Data, learning and privacy in recommendation systems

Nupur Mittal

ANNÉE 2016

UNIVERSITÉ DE RENNES 1

UNIVERSITE BRETAGNE LOIRE

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Bretagne Loire

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Informatique

Ecole doctorale Matisse

présentée par

Nupur Mittal

préparée à l'unité de recherche Inria Rennes - Bretagne Atlantique
Institut national de recherche en informatique et en automatique
Université de Rennes 1

DATA, LEARNING & PRIVACY in RECOMMENDATION SYSTEMS

Thèse soutenue à Rennes
le 25 Novembre 2016

devant le jury composé de :

Esther PACITTI
Professeur à l'Université de Montpellier / rapporteur

Pascal MOLLI
Professeur à l'Université de Nantes / rapporteur

David GROSS-AMBLARD
Professeur à l'Université de Rennes 1 / examinateur

Erwan le MERRER
Chercheur à Technicolor Rennes / examinateur

Anne-Marie KERMARREC
Directrice de recherche à Inria Rennes / Directrice de thèse

George GIAKKOUPIS
Chargé de recherche à Inria Rennes / co-directeur de thèse

Acknowledgments

This thesis is a result of the efforts of many people. First of all, I would like to extend my gratitude to all the jury members for accepting to be a part of my work's evaluation. I am very thankful to Prof Esther Pacitti and Prof Pascal Molli for accepting to report on this work, and Prof David Gross-Amblard and Erwan le Merrer for being the examiners in the jury. Special thanks to my co-supervisor George Giakkoupis. Working with him has definitely imparted me a different outlook towards problems and techniques for finding their solutions. Other than being a constant motivation, I also thank George for letting me steal his Latex tricks, which have been extremely helpful in writing this thesis.

This work would have been a mere concept if it were not for the efforts of my supervisor Dr. Anne-Marie Kermarrec. She has been an inspiration not only as a mentor but also as a person. I would like to thank her for her efforts, patience, and time over the past three years. I thank her for replying to all 1,432 of my emails, however irrelevant and unimportant they were. A complete statistical analysis of the email exchanges can be found in Appendix C.

French immigration and visa department hold special credits in my work. Thank you for pushing me beyond my patience and tolerance boundaries. If it were not for you, and the huge pile of documents submitted almost monthly in your offices, I wouldn't have been able to kill 20 extra trees.

Finally I would like to thank my friends and family who have shown their immense support. Special thanks to Samuel, without you and your car, I wouldn't have been able to spend the 12% of my time traveling and eating. I must thank Arnaud for organizing enough game nights to consume around 2% of the time of past three years. The strategy that I made during games might not have helped me win the games against you, but they certainly helped in my thesis. I would like to conclude this by thanking my parents and my siblings back home in India, for trusting in me and my decisions and for their constant support. For all the 8 food parcels that you sent me in the past three years, with the spicy food that almost burnt some of my colleagues.

Thank you for all for your support and patience.

Table of Contents

List of Figures

List of Tables

1

INTRODUCTION

Hiding within those mounds of data is knowledge that could change the life of a patient or even change the world

Atul Butte
Stanford School of Medicine

The last decade has witnessed an overwhelming increase in the accessibility of internet, both at an organizational and at an individual level. In these years, world wide web has grown from being a mere information retrieval/search service to incorporating ideas like knowledge sharing, online social networking, micro-blogging, advanced e-commerce services, opinion sharing, news publishing etc. This new spectrum of services is constantly flooding the www with *data*. This data in not only restricted to the types of products/services and items but also consists of opinions, trust scores, and an individual's inclination towards various topics. Furthermore, the data can be ready-to-use or might require pre-processing and cleansing before it can be used. With approximately 1.09 billion daily active users on Facebook [62], over 488 million products sold on Amazon USA [73], and around 500 million tweets per day [4], one can easily get the idea of the "data-explosion" taking place on the world wide web.

The above numbers confirm the notion that the rate at which data is generated on internet is far greater than the rate at which an individual or even an organization can consume it rationally, i.e., within acceptable time limit constrained by computing resources. With so much information all around, it is not greedy to affirm that a user will need more time to find the information relevant to her needs. For example, a simple search on Google for recipes with cheese results in around 74,500,000 results. A user looking to select a comedy movie to watch, on IMDB is returned with around 72,938 options as shown in Figure 1.1. Therefore, with this exponentially increasing amount of data comes the challenge of facilitating *timely access* to the information *relevant* to the user. In other words, there is a desperate need of *filtering* and *efficiently prioritizing* the data available on internet, *customized* to users' needs to overcome the side-effects of the phenomenon of data-explosion. In a situation like this, involving vast amount of data to output selected customized information, *Recommendation Systems* come as a rescue.

