithm. Thus, a feature combination hybrid methods borrow the recommendation logic from another recommendation algorithm rather than implementing several algorithms together [39]. For example: LIBRA recommender system [135] is a feature combination hybrid, it is used to recommend books from Amazon according to their content, such as: title and authors. And according to the collaborative information about Amazon's related users, which are treated as additional content about the book.

- *Feature augmentation:* A first recommender system produces ratings for each item in the system. Then, a second recommender system uses the obtained information as inputs for its recommendation process. So, in feature augmentation hybrid the output of the first recommender system is considered as an input to the second recommender algorithm. In [173] authors proposed to integrate the content-based ratings into the collaborative filtering model using robots named filterbots, these robots act as artificial users. Filterbots evaluate and rate new documents, to finally use their ratings in the collaborative filtering process besides to the real users ratings.

- *Cascade:* Cascade hybrid recommenders. These systems involve a staged sequential process. A first recommender produces a coarse ranking of candidates. Next, a second recommender starts from the previously filtered list as the set of candidate items, and produces a refined set of final suggestions. The benefit of these methods is that they avoid employing the second, lower- priority technique on items that are well differentiated by the first technique, or are sufficiently poorly-rated so that they will never be recommended. By doing this, cascade recommenders achieve more computationally efficient

Table 3.6: Hybrid Recommender Systems

Type	Definition	Example
Weighted	CB and CF are combined together.	Netflix prize challenge [18]
Switched	Switching between different recommenders based on predefined criterions.	DailyLearner news recommender [25]
Mixed	Mixing the results given by applying different kinds of recommenders.	PTV to recommend TV shows [181]
Feature Combination	Adding the CF as extra features to the CB algorithm.	LIBRA book recommender [135]
Feature augmentation	The output of the first system is used as an input to the second one	Filterbot for news recommendation [173]
Cascade	Refining the recommendations given by other systems	Entree restaurant recommender [38]
Meta-Level	Generated model from the first algorithm is used an input for the second one.	Collaboration via content recommender [14]

recommendations than, for example, a weighted hybrid recommender that has to apply all its techniques to all items. In addition, the cascade approach is by its nature tolerant to noise in the low-priority technique, since recommendations given by the high-priority recommender can only be refined.

- *Meta-Level:* These systems use two recommendation algorithms: the first one is used to generate a model, and the second one uses the previously generated model as an input. In [152] a collaborative via content method works as follows: at the beginning, the first recommender system generates users profiles, these profiles are represented via vectors of users features. Then the generated vectors are used as input to a collaborative filtering algorithm.

Table 3.7: Accuracy metrics used in recommender systems

Accuracy Metrics	Definition	Example
Predictive	Evaluating the proximity between the predicted ratings and the real ratings	MAE [177, 34, 80], RMSE [20]
Classification	Evaluating the frequency of having accurate and inaccurate ratings	Precision, Recall and F-measure [23, 170, 171]
Rank	Evaluating ranked lists of items	NDPM [15]

3.5 Evaluating Recommender Systems: Accuracy Metrics

Evaluating recommender systems is one of the fundamental aspects in the recommendation domain, because it determines the effectiveness of the ratings predicted by the system. For that, several metrics have been proposed. Accuracy metrics are one of the most popular metrics, that are used to measure the proximity between the system predicted ratings and the real ratings given by users. Accuracy metrics are grouped into three main groups [81]: Predictive accuracy metrics, Classification accuracy metrics and Rank accuracy metrics. Table 3.7 shows the differences between these three metrics.

3.5.1 Accuracy Metrics

Accuracy metrics are very well known metrics, that are used to measure the proximity between the system predicted ratings and the real ratings given by users. Accuracy metrics are grouped into three main groups [81]: Predictive accuracy metrics, Classification accuracy metrics and Rank accuracy metrics. Table 3.7 shows the differences between these three metrics.

3.5.1.1 Predictive accuracy metrics

Predictive accuracy metrics measure the proximity between the system's predicted ratings and the real ratings given by the user. Thus, these metrics measure the difference between the predicted ratings and the real ratings given by the users. For that several metrics have been proposed, such as: mean absolute error and root mean squared error.

- *Mean Absolute Error MAE:* finds the mean average deviation between the predicted rating and the real rating, and it is given by Formula 3.25:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|\hat{r}_i - r_i| \tag{3.25}$$

  MAE has been used in [177, 34, 80].

- *Root Mean Squared Error RMSE:* is another predictive accuracy measure, in which errors are squared before summing them, larger errors means larger RMSE. This measure has been used in Netflix prize [20] and it is denoted by Formula 3.26:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{r}_i - r_i)^2} \tag{3.26}$$

  In Formula 3.25 and Formula 3.26: N denotes the number of recommended items, $\hat{r}_i$ is the real rating given by the user and r_i is the predicted rating given by the recommendation algorithm. Moreover, minimum values of MAE or RMSE means maximum recommendation accuracy.

3.5.1.2 Classification accuracy metrics

Classification accuracy metrics, also named decision-support metrics are used to measure the frequency of having accurate or inaccurate ratings about items. Furthermore, Precision and Recall are good examples of classification accuracy metrics, and they are widely used in the classical information retrieval methods. Precision and recall are defined as follows:

- *Precision*: according to information retrieval methods, precision equals to the ratio of the number of true positive values tp (relevant recommended items), to the sum of the number of true positive values tp and the number of false positive values fp (the sum of relevant and irrelevant recommended items). Thus, according to recommender system methods, precision estimates the probability of recommending relevant items [23, 170, 171], as described in Formula 3.27:

$$P = \frac{tp}{tp + fp} \tag{3.27}$$

- *Recall* according to information retrieval methods, recall equals to the number of true positive values tp (relevant recommended items)divided by the sum of the number of true positive values tp and the number of false negative values fn (sum of relevant recommended items and relevant non-recommended items, user's preferred items). According to recommender systems, recall represents the probability that a relevant item will be recommended. Recall is denoted by Formula 3.28:

$$R = \frac{tp}{tp + fn} \tag{3.28}$$

- *F-Measure*: as precision and recall are inversely related, and as precision and recall should be studied together in order to understand the performance of recommendation algorithms, f-measure is proposed as a single measure that combines precision and recall in one formula[170, 171]. F-measure is denoted by Equation 3.29:

$$F = 2 \times \frac{P \times R}{P + R} \tag{3.29}$$

Figure 3.8 explains how precision and recall are calculated.

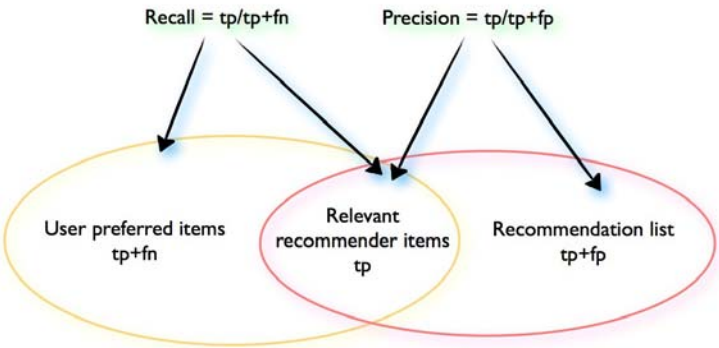

Figure 3.8: Preicsion and recall

3.5.1.3 Rank accuracy metrics

Rank accuracy metrics are used to propose ranked recommendation lists to users. Thus, such metrics [81] matches a list of recommended

items with a user's list of ordered items. Normalized Distance-based Performance Measure NDPM is one of the common metrics that are used as rank accuracy measure.

- Normalized Distance-based Performance Measure NDPM: is used to compare two different weakly ordered ratings [15] and it denoted by Equation 3.30:

$$NDPM = \frac{2C^{-} + C}{2C^{i}} \tag{3.30}$$

Where C^{-} is the number of contradictory preferences between system's ranking and user's ranking, C^{i} is the number of preferred items in the user's ranking and C is the number of compatible preferences between user's ranking and system's ranking.

3.5.2 Non-Accuracy Metrics

As mentioned in [81] recommendation accuracy is not enough to determine whether the recommendation is effective or not. For that, recommender system usefulness is proposed to move beyond the accuracy measure to include the suitability of the recommendations to users [81]. Recommender system suitability or usefulness comprises several measures as described in [81]:

- Coverage: measures the percentage of a dataset that the recommender system is able to provide prediction for.
- Confidence: this measure determines how sure is the recommender system that its recommendation is accurate.
- Learning rate: measures how quickly an algorithm can produce good recommendations.
- Novelty or Serendipity: some recommender systems propose a very accurate recommendations but still are not useful for users. Novelty metrics help to measure how much is the user aware of new and unknown items.
- User satisfaction: measures the satisfaction of the user on the recommender system performance.

3.6 Conclusion

Recommender systems are very important, because they help users to find interesting items based on their tastes and preferences. In fact recommender systems have changed the classical methods in which queries are applied on static datasets to extract information, by proposing new algorithms which are oriented on user's tastes and opinions. In recommender systems user's interests are classified according to their historical ratings about different items.

In this chapter we introduced a brief review about recommender systems. Recommender systems have three main types: Content based, Collaborative filtering and hybrid recommendation. In content based recommendation the content of user interests and item features are involved in the recommendation process. In collaborative filtering recommender systems, recommendations are based on the ratings of like minded users. Hybrid recommender systems combines content based methods with collaborative filtering methods. Moreover, content-based and collaborative-filtering algorithms have two main types model-based and memory-based, and each of these type has its own methods.

Also, we introduced the user-based and the item-based collaborative filtering and we compared them. In fact, memory-based collaborative filtering use some of the well known similarity measures such as cosine similarity, pearson correlation and more. Moreover, memory-based collaborative filtering uses graph-based methods, which are very interesting for our thesis. Also, we introduce the limitations of each type of recommender system and finally we present some of the well known measures which are used to evaluate the recommender systems.

Chapter 4

Semantic-Social Recommender System

4.1 Introduction

In this chapter we present a new approach of recommender systems called *semantic-social recommender system*.
Semantic social recommender system includes two types of information *semantic information* and *social information*.
Semantic information is related to the semantic representation and semantic relevance between user profile and item profile. We propose, for that, using ontology to represent user profile and item profile.
Social information is related to users position and role (centrality) in the social network, besides to the type of connections and social ties between these users.

Our motivation, in this approach, is to *recommend* an input item to a group of users connected via a social network or collaboration network, using semantic information and social information.

Figure 4.1 illustrates the main axes of our model, which we will detail later in this chapter.

Figure 4.2 gives a global view of our approach in recommendation. In this Figure we notice that users and items are attached to a taxonomy, which is a special case of ontology. Moreover, users are represented as vertices of the collaboration network. The rec-

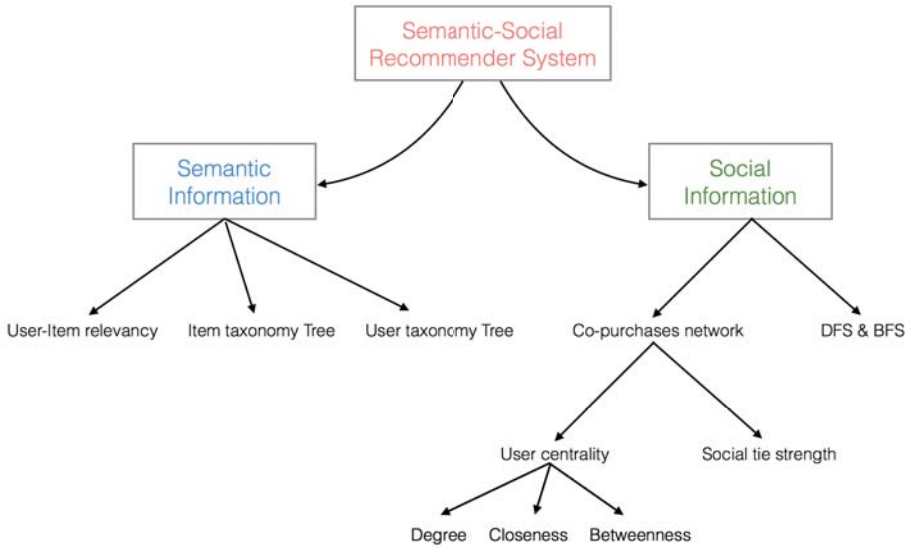

Figure 4.1: Semantic-social recommendation global details

ommendation query (the algorithms input) is defined as an item to be recommended to users connected via a collaboration social network. The output of the algorithms is a recommendation list which contains all the relevant users to the input item.

This chapter is organized as follows, Section 4.2 and Section 4.3 represent a detailed explications about semantic information and social information proposed in our approach. Section 4.4 describes our proposed algorithms, and finally we conclude in Section 4.5.

4.2 Semantic information

In this section we investigate the semantic part of our new approach, and we show how the *semantic information* can be used in the semantic-social recommender system.

According to our approach, the *semantic information* depends on two points: the first point is related to the taxonomy representation of information, particularly information about user preferences

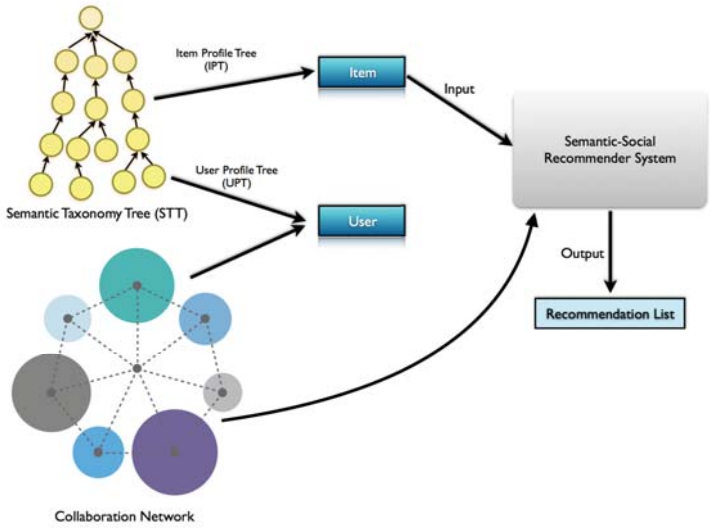

Figure 4.2: Semantic-social recommendation a global view

(user profile) and item features (item profile); while the second point is related to the semantic similarity between user preferences and item features.

This section contains a brief background about taxonomy, and a detailed explanations about the semantic part of semantic-social recommender system.

4.2.1 Background

4.2.1.1 Taxonomy

Taxonomy is a class/subclass relationship between a collection of entities, describing a certain object or a certain domain of objects. Taxonomy has a hierarchical structure, characterized by the 'is-a' hierarchy, and it is organized in a tree in which nodes and edges represent connected concepts or categories. In most cases, taxonomy is established according to a predefined human design or some other criteria [86]. The well known linguistic database *Wordnet* is an example of a taxonomy [130, 88, 92]

In taxonomy the nodes are placed in several levels, from level 0 to level $n-1$. Level 0 contains the most general nodes (concepts) in the domain (0 out-degree), while level $n-1$ contains the most specific nodes (concepts) in the domain (0 in-degree). Thus, moving deeper in taxonomy means moving from general concepts to more specific concepts.
Figure 4.3, is an example of fruit taxonomy.This taxonomy has been extracted from food domain taxonomy, and it has 5 levels, starting from the concept "Thing", the most general concept in the taxonomy, in level 0, and ending by the concept "Orange", one of the most specific concepts, in level 4.

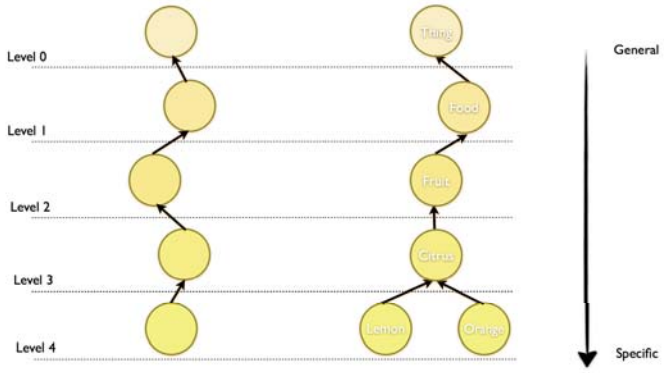

Figure 4.3: Taxonomy Example

Taxonomy as a special case of ontology The origins of ontology come from philosophy, where ontology was firstly introduced by *Aristotle* as the study of the characteristics of being or existence. Computer science has borrowed the notion of ontology from philosophy and applied it on several domains such as natural language applications, knowledge representation, database design, object-oriented analysis, information retrieval and other domains [75]. According to computer science, *ontology* is defined as an explicit specification of conceptualization [72, 73]. In ontology the concepts are grouped in a semantic graph, the ties between these concepts represent semantic relations or specification relations, such as 'is-a' relation in taxonomy. In ontology concepts or terms are grouped in classes, these

classes can have several kinds of relationships between them. Moreover, each class can have several kinds of properties, several kinds of attributes and values correlated to the attributes. Furthermore, ontology has the reasoning characteristics that is used to analyze the relationships between ontology classes and to reason their properties and attributes.

Ontology definition is very generic and it can carry out several interpretations. Moreover, in certain cases taxonomy is considered as a special case of ontology; and ontology can be defined as a taxonomy with additional properties [160]. Generally, taxonomy has a tree structure while ontology has a forest structure. Also, taxonomy is easy to read and interpret, while ontology needs a special specification languages in order to define the concepts and their relationships and characteristics. Furthermore, the reasoning process in taxonomy is little limited compared to ontology.

Taxonomy in Recommender Systems In recommender systems taxonomy can be used to represent user profile, where user preferences can be classified according to a semantic taxonomy tree, derived from a global domain taxonomy. Moreover, the same thing can be done on item profile by representing item features using a semantic taxonomy tree. Furthermore, *classification* and *specification* characteristics of taxonomy can be very helpful methods used for recommendation. Thus, classification and specification are a very useful methods that are used to estimate the users' preferences, especially in the case of having lack of information about user preferences or item features [200, 43, 202].

4.2.1.2 Semantic relevancy meausres

Semantic similarity measures are widely used to compute the semantic relevance between any given pair of concepts in the ontology. In recommender systems, using taxonomy, semantic similarity measures are very important to find out the relevance between users and items.

The existing semantic similarity measures have three principal types [176]: Information content or node-based [163, 111], distance-based or edge-based [163, 154, 103], and hybrid [88].

Information Content (Node-Based) approach *Information Content* approach or *Node-Based* approach is used to compute the similarity between two concepts in any given taxonomy.
Let T be an 'is-a' taxonomy and C the set of all the concepts of this taxonomy, the similarity between two concepts $c, c' \in C$ equals to the extent to which c and c' share information in common. This can be identified by determining a specific concept in the taxonomy representing the lowest common ancestor between the two concepts c and c'.

According to information theory [166], the information content of a concept c can be quantified as the negative value of the log likelihood $IC(c) = -\log p(c)$, where $p(c)$ is the probability that the concept c has an instance. Based on $IC(c)$ if the probability increases, then informativeness decreases. That means the more abstract a concept, the lower its information content. In [163] the information content similarity between two concepts c and c' in any given taxonomy is denoted by Formula 4.1:

$$sim(c, c') = max_{c'' \in S(c,c')}(-\log \frac{1}{p(c'')}) \tag{4.1}$$

where $S(c, c')$ is the set of common ancestors subsuming c and c'. However, this measure indicates that: the more information c and c' share in common, the more similarity they have. Moreover, the number of common information between c and c' is correlated to the number of common ancestors between them.

In [111] the author proposed another variant of node-based similarity measure, by finding the ratio between the similarity given by Formula 4.1 and the sum between the information needed to describe c given by $IC(c) = -\log \frac{1}{p(c)}$ and the information needed to describe c' given by $IC(c') = -\log \frac{1}{p(c')}$. As denoted in Formula 4.2:

$$sim(c, c') = \frac{-2 \times max_{c'' \in S(c,c')}(\log \frac{1}{p(c'')})}{\log \frac{1}{p(c)} + \log \frac{1}{p(c')}} \tag{4.2}$$

where $S(c, c')$ is the set of common ancestors subsuming c and c'.

Distance-Based (Edge-Based) approach or *Edge-Based* approach is a very simple measure used to compute the similarity between taxonomy's concepts. This measure counts the number of edges of the

shortest path connecting the two concepts of the given taxonomy. Let T be a taxonomy, with a set of concepts C, the similarity between the concepts $c, c' \in C$ is given by Equation 4.3 as follows:

$$sim(c, c') = (2 \times D) - len(c, c') \tag{4.3}$$

where D is the maximum depth of the taxonomy, and $len(c, c')$ is the shortest path between the two concepts c and c'.

In [103] authors proposed another measure considered as a variant of distance-based similarity measure. This measure scales the shortest path between c and c' by dividing it by twice of the maximum path length in the taxonomy, as denoted by Formula 4.4:

$$sim(c, c') = -\log \frac{len(c, c')}{2 \times D} \tag{4.4}$$

Hybrid approach Hybrid approach combines the two previous similarity measures, node-based and edge-based, in one similarity measure by adding the information content as a decision factor [88]. This similarity is given as follows:

$$sim(c, c') = -\log \frac{1}{p(c)} - \log \frac{1}{p(c')} - 2 \times (max_{c \in S(c,c')}(-\log \frac{1}{p(c)}))$$

where $-\log(\frac{1}{p(c)})$ and $-\log(\frac{1}{p(c')})$ are the information content of c and c'.

4.2.2 Semantic infromation in semantic-social recommender system

The semantic information part of our contribution includes three main aspects, as follows: semantic-taxonomy tree or domain taxonomy, semantic-tree for user profile and semantic-tree for item profile.

4.2.2.1 Taxonomy

In our approach we propose to represent all the knowledge in the recommender system in a form of taxonomy named semantic taxonomy tree STT or domain taxonomy. However, the taxonomy representation of knowledge in the recommender system contains the conceptual representation of all the items in the recommender system.

Definition 15 (Semantic Taxonomy Tree STT)**.** *Semantic Taxonomy Tree* (STT) *is a tree in which* nodes *are the domain terms, and* edges *represent the hierarchy between these terms. An STT is represented as a set of pairs of the form* $(term, level)$*, where level is an integer. When an STT has n levels, the term of level* 0 *is the most general term, while the terms of level* $n-1$ *are the most specific terms in the domain. Moreover, we assume that every item is associated with a* unique *leaf of the STT.*

Figure 4.4 is an example of a Movies Semantic Taxonomy Tree (Movies STT). In this example, domain terms represent the genres of movies, and an 'is-a' hierarchy is defined between these genres. According to Definition 15:
$STT = \{(Thing, 0), (Film, 1), (Adventure, 2), (Thriller, 2),$
$(Crime, 3), (Drama, 2), \dots\}$.
In Movies STT the maximum number of levels is equal to 5, genre in level 0 is the most general genre 'Thing'; and genres in level 4 are the most specific ones e.g. the genre 'Western'.

Figure 4.5 is an example of Amazon Semantic Taxonomy Tree (Amazon STT). In this example the domain taxonomy represents books categories, and an 'is-a' hierarchy is defined between these categories. According to Definition 15
$STT = \{(Thing, 0), (Book, 1), (HumanScience, 2), (Philosophy, 3),$
$(JavaScript, 5), \dots\}$.
In Amazon STT the maximum number of levels is equal to 5, the category in level 0 is the most general category in the domain 'Thing'; while categories in level 5 are the most specific ones, such as 'PHP'.

4.2.2.2 Item profile

Item profile holds on all the possible information describing the item. These information are mainly related to the features and the characteristics of this item. Item profile is a conceptual profile, where the topics of item's features are represented via a semantic tree. Item profile is defined as follows:

Definition 16 (Item Profile Tree IPT(x))**.** *Given an STT and an item x, we associate the item x with a subset of STT. This subset, called* Item Profile Taxonomy Tree *and denoted by* $IPT(x)$*.* $IPT(x)$

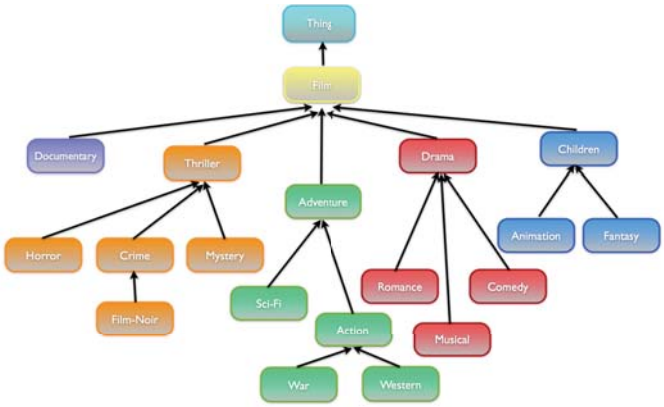

Figure 4.4: Movies semantic taxonomy tree *STT*

is the path of the STT containing all the pairs (τ, λ) *connecting the root of the STT with the leaf to which item* x *is associated.*

Figure 4.6 is an example of a western movie profile. Obviously, this movie is associated with the movie semantic taxonomy tree described by Figure 4.4. According to this example:

$$IPT(x) = \{(Thing, 0), (Film, 1), (Adventure, 2), (Action, 3), (Western, 4)\}$$

Figure 4.7 is another example of a book about java script associated with books semantic taxonomy tree described by Figure 4.5. According to this example:

$$IPT(x) = \{(Thing, 0), (Book, 1), (ComputerScience, 2), (Programming, 3), (WebProgramming, 4), (JavaScript, 5)\}$$

4.2.2.3 User profile

User Profile contains all the possible information about user's preferences and activities. User profile has a conceptual representation, in which all the topics of user's preferences are grouped in a semantic tree. User profile is defined as follows:

Definition 17 (User Profile Tree UPT(u))**.** *Let* u *be a user and* $I(u)$ *its associated set of items, that the user has liked or bought in*

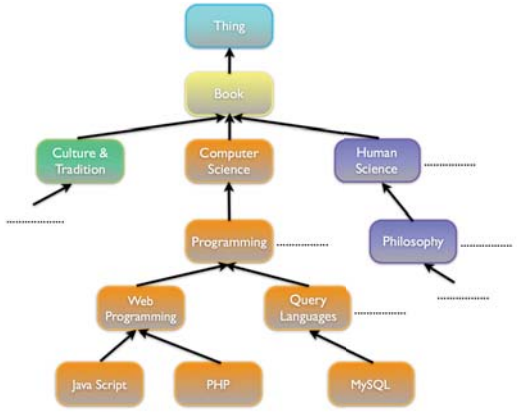

Figure 4.5: Amazon semantic taxonomy tree STT

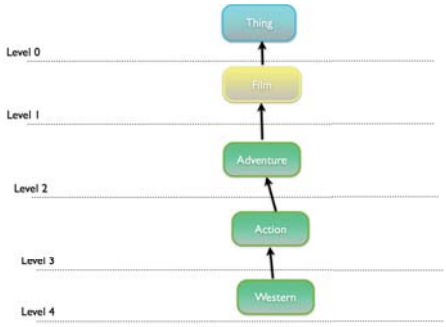

Figure 4.6: Example of movie item profile tree

the past. User Profile Tree *of user u, denoted by* $UPT(u)$*, is the union of all the item profile trees of all the items in* $I(u)$*. In other words* $UPT(u) = \bigcup_{x \in I(u)} IPT(x)$.

Figure 4.8 is an example of user profile tree. From this figure we can find that user u has two items (movies) on his/her list of preferred items. According to our proposed definition: UPT(u) is generated by finding the union of the item profile trees of movie1 and movie2, as follows:
$UPT(u) = IPT(movie1) \cup IPT(movie2)$.

Figure 4.9 is another example of user profile tree. From this figure

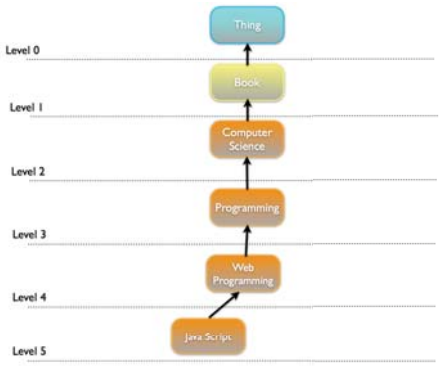

Figure 4.7: Example of book item profile tree

the user u has three items (books) in his/her list of preferred items. So according to our proposed definition, user profile tree of the user u is given as follows:
$UPT(u) = IPT(Book1) \cup IPT(Book2) \cup IPT(Book3)$.

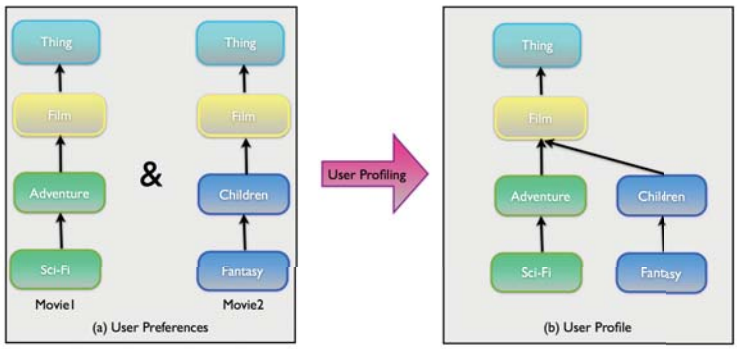

Figure 4.8: User profile tree of movies example

4.2.2.4 User-item semantic relevancy measure

Based on the previous definitions, we present a user-item semantic relevancy measure between users and items. To do so, we introduce a similarity measure between two profile trees P_1 and P_2, denoted

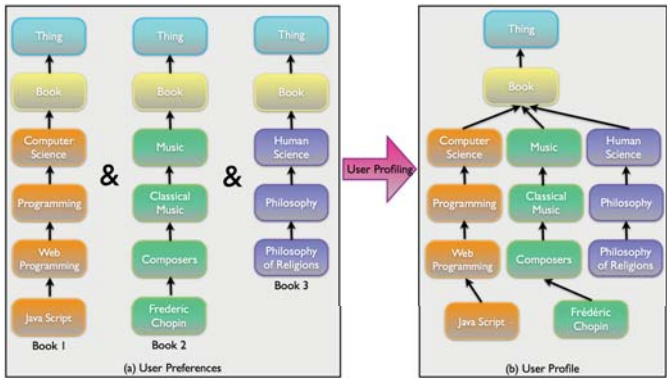

Figure 4.9: User profile tree of books example

by $\sigma(P_1, P_2)$, as follows:

$$\sigma(P_1, P_2) = \frac{1}{\mu}\left(\sum_{(\tau,\lambda)\in P_1\cap P_2} \lambda\right)$$

where $\mu = \min\left(\sum_{(\tau,\lambda)\in P_1} \lambda, \sum_{(\tau,\lambda)\in P_2} \lambda\right)$, τ is the common concepts of the profiles P_1 and P_2 and λ is the level of the common concept.

Using σ, we now define the relevancy between a user u and an item x as the similarity between their profile trees.

Definition 18. *Let u be a user and x an item. The user-item relevancy measure between u and x, denoted by $sim(u, x)$, is defined as follows:*

$$sim(u, x) = \sigma(UPT(u), IPT(x))$$

As can be seen from Definition 18, $sim(u, x)$ is not a standard similarity measure since it applies two arguments of different types (namely a user and an item). In fact, $sim(u, x)$ measures to which extent items 'similar' to x can be found in $UPT(u)$. Notice in this respect that if u bought and liked x, then $ITP(x)$ is a subtree of $UPT(u)$, in which case $sim(u, x)$ is maximal, that is, equal to 1.

Example 8. Figure 4.9 shows an example of user u, who likes the items $Book1$, $Book2$ and $Book3$.
In this case, we have $I(u) = \{Book1, Book2, Book3\}$ and $UPT(u) = IPT(Book1) \cup IPT(Book2) \cup IPT(Book3)$.

For an item x such that:
$IPT(x) = \{(Books, 0), (ComputerScience, 1), (Programming, 2), (ObjectOriented, 3)\}$.
The relevance between u and x $sim(u, x)$ is computed by the following steps:

1. $UPT(u) \cap IPT(x) = \{(Books, 0), (ComputerScience, 1), (Programming, 2)\}$, and $\sum \lambda = 3$
2. $\sum_{(\tau,\lambda) \in UPT(u)} \lambda = 38$
3. $\sum_{(\tau,\lambda) \in IPT(x)} \lambda = 6$

As a result, the user-item relevancy between x and u is equal to the ratio between the number of the common terms between x and u, and the minimum sum of levels between $UPT(u)$ and $ITP(x)$, $sim(u, x) = 3/6 = 0.5$.

Example 9. The same thing for Figure 4.8. The user u likes items $Movie1$ and $Movie2$.
So, $I(u) = \{Movie1, Movie2\}$ and $UPT(u) = IPT(Movie1) \cup IPT(Movie2)$.
For an item x such that:
$IPT(x) = \{$(Thing,0),(Film,1),(Adventure,2),(Action,3),(Western,4)$\}$.
$sim(u, x)$ is computed as follows:

1. $UPT(u) \cap IPT(x) = \{(Thing, 0), (Filme, 1), (Adventure, 2)\}$
2. $\sum_{(\tau,\lambda) \in UPT(u)} \lambda = 11$
3. $\sum_{(\tau,\lambda) \in IPT(x)} \lambda = 10$

The relevance between u and x is $sim(u, x) = 3/10 = 0.3$

4.3 Social information

In this section we present a detailed description about *social information* that we propose to use in our semantic-social recommender approach. For that, at the beginning, we present a general background about graph and social network analysis, then we explain how we use centrality measures and social ties between users in the semantic-social recommender system.

4.3.1 Background

A *graph* G is composed of two sets, a set of vertices and a set of edges. Every edge from the set of edges connects two vertices from the set of vertices.
Social network is defined as a graph in which vertices represent actors or users and edges represent the social ties or relationships between these users [145]. Furthermore, *Collaboration Networks* are known as a special type of social networks, obtained from bipartite graphs.
On the other hand, *Graph Searching* algorithms are used to explore the graph by visiting all its vertices in order to search, verify or change the values attached to these vertices. In our approach we use two graph searching algorithms. These algorithms are Depth-First Search algorithm (DFS) and Breadth-First Search (BFS) algorithm. Furthermore, we propose to apply these algorithms on collaboration networks extracted from user-item bipartite graph representing user item relationship in the recommender systems.

User-item bipartite graph and co-purchases collaboration network, that we use in our approach, are defined by the following definitions:

Definition 19 (User-item biparite graph)**.** *The user-item bipartite graph G is a graph, with a set of vertices $V(G)$ and a set of edges $E(G)$. $V(G) = \{U \cup I\}$, where U is the set of all the users, in the recommender system, and I is the set of all the items, in the recommender system, U and I are disjoint sets. In G, every edge e from the set $E(G)$ connects users u with items i if the user u has purchased, rated or liked the item i.*

Definition 20 (Co-purchases collaboration network)**.** *Co-purchases network is the weighted graph G which represents the users one-mode projection of the user-item bipartite graph G (defined in Definition 19). $U(G)$ is the set of users and $E(G)$ is the set of edges between these users. An edge e from $E(G)$ connects two users u and u' from $U(G)$ if u and u' have purchased at least one item in common, and the weight of the edge is correlated to the number and the rating of the common purchased items between u and u'.*

We can also call the co-purchases collaboration network, the co-rated collaboration network. This depends on the types of social

ties between users e.g. if users buy items then we call the network co-purchases network, and if users rate items then we call the network co-rated network.

Example 10. Figure 4.10 illustrates the user-item bipartite graph $G(V, E)$, where V is the set of graph vertices and E is the set of graph edges. $G(V, E)$ is represented in (a); the users one-mode projection is represented in (b); and the items one-mode projection is represented in (c).
The vertices set $V(G)$ is combined of two disjoint sets the set of users $U = \{A, B, C, D, E\}$ and the items $I = \{1, 2, 3, 4\}$. In the bipartite graph G, only user-item connections are possible, and a user u is connected to an item i if u has purchased or rated i (these connections are determined according to the recommender system). Moreover, the users one-mode projection is obtained if users u_1 and u_2 have purchased a same item i, that is if there exists an item i such that $e(u_1, i)$ and $e(u_2, i)$ are edges in $G(V, E)$. Similarly, the items one-mode projection is obtained from $G(V, E)$ by connecting items i_1 and i_2 if there exists a user u such that $e(u, i_1)$ and $e(u, i_2)$ are edges in $G(V, E)$.

In our approach, after constructing the users co-purchases network, we explore this network, using DFS and BFS algorithms, in order to achieve the recommendation; and we profite from the social information related to user centrality in the network and the weights of the social ties (edges) in the network.

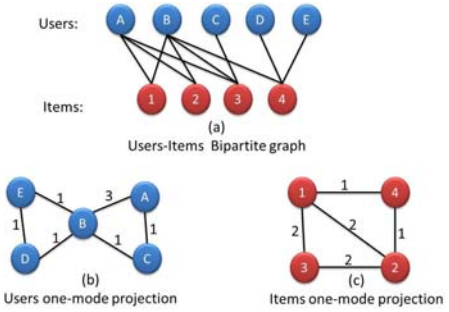

Figure 4.10: User-item bipartite graph (a), users one-mode projection (b), and items one-mode projection (c)

Table 4.1: User centrality & recommender systems (RS)

Centrality	User characteristics in (RS)
Degree	Number of direct connections Immediate risk and influence Information broadcasting
Closeness	Number of connected shortest paths Information diffusion Information reception
Betweenness	Frequency of passing through shortest paths Information broadcast inter communities

4.3.2 Social information in recommender system

In the *social information* part of our approach, we take into consideration (a) user centrality in the co-purchases network and (b) weight of the social ties (edge weight) in the co-purchases network. User centrality determines the role of the user in the network, such as degree centrality, closeness centrality and betweenness centrality; while edge weight determines the strength of the connections between the users in the network.

4.3.2.1 User centrality

User or vertex centrality in social networks is important, because it determines the role and the influence of the users in the network. Thus, user centrality is helpful to be used in recommender systems. In our approach, we use and compare three types of user centralities: degree centrality, closeness centrality and betweenness centrality. Moreover, we study the case of combining two different centralities in one recommender system e.g. combining degree centrality with betweenness centrality.
Table 4.1 synthesizes users centralities and their role in recommender systems.

Degree centrality of a given vertex (user) counts the number of vertices (users), which are in direct connections with this user. Degree centrality is used to measure the immediate risk or influence of a user in the social network, and to measure the capacity of information broadcasting [31, 61, 145]. In recommender systems, degree centrality can be very useful where a higher degree centrality of a

user means a higher recommendation influence on other users in the system.

Closeness centrality counts the number of the shortest paths connecting a given vertex (user) with other vertices (users) in the network. Closeness centrality plays an important role in information diffusion and reception. In fact, users with high closeness have a good position in the network, with large number of shortest paths connecting them with other users [31, 61, 145]. In recommender systems applied on social networks, users having high closeness centrality are most likely to receive and broadcast recommendations easily and in a very short time.

Betweenness centrality indicates the frequency of the shortest paths passing through a given vertex (user) in the network [145]. In fact, betweenness centrality refers to the user ability to facilitate information flow in the network [31]. Often, vertices (users) having high betweenness centrality connect the network communities. Therefore, in recommender systems, betweenness centrality can be used as a mediator to transmit the information between different communities in the network.

4.3.2.2 Social ties and edge weights

In most of the collaboration networks social ties (edges) are weighted. These weights correspond to the number of common preferences between users. Therefore, edges weights in collaboration networks can play an important role in semantic-social recommender system, because edges weights depend on the number of common preferred items between users. So, the higher the weight of edges connecting users, the more likely that users have similar tastes and similar opinions.

According to our approach, the edge weight between two users depends on the number and the rating of common preferred items between them. In semantic-social recommender system edge weights are defined as follows:

Definition 21. *Let u and u' be two users, and let $I(u)$ be the set of items purchased by the user u, and let $I(u')$ be the set of items*

purchased by the user u', such that $I(u) \cap I(u') \neq \emptyset$. The weight *of the edge $e(u, u')$, denoted by $w(u, u')$ is defined by:*

$$w(u, u') = \frac{1}{|I(u) \cap I(u')|} \sum_{i \in I(u) \cap I(u')} (r(u, i) + r(u', i))$$

In our approach, we propose to connect users having rating values of the range $[3 - 5]$, from the total range $[1 - 5]$.

Intuitively, Definition 21 means that 'the more users u and u' are connected (the higher the weight of the edge $e(u, u')$), the more u and u' purchased and liked items in common'. Therefore, it makes sense to state that recommending item x to u should imply recommending x to u'.

4.4 Semantic-social recommender system: Algorithms

In this section we present our proposed recommendation algorithms, named semantic-social depth-first search algorithm (SSDFS) and semantic-social breadth-first search algorithm (SSBFS). Each of these algorithms have two versions vertex-based and vertex-edge-based. Also, we discuss the cases of having several centralities, such as degree centrality, closeness centrality, betweenness centrality or even combining two different types of centralities. Our proposed algorithms are summarized in Figure 4.11, and they are presented in the following subsections.

4.4.1 Recommendation query

Let x be an item described by item profile tree $IPT(x)$, and let G be the users co-purchases collaboration network in which each user has user profile tree $UPT(u)$.
The recommendation query is the item x described by its profile $IPT(x)$ submitted to the co-purchases network G. Our goal is to recommend the item x to a group of users u from the network G.

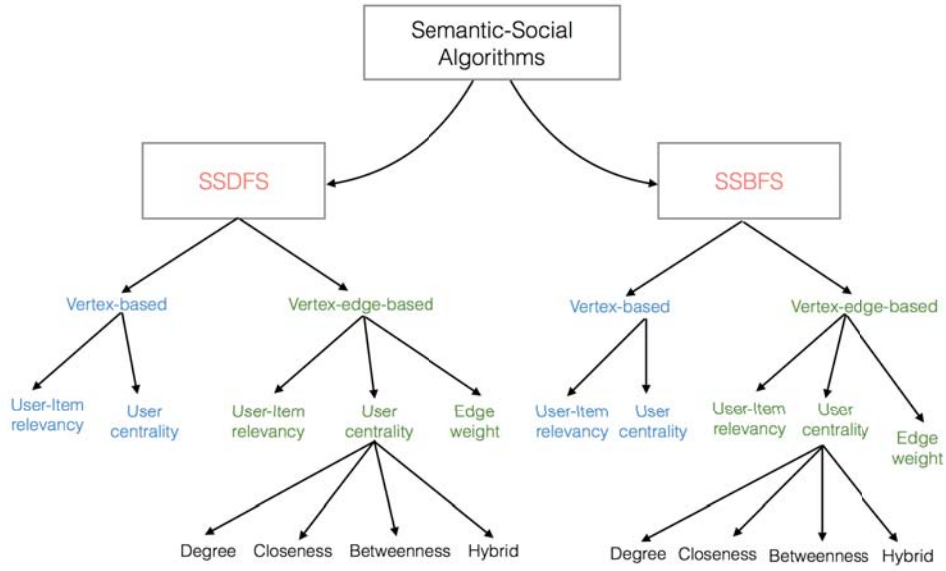

Figure 4.11: A summary of SSDFS and SSBFS algorithms

4.4.2 Semantic-social depth-first search SSDFS

Depth-First Search algorithm (DFS) is one of the fundamental graph searching algorithms. This algorithm searches deeper in the graph whenever it is possible, and it explores *all* the graph vertices [49].