Figure 1.1 – Searching for comedy movies on IMDB results in 72,938 *options.*

Recommendation Systems, initially targeting to combat the data-explosion, surfaced as an independent research area in the 90s [14, 70, 104]. They originated to mimic the idea of "word-of-mouth", where individuals generally seek each others' endorsements and experiences over routine tasks. It was only the initial idea behind recommendations and it later became the base for what is now known as *collaborative filtering* : the most used technique of recommendation systems. Over the years, recommendation systems have drastically evolved to grow beyond the "word-of-mouth" concept. Instead, recommendation systems leveraged other content available vastly over internet, like description of products, user attributes and so on. Regardless of the technique used to make recommendations, they aim to provide a filtered list of items (products, services, opinions, reviews etc.) *personalized* to a user's requirements. Generally, this filtered and personalized list is *ranked* according to a *predicted* user preferences.

They quickly grew in popularity, soon becoming one of the most researched areas and attracting applications from all over the internet. Attributing to their novel idea of information filtering and returning customized results, they were indisputably loved by the e-commerce industry. As a consequence, recommendation systems were, in no time, employed on all major e-commerce portals like Amazon, IMDB, Netflix, CD-NOW, Last.fm to name a few. The popularity and use of recommendation systems continue to evolve, along with its tools and techniques. From having an easy ac-

Figure 1.2 – Making recommendations is like passing the data through a sieve.

cess to the large data/information to be filtered to presenting the personalized filtered results to the user, recommendation systems involve various tools, algorithms and techniques, undoubtedly spanning multiple fields, like machine learning, data-mining, information retrieval and natural language processing. Depending on the tools and techniques involved in the working of a recommendation system/engine, they can be classified into various categories. Different categorization and subsequently sub-categorization [10, 25, 118, 122] is available in the rich literature of recommendation systems and we chose to follow the one shown in Figure 1.3.

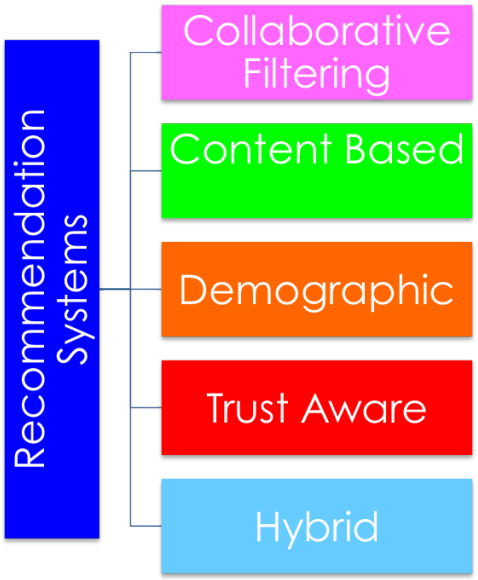


Figure 1.3 – Broad categorization of recommendation system techniques.

Process of recommending, by a recommendation system, can be easily pictured as a careful and accurate combination of various phases namely: *data acquisition*, *data processing*, *data modeling* and *data interpretation*. Numerous techniques to achieve each of these phases are researched upon constantly. The changes in them are directly inflicted upon the performance of the recommendation system. While it is difficult to focus on all of these phases and techniques involved in them separately and exclusively, we cover some of them in the subsequent chapters of this document.

Recommendation Systems as intended originally, do help the internet users to filter and access useful information from vast data mounds but this is not their only feature. One of the significant reasons for their growing popularity is their ability to include *serendipity* in a user's search, while giving her access to useful and relevant information. This feature of including new, surprising results, is as beneficial to the service providers as it is dear to the customers. For customers, it acts as a tool to discover hidden desirable products that they don't know about. It helps the service providers to expose new products and services to targeted customers who, otherwise, will not find the product. Showing these new products only to targeted users is another feature of recommendation systems and is called *personalization*. It means that the system takes care of the preferences of each of its users individually and provides them with what they like the most. Attributing to these unique features, recommendation systems have become very popular and find their applications in almost every field. Online social networks use them to make the network : predicting desirable connections for users. They also use recommendations to spread information in the network, by exposing new information to users who will find it interesting and will contribute to spread it forward. Similarly, they are preva-

lent in all e-businesses where number of new products are increasing exponentially. Search engines have adapted themselves to include personalized search results based on recommendation techniques, rather than relying only on filtering mechanisms. They find them equally advantageous in domains like loyalty programs where recommendations are made pertaining to specific programs, redemption suggestions and upgrade schemes. Some other application areas include email campaigns, product discovery and dating services.

To conclude the applications, relevance and popularity of recommendation systems, we quote CNN money:

> The Web, they say, is leaving the era of search and entering one of **discovery**. What's the difference? Search is what you do when you're looking for something. Discovery is when something wonderful that you didn't know existed or didn't know how to ask for, finds you.