In our approach, the graph to be searched is the *social network* represented by the *co-purchases collaboration network*.
Using DFS strategy our algorithms explore the co-purchases collaboration network. However, since this social network is huge, we apply heuristics in order to avoid exploring *all* the network vertices. The heuristics we propose to use with DFS are based on semantic information and social information as follows:

1. Semantic Information represented by:
   (a) Item profile tree $IPT(i)$ as defined in Definition 16.
   (b) User profile tree $UPT(u)$, as defined in Definition 17.
   (c) User-item semantic relevancy measure used to evaluate the

relevance between user preferences and item features, as defined in Definition 18.

2. Social Information represented by:
   (a) The co-purchases collaboration social network, as defined in Definition 20.
   (b) Users role and position in the network according to user centrality.
   (c) The weights of social ties in the network as defined in Definition 21.

Our proposed algorithm uses semantic information and social information as heuristics integrated with depth-first search algorithm, for that we suggest to call this algorithm *semantic-social depth-first search* SSDFS. This algorithm can be used with two heuristics vertex-based and vertex-edge-based.

In this subsection we introduce *semantic-social depth-first search* algorithm SSDFS with the two distinct heuristics *vertex-based* and *vertex-edge-based*.
According to SSDFS these heuristics are seen as recursive procedures that are called by the same main algorithm that we present next in Algorithm 5.

Algorithm 5: SSDFS Main Algorithm

Input: (i) An item x having $IPT(x)$
(ii) A positive integer n
(iii) A user-item similarity threshold θ
(iv) An edge weight threshold δ
Output: Recommendation list: *user_list*

```
1: for all vertices v in V(G) do
2:     v.label = unvisited
3: end for
4: user_list = empty list
5: Compute the centrality of every vertex in V(G)
6: vertex-centrality-vector = n vertices of V(G) with the highest centrality
7: for all v in vertex-centrality-vector do
8:     call one of the two heuristic algorithms with input v
9: end for
10: return user_list
```

4.4.2.1 SSDFS the main algorithm

Here we present a detailed explanation about SSDFS algorithm, considering its parameters, its input and output. Moreover, we discuss the proposed heuristics attached to SSDFS algorithm vertex-based and vertex-edge-based.

Algorithm Parameters SSDFS algorithm requires three types of parameters, as follows: n which is related to vertex-centrality-vector N, δ which is related to the edges weights and θ which is related to the user-item relevancy measure. These parameters are defined as follows:

- Vertex-centrality-vector N: is a vector of size equals to n. n is the number of users from the co-purchases collaboration network, having highest centrality value in the network.
- User-Item relevancy threshold δ: which determines the minimum accepted similarity between user preferences and item features in co-purchases network, that lets the algorithm continues the network searching.
- Edge weight threshold θ: which determines the minimum accepted edge weight in the co-purchases network, that lets the algorithm continues the search in the network.

In SSDFS algorithm with vertex-based heuristic we use two thresholds n and δ; while in SSDFS with vertex-edge-based heuristic we use the three thresholds n, δ and θ.

Algorithm input An item x with tree profile $ITP(x)$ as in Definition 16; and the algorithm's thresholds n, δ and θ

Algorithm output Recommendation list containing all the recommended users, whose preferences are relevance to the item x (the input item).

SSDFS algorithm It should be first noted that, in our algorithms, the vertices of the co-purchases collaboration network G to be explored are labeled in order to visit each of them at most once. More

precisely, every vertex v of G is associated with a label whose possible values are *unvisited* or *visited*.
As we would like to avoid visiting all the vertices of G, but in order to visit as many relevant vertices as possible, we choose to start the exploration of G through the vertices that have the highest *centrality* in the network, that is, the vertices that are connected to a high number of other vertices (e.g. the case of degree centrality).
To this end, the centrality of every vertex is computed (see line 5 of Algorithm 5) and the n vertices having the highest centrality are stored in a vector called vertex-centrality-vector N (see line 6 of Algorithm 5). Starting form the vertices in vertex-centrality-vector, the co-purchases network is explored using DFS strategy and according to two heuristics called Vertex-Based and Vertex-Edge-Based heuristics, as will be presented next (see line 8 of Algorithm 5).

4.4.2.2 Vertex-Based Heuristic

Considering a predefined parameter n and a predefined user-item semantic relevancy threshold δ.
SSDFS algorithm starts the search from the first vertex v in vertex-centrality-vector N (vertex with heights centrality), v is processed if it is still unvisited and if its similarity to the item x is greater than the threshold δ. Then, all successors of v are recursively processed in the same way, until reaching a vertex that fails to satisfy the similarity requirement or until the vertex-centrality-vector becomes empty. The corresponding procedure is shown in Algorithm 6.
Example 11 details how vertex-based SSDFS achieves the recommendation, and Figure 4.12 illustrates how Algorithm 6 is applied on the co-purchases collaboration network.

Algorithm 6: Vertex-based-visit

```
1: if v.label = unvisited then
2:    v.label = visited
3:    if sim(v, x) > δ then
4:       Add v to the current value of user_list
5:       for all e = (v, v') and (v', v) in E do
6:          Vertex-based-visit(G, v')
7:       end for
8:    end if
9: end if
```

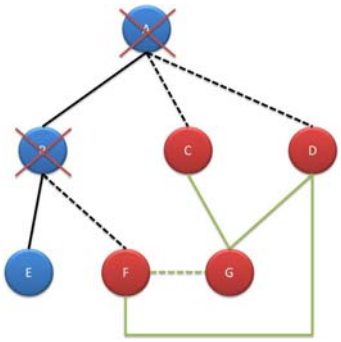

Figure 4.12: Vertex-based SSDFS example

Example 11. In Figure 4.12 blue vertices represent the non visited users (E), while red and crossed out vertices represent the visited users (A, B, F, D, G, C). Black edges are the non explored edges while green and dashed edges are the explored ones. Moreover, red vertices represent the recommended users.
Furthermore, we consider degree centrality as the chosen centrality in the vertex-centrality-vector and we call it vertex-degree-vector.
As shown in Figure 4.12, and based on this social network we assume that $n = 5$. Thus, vertex-degree-vector contains the vertices A, B, F, G and D, whose degree centrality is the highest in the social network. Moreover, calling x the item to be recommended, we also assume that $sim(A, x)$ and $sim(B, x)$ are below the threshold δ, whereas for all other vertices v in the graph $sim(v, x) > \delta$.

In this case, the main algorithm first marks all the vertices as *unvisited* and builds the vertex-degree-vector. Then, A is first considered, but as this vertex does not satisfy the similarity criterion, the call to Algorithm 6 just changes the label of A from *unvisited* to *visited*. The processing of B being similar, we now turn to F. As $sim(F, x) > \delta$, F is added to the user recommendation list, and vertices B, D and G are considered recursively. Notice that as B has already been visited, no further test is done.

It is easy to see that, in this example, E is not visited and that the output of our algorithm, the recommendation list, contains the vertices F, D, G and C. The algorithm stops searching the social network when the vertex-degree-vector becomes empty.

4.4.2.3 Vertex-edge-based heuristic

The only difference between vertex-based heuristic and vertex-edge-based heuristic is that, in vertex-edge-based heuristics, the weight θ of the edges is taken into account (see line 6 of Algorithm 7), which is not the case in the vertex-based heuristics. Example 12 details

Algorithm 7: Vertex-edge-based-visit

```
1:  if v.label = unvisited then
2:    v.label = visited
3:    if sim(v, x) > δ then
4:      Add v to the current value of user_list
5:      for all e = (v, v') and (v', v) in E do
6:        if e.weight > θ then
7:          vertex-edge-based-visit(G, v')
8:        end if
9:      end for
10:   end if
11: end if
```

Vertex-Edge-Based SSDFS as described in Algorithm 5 and Algorithm 7, and Figure 4.13 illustrates the application of this algorithm on co-purchases collaboration network.

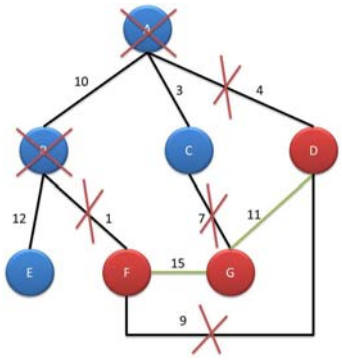

Figure 4.13: Vertex-edge-based SSDFS example

Example 12. Figure 4.13 shows a graph similar to the one in Figure 4.12, in which weights are associated to edges. In this Figure, blue vertices are the non visited users (E, C), while red and crossed out blue vertices are the visited ones (A, B, F, D, G). Black edges

Table 4.2: Vertex-based and vertex-edge-based heuristics a comparision

Heuristic	Recommended users	Non-visited users	Non-explored edges
Vertex-Based	4/7	1/7	2/9
Vertex-Edge-Based	3/7	2/7	3/9

are the non explored edges while green, dashed and crossed out edges are the explored ones. Moreover, red vertices represent the recommended users.
We assume that the weight threshold θ has been set to 10, and we consider the degree centrality in the vertex-centrality-vector as in Example 11, which contains the vertices A, B, F, G and D.
According to Algorithm 7, the vertices A and B are processed as in Example 11. Regarding the processing of F, we note that G is reached from F by Algorithm 7, whereas D is not. This is so because the weight of $e(F, G)$ is greater than 10 and the weight of $e(F, D)$ is less than 10. In this example, the users F, D and G are output of the algorithm, the recommendation list, and vertices C and E are the non visited users.

Table 4.2 compares the number of vertices and edges explored by vertex-based and vertex-edge-based heuristics.

4.4.3 Semantic-social breadth-first search SSBFS

Like depth-first search algorithm (DFS), breadth-first search algorithm (BFS) is considered as another important algorithm used for graph searching.
For a given graph G and a distinguished source vertex $v \in V(G)$, BFS explores all the graph vertices that are reachable from v, level by level, and it computes the distance from v to these vertices. At the beginning, BFS explores all the vertices at level $l = 1$ from the source vertex v, then it explores all the vertices at level $l + 1$ from this source. The algorithm repeats these steps until visiting all the graph vertices at all the levels [49].
As SSDFS, semantic-social breadth-first search SSBFS explores the co-purchases collaboration network, and it avoids exploring *all* its vertices (users) and *all* its edges (social ties), by applying heuristics depend on *semantic information* and *social information*, as follows:

- Semantic Information: represented by user profile tree $UPT(u)$, item profile Tree $IPT(x)$ and the semantic relevancy between

user preferences and item features.

- Social Information: by considering user centrality and social ties strength.

In this subsection we introduce the *semantic-social breadth-first search* algorithm SSBFS. Although, we study and analyze two versions of this algorithm: vertex-based SSBFS and vertex-edge-based SSBFS.

Algorithm thresholds SSBFS algorithm requires the same thresholds as SSDFS. These thresholds are n which is related to the size of the vertex-centrality-vector, the threshold δ which is related to user-item relevancy measure and θ which represents the edges weight threshold.

We use n and δ as thresholds for vertex-based SSBFS algorithm, and we use n, δ and θ as thresholds for vertex-edge-based SSBFS.

4.4.3.1 Vertex-based SSBFS

Algorithm input an item x having item profile tree $IPT(x)$, and the algorithm thresholds n and δ.

Algorithm output *recommendation list* containing all the recommended users.

Vertex-based SSBFS algorithm in order to avoid visiting all the vertices of G, while visiting as many relevant vertices as possible, the exploration of G starts from the vertices that have high centrality. To do so, a predefined n number of vertices with the highest centrality in the co-purchases network are computed and stored in a vector called vertex-centrality-vector (lines 3-4 of Algorithm 8). Starting form the vertices in vertex-centrality-vector, the graph is explored using BFS strategy. However, all vertices from vertex-centrality-vector whose similarity with $(IPT(x))$ of the input item x is greater than δ are first inserted into queue Q and added to the output list (lines 4-8 of Algorithm 8). Then while Q is not empty, and starting from the first node v in Q, all the successors v' of v are processed in the same manner using BFS algorithm with respect to

the user-item relevancy threshold δ.
The algorithm stops the search when the queue Q becomes empty, (from line 12 to line 19 of Algorithm 8)

Algorithm 8: Vertex-based SSBFS

Input: (i) An item x having $IPT(x)$
(ii) A positive integer n
(iii) A user-item similarity threshold δ
Output: Recommendation list $user\_list$

```
1: Q = empty queue
2: Compute the centrality of every vertex in V(G)
3: Order every v ∈ V(G) in a descending order acceding to their centrality
4: vertex-centrality-vector = the first n vertices of V(G) having the highest
   centrality
5: for all v in vertex-centrality-vector do
6:    v.label = visited
7:    if sim(v, x) > δ then
8:       Add v to the current value of user_list
9:       enqueue(Q, v)
10:   end if
11: end for
12: while Q ≠ ∅ do
13:   v = dequeue(Q)
14:   for all edge e = (v, v') in G do
15:      if v'.label = unvisited then
16:         v'.label = visited
17:      end if
18:   end for
19: end while
```

4.4.3.2 Vertex-edge-based SSBFS

Algorithm input an item x having item profile tree $IPT(x)$, and the algorithm thresholds n, δ and θ.

Algorithm output *recommendation list* containing all the recommended users.

Vertex-edge-based SSBSF algorithm details the exploration of the co-purchases network starts from the users having high centrality values. so, n users with highest centrality are stored in the vertex-centrality-vector (lines 2-4 of Algorithm 9). Starting form the users

in vertex-centrality-vector, the graph is explored using BFS algorithm.
As in vertex-based SSBFS, all vertices from the vertex-centrality-vector whose similarity with the input item x is greater than δ are firstly inserted into the queue Q and added to the output list (lines 4-8 of Algorithm 9). Then while Q is not empty, and starting from the first vertex v in Q, all the successors v' of v whose connection with v has a weight greater than θ are processed as follows: if v' is unvisited and if $sim(v', x) > \delta$, then v' is added to the output list and v' is inserted into Q (lines 14-19 of Algorithm 9).
The algorithm stops searching the graph when the queue Q becomes empty.

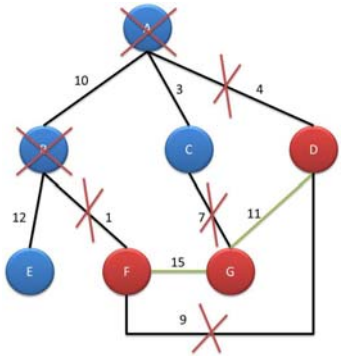

Figure 4.14: Vertex-edge-based SSBFS example

Example 13. Figure 4.14 illustrates an example of vertex-edge-based SSBFS with degree centrality (vertex-centrality(degree)-vector).
In this example we assume that edge weight threshold $\theta = 10$, and the size of vertex-degree-vector $n = 4$ containing the vertices A, B, F, and G, which are having the highest *degree centrality*. Moreover, we suppose that $sim(A, x) < \delta$ and $sim(B, x) < \delta$ while $sim(v, x) > \delta$ for all the other vertices in the co-purchases network.
SSBFS adds the vertices of vertex-degree-vector to the queue Q and to the recommendation list, except the vertices A and B, because $sim(A, x) < \delta$ and $sim(B, x) < \delta$, and it starts the breadth-first search from the vertex F in order to visit the vertices G and D; according to SSBFS the vertex G is reachable because $w(F, G) > 10$ and the vertex D is not reachable because $w(F, D) < 10$, so SSBFS

Algorithm 9: Vertex-edge-based SSBFS

Input: (i) An item x having $IPT(x)$
(ii) A positive integer n
(iii) A user-item similarity threshold δ
(iv) An edge weight threshold θ
Output: Recommendation list $user\_list$

```
1: Q = empty queue
2: Compute the centrality of every vertex in V(G)
3: Order all the vertices of V(G) in a descending order acceding to their
   centrality
4: vertex-centrality-vector = the first n vertices v ∈ V(G) having the highest
   centrality
5: for all v in vertex-centrality-vector do
6:    v.label = visited
7:    if sim(v, x) > δ then
8:       Add v to the current value of user_list
9:       enqueue(Q, v)
10:   end if
11: end for
12: while Q ≠ ∅ do
13:   v = dequeue(Q)
14:   for all edge e = (v, v') in G do
15:      if e.weight > θ and v'.label = unvisited then
16:         v'.label = visited
17:         if sim(v', x) > δ then
18:            Add v' to the current value of user_list
19:            enqueue(Q, v')
20:         end if
21:      end if
22:   end for
23: end while
```

restart the breadth-first search from G. The vertex D is reachable from G so it is added to the recommendation list and to the queue, while the vertex C is not. SSBFS resumes the graph searching from the queue starting from the vertex D, to find that there are no more vertices to be visited (vertex A is not reachable from D, and G has already been visited) so the algorithm stops the search and the output of the algorithm is represented by the recommendation list which contains the vertices F, G and D.

4.4.4 Users centralities in the co-purchases network

Previously, in this section, we discussed vertex-based and vertex-edge-based SSDFS algorithms; also we discussed vertex-based and vertex-edge-based SSBFS. In these algorithms we considered the general case of *centrality*, which means we did not determine the type of user centrality. For instance, in the algorithms Algorithm 5, Algorithm 8 and Algorithm 9 we considered user's centrality as a general centrality, by referencing the *vertex-centrality-vector* without specifying which centrality the algorithm tests or uses.

In this subsection, we specify the centralities used in vertex-centrality-vector of each algorithm, Algorithm 5, Algorithm 8 and Algorithm 9. Moreover, we give some examples about these algorithms, also we propose a new algorithm, a hybrid algorithm that combines *degree centrality* with *betweenness centrality* in one algorithm.

Figure 4.16, illustrates a collaboration social network; in which we suppose $n = 3$ the size of the vertex-centrality-vector, and we suppose that $sim(D, x) < \delta$ and $sim(F, x) < \delta$. Furthermore, we set the edge weight threshold $\theta = 10$. Moreover, we study and analyze the attitude of our proposed algorithms according to each centrality: degree centrality, closeness centrality, betweenness centrality and the hybrid degree-betweenness, on all of our proposed algorithms.

Table 4.3 contains the degree, closeness and betweenness centrality values of all the users (vertices) of the co-purchases collaboration network given by Figures 4.15.

Table 4.3: Co-purchases network users centralities

User	Degree	Closeness	Betweenness
A	4	1.667	13.5
B	3	2.111	8
C	2	2.222	0.5
D	3	1.778	10
E	1	3	0
F	3	2	3.5
G	3	1.889	5.5
H	3	2	5
I	3	2.222	8
J	1	3.111	0

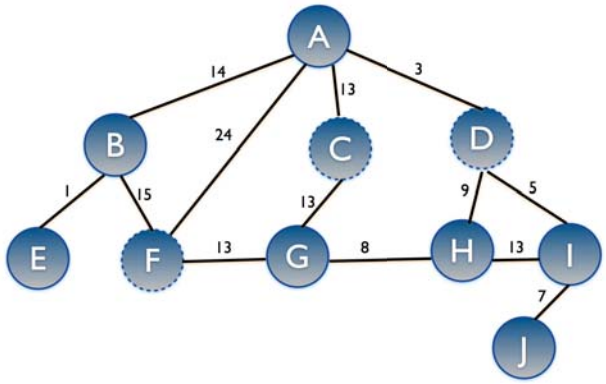

Figure 4.15: Co-purchases network: example

4.4.4.1 Degree centrality

We replace the vertex-centrality-vector in the algorithms, Algorithm 5 (lines 5-6-7), Algorithm 8 (lines 2-4-4) and Algorithm 9 (lines 2-4-5), by the vertex-degree-vector. In this case we compute degree centrality of all the users in the co-purchases network and we add n users, having the heights degree centrality, to the vertex-degree-vector.

Example 14 (Degree Centrality). We apply SSDFS algorithm on the social network represented by Figure 4.16.

In this example we suppose that the size of vertex-degree-vector is $n = 3$ so this vector contains the vertices: A, B and G, these vertices have the largest degree values in the network.
SSDFS starts exploring the network from the vertex A and it adds A to the recommendation list. Starting form A, the algorithm explores all the reachable paths from A. Reachable paths are determined according to two values: the value of edge weight, edges should have weight $w > \theta$, and the value of the similarity between the input item and the vertex (representing user profile).
In this example, the only reachable path from A is: $A \rightarrow B \rightarrow F$ in which B is recommended and no edges are explored passing by F because $sim(F, x) < \delta$. After exploring all the reachable paths from A, SSDFS stops searching the network and it reprises the search from B. As the vertex B has already been visited the algorithm starts another exploration form G. SSDFS adds G to the recommendation list and it explores all the reachable paths form G. At this end, SSDFS terminates exploring the network because there are no reachable paths from G and all the vertices in vertex-degree-vector have been visited.
The output of the algorithm is the recommendation list containing the vertices: A, B, and G. Moreover, the algorithm explores a small part of the graph and there are some unvisited vertices as: D, I, H, J and E.

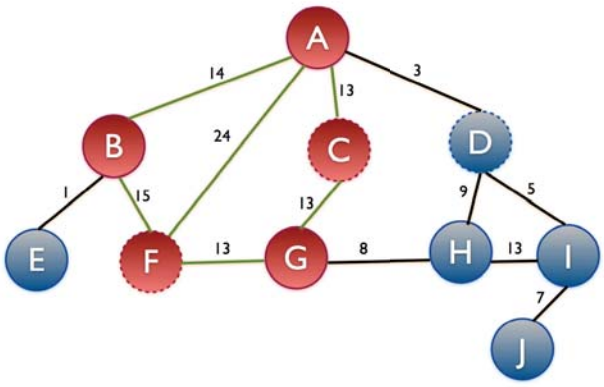

Figure 4.16: SSDFS with degree centrality

4.4.4.2 Closeness centrality

We replace the vertex-centrality-vector in the algorithms, Algorithm 5 (lines 5-6-7), Algorithm 8 (lines 2-4-4) and Algorithm 9 (lines 2-4-5), by the vertex-closeness-vector. In this case we compute the closeness centrality of all the users in the co-purchases network and we add n users, having the heights closeness centrality, to the vertex-closeness-vector.

Example 15 (Closeness Centrality). In Figure 4.17, the vertex-closeness-vector contains the vertices: J, E and I.
We apply SSDFS algorithm on the social network represented by Figure 4.17.
In this example we suppose that the size of vertex-degree-vector is $n = 3$ and this vector contains the vertices: J, E and I, these vertices have the largest *closeness centrality* values in the network.
SSDFS starts exploring the network from the vertex J and it adds J to the recommendation list. Starting form J, the algorithm explores all the reachable paths from J. Reachable paths are determined according to two values: the value of edge weight, edges should have weight $w > \theta$, and the value of the similarity between the input item and the vertex (representing user profile).
In this example, there is no reachable path from J, so SSDFS stops and it reprises the search from E and it adds E to the recommendation list. As J, there is no reachable paths from E and the algorithm stops the search to reprise another research from I. Starting from I the only possible path to be explored is the one connecting I with H. So the algorithm adds I and H to the recommendation list and it terminates the search. The output of the algorithm is the recommendation list containing the vertices: J, E, I and H. Moreover, the algorithm explores a small part of the graph and there are some unvisited vertices as: A, B, C, D, F and G.

4.4.4.3 Betweenness centerality

We replace the vertex-centrality-vector in the algorithms, Algorithm 5 (lines 5-6-7), Algorithm 8 (lines 2-4-4) and Algorithm 9 (lines 2-4-5), by the vertex-betweenness-vector. In this case we compute the betweenness centrality of all the users in the co-purchases network and we add n users, having the heights betweenness centrality, to the vertex-betweenness-vector.

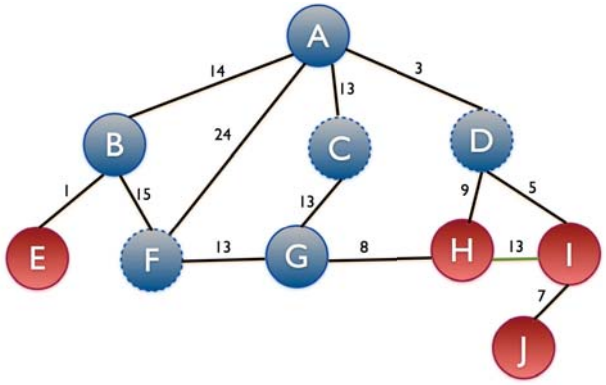

Figure 4.17: SSDFS with closeness centrality

Example 16 (Betweenness Centrality). We apply SSDFS algorithm on the social network represented by Figure 4.19.
In this example we suppose that the size of vertex-betweenness-vector is $n = 3$ and this vector contains the vertices: A, I and B, these vertices have the largest degree values in the network.
SSDFS starts exploring the network from the vertex A and it adds A to the recommendation list. Starting form A, the algorithm explores all the reachable paths from A. Reachable paths are determined according to two values: the value of edge weight, edges should have weight $w > \theta$, and the value of the similarity between the input item and the vertex (representing user profile).
In this example, the only reachable path from A is: $A \rightarrow B \rightarrow F$ in which B is recommended and no edges are explored passing by F because $sim(F, x) < \delta$. After the vertex A, the algorithm starts another exploration form I. SSDFS adds I to the recommendation list and it explores all the reachable paths form I. The only reachable path is to the vertex H, which is added to the recommendation list. As the vertex B has been visited, SSDFS terminates exploring the network, and the output of the algorithm is the recommendation list containing the vertices: A, B, I, and H.

4.4.4.4 Hybrid centrality Degree-Betweenness

In this algorithm we propose to combine degree centrality with betweenness centrality of users in one algorithm in order to benefit

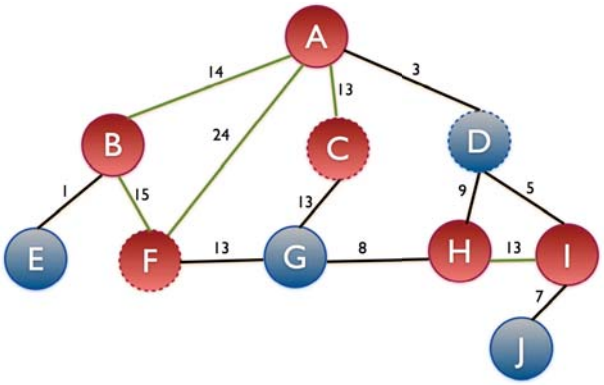

Figure 4.18: SSDFS with betweenness centrality

from the characteristics of both centralities. This algorithm is applied on vertex-edge-based SSDFS algorithm, Algorithm 10, and on vertex-edge-based SSBFS algorithm, Algorithm 11.

For vertex-edge-based SSDFS, the algorithm orders the graph vertices, in a descending order, according to their degree centrality in the vertex-degree-vector in the main algorithm Algorithm 5, considering the degree centrality case. Then it adds to the semantic heuristic another condition related to betweenness centrality of the discovered vertex, in Algorithm 10 (line 6). If the vertex satisfies the betweenness centrality threshold σ, then the algorithm continues the graph exploration. If not then the algorithm resumes the search form vertices in the vertex-degree-vector. This algorithm is described in Algorithm 10, which is the visit procedure of the main algorithm Algorithm 5.

In the same manner, we apply vertex-edge-based SSBFS with hybrid centralities. This algorithm uses vertex-degree-vector and it adds the betweenness centrality of the vertex as a condition to decide if the algorithm will continue the network exploration or not. Hybrid vertex-edge-based SSBFS is explained by Algorithm 11.

Example 17 (Degree-Betweenness Centrality). In this example we suppose that the size of vertex-degree-vector is $n = 3$ and this vector contains the vertices: A, B and G, these vertices have the largest

Algorithm 10: Hybrid-vertex-edge-based-Visit

```
1: if v.label = unvisited then
2:    v.label = visited
3:    if sim(v, x) > δ then
4:       Add v to the current value of user_list
5:       for all e = (v, v') and (v', v) in E do
6:          if e.weight > θ and v'.betweenness > σ then
7:             Hybrid-Vertex-Edge-Based-Visit(G, v')
8:          end if
9:       end for
10:   end if
11: end if
```

Algorithm 11: Hybrid vertex-edge-based SSBFS

Input: (i) An item x and its Item Preference Tree $IPT(x)$
(ii) A positive integer n
(iii) A user-item similarity threshold δ
(iv) An edge weight threshold θ
(v) Vertex Betweenness threshold σ
Output: List of recommended users *user_list*

```
1: Q = empty queue
2: Compute the degree centrality of every vertex in V
3: vertex-degree-vector = vertices in a descending order according to degree
   centrality
4: for all v in vertex-degree-vector do
5:    if sim(v, x) > δ and i ≤ n then
6:       Add v to the current value of user_list
7:       v.label = visited
8:       enqueue(Q, v)
9:       i = i + 1
10:   end if
11: end for
12: while Q ≠ ∅ do
13:   v = dequeue(Q)
14:   for all edge e = (v, v') in G do
15:      if v'.label = unvisited and e.weight > θ then
16:         v'.label = visited
17:         if sim(v', x) > δ and v'.betweenness > σ then
18:            Add v' to the current value of user_list
19:            enqueue(Q, v')
20:         end if
21:      end if
22:   end for
23: end while
```

Table 4.4: Social network analysis measure as parameters

Centrality	Visited users	Visited edges	Recommended users
Degree	5/10	6/13	3
Closeness	4/10	1/13	4
Betweenness	6/10	4/13	4
Degree-betweenness	2/10	3/13	2

degree values in the network. We suppose that the vertex betweenness centrality threshold $\sigma = 6$, the edge weight threshold $\theta = 10$ and $sim(D, x) < \delta$ and $sim(F, x) < \delta$.
SSDFS starts the search from A. The reachable vertices from A is B. The other vertices are not reachable and not processed by SSDFS.The output of SSDFS using degree-betweenness centrality is the recommendation list containing the vertices A and B.

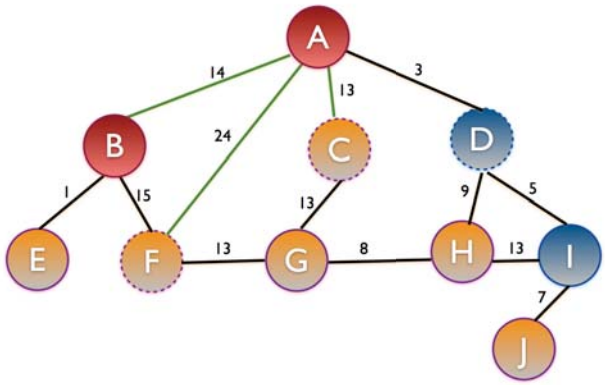

Figure 4.19: SSDFS with degree-betweenness centrality

Table 4.4 compares how many vertices and edges the different algorithms with different centrality parameters explore the co-purchases collaboration network.

4.5 Conclusion

In this chapter we introduced our proposed approach: the semantic-social recommender system. This new approach uses two types of information, semantic information and social information.

In semantic information we proposed to represent user preferences and item features in a form of semantic taxonomy tree. Furthermore, we proposed a new user-item relevancy measure to find out the semantic relevance between users and items in the system.

On the other hand, according to social information we proposed to represent users via a collaboration social network extracted from user-item bipartite graph; and to use information related to users centrality and the strength of their social ties.

For that, we proposed a new approach which integrates semantic information and social information with a graph searching algorithm, such as depth-first search DFS algorithm and breadth-first search BFS algorithm. We called our new proposed algorithms semantic-social depth-first search SSDFS and semantic-social breadth-first search SSBFS.

Both of SSDFS and SSBFS have two versions: vertex-based and vertex-edge-based. In vertex-based we only consider user centrality and user-item semantic relevancy; while in vertex-edge-based we consider user centrality, edge weights and user-item semantic relevancy.

Moreover, we applied our proposed semantic-social approach with several types of centralities: degree centrality, closeness centrality, betweenness centrality and a hybrid model that combines degree centrality with betweenness centrality.

In the next chapter we will provide a detailed experimental study about the different types of our proposed algorithms, and the different datasets and algorithms that are used to test and validate our semantic-social recommendation approach.

Chapter 5

Experimental Results

5.1 Introduction

In this chapter we present in details our experiments and our obtained results. In fact, we apply our proposed algorithms on two real datasets, the first one is extracted from Amazon.com and the second one is extracted from MovieLens. Then we compare our algorithms with two of the classical recommendation algorithms: item-based collaborative filtering and hybrid algorithm. Finally, in order to compare and validate our algorithms we use two types of measures: accuracy measures, including precision, recall and f-measure; and performance measures including graph coverage and execution time (recommendation time).

Our tests are organized as follows:

1. A comparative study of semantic-social depth-first search SSDFS with two heuristics: vertex-based and vertex-edge-based, considering user-item semantic relevancy, user degree centrality and edges weight in the case of vertex-edge-based heuristic.

2. A comparative study of semantic-social breadth-first search SSBFS with two heuristics: vertex-based and vertex-edge-based, considering user-item semantic relevancy, user degree centrality and edges weight in the case of vertex-edge-based heuristic.

3. A comparative study of semantic-social depth-first search SSDFS and semantic-social breadth-first search SSBFS.

4. A comparative study of semantic-social depth-first search SSDFS and semantic-social breadth-first search SSBFS, with different types of user centralities (closeness, betweenness and degree-betweenness).

This chapter is organized as follows: in the second section we represent the datasets we use, the third section details the evaluation algorithms, the fourth section explains the evaluation metrics and the fifth section introduce the algorithm parameters and we expose our results in the sixth section to finally conclude in the seventh section.

5.2 Datasets

We use two datasets from *real world*. (a) *MovieLens*[1] the well known dataset in the world of movie recommendation, and (b) *Amazon.com*[2] the well known dataset in the world of e-commerce.

MovieLens dataset has been collected by GroupLens research lab at the university of Minnesota. This dataset describes the users and their preferred movies, and it has been used to test and validate several types of collaborative-filtering recommendation algorithms since 1997.

Amazon is an international company for e-commerce, that have been created in 1994. Datasets from Amazon.com contain several types of information about users and their purchases.

In our experiments, we use one dataset from MovieLens and four datasets from Amazon.com. Table 5.1 shows the features of these datasets, mainly the number of users and the number of items.

Our motivation of using several types of datasets with different sizes, is to study the stability and the scalability of our algorithms and to analyze their ability to be used on different types of datasets with different sizes.

This section discusses the datasets that we used in our experiments. Moreover, we show how to shift from user-item classical dataset to a social collaboration network connecting users.

[1] http://movielens.org
[2] http://www.amazon.fr

Table 5.1: Datasets features

Social Network	Number of users	Number of items
MovieLens	999	1682
Amazon(1)	1,253	500
Amazon(2)	2,802	1000
Amazon(3)	38,982	15,000
Amazon(4)	51,220	20,000

Table 5.2: Social Networks Features

Social Network	$\|V\|$	$\|E\|$	Dia	D	CC
MovieLens	999	46,689	4	0.094	0.436
Amazon(1)	1,253	36,802	4	0.047	0.604
Amazon(2)	2,802	117,924	4	0.03	0.74
Amazon(3)	38,982	5,005,398	5	0.139	0.588
Amazon(4)	51,220	6,893,029	17	0.002	0.878

Table 5.3: Social Networks Centralities

Social Network	(av) Degree	(av) Closeness	(av) Betweenness
MovieLens	93.472	0.608	295.3
Amazon(1)	58.7	1.623	411.38
Amazon(2)	84.171	1.635	93667
Amazon(3)	187	4.531	13605
Amazon(4)	269	6.56	18276

5.2.1 MovieLens Dataset

MovieLens dataset has been collected by GroupLens research project at the University of Minnesota [1]. MovieLens has several types of datasets, the one we have chosen includes information about 100,000 ratings, these ratings have interval of $[1, 5]$, from about 1000 users on 1682 movies. In this dataset, each user has rated 20 movies at least.

Users have extra information about their age, gender, occupation and location. Moreover, Movies are described via their genres, these genres are defined as follows: Action, Adventure, Animation, Children's, Comedy, Crime, Documentary, Drama, Fantasy, Film-Noir, Horror, Musical, Mystery, Romance, science- fiction (Sci-Fi), Thriller, War and Western.

We handled MovieLens dataset by: firstly building user-movie bi-

[1] http://grouplens.org/datasets/movielens

partite graph, secondly extracting users co-rated network and finally using a semantic taxonomy tree describing the hierarchical relations among movies genres. These steps are described as follows:

- User-movie bipartite graph G is a graph in which the set of vertices $V(G)$ is composed of two disjoint sets, the set of users U and the set of movies M. Edges connect a user $u \in U$ with a movie $m \in M$ if the user u has already rated m.

- Users co-rated network (co-rated movies), is the users weighted one-mode projection network extracted from user-movie bipartite graph. In our approach, we submit the recommendation query on this network. Figure 5.1 shows this network, as we plotted it using Gephi software, and Table 5.2 gives some statistics about this network as follows: number of network vertices $|V|$, number of network edges $|E|$, network diameter Dia, network density D and clustering coefficient CC. Furthermore, Figure 5.2 shows the degree distribution of this network.

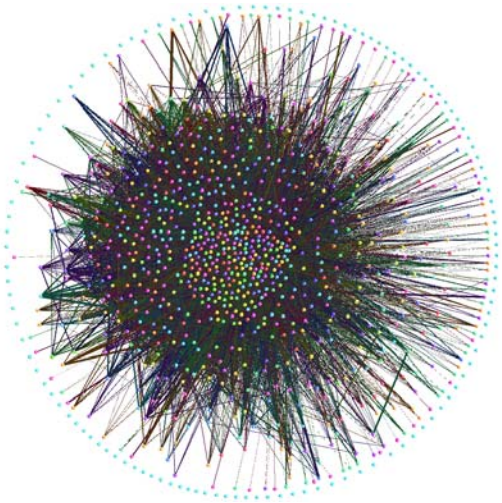

Figure 5.1: MovieLens collaboration network (co-rated network) extracted form user-movie bipartite graph.

- Semantic taxonomy Tree (STT) is defined in Definition 15 of Chapter 4. In our approach we use the taxonomy of the genres of the movies as proposed in [176]. In this taxonomy, the gen-

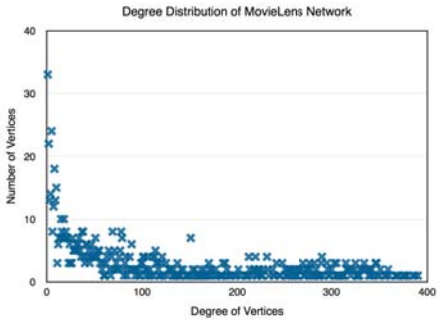

Figure 5.2: Degree distribution of the vertices of MovieLens collaboration network.

res of movies are organized in a hierarchy starting from level 0, which contains the most general concept describing movie genres, to the level 4 (the deepest level in STT) containing the most specific concepts describing movies genres. This taxonomy is given by Figure 5.3. Moreover, as defined in Chapter 4, item profile tree of a movie m $IPT(m)$ is the subtree from the semantic taxonomy tree starting from the concept 'Thing' in the taxonomy and ending by the concept describing the genre of the movie m. Furthermore, user profile tree $UPT(u)$ of user u is defined as the union of item profile trees of all the user's rated movies.

In our approach we apply the semantic-social recommendation algorithms on users co-purchases network showed in Figure 5.1, and we consider the semantic information of user profile tree $UPT(u)$ and item profile tree $IPT(m)$.
Table 5.3 shows the average degree, closeness and betweenness centralities in users co-rated network.

5.2.2 Amazon Dataset

We use *Amazon* data as another real dataset to test and validate our approach. Amazon dataset contains detailed information about 7 million users and more than 500 thousand items. This data was collected, in the summer of 2006, by crawling Amazon website, and it contains metadata and review information about different prod-

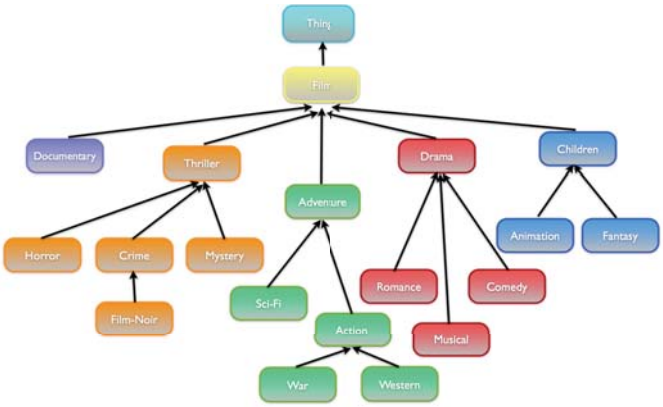

Figure 5.3: Movies Semantic Taxonomy Tree

ucts such as books, music CDs, DVDs and VHS video tapes [107]. Amazon dataset is available on the web page of Stanford University[3].

In our approach, we handle Amazon dataset in the same way as MovieLens dataset, by establishing the user-item bipartite graph from Amazon classical dataset, then we extract users co-purchases network from user-item bipartite graph, also we use the semantic information attached to users and items in the dataset. These steps are described as follows:

- User-item bipartite graph G, which is extracted from amazon dataset. The vertices of this graph are divided into two disjoint sets, the set of users U and the set of items I. Edges connect a user $u \in U$ with an item $i \in I$, if the user u has purchased the item i.

- Users co-purchases collaboration network, is the users weighted one-mode projection extracted from user-item bipartite graph. This network is shown in Figure 5.4, and its features are given in Table 5.2. In fact, we have extracted several users co-purchases networks named Amazon(1), Amazon(2), Amazon(3) and Amazon(4). The different features of these networks are indicated in Table 5.2, these features are the number of network vertices $|V|$, the number of network edges $|E|$, network diameter Dia,

[3]http://snap.stanford.edu/data/amazon-meta.html

network density D and clustering coefficient CC. Moreover, the degree distribution of each one of these datasets is shown in Figure 5.5 [4].

- In Amazon dataset, items have a profile tree describing their features; and users have a history of their previous purchases. Thus, we can extract item profile tree IPT and user profile tree UPT, as defined in Definition 16 and Definition 17 of Chapter 4.

Moreover, Table 5.3 shows the average degree, closeness and betweenness centralities in users co-purchases network.

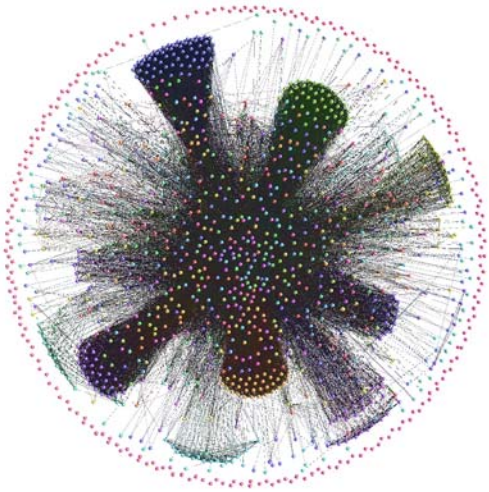

Figure 5.4: Amazon(1) users co-purchases network.

5.3 Evaluation Algorithms

In order to evaluate and validate our approach, we propose to compare it with two classical algorithms from the literature of recommender systems [5, 38, 172]. These algorithms are the item-based collaborative filtering algorithm and the hybrid recommendation algorithm.

[4] We generated this figure using Gephi software.

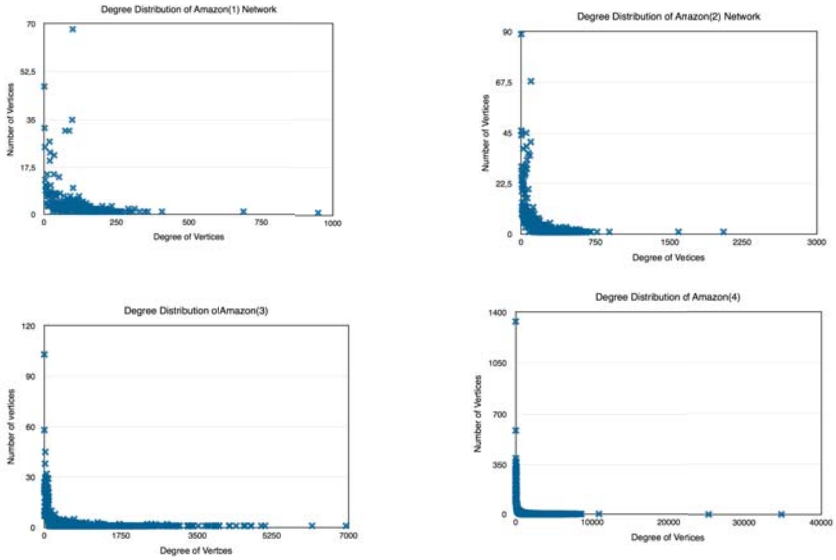

Figure 5.5: Degree Distribution of Amazon(1), Amazon(2), Amazon(3) and Amazon(4) co-purchases network.

5.3.1 Item-based collaborative filtering algorithm

Item-based Collaborative Filtering algorithm explores the similarity between the items in the system with regards to the users ratings [172, 171, 113].

In any item-based collaborative filtering system and for the input item i the algorithm finds all the items that are similar to i; and it recommends it to the users who have bought or rated these similar items. Similarity between items is computed using several measures such as cosine similarity, pearson correlation and adjusted cosine similarity [172, 171].

In our experiments we propose to use cosine similarity to compute the similarity between items. Cosine similarity is denoted by Formula 5.1:

$$Cosine(u_i, u_j) = \frac{\sum_{k=1}^{n} r_{u_{i,k}} . r_{u_{j,k}}}{\sqrt{\sum_{k=1}^{n} r_{u_{i,k}}^2 . \sum_{k=1}^{n} r_{u_{j,k}}^2}} \tag{5.1}$$

where $r_{u_{i,k}}$ is the rating of the user u_i on the item i_k, $r_{u_{j,k}}$ is the rating of the user u_j on the same item i_k and n is the number of items in the system.

We implemented the item-based collaborative filtering algorithm on *MovieLens* and *Amazon* datasets. The output of this algorithm is a recommendation list containing all the relevant users to the recommended item.

5.3.2 Hybrid recommendation algorithm

Hybrid recommender systems have several types [38, 39]. In our experiments we use the weighted hybrid recommender system [20]. So, we apply the item-based collaborative filtering then we apply the content-based algorithm. In this hybrid recommendation the Item-based recommendation uses the cosine similarity as described in 5.3.1, while content-based recommendation uses our proposed user-item semantic relevancy measure which is described as follows:

$$\sigma(P_1, P_2) = \frac{1}{\mu} \left(\sum_{(\tau,\lambda) \in P_1 \cap P_2} \lambda \right)$$

where $\mu = \min \left(\sum_{(\tau,\lambda) \in P_1} \lambda, \sum_{(\tau,\lambda) \in P_2} \lambda \right)$, τ is the common concepts of the profiles P_1 and P_2 and λ is the level of the common concept. Using σ, we define the relevancy between a user u and an item x as the similarity between their profile trees. The user-item relevancy measure is given as follows:

$$sim(u, x) = \sigma(UPT(u), IPT(x))$$

The output of the algorithm is a recommendation list containing all the relevant users found by the recommendation algorithm.

5.4 Evaluation Metrics

In order to evaluate our proposed algorithms, we use three types of metrics (a) accuracy metrics including precision, recall, and F-measure, (b) graph coverage (c) and execution time.

5.4.1 Accuracy

We use *precision*, *recall* and *F-measure* to evaluate the accuracy of our proposed recommendation algorithms [81]. We apply these measures on the algorithms outputs the "recommendation list". The output of each recommendation algorithm is a recommendation list containing all the recommended users. In fact, the recommendation list includes two types of users the positive relevant users and the relevant users, as follows:

1. Positive relevant users are the users from the dataset who have already bought or rated the input item and who have been recommended by the recommendation algorithm. Referring to the usual way of assessing the accuracy of algorithms, positive relevant users can seen as the *true positive* part of the outputs of the algorithms. This is why, in what follows, we denote by TP the number of positive relevant users returned by the algorithms in our experiments.

2. Relevant users are the users who have been found by the recommendation algorithm, and who have not rated or bought the item. In fact, these users represent the algorithm's suggested users, who have relevant interests to the input item (based on the item-user relevancy measure). Relevant users can seen as the *true negative* part of the outputs of the algorithms. This is why, in what follows, we denote by TN the number of relevant users returned by the algorithms in our experiments.

Precision is defined as the probability that a user, to whom we propose the recommendation, is positive relevant; and recall is defined as the probability that a positive relevant user is recommended [81].

The definitions of these two measures is given as follows:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + TN}$$

where TP (true positive) is the number of the positive relevant users who have been recommended, TN (true negative) is the number of the positive relevant users who have not been recommended, and

FP (false positive) is the number of the relevant users, who have been recommended.
Moreover, we use F-measure which combines precision and recall in one formula, as follows:

$$F - Measure = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

The validation algorithm finds:

1. The number of the common users between the recommendation list and the list of all the other users who have bought or rated the items in the whole system, this number represents the true positive TP.
2. The size of the recommendation list which is equal to $TP+FP$
3. The number of all the users who have already bought or rated the items from the whole dataset, this number is equal to $TP+TN$

Then we compute and compare the precision, recall and f-measure of the all recommendation algorithms, as previously described in this section.

5.4.2 Data coverage

Data coverage is defined as the percentage of the *dataset* or the *graph vertices* (users) who have been visited by the recommendation algorithms. The percentage of graph visited vertices is very important, especially in the case of big graphs, because visiting all the graph vertices influences the algorithm performance.

5.4.3 Execution time

Execution time is defined as the time each algorithm takes to search the graph or the dataset, in order to recommend an item to users.

5.5 Algorithms: Thresholds

In this section we study the impact of the different values of the algorithms thresholds on the results. For that, we investigate the

different values of the semantic threshold δ, the edge weight threshold θ and the size of the centrality vector n and their effect on the results. Actually, we study the case of Amazon dataset and we are concerned by the values that give our algorithms the best accuracy and performance results.

5.5.1 F-Measure and Graph coverage of algorithms thresholds

We achieved several tests on different values of each threshold in the system. Then we selected the best values of these thresholds that give good accuracy and performance results.

- Semantic threshold δ: used in the user-item semantic relevancy measure. Figure 5.6 shows the best values of δ giving the best F-measure and graph coverage for SSDFS algorithm. As well, Figure 5.7 shows the best values for SSBFS algorithm. According to these figures we choose $\delta = 3$

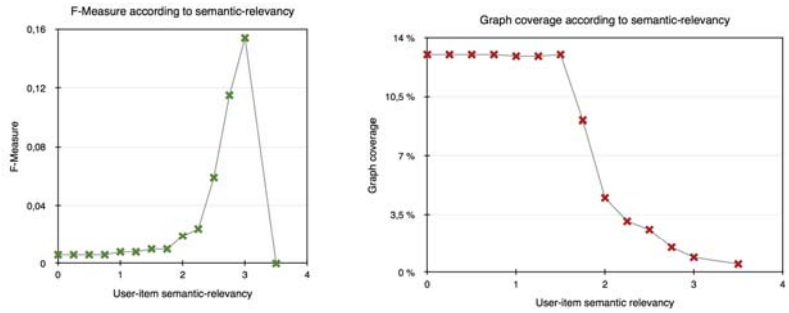

Figure 5.6: User-item semantic relevancy threshold δ effects on F-Measure and graph coverage for SSDFS algorithm.

- Edge weight threshold θ: determines the edges weights in the network. Figure 5.8 shows the best values of θ giving better F-measure and graph coverage for SSDFS algorithm. Likewise, Figure 5.9 determines the same for SSBFS algorithm. According to these figures we choose $\theta = 30$.
- Centrality-vector size threshold n that determines the maximum number of users in the vertex-centrality-vector. Figure 5.10 shows the best value of centrality-vector size giving

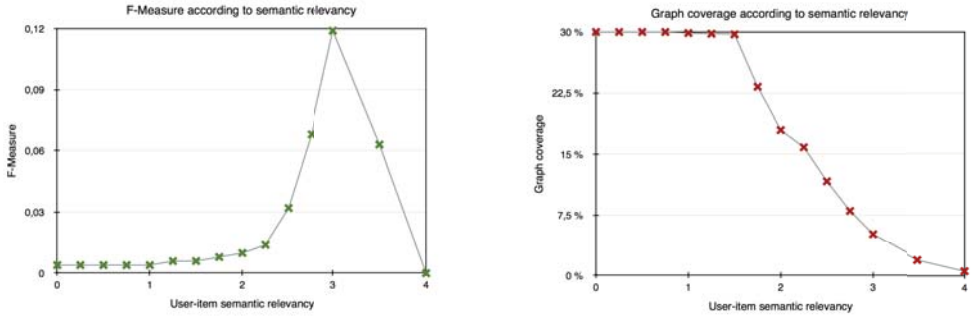

Figure 5.7: User-item semantic relevancy threshold δ effects on F-Measure and Graph coverage for SSBFS algorithm.

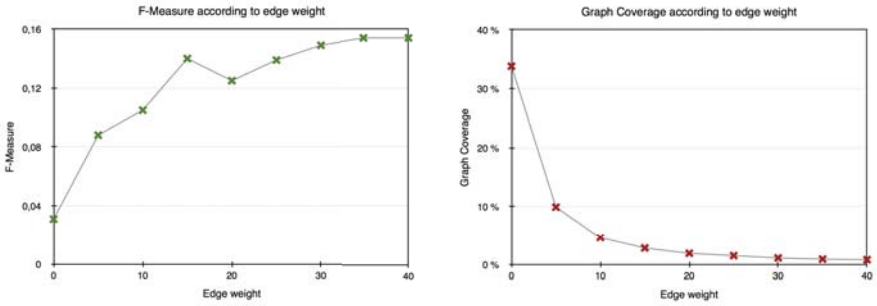

Figure 5.8: Edge weight threshold θ effects on F-Measure and Graph coverage for SSDFS algorithm.

better F-measure and graph coverage for SSDFS algorithm. Also, Figure 5.11 shows the best values for SSBFS algorithm. According to these figures we choose $n = 200$.

Summarizing the previous discussion we fixed the semantic threshold $\delta = 3$, the edge weight threshold $\theta = 30$ and the centrality vector size $n = 200$ for the both algorithms SSDFS and SSBFS.

5.6 Recommendation queries

The recommendation query is an item x to be recommended to the users who are connected via the collaboration network. The item x

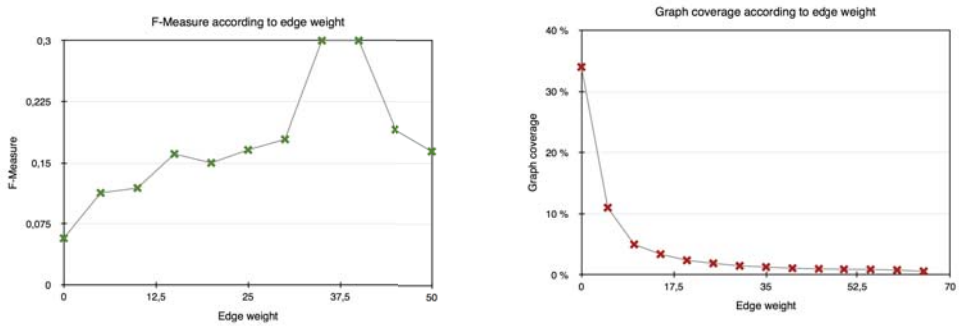

Figure 5.9: Edge weight threshold θ effects on F-Measure and Graph coverage for SSBFS algorithm.

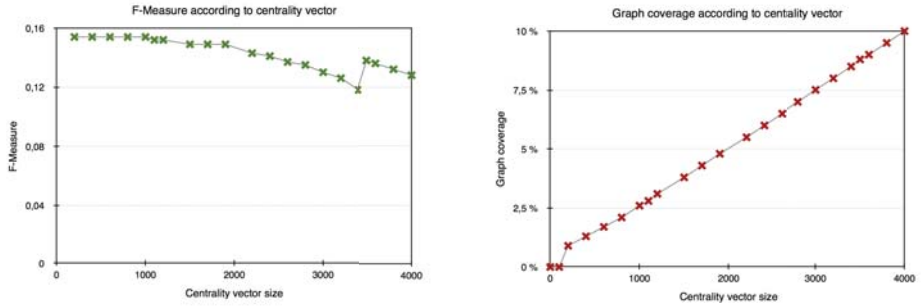

Figure 5.10: Centrality-vector size n effects on F-Measure and Graph coverage for SSDFS algorithm.

has an item profile tree $IPT(x)$ and each user u in the collaboration network has a user profile tree $UPT(u)$.

For MovieLens dataset, we have randomly selected and submitted 100 queries. Moreover, we have selected the recommendation queries as items that have been rated by more than 20 users in the dataset. Each one of the 100 items is submitted to the SSDFS, SSBFS, collaborative filtering, and hybrid algorithms. Our results represent the average precision, average recall and average f-measure of the 100 queries.

For Amazon dataset, we have randomly selected and submit-

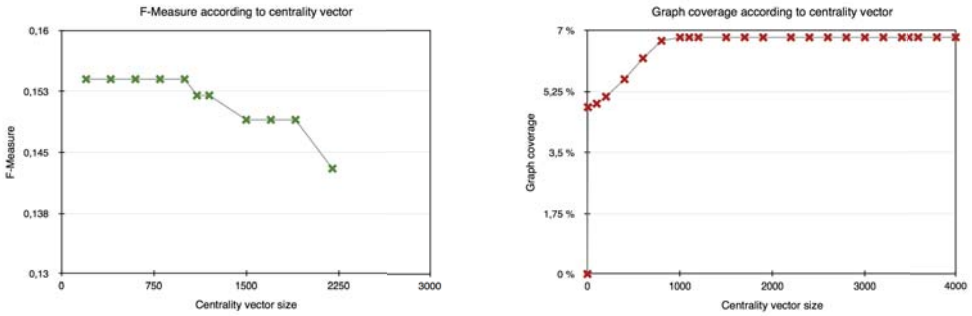

Figure 5.11: Centrality-vector size n effects on F-Measure and Graph coverage for SSBFS algorithm.

ted 54 queries. Furthermore, we have selected the recommendation queries as items that have been bought by more than 20 users in the dataset. Each one of the 54 items is submitted to the SSDFS, SSBFS, collaborative filtering and hybrid algorithms. Our results represent the average precision, average recall and average f-measure of the 54 queries.

5.7 Results and Validation

We developed our algorithms using java 6, we also used JUNG (Java Universal Network/Graph), NetworkX (Python), and Gephi. These frameworks are used for social network analysis and visualization. Moreover, we performed our experiments on an Intel(R) Xeon(R) CPU E5520 2.27GHz with 12 Giga of RAM, using Debian Linux as operating system.

In this section Table 5.4 and Table 5.5 show how our tests and validation processes are organized. In fact, we use three different datasets Amazon(3), Amazon(4) and MovieLens, these datasets are described in Table 5.2.

Our motivation is to study the accuracy and the performance of the semantic-social recommender system using different types of datasets with different sizes. In doing so we study the stability and the scalability of our approach, and we show the ability of our algorithms to be applied on several datasets.

Table 5.4: Experiments and their validation

Experiments	Algorithms	Validation	Dataset
SSDFS	Vertex-based Vertex-Edge-based	CF & Hybrid Accuracy & Performance	Amazon(3)
SSBFS	Vertex-based Vertex-Edge-based	CF & Hybrid Accuracy & Performance	Amazon(3)
SSDFS (vs) SSBFS	Vertex-based Vertex-Edge-based	CF & Hybrid Accuracy & Performance	Amazon(3)
SSDFS & SSBFS Centrality variation	Vertex-Edge-based	Accuracy & Performance	Amazon(4) & MovieLens

Table 5.5: Experimented algorithms details

Experiments	Algorithms	Heuristic
SSDFS	Vertex-based	Semantic relevancy Degree centrality
	Vertex-Edge-based	Semantic relevancy Degree centrality Edge weight
SSBFS	Vertex-based	Semantic relevancy Degree centrality
	Vertex-Edge-based	Semantic relevancy Degree centrality Edge weight
SSDFS (vs) SSBFS	Vertex-based	Semantic relevancy Degree centrality
	Vertex-Edge-based	Semantic relevancy Degree centrality Edge weight
SSDFS & SSBFS Centrality variation	Vertex-Edge-based	Semantic relevancy closeness centrality Edge weight
		Semantic relevancy Betweenness centrality Edge weight
		Semantic relevancy Betweenness-Degree centrality Edge weight

Our experiments include the following comparative tests:

- Semantic-social depth-first search SSDFS with two heuristics: vertex-based heuristic and vertex-edge-based heuristic. We compare these algorithms with the item-based collaborative filtering recommendation algorithm and the hybrid recommendation algorithm.
- Semantic-social breadth-first search SSBFS with two heuristics: vertex-based and vertex-edge based. We compare these algorithms with the item-based collaborative filtering recommendation algorithm and the hybrid recommendation algorithm.
- We compare the semantic-social depth-first search SSDFS with the semantic-social breadth-first search SSBFS algorithms.
- We compare the semantic-social depth-first search SSDFS with the semantic-social breadth-first search SSBFS using different centralities: degree centrality, closeness centrality and betweenness centrality. Moreover we propose to combine degree and betweenness centralities in one algorithm.

5.7.1 Semantic-Social Depth-First Search SSDFS

In this experiment, we test two classes of the semantic-social recommendation approach as explained in Chapter 4, Subsection 4.4.2:

1. Semantic-social depth-first search with vertex-based heuristic, based on user degree centrality and user-item semantic relevancy measure.
2. Semantic-social depth-first search with vertex-edge-based heuristic, based on user degree centrality, edge weight and user-item semantic relevancy measure.

Vertex-based and vertex-edge-based heuristics are based on the users *degree* centrality, so the centrality vector of these algorithms contains users with high values of degree centrality.

In this test we use Amazon(3) dataset which has $38,982$ users and $5,005,398$ edges; and we submit 54 different recommendation queries for 54 different items.

Figure 5.12 shows a comparison of the average values of precision, recall and F-measure of the 54 recommendation quires submitted to the four recommendation algorithms (vertex-based, vertex-edge-based, collaborative filtering and the hybrid); and Figure 5.13 shows

a comparison of the data coverage and the execution time of these algorithms.

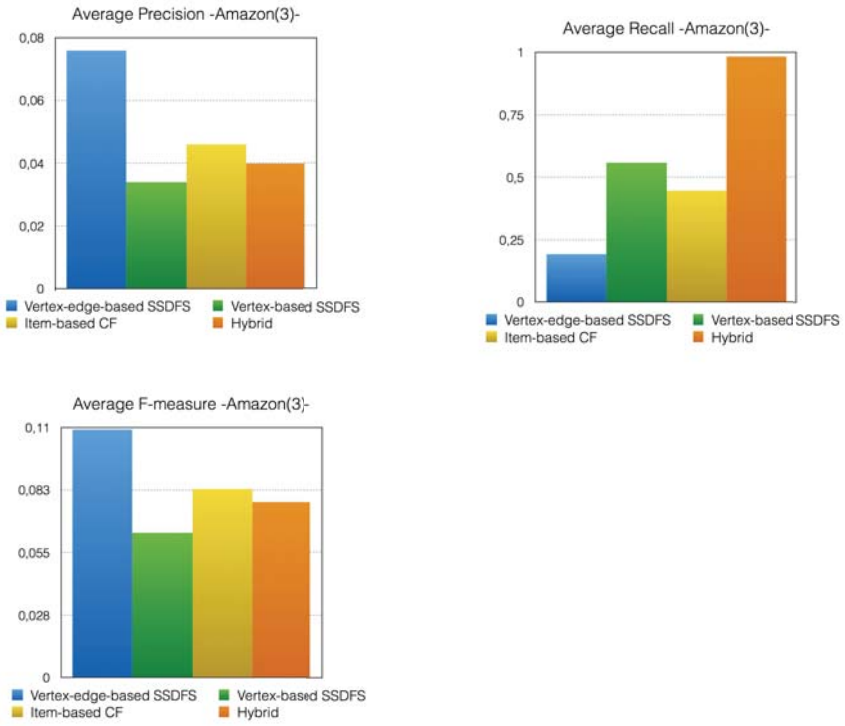

Figure 5.12: Average precision recall and F-measure, applied on Amazon(3) dataset.

Accuracy measures Figure 5.12 shows the average precision, the average recall and the average F-measure of the 54 submitted recommendation queries.

- Precision: Figure 5.12 shows that vertex-edge-based SSDFS gives the best precision value. In fact the range between the best precision value and the worst one is not very wide, but yet we can say that vertex-edge-based SSDFS gives good precision compared to the other algorithms.
- Recall: Hybrid recommendation algorithm gives the best recall value compared to the other algorithms, then the vertex-

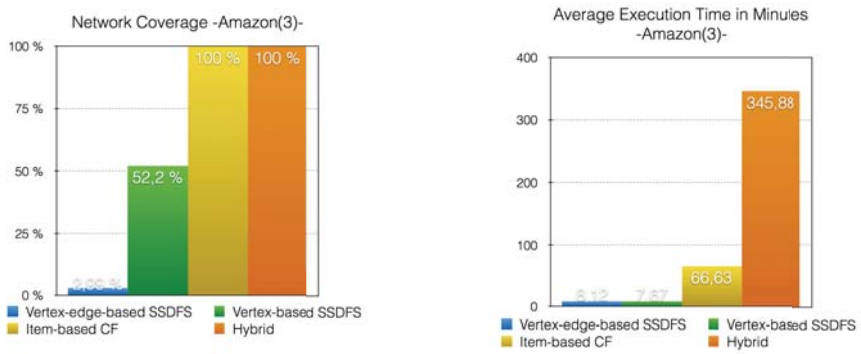

Figure 5.13: Data coverage and execution time in minutes

based SSDFS comes next after the hybrid algorithm. From Figure 5.12 and according to recall values the range between the best value and the worst value is pretty wide compared to the precision.

- F-measure: we notice that, the vertex-edge-based algorithm gives the best F-measure compared to the other three algorithms. Moreover, vertex-based algorithm, collaborative filtering algorithm and hybrid algorithm show close values of F-measure but not enough to be as good as the vertex-edge-based algorithm.

Graph coverage Figure 5.13 shows the average data coverage overall the tested algorithms. We notice that vertex-edge-based SSDFS algorithm covers a very small part of the graph (2.99%) compared to the vertex-edge SSDFS algorithm which covers 52.2%. Moreover, item-based collaborative filtering and the hybrid algorithm explore 100% of the dataset.

Execution time Figure 5.13 shows that hybrid recommendation algorithm takes a very long time to answer the recommendation query, and it is the same case for item-based algorithm. On the other hand, vertex-based SSDFS and vertex-edge-based SSDFS answers the recommendation query in a very short time, less than 9 minutes.

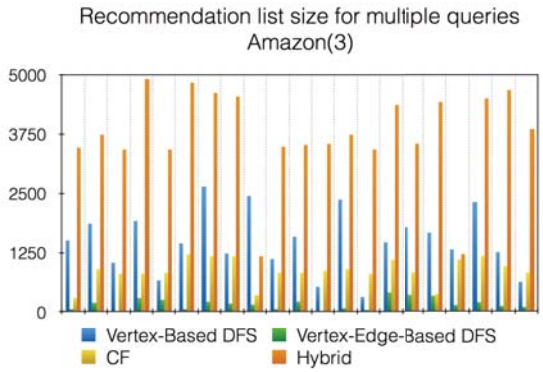

Figure 5.14: The recommendation list size of 21 queries of the compared algorithms

Discussion Our results show that vertex-edge-based heuristic gives better accuracy and performance results than the other algorithms; also it is better than vertex-based algorithm, even if the range between the largest value and the smallest value is not wide.

Moreover, the importance of our approach comes form the fact that, it explores a small part of the data, 2.99% in the case of vertex-edge-based SSDFS, and in a very short time; and it still gives good accuracy values compared to the other methods that search all the data (and these take a very long time as shown in Figure 5.13). So far, the semantic-social depth-first search shows an acceptable accuracy and a significantly improved performance compared to the other algorithms.

Moreover, Figure 5.14 shows the size of some of the recommendation lists, the outputs of the tested recommendation algorithms. Obviously, as vertex-edge-based SSDFS explores a small part of the network, it has the smallest size of the recommendation list compared to the other recommendation lists given by the other algorithms, this can explain the high precision values and the small recall value. Furthermore we notice that, the recommendation list size of the vertex-based SSDFS algorithm, in some queries, is near to the one of the collaborative filtering algorithm.

5.7.2 Semantic-Social Breadth-First search SSBFS

Here we test two versions of the semantic-social recommendation using breadth-first search algorithm as explained in Chapter 4, Subsection 4.4.3:

1. Semantic-social breadth-first search, with user centrality (degree) and item-user relevancy measure.

2. Semantic-social breadth-first search with user centrality (degree), item-user relevancy measure and edge weight.

We use exactly the same recommendation quires, the same parameters and the same dataset Amazon(3) as described in (Subsection 5.7.1).

Figure 5.15 shows a comparison of the average precision, recall and F-measure of the 54 recommendation queries submitted to the four recommendation algorithms (vertex-based, vertex-edge-based, collaborative filtering and the hybrid). Figure 5.16 shows a comparison of the data coverage and the execution time of these algorithms.

Accuracy Figure 5.15 shows that item-based collaborative filtering algorithm gives the best precision value and vertex-based SSBFS gives the worst precision value (the range between the two values is not very wide).

Moreover, the hybrid recommendation algorithm shows the best recall value compared to the other algorithms, then the vertex-based algorithm, while collaborative filtering algorithm shows the worst recall value.

Furthermore, according to F-measure collaborative filtering shows the best results, while the vertex-based algorithm shows the worst values. We can also notice that, the range of the differences of F-measure values is not wide enough between the four algorithms.

Graph coverage Figure 5.16 shows that vertex-based and vertex-edge-based algorithms explore less data than the item-based collaborative filtering and the hybrid recommendation. In fact, our proposed semantic-social recommendation algorithms explore about 50% of the dataset while the other algorithms explore 100% of the dataset.

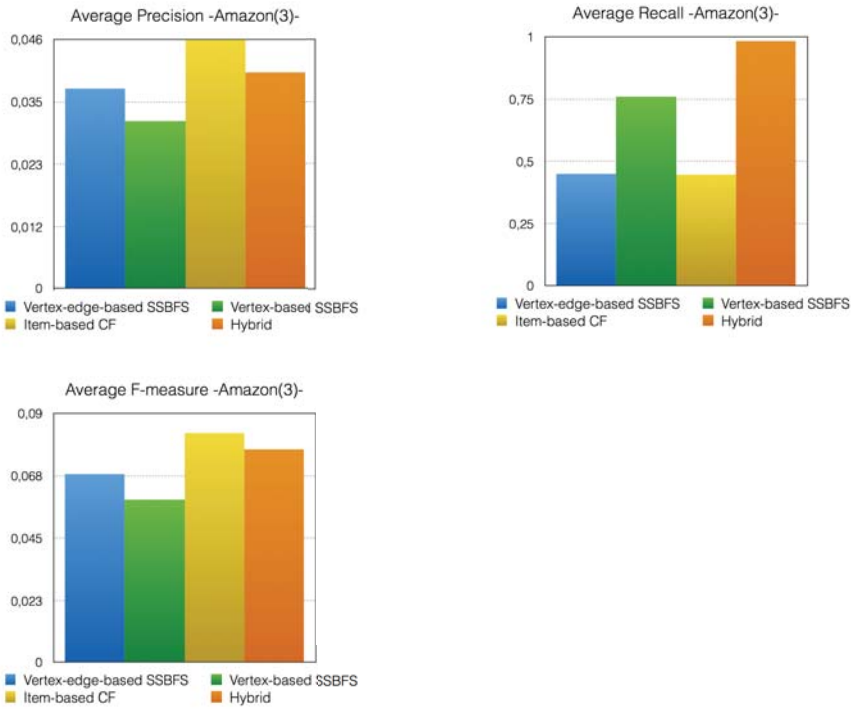

Figure 5.15: Average precision recall and F-measure, applied on Amazon(3) dataset for SSBFS algorithm.

Execution time Figure 5.16 shows that vertex-based and vertex-edge-based algorithms are very rapid in answering the recommendation query, they take less than 12 minutes, which is a very significant time compared to the other classical algorithms which takes more than 300.

Discussion Our results show that the item-based collaborative filtering and the hybrid algorithm give better accuracy values than vertex-edge-based SSBFS and vertex-based SSBFS. But these differences in accuracy values are not significant, because the range between the minimum value and the maximum value is not wide enough.

Moreover, our proposed approach shows a significant improve-

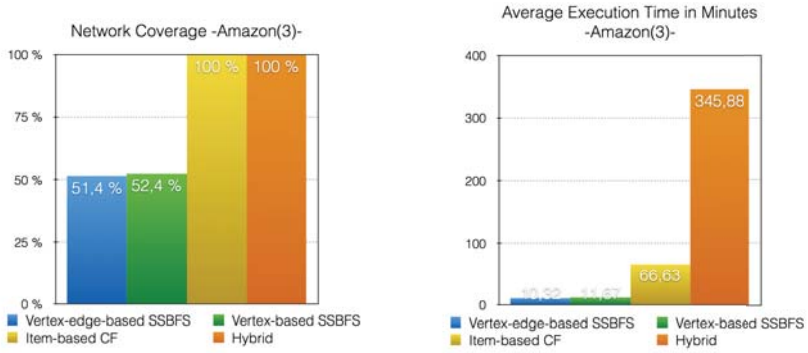

Figure 5.16: Data coverage and execution time in minutes

ment in the performance compared to the two the other classical algorithms.

Figure 5.17 shows the size of the recommendation lists, generated by submitting several recommendation queries on the four algorithms. Obviously the size of the recommendation lists of vertex-edge-based algorithm is the minimum compared to the others, because it explores a small part of the network.

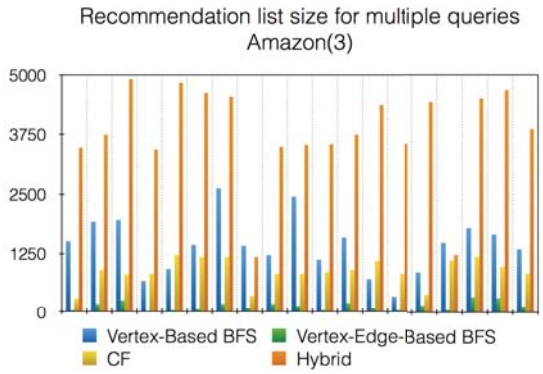

Figure 5.17: The recommendation list size of 21 queries applied on the compared algorithms

5.7.3 Comparasion between SSDFS and SSBFS

We propose to compare SSDFS and SSBFS algorithms in their two versions vertex-based and vertex-edge-based according to the accuracy and the performance. We use the same recommendation queries, the same parameters and the same dataset Amazon(3) as described in (Subsection 5.7.1) and (Subsection 4.4.3) of Chapter 4. Also, we submit the same 54 recommendation queries. Our results are described as follows:

Accuracy Figure 5.18 shows that the vertex-edge-based SSDFS shows better precision than the vertex-edge-based SSBFS, similarly vertex-based SSDFS algorithm shows better precision than vertex-based SSBFS algorithm. Moreover, vertex-edge-based SSBFS and vertex-based SSBFS show better recall than vertex-edge-based SSDFS and vertex-based SSDFS. On the other hand, the F-measure shows that vertex-edge-based SSDFS gives the most significant values compared to the other algorithms.

Graph coverage and execution time Figure 5.19 shows that vertex-edge-based SSDFS explores a very small part of the co-purchases network 3% of the network, while vertex-edge-based SSBFS explores 52% of the network.

For the case of execution time the range of differences between SSDFS and SSBFS is about 4 minutes, and the vertex-edge-based SSDFS algorithm achieves the recommendation in the shortest time compared to the other algorithms.

Discussion Figure 5.20 shows the size of the recommendation lists of the four algorithms, and we notice that vertex-edge-based SSDFS has the minimum recommendation list size as it explores a small part of the network.

5.7.4 Social network analysis measures as parameters

In these experiments we apply and compare the semantic-social depth-first search SSDFS and the semantic-social breadth-first search SSBFS, as follows:

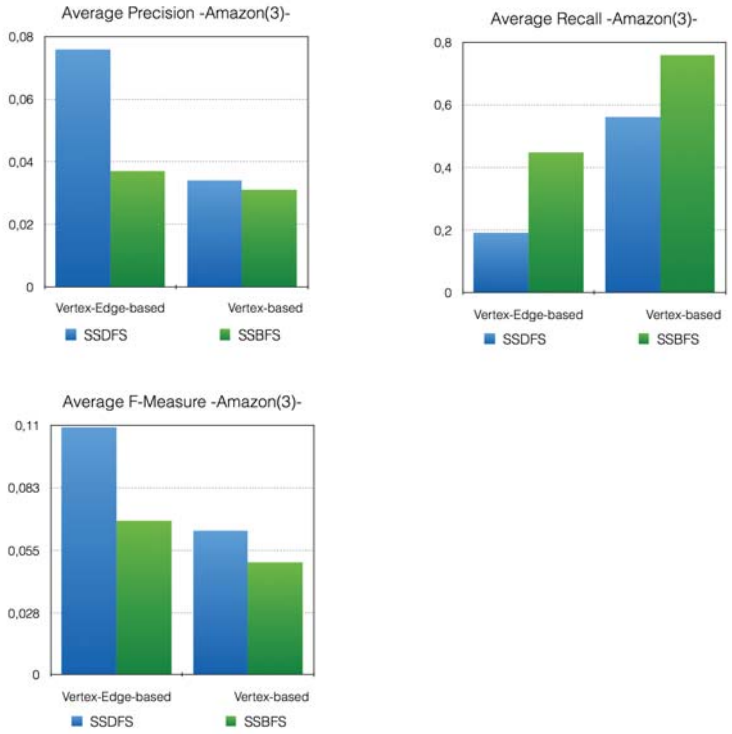

Figure 5.18: Average precision recall and F-measure, applied on Amazon(3) dataset.

1. SSDFS and SSBFS with semantic relevancy, closeness centrality and edge weight.

2. SSDFS and SSBFS with semantic relevancy, betweenness centrality and edge weight.

3. SSDFS and SSBFS with semantic relevancy, degree-betweenness centrality (in this test we combine the degree centrality with the betweenness centrality in one algorithm) and edge weight.

In these tests, we submit the recommendation queries to two types of datasets: Amazon(4) dataset and MovieLens dataset. For Amazon(4) we use the same parameters as Amazon(3). For MovieLens dataset we take the average value of δ of several queries; and the average value of all the edge weights θ in the co-rated graph. We

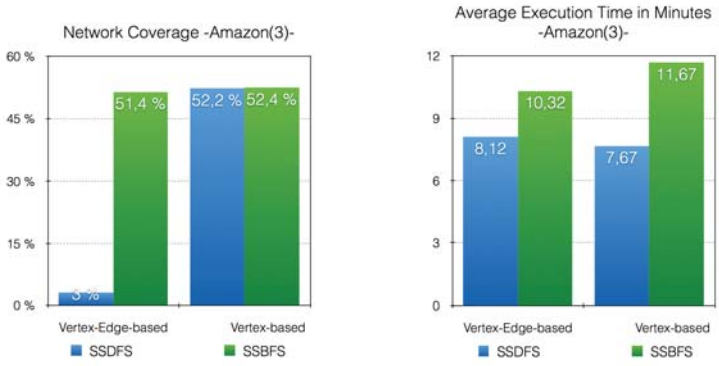

Figure 5.19: Data coverage and execution time in minutes

submit 54 queries to Amazon(4) dataset, and we submit 100 queries to MovieLens dataset.

The accuracy values and the performance values of our results are described, as follows:

Accuracy Figure 5.21 shows the values of precision, recall and F-measure of the 100 queries submitted to the SSDFS and SSBFS algorithms, using MovieLens dataset. From this figure, we notice that SSDFS with betweenness centrality gives the best precision value; and SSDFS with closeness centrality gives the worst precision value. Moreover, SSDFS with degree-betweenness centrality gives the best recall value; while SSDFS with betweenness centrality gives the worst recall value. Furthermore, for F-measure SSDFS with degree-betweenness gives the best value while SSDFS with betweenness gives the worst value.

For SSBFS algorithm, we notice the same thing as SSDFS algorithm in recall and F-measure, except precision where closeness gives the best precision value and degree gives the worst one.

Figure 5.22 shows the values of precision, recall and F-measure of the 54 queries submitted to the two algorithms SSDFS and SSBFS, using Amazon dataset. In SSDFS algorithm degree-betweenness centrality gives the best precision value and closeness gives the worst one, also degree centrality gives the best recall value while degree-betweenness centrality gives the worst one. According to F-measure degree centrality has the best value while closeness has the worst

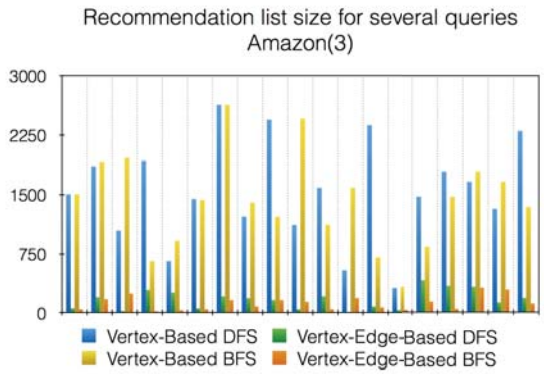

Figure 5.20: The recommendation list size of the different compared algorithms

value.

For SSBFS algorithm we notice that degree-betweenness centrality has the best precision, and degree centrality has the best recall and F-measure values.

Graph coverage The graph coverage of MovieLens graph is shown in Figure 5.23, from this figure we notice that the algorithms explore a small part of the graph, the range of graph exploration is between 16% in the case of betweenness centrality and 53% in the case of degree-betweenness centrality for SSDFS algorithm. In the case of SSBFS algorithm the algorithm explores a larger part of the graph and the range is between 92% in the case of degree centrality and 86% in the case of degree-betweenness centrality.

The graph coverage in Amazon(4) dataset is shown in Figure 5.24, and it is very small, its range is between 0.21% in the case of degree-betweenness and 2.3% in the case of degree centrality for SSDFS algorithm. Furthermore, the graph coverage range for SSBFS algorithm is between 3.1% in the case of closeness centrality and 37.3% in the case of degree centrality.

Discussion MovieLens dataset and Amazon(4) dataset has no wide range between the best accuracy values and the worst accuracy val-

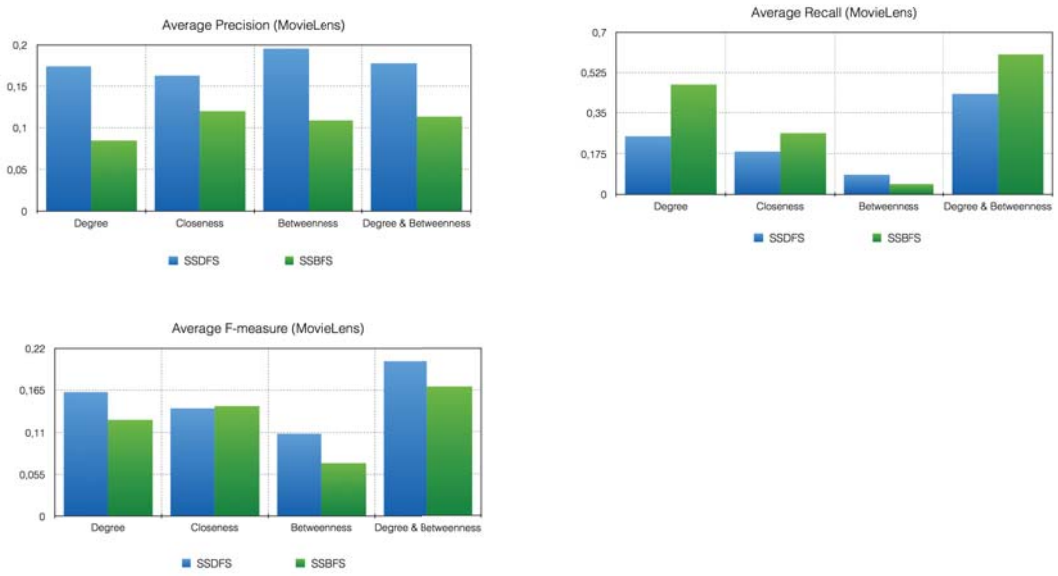

Figure 5.21: SSDFS algorithm vs SSBFS algorithm Precision Recall and F-measure Degree, Closeness, Betweenness and Degree-Betweenness centrality

ues using different centralities. We can say that our algorithms give an acceptable accuracy but they have a very important performance.

5.8 Conclusion

In this chapter we introduced the experiments and the validation processes of our proposed approach the *semantic-social recommender system*.

To validate our approach we used different datasets with different sizes and different natures, these datasets are the co-purchases dataset from Amazon.com and movies co-ratings dataset from MovieLens.

We compared our proposed algorithms with the classical recommendation algorithms mainly the item-based and the hybrid algorithm. Then we used precision, recall and F-measure as accuracy measures to assess the accuracy of our algorithms. Moreover, we assessed the performance of the algorithms by comparing the graph

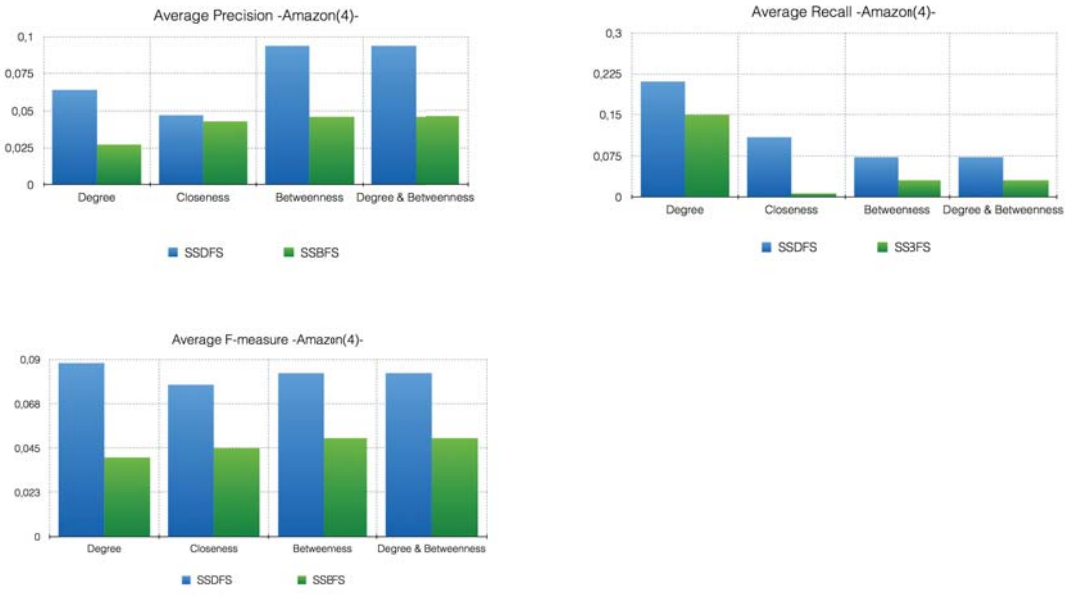

Figure 5.22: SSDFS algorithm vs SSBFS algorithm Precision Recall and F-measure

coverage and the execution time.

Also we compared the semantic-social depth-first search with the semantic-social breadth-first search, and we found that semantic-social depth-first search gives better accuracy and performance than the semantic-social breadth-first search.

It has been seen that our approach gives a comparable and sometimes better accuracy with regards to the two existing approaches considered in our experiments. On the other hand, our experiments show a significant improvement regarding to the performance, compared to the two considered existing approaches.

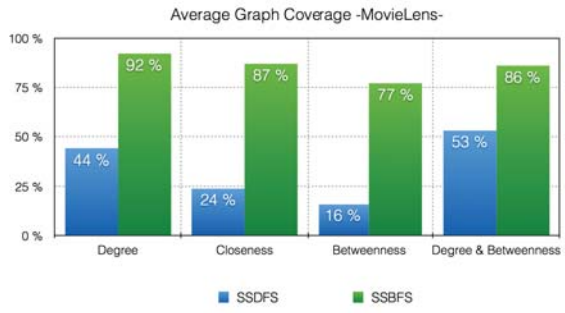

Figure 5.23: SSDFS algorithm vs SSBFS algorithm Data Coverage

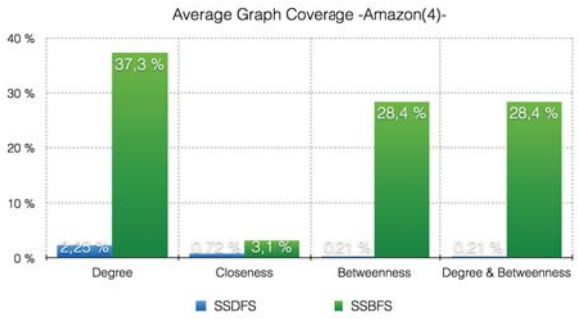

Figure 5.24: SSDFS algorithm vs SSBFS algorithm Data Coverage and Execution time

Chapter 6

Conclusion

6.1 Summary

Social networks have a vast range of applications in several domains such as: marketing, e-commerce, finding users with similar tastes for *recommendation* and searching for expertise. These applications are evident in the recent social networking sites like Facebook, LinkedIn and google+.

For long time ago, social networks have attracted the attention of scientists from several domains such as: psychology, social science, physics, mathematics and computer science; and they have been used in several studies mainly the analysis of different types of relationships between users (friendship, professional, collaborative).

Social Network Analysis SNA is used to study the structure of social networks. In fact SNA is very important because it helps to investigate the different characteristics of social networks, and to study the connectivity and the relationships between different users. Due to that, social network analysis can be implemented in *recommender systems*.

Recommender systems have attracted the attention of the researchers in the last two decades. Recommender systems recommend an item to a group of users by generating recommendation lists that contain all the possible relevant users to the input item, and they use methods from domains such as information filtering and information retrieval.
Recommender systems have several applications, in our modern life,

as movies recommendation, music recommendation and books recommendation.
Recommender systems have three main classes: content-based, collaborative filtering and hybrid. Content-based recommendation is based on the relevance between the item features and the user profile. Collaborative filtering is based on the common preferences between users. Hybrid recommendation combines content-based recommendation with collaborative filtering.

In this thesis, our focus is on hybrid recommendation applied on social collaboration networks or co-purchases networks. In this system the *collaborative filtering* depends on (a) user centralities and (b) the strength of social ties between social network users. Moreover, the content-based depends on the contents of user preferences profile and the content of item features profile, for that we use user-item relevancy measure, based on taxonomy, to find the semantic relevance between the input item and the users .

Based on social network analysis algorithms, and the taxonomy used in user profile and item profile, we propose a new approach named *Semantic-Social recommender system* which recommends an input item to users connected by a collaboration social network, using two types of information: *semantic information* using taxonomy representation of user profile and item profile; and *social information* using information about users and their connections in the co-purchases network.

6.2 Contribution

We proposed a new approach of recommendation, named *semantic-social recommender system.* In this approach we employed two types of information: *semantic information* and *social information.*
Semantic information depends on the semantics that describe user profile and item profile.
Social information depends on the role and the position of the users in the co-purchases collaboration networks. Semantic and social information of our approach are described as follows:

1. Semantic information including:

   (a) User profile tree using ontology (taxonomy).
   (b) Item profile tree using ontology (taxonomy).
   (c) User-item semantic relevancy, which is used to find out the semantic relevance between user taxonomy tree and item taxonomy tree.

2. Social information including:
   (a) Co-purchases network, in which vertices represent users in the recommender systems and edges represent the weighted relations between users.
   (b) Graph searching algorithms based on Depth-first search DFS and Breadth-first search BFS.

For that we proposed:

1. semantic measure, which is used to find the semantic relevance between user preferences and item features in the system;
2. social heuristics, which are based on (a) user centrality in the co-purchases network and (b) social ties between users in the co-purchases network.

In our methods, we proposed to integrate semantic information and social information with graph searching algorithms, mainly DFS and BFS. Thus we called our algorithms semantic-social depth first search SSDFS and semantic-social breadth-first search SSBFS. These algorithms are designed to search a *small* part of the graph while keep having a significant recommendation accuracy, which is not the case of the classical recommendation algorithms (collaborative-filtering and content-based). In fact, classical recommendation algorithms search *all* the dataset in order to have a significant accuracy.

6.2.1 Tests and Results

We tested our methods on several datasets with several sizes, these datasets are: Amazon dataset, from Amazon.com, and MovieLens. We compared our methods with the classical recommendation algorithms: item-based collaborative filtering and hybrid recommendation.

Our results showed a significant improvement in performance and accuracy, compared to item-based collaborative filtering and hybrid recommender systems.

The main improvement of our approach is to significantly reduce the size of explored data, while ensuring a slightly better accuracy. For instance, SSDFS algorithm with vertex-edge-based heuristic explores (2.99%) of the dataset while item-based collaborative-filtering and hybrid recommendation explore (100%) of the dataset, in addition SSDFS algorithm gives a slightly better accuracy than item-based and hybrid recommendations.

6.3 Perspectives and Future work

We have several proposes and perspectives for future work. In this section, we mention some of these perspectives grouping them in two groups: short-term perspectives and long-term perspectives.

6.3.1 Short-term perspectives

Studying the impact of other types of users centralities in the recommendation in our approach we use degree centrality, closeness centrality, betweenness centrality and we proposed a hybrid centrality, by combining degree with betweenness. For future, we can use other types of centralities such as PageRank.

Applying our algorithms on the network communities This proposed approach is described as follows:

1. Group the users, of the collaboration network, according to their communities. By applying one of the well known community detection algorithms.
2. Create a community profile for each specific community.
3. Submit the recommendation query on each community instead of submitting the recommendation query on the global network.

For that, we propose to define the community profile as the union of all the user tree profiles $UTP(u)$ of all the users, that are members of this community. As a result, the community profile will have a

weighted tree structure, in which weights are attached to the concepts of this tree and their weight values equal to the number of their occurrence in all the $UTP(u)$ of all the users of this community.
In this case, we will need a community-item relevancy measure to determine the relevancy between the input item and the community profile. Community profile summarizes all the users preferences in this community. If the community-item relevancy measure is acceptable (according to a predefined criterion), then the SSDFS or the SSBFS will start the semantic-social recommendation in this community; and the semantic-social recommender system will do the same process for all the communities in the network.

6.3.2 Long-term perspectives

Big Data by applying and testing our algorithms on big data and real social networks such as Facebook, LinkedIn and Twitter.

Studying the impact of using different types of user-item relevancy measures in this thesis we proposed an intuitive user-item relevancy measure, based on a taxonomy tree using 'is-a' hierarchy. So, studying more complex user-item relevancy measures and using more complex ontology than a taxonomy could lead us to a better user-item semantic relevancy measure.

Using tripartite graph tripartite graphs have three disjoint sets of vertices. Thus, we can employ tripartite graphs in the semantic-social recommender system by using three different sets of graph vertices, as follows: a set for users, a set for items and a third set for semantic concepts connected via taxonomy. As a consequence, we will have the relations: user-item, user-semantic-taxonomy and item-semantic-taxonomy. Furthermore, we can adapt our algorithms on such types of graphs in order to achieve the recommendation.

Bibliography

[1] *GroupLens: An Open Architecture for Collaborative Filtering of Netnews*, Chapel Hill, North Carolina, 1994. ACM.

[2] *Efficient Identification of Web Communities*, Boston, MA, August 2000.

[3] Silvana Aciar, Debbie Zhang, Simeon Simoff, and John Debenham. Informed recommender: Basing recommendations on consumer product reviews. *IEEE Intelligent Systems*, 22(3):39–47, May 2007.

[4] Lada A. Adamic, Rajan M. Lukose, Amit R. Puniyani, and Bernardo A. Huberman. Search in power-law networks. *Physical Review E*, 64(4):046135+, September 2001.

[5] G. Adomavicius and A. Tuzhilin. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734–749, 2005.

[6] C. C. Aggarwal, J. L. Wolf, K. Wu, and P. S. Yu. Horting Hatches an Egg: A New Graph-Theoretic Approach to Collaborative Filtering. In *KDD '99: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 201–212, San Diego, California, United States, 1999. ACM.

[7] Jae-Wook Ahn, Peter Brusilovsky, Jonathan Grady, Daqing He, and Sue Y. Syn. Open User Profiles for Adaptive News Systems: Help or Harm? In *Sixteenth International World Wide Web Conference, Banff, Alberta, Canada*, May 2007.

[8] Richard D. Alba and Charles Kadushin. The Intersection of Social Circles: A New Measure of Social Proximity in Networks. *Sociological Methods Research*, 5(1):77–102, August 1976.

[9] R. Albert, H. Jeong, and A. L. Barabasi. The diameter of the world wide web. *Nature*, 401:130–131, 1999.

[10] Antti Autere. Aas an optimal resource allocation policy in path finding problems. In *Proceedings of the Fourteenth International Florida Artificial Intelligence Research Society Conference*, pages 139–144. AAAI Press, 2001.

[11] Yoram Bachrach and Ralf Herbrich. Fingerprinting Ratings for Collaborative Filtering — Theoretical and Empirical Analysis String Processing and Information Retrieval. In Edgar Chavez and Stefano Lonardi, editors, *String Processing and Information Retrieval*, volume 6393 of *Lecture Notes in Computer Science*, chapter 3, pages 25–36. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2010.

[12] Lars Backstrom and Jure Leskovec. Supervised random walks: Predicting and recommending links in social networks. *CORR*, abs/1011.4071, 2010.

[13] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 1st edition, May 1999.

[14] Marko Balabanović. Exploring versus exploiting when learning user models for text recommendation. *User Modeling and User-Adapted Interaction*, 8(1-2):71–102, January 1998.

[15] Marko Balabanović and Yoav Shoham. Fab: content-based, collaborative recommendation. *Commun. ACM*, 40(3):66–72, March 1997.

[16] Murray A. Beauchamp. An improved index of centrality. *Behavioral Science*, 10(2):161–163, 1965.

[17] Robert Bell, Yehuda Koren, and Chris Volinsky. Modeling relationships at multiple scales to improve accuracy of large recommender systems. In *KDD '07: Proceedings of the 13th*

ACM SIGKDD international conference on Knowledge discovery and data mining, pages 95–104, New York, NY, USA, 2007. ACM.

[18] Robert M. Bell and Yehuda Koren. Lessons from the Netflix prize challenge. *SIGKDD Explor. Newsl.*, 9(2):75–79, December 2007.

[19] Nesserine Benchettara, Rushed Kanawati, and Céline Rouveirol. A supervised machine learning link prediction approach for academic collaboration recommendation. In *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, pages 253–256, New York, NY, USA, 2010. ACM.

[20] J. Bennett and S. Lanning. The Netflix Prize. In *Proceedings of the KDD Cup Workshop 2007*, pages 3–6, New York, 2007. ACM.

[21] Pavel Berkhin. A Survey on PageRank Computing. *Internet Mathematics*, 2(1):73–120, July 2005.

[22] Rajendra Bhatia. *Matrix Analysis*. Springer, 1997.

[23] Daniel Billsus and Michael J. Pazzani. Learning collaborative information filters. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, pages 46–54, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[24] Daniel Billsus and Michael J. Pazzani. A hybrid user model for news story classification. In *Proceedings of the seventh international conference on User modeling*, UM '99, pages 99–108, Secaucus, NJ, USA, 1999. Springer-Verlag New York, Inc.

[25] Daniel Billsus and Michael J. Pazzani. User Modeling for Adaptive News Access. *User Modeling and User-Adapted Interaction*, 10(2-3):147–180, 2000.

[26] Y. Blanco-Fernandez, J. Pazos-arias, A. Gil-Solla, M. Ramos-Cabrer, and M. Lopez-Nores. Providing entertainment by content-based filtering and semantic reasoning in intelligent recommender systems. *IEEE Trans. on Consum. Electron.*, 54(2):727–735, May 2008.

[27] Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008+, October 2008.

[28] Kurt Bollacker, Steve Lawrence, and Lee C. Giles. CiteSeer: An Autonomous Web Agent for Automatic Retrieval and Identification of Interesting Publications. In Katia P. Sycara and Michael Wooldridge, editors, *Proceedings of the Second International Conference on Autonomous Agents*, pages 116–123, New York, 1998. ACM Press.

[29] Phillip Bonacich. Power and Centrality: A Family of Measures. *American Journal of Sociology*, 92(5):1170–1182, 1987.

[30] Phillip Bonacich. Some unique properties of eigenvector centrality. *Social Networks*, 29(4):555–564, October 2007.

[31] Stephen P. Borgatti. Centrality and network flow. *Social Networks*, 27(1):55–71, January 2005.

[32] Allan Borodin, Gareth O. Roberts, Jeffrey S. Rosenthal, and Panayiotis Tsaparas. Link analysis ranking: algorithms, theory, and experiments. *ACM Trans. Inter. Tech.*, 5(1):231–297, February 2005.

[33] U. Brandes. A faster algorithm for betweenness centrality, 2001.

[34] John S. Breese, David Heckerman, and Carl Kadie. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. pages 43–52, 1998.

[35] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the seventh international conference on World Wide Web 7*, WWW7, pages 107–117, Amsterdam, The Netherlands, The Netherlands, 1998. Elsevier Science Publishers B. V.

[36] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the seventh international conference on World Wide Web 7*, WWW7, pages 107–117, Amsterdam, The Netherlands, The Netherlands, 1998. Elsevier Science Publishers B. V.

[37] D. Bu, Y. Zhao, L. Cai, H. Xue, X. Zhu, H. Lu, J. Zhang, S. Sun, L. Ling, N. Zhang, G. Li, and R. Chen. Topological structure analysis of the protein-protein interaction network in budding yeast. *Nucleic Acids Research*, 31:2443–2450, 2003.

[38] Robin Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, November 2002.

[39] Robin Burke. Hybrid Web Recommender Systems The Adaptive Web. In Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl, editors, *The Adaptive Web*, volume 4321 of *Lecture Notes in Computer Science*, chapter 12, pages 377–408. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2007.

[40] Laurent Candillier, Frank Meyer, and Marc Boullé. Comparing State-of-the-Art Collaborative Filtering Systems. In *MLDM '07: Proceedings of the 5th international conference on Machine Learning and Data Mining in Pattern Recognition*, pages 548–562. Springer-Verlag, Berlin, Heidelberg, 2007.

[41] P. Cano, O. Celma, M. Koppenberger, and Martin J. Buldú. Topology of music recommendation networks. *Chaos An Interdisciplinary Journal of Nonlinear Science*, 16, 2006.

[42] Iván Cantador, Alejandro Bellogín, and Pablo Castells. News@hand: A semantic web approach to recommending news. In *Proceedings of the 5th international conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, AH '08, pages 279–283, Berlin, Heidelberg, 2008. Springer-Verlag.

[43] Ivan Cantador, Pablo Castells, and David Vallet. Enriching group profiles with ontologies for knowledge-driven colalborative content retrieval. In *Proceedings of the 15th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, WETICE '06, pages 358–363, Washington, DC, USA, 2006. IEEE Computer Society.

[44] Sylvain Castagnos and Anne Boyer. A client/server user-based collaborative filtering algorithm: Model and implementation. In *Proceedings of the 2006 conference on ECAI 2006: 17th European Conference on Artificial Intelligence August 29 –*

September 1, 2006, Riva del Garda, Italy, pages 617–621, Amsterdam, The Netherlands, The Netherlands, 2006. IOS Press.

[45] íscar Celma and Xavier Serra. Foafing the music: Bridging the semantic gap in music recommendation. *Web Semant.*, 6(4):250–256, November 2008.

[46] Ò. Celma, Rafael Ramirez, and P. Herrera. Foafing the music: A music recommendation system based on rss feeds and user preferences. In *6th International Conference on Music Information Retrieval (ISMIR)*, London, UK, 2005.

[47] S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, P. Raghavan, and S. Rajagopalan. Automatic resource list compilation by analyzing hyperlink structure and associated text. In *WWW7*, 1998.

[48] M. Connor and J. Herlocker. Clustering items for collaborative filtering, 2001.

[49] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, third edition edition, July 2009.

[50] Fabio Crestani and Puay L. Lee. Searching the Web by constrained spreading activation. *Inf. Process. Manage.*, 36(4):585–605, July 2000.

[51] Abhinandan S. Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 271–280, New York, NY, USA, 2007. ACM.

[52] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, and Dasarathi Sampath. The YouTube video recommendation system. In *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, pages 293–296, New York, NY, USA, 2010. ACM.

[53] Mukund Deshpande and George Karypis. Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177, January 2004.

[54] C. Desrosiers and G. Karypis. A comprehensive survey of neighborhood-based recommendation methods. *Recommender Systems Handbook*, 2011.

[55] Byron Dom, Iris Eiron, Alex Cozzi, and Yi Zhang. Graph-based ranking algorithms for e-mail expertise analysis. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, DMKD '03, pages 42–48, New York, NY, USA, 2003. ACM.

[56] J. C. Dunn. A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters. *Journal of Cybernetics*, 3(3):32–57, 1973.

[57] M. Eirinaki, M. Vazirgiannis, and I. Varlamis. SEWeP: using site semantics and a taxonomy to enhance the Web personalization process. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 99–108, New York, NY, USA, 2003. ACM Press.

[58] P. Erdős and A. Rényi. On random graphs. I. *Publ. Math. Debrecen*, 6:290–297, 1959.

[59] Santo Fortunato. Community detection in graphs. *CoRR*, abs/0906.0612, 2009.

[60] Francois Fouss, Alain Pirotte, and Marco Saerens. A novel way of computing similarities between nodes of a graph, with application to collaborative recommendation. In *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, WI '05, pages 550–556, Washington, DC, USA, 2005. IEEE Computer Society.

[61] L. Freeman. Centrality in social networks conceptual clarification. *Social Networks*, 1(3):215–239, 1979.

[62] L. Freeman. Centrality in valued graphs: A measure of betweenness based on network flow. *Social Networks*, 13(2):141–154, June 1991.

[63] Noah E. Friedkin. Theoretical Foundations for Centrality Measures. *The American Journal of Sociology*, 96(6):1478–1504, 1991.

[64] Susan Gauch, Mirco Speretta, Aravind Chandramouli, and Alessandro Micarelli. User Profiles for Personalized Information Access The Adaptive Web. In Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl, editors, *The Adaptive Web*, volume 4321 of *Lecture Notes in Computer Science*, chapter 2, pages 54–89. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2007.

[65] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, June 2002.

[66] Jennifer Golbeck and James Hendler. Inferring binary trust relationships in web-based social networks. *ACM Trans. Internet Technol.*, 6(4):497–529, November 2006.

[67] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35(12):61–70, December 1992.

[68] Marco Gori and Augusto Pucci. ItemRank: a random-walk based scoring algorithm for recommender engines. In *Proceedings of the 20th international joint conference on Artifical intelligence*, pages 2766–2771, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[69] Roger V. Gould and Roberto M. Fernandez. Structures of Mediation: A Formal Approach to Brokerage in Transaction Networks. *Sociological Methodology*, 19:89–126, 1989.

[70] Mark S. Granovetter. The Strength of Weak Ties. *American Journal of Sociology*, 78(6):1360–1380, 1973.

[71] Miha Grčar, Blaž Fortuna, Dunja Mladenič, and Marko Grobelnik. knn versus svm in the collaborative filtering framework. *Data Science and Classification*, pages 251–260, 2006.

[72] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowl. Acquis.*, 5(2):199–220, June 1993.

[73] Thomas R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. *Int. J. Hum.-Comput. Stud.*, 43(5-6):907–928, December 1995.

[74] Jelena Grujić. Movies recommendation networks as bipartite graphs. In *Proceedings of the 8th international conference on Computational Science, Part II*, ICCS '08, pages 576–583, Berlin, Heidelberg, 2008. Springer-Verlag.

[75] Nicola Guarino. Formal Ontology and Information Systems, 1998.

[76] Harary, Frank. *Graph theory.* Addison-Wesley, 1969.

[77] Peter Hart, Nils Nilsson, and Bertram Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, July 1968.

[78] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer, corrected edition, August 2003.

[79] Jon Herlocker, Joseph A. Konstan, and John Riedl. An Empirical Analysis of Design Choices in Neighborhood-Based Collaborative Filtering Algorithms. *Inf. Retr.*, 5(4):287–310, October 2002.

[80] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '99, pages 230–237, New York, NY, USA, 1999. ACM.

[81] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, January 2004.

[82] Thomas Hofmann. Collaborative filtering via gaussian probabilistic latent semantic analysis. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 259–266, New York, NY, USA, 2003. ACM.

[83] Andreas Hotho, Robert Jäschke, Christoph Schmitz, and Gerd Stumme. FolkRank: A Ranking Algorithm for Folksonomies. In *Proc. FGIR 2006*, 2006.

[84] Zan Huang, Hsinchun Chen, and Daniel D. Zeng. Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. *ACM Trans. Inf. Syst.*, 22(1):116–142, January 2004.

[85] Zan Huang, Wingyan Chung, Thian-Huat Ong, and Hsinchun Chen. A Graph-based Recommender System for Digital Library. *JCDL '02*, pages 65–73, 2002.

[86] Sung Ju Hwang, Kristen Grauman, and Fei Sha. Semantic kernel forests from multiple taxonomies. In P. Bartlett, F.C.N. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1727–1735. 2012.

[87] Mohsen Jamali and Martin Ester. Trustwalker: a random walk model for combining trust-based and item-based recommendation. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 397–406, New York, NY, USA, 2009. ACM.

[88] Jay J. Jiang, David W. Conrath. Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy. *CoRR*, cmp-lg/9709008, 1997.

[89] Glen Jeh and Jennifer Widom. Scaling personalized web search. In *Proceedings of the 12th international conference on World Wide Web*, WWW '03, pages 271–279, New York, NY, USA, 2003. ACM.

[90] Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, March 1953.

[91] B. W. Kernighan and S. Lin. An Efficient Heuristic Procedure for Partitioning Graphs. *The Bell system technical journal*, 49(1):291–307, 1970.

[92] Larry Kerschberg, Wooju Kim, and Anthony Scime. A Semantic Taxonomy-Based Personalizable Meta-Search Agent.

In *WISE '01: Proceedings of the Second International Conference on Web Information Systems Engineering (WISE'01) Volume 1*, Washington, DC, USA, December 2001. IEEE Computer Society.

[93] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, September 1999.

[94] Jon M. Kleinberg. Hubs, authorities, and communities. *ACM Comput. Surv.*, 31(4es), December 1999.

[95] D. E. Knuth. *The Stanford GraphBase: a platform for combinatorial computing*. Addison-Wesley, 1993.

[96] Joseph A. Konstan, Bradley N. Miller, David Maltz, Jonathan L. Herlocker, Lee R. Gordon, and John Riedl. Grouplens: applying collaborative filtering to usenet news. *Commun. ACM*, 40(3):77–87, March 1997.

[97] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 426–434, New York, NY, USA, 2008. ACM.

[98] R. Korf. Depth-first iterative-deepening An optimal admissible tree search. *Artificial Intelligence*, 27(1):97–109, September 1985.

[99] R. Korf. Depth-first iterative-deepening An optimal admissible tree search. *Artificial Intelligence*, 27(1):97–109, September 1985.

[100] Bruce Krulwich and Chad Burkey. The infofinder agent: Learning user interests through heuristic phrase extraction. *IEEE Expert: Intelligent Systems and Their Applications*, 12(5):22–27, September 1997.

[101] Ravi Kumar, Jasmine Novak, and Andrew Tomkins. Structure and evolution of online social networks. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 611–617, New York, NY, USA, 2006. ACM.

[102] Lada A. Adamic and Eytan Adar. How To Search a Social Network. *Social Networks*, 27, 2005.

[103] Claudia Leacock and Martin Chodorow. *Combining Local Context and WordNet Similarity for Word Sense Identification*, chapter 11, pages 265–283. The MIT Press, May 1998.

[104] Sangkeun Lee. A generic graph-based multidimensional recommendation framework and its implementations. In *Proceedings of the 21st international conference companion on World Wide Web*, WWW '12 Companion, pages 161–166, New York, NY, USA, 2012. ACM.

[105] Sangkeun Lee, Minsuk Kahng, and Sang-goo Lee. Flexible recommendation using random walks on implicit feedback graph. In *Proceedings of the 5th International Conference on Ubiquitous Information Management and Communication*, ICUIMC '11, pages 3:1–3:6, New York, NY, USA, 2011. ACM.

[106] Sangkeun Lee, Sang-il Song, Minsuk Kahng, Dongjoo Lee, and Sang-goo Lee. Random walk based entity ranking on graph for multidimensional recommendation. In *Proceedings of the fifth ACM conference on Recommender systems*, RecSys '11, pages 93–100, New York, NY, USA, 2011. ACM.

[107] Jure Leskovec, Lada A. Adamic, and Bernardo A. Huberman. The dynamics of viral marketing. In *EC '06: Proceedings of the 7th ACM conference on Electronic commerce*, pages 228–237, New York, NY, USA, 2006. ACM.

[108] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. Predicting Positive and Negative Links in Online Social Networks. March 2010.

[109] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, KDD '05, pages 177–187, New York, NY, USA, 2005. ACM.

[110] Henry Lieberman. Letizia: an agent that assists web browsing. In *Proceedings of the 14th international joint conference on Artificial intelligence - Volume 1*, IJCAI'95, pages 924–929,

San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[111] Dekang Lin. An Information-Theoretic Definition of Similarity. In *Proc. International Conference on Machine Learning (ICML)*, pages 296–304, 1998.

[112] Weiyang Lin, Sergio A. Alvarez, and Carolina Ruiz. Efficient adaptive-support association rule mining for recommender systems. *Data Min. Knowl. Discov.*, 6(1):83–105, January 2002.

[113] G. Linden, B. Smith, and J. York. Amazon.com recommendations: item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80, 2003.

[114] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, January 2003.

[115] Pasquale Lops, Marco Gemmis, and Giovanni Semeraro. Content-based Recommender Systems: State of the Art and Trends Recommender Systems Handbook. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, *Recommender Systems Handbook*, chapter 3, pages 73–105. Springer US, Boston, MA, 2011.

[116] László Lovász. *Random Walks on Graphs: A Survey*, pages 353–397. 1993.

[117] Ulrike Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, December 2007.

[118] J. B. Macqueen. Some methods of classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[119] Bernardo Magnini and Carlo Strapparava. Improving user modelling with content-based techniques. In *Proceedings of the 8th International Conference on User Modeling 2001*, UM '01, pages 74–83, London, UK, UK, 2001. Springer-Verlag.

[120] Harry Mak, Irena Koprinska, and Josiah Poon. Intimate: A web-based movie recommender using text categorization. In

Proceedings of the 2003 IEEE/WIC International Conference on Web Intelligence, WI '03, pages 602–, Washington, DC, USA, 2003. IEEE Computer Society.

[121] Massimo Marchiori. The quest for correct information of the Web: hyper search engines. In *Proc. of the sixth international conference on the Web*, Santa Clara, USA, April 1997.

[122] Sergei Maslov and Kim Sneppen. Specificity and Stability in Topology of Protein Networks. *Science*, 296(5569):910–913, May 2002.

[123] Matthew Richardson and Pedro Domingos. The Intelligent Surfer: Probabilistic Combination of Link and Content Information in PageRank. *In Advances in Neural Information Processing Systems (NIPS)*, 14, 2002.

[124] J. J. McAuley and J. Leskovec. Discovering social circles in ego networks. In *TKDD*, 2013.

[125] David W. McDonald and Mark S. Ackerman. Expertise recommender: a flexible recommendation system and architecture. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, CSCW '00, pages 231–240, New York, NY, USA, 2000. ACM.

[126] Stuart E. Middleton, Nigel R. Shadbolt, and David C. De Roure. Ontological user profiling in recommender systems. *ACM Trans. Inf. Syst.*, 22(1):54–88, January 2004.

[127] Peter Mika. Social networks and the semantic web. In *Proceedings of the 2004 IEEE/WIC/ACM International Conference on Web Intelligence*, WI '04, pages 285–291, Washington, DC, USA, 2004. IEEE Computer Society.

[128] Bradley N. Miller, Istvan Albert, Shyong K. Lam, Joseph A. Konstan, and John Riedl. Movielens unplugged: experiences with an occasionally connected recommender system. In *Proceedings of the 8th international conference on Intelligent user interfaces*, IUI '03, pages 263–266, New York, NY, USA, 2003. ACM.

[129] George A. Miller. The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological Review*, 63(2):81–97, 1956.

[130] George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. Introduction to WordNet: an On-line Lexical Database*. *International Journal of Lexicography*, 3(4):235–244, December 1990.

[131] Batul J. Mirza, Benjamin J. Keller, and Naren Ramakrishnan. Studying recommendation algorithms by graph analysis. *J. Intell. Inf. Syst.*, 20(2):131–160, March 2003.

[132] Dunja Mladeni'c and J. stefan. Machine learning used by Personal WebWatcher. 1999.

[133] Bamshad Mobasher, Honghua Dai, Tao Luo, and Miki Nakagawa. Effective personalization based on association rule discovery from web usage data. In *Proceedings of the 3rd international workshop on Web information and data management*, WIDM '01, pages 9–15, New York, NY, USA, 2001. ACM.

[134] James Moody. Race, School Integration, and Friendship Segregation in America. *The American Journal of Sociology*, 107(3):679–716, 2001.

[135] Raymond J. Mooney and Loriene Roy. Content-based book recommending using learning for text categorization. In *Proceedings of the fifth ACM conference on Digital libraries*, DL '00, pages 195–204, New York, NY, USA, 2000. ACM.

[136] Jacob L. Moreno. *Who Shall Survive? Foundations of Sociometry, Group Psychotherapy and Sociodrama.* Beacon House, Beacon, NY, 1953,1978.

[137] M. E. J. Newman. The Structure and Function of Complex Networks. *SIAM Review*, 45(2):167–256, 2003.

[138] M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69(6):066133+, June 2004.

[139] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74(3):036104+, July 2006.

[140] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2):026113+, August 2003.

[141] Mark Newman. Who Is the Best Connected Scientist?A Study of Scientific Coauthorship Networks. In *Complex Networks*, pages 337–370. 2004.

[142] M.E.J. Newman. Scientific collaboration networks. I. Network construction and fundamental results. *Phys Rev E Stat Nonlin Soft Matter Phys*, 64(1 Pt 2), July 2001.

[143] M.E.J. Newman. Scientific collaboration networks. II. Shortest paths, weighted networks, and centrality. *Physical Review E*, 64(1), June 2001.

[144] M.E.J. Newman. A measure of betweenness centrality based on random walks. *Social Networks*, 27:39–54, 2005.

[145] M.E.J. Newman. *Networks An Introduction*. Oxford University Press, 2010.

[146] Andrew Y. Ng, Alice X. Zheng, and Michael I. Jordan. Stable algorithms for link analysis. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '01, pages 258–266, New York, NY, USA, 2001. ACM.

[147] J. Nieminen. On centrality in a graph. *Scandinavian Journal of Psychology*, 15(1):322–336, 1974.

[148] David L. Nowell and Jon Kleinberg. The link prediction problem for social networks. In *Proceedings of the twelfth international conference on Information and knowledge management*, CIKM '03, pages 556–559, New York, NY, USA, 2003. ACM.

[149] Derry Oasullivan, Barry Smyth, David C. Wilson, Kieran Mcdonald, and Alan Smeaton. Improving the Quality of the Personalized Electronic Program Guide. *User Modeling and User-Adapted Interaction*, V14(1):5–36, February 2004.

[150] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford Digital Library Technologies Project, 1998.

[151] Michael Pazzani and Daniel Billsus. Learning and Revising User Profiles: The Identification of Interesting Web Sites. *Machine Learning*, 27(3):313–331, June 1997.

[152] Michael J. Pazzani. A Framework for Collaborative, Content-Based and Demographic Filtering. *Artificial Intelligence Review*, 13(5):393–408, December 1999.

[153] Michael J. Pazzani, Jack Muramatsu, and Daniel Billsus. Syskill & webert: Identifying interesting web sites. In William J. Clancey, Daniel S. Weld, William J. Clancey, and Daniel S. Weld, editors, *AAAI/IAAI, Vol. 1*, pages 54–61. AAAI Press / The MIT Press, 1996.

[154] Philip Resnik. Semantic Similarity in a Taxonomy: An Information-Based Measure and its Application to Problems of Ambiguity in Natural Language. *CoRR*, abs/1105.5444, 1999.

[155] Huseyin Polat and Wenliang Du. Privacy-Preserving Collaborative Filtering Using Randomized Perturbation Techniques. In *ICDM '03: Proceedings of the Third IEEE International Conference on Data Mining*, 2003.

[156] Ramasco, José J. and Dorogovtsev, S. N. and Pastor-Satorras, Romualdo. Self-organization of collaboration networks. *Phys. Rev. E*, 70(3), September 2004.

[157] Al M. Rashid, Istvan Albert, Dan Cosley, Shyong K. Lam, Sean M. McNee, Joseph A. Konstan, and John Riedl. Getting to know you: learning new user preferences in recommender systems. In *Proceedings of the 7th international conference on Intelligent user interfaces*, IUI '02, pages 127–134, New York, NY, USA, 2002. ACM.

[158] Matthew J. Rattigan, Marc Maier, and David Jensen. Graph clustering with network structure indices. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 783–790, New York, NY, USA, 2007. ACM.

[159] Raymond J. Mooney and Paul N. Bennett and Loriene Roy. Book Recommending using Text Categorization with Extracted Information. In *IN RECOMMENDER SYSTEMS.*

PAPERS FROM 1998 WORKSHOP, pages 49–54. AAAI Press, 1998.

[160] Reinout V. Rees. Clarity in the usage of the terms ontology , taxonomy. *Civil Engineering*, 2003.

[161] Albert Reka and Barabási. Statistical mechanics of complex networks. *Rev. Mod. Phys.*, 74:47–97, June 2002.

[162] Paul Resnick and Hal R. Varian. Recommender systems. *Commun. ACM*, 40(3):56–58, March 1997.

[163] Philip Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the 14th international joint conference on Artificial intelligence - Volume 1*, IJCAI'95, pages 448–453, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[164] Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to Recommender Systems Handbook. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, *Recommender Systems Handbook*, chapter 1, pages 1–35. Springer US, Boston, MA, 2011.

[165] A. Rice. The identification of blocs in small political bodies. *The American Political Science Review*, 3(21):619–627, 1927.

[166] Sheldon Ross. *First Course in Probability, A (8th Edition)*. Prentice Hall, 8 edition, January 2009.

[167] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2003.

[168] Donald S. Sade. Sociometrics of Macaca Mulatta III: n-path centrality in grooming networks. *Social Networks*, 11(3):273–292, September 1989.

[169] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted Boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, volume 227 of *ICML '07*, pages 791–798, New York, NY, USA, 2007. ACM.

[170] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Application of dimensionality reduction in recommender systems–a case study, 2000.

[171] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Analysis of recommendation algorithms for e-commerce. In *Proceedings of the 2nd ACM conference on Electronic commerce*, EC '00, pages 158–167, New York, NY, USA, 2000. ACM.

[172] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John Reidl. Item-based collaborative filtering recommendation algorithms. In *World Wide Web*, pages 285–295, 2001.

[173] Badrul M. Sarwar, Joseph A. Konstan, Al Borchers, Jonathan L. Herlocker, Bradley N. Miller, and John Riedl. Using Filtering Agents to Improve Prediction Quality in the GroupLens Research Collaborative Filtering System. In *Proceedings of the 1998 ACM conference on Computer supported cooperative work (CSCW '98)*, pages 345–354, 1998.

[174] Thomas Schank and Dorothea Wagner. Approximating clustering coefficient and transitivity. *Journal of Graph Algorithms and Applications*, 9, 2005.

[175] Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '02, pages 253–260, New York, NY, USA, 2002. ACM.

[176] Vincent Schickel-Zuber. *Ontology Filtering Inferring Missing User's preferences in eCommerce Recommender Systems*. PhD thesis, École Polytechnique Fédérale de Lussanne, Switzerland, October 2007.

[177] Upendra Shardanand and Pattie Maes. Social information filtering: algorithms for automating "word of mouth". In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, pages 210–217, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[178] B. Sheth and P. Maes. Evolving agents for personalized information filtering. pages 345–352, March 1993.

[179] Jianbo Shi and Jitendra Malik. Normalized Cuts and Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[180] Jiri Sima and Satu E. Schaeffer. On the NP-Completeness of Some Graph Cluster Measures, June 2005.

[181] B. Smyth and P. Cotter. A Personalized TV Listings Service for the Digital TV Age. *Knowledge-Based Systems*, 13(2-3):53–59, 2000.

[182] Karen Stephenson and Marvin Zelen. Rethinking centrality: Methods and examples. *Social Networks*, 11(1):1–37, March 1989.

[183] Xiaoyuan Su and Taghi M. Khoshgoftaar. A survey of collaborative filtering techniques. *Adv. in Artif. Intell.*, 2009:1–19, January 2009.

[184] Dalia Sulieman, Maria Malek, Hubert Kadima, and Dominique Laurent. Graph searching algorithms for semantic-social recommendation. In *Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012)*, ASONAM '12, pages 733–738, Washington, DC, USA, 2012. IEEE Computer Society.

[185] Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk. Investigation of various matrix factorization methods for large recommender systems. In *Proceedings of the 2nd KDD Workshop on Large Scale Recommender Systems and the Netflix Prize Competition*, August 2008.

[186] Robert Tarjan. Depth-First Search and Linear Graph Algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

[187] John A. Tomlin. A new paradigm for ranking pages on the world wide web. In *Proceedings of the 12th international conference on World Wide Web*, WWW '03, pages 350–355, New York, NY, USA, 2003. ACM.

[188] Jeffrey Travers and Stanley Milgram. An Experimental Study of the Small World Problem. *Sociometry*, 32(4):425–443, December 1969.

[189] S. Upendra. Social Information Filtering for Music Recommendation, 1994.

[190] Jun Wang, Arjen P. de Vries, and Marcel J. T. Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '06, pages 501–508, New York, NY, USA, 2006. ACM.

[191] D. J. Watts and S. H. Strogatz. Collective dynamics of'small-world'networks. *Nature*, 393(6684):409–10, 1998.

[192] Robert S. Weiss and Eugene Jacobson. A method for the analysis of the structure of complex organizations. *American Sociological Review*, 20(6), December 1955.

[193] Jaewon Yang and Jure Leskovec. Defining and Evaluating Network Communities based on Ground-truth, November 2012.

[194] Bin Yu and Munindar P. Singh. Searching social networks. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, AAMAS '03, pages 65–72, New York, NY, USA, 2003. ACM.

[195] Kai Yu, Anton Schwaighofer, Volker Tresp, Xiaowei Xu, and Hans-Peter Kriegel. Probabilistic memory-based collaborative filtering. *IEEE Trans. on Knowl. and Data Eng.*, 16(1):56–69, January 2004.

[196] Wayne W. Zachary. An Information Flow Model for Conflict and Fission in Small Groups. *Journal of Anthropological Research*, 33(4):452–473, 1977.

[197] Jun Zhang and Mark S. Ackerman. Searching for expertise in social networks: a simulation of potential strategies. In *GROUP '05: Proceedings of the 2005 international ACM SIGGROUP conference on Supporting group work*, pages 71–80, New York, NY, USA, 2005. ACM.

[198] Jun Zhang, Mark S. Ackerman, and Lada Adamic. Expertise networks in online communities: structure and algorithms. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 221–230, New York, NY, USA, 2007. ACM.

[199] Tao Zhou, Jie Ren, Matúš Medo, and Yi C. Zhang. Bipartite network projection and personal recommendation. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 76(4):046115+, 2007.

[200] Cai N. Ziegler, Georg Lausen, and Schmidt T. Lars. Taxonomy-driven computation of product recommendations. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, CIKM '04, pages 406–415, New York, NY, USA, 2004. ACM.

[201] C. Lawrence Zitnick and Takeo Kanade. Maximum Entropy for Collaborative Filtering. In David M. Chickering, Joseph Y. Halpern, David M. Chickering, and Joseph Y. Halpern, editors, *UAI*. AUAI Press, 2004.

[202] Vincent S. Zuber and Boi Faltings. OSS: A Semantic Similarity Function based on Hierarchical Ontologies. In *Proceedings of IJCAI 07*, pages 551–556, 2007.