1.1 Challenges in Recommendation Systems

The most popular technique of recommendations, collaborative filtering, roots to the simple idea of relying on others' opinions and reviews about products and services to serve users who are new to these products and services. There are several ways of using this information, which also forms the basis of the categorization of recommendation systems, i.e., user-user or item-item collaborative filtering or knowledge-based etc. Though depending on the category of the recommendation system, the challenges in its design, approach and usage vary yet there are some basic challenges common to all of them. For example, while limited analysis of the content is among one of the major problems faced by content-based recommendation systems, the sparsity of ratings concerns only the collaborative recommendation systems. On the same note, new user problem is more or less common to the both of them, and also to the other recommendation systems, like demographic or hybrid. Attributed to the widespread use and promising results, recommendation system is a highly researched field. As a consequence, a lot of work has been dedicated to outline the issues and challenges in the same [7, 89, 101, 128, 134, 141]. Tracing to the multi-disciplinary and evolving nature of the recommendation systems, the diverse tools and techniques involved in their design, development and implementation, we can surmise that their issues and challenges bear the same nature : of varied nature, spanning different domains and continuously unfolding. Therefore, the challenges involved in the field of recommendation systems are not confined to the ones elaborated in previous works [7, 89, 101, 128, 134, 141] and are far more widespread in nature and hence, it is very difficult to mention them all. There is a very explicit and elaborate mention of the challenges in recommendation systems by Ricci et al. [128] with several pointers to many of the other works, in a well categorized fashion. It is of no importance to reinvent the wheel, therefore, in this work we will identify some of the crucial sub-challenges from these major challenges and propose efficient and carefully designed solutions to them. While outlining these challenges, we incline them towards the four phases constituting the design of recommenda-

tion systems.

1.1.1 DATA: Too much or Too little? Too raw or Too processed?

From the brief introduction and background above, it is evident that the data and information about users and products involved in a recommendation system are of utmost importance. *Without data, there is no learning and without learning, there is no recommendation.* While at the same time, in this era of *big data*, too much data without proper insight into it can lead to confusion and hence wrong approaches for learning and analyzing it. Furthermore, there are challenges concerning the forms in which this data can exist. It can range from personal opinions and sentiments to user *attributes* like age or gender, in plain text, geo-location data, encrypted, scraped in form of xml etc. Recommendation Systems are as dear to the consumers as they are to the service providers. In order to give the consumers the best of recommendations and to render the providers with the best of profits, recommendation systems have to be fed with the right *amount* and *kind* of data. With recommendation systems spanning almost every domain, and being used by almost every internet based service and application, it is a challenge to filter and feed them with the right data. Therefore, we consider it one of the basic, yet important issue. Now we will discuss the two **Q**s of the data : *Quantity* and *Quality*.

Use, do not abuse... neither abstinence nor excess ever renders man happy.

1. **Quantity** Regardless of their type, all the recommendation systems rely on data obtained directly or indirectly from users and/or service providers. In the scope of this thesis, we will majorly focus on the three principal types of recommendation systems: collaborative filtering, content-based and hybrid, unless specified.

   It is a well known fact that any machine learning model needs a lot of data to learn from, to perform efficiently. Recommendation system is no exception, as it generally relies on a machine learning based data-learning model to learn from the past user behavior or from the natural language text chunks (in case of content based systems). This requirement of recommendation systems is burdensome to fulfill. Small e-commerce services or applications with a limited user base are the ones mostly concerned. On the contrary, big players like Google, Amazon, Facebook, IMDB are known for their recommendation systems, the credit of which can be, in part, attributed to their huge user bases.

   Hindu philosopher Chankya said *even nectar is poison if taken to excess.* So is the case with the excessive data in recommendation systems. One very careful parameter in recommendation systems is to deal with user preferences, which is reflected from the collected data. User preferences continue to change and hence a careful decision and choice needs to be made to avoid using outdated data, as it can lead to unsought learning, and hence bad recommendations. citation This problem is visibly aggravated when the size of collected data increases. The larger the size of the data, the more are the problems to store, to process and to analyze it.

   In conclusion, we need good amount of data to feed to the recommendation systems: not too big, not to small. But the data available is not always in this required quantities. It is therefore a challenge to alter the size of data to suit the recommendation needs.

2. **Quality** A lot of recent works shed light on the significance of quality of data, specially relating it to the practicality of recommendation systems [11, 21, 130]. While most of the collaborative filtering recommendation systems use the data available in the form of user preferences: affinity of users towards a particular item, most of the content-based and

Figure 1.4 – Quality and Quantity are two important aspects of data in recommendation systems.

demographic recommendation systems employ the user (and/or item) attributes : like users' name, age, gender etc. With the increasing popularity and use of services like news feed, micro-blogging etc, a new range of data content is available in the form of natural language chunks. Content based recommendation systems techniques are constantly evolving to make use of such data. With such increasing diversity in the nature of data, the issues of data quality become more prominent. Data quality can have several aspects related to it. It can be erroneous, or with incomplete details. For example, dataset containing ages of only 60% of its users. Similarly, data can be either too raw and hence not directly usable or too processed, i.e., very difficult to extract any kind of information from it.

In conclusion, data quality is an indispensable aspect to deal with in the context of recommendation systems. There are various aspects that can be associated to the quality of data, and it is generally impractical to consider them all at the same time. Therefore, in this work, we orient the quality of data in terms of preprocessing required to make it useful and powerful.

1.1.2 HOW TO LEARN

The heart of any recommendation system is the black box to which the data is inputted to have predicted ordered outputs. While *designing* the black box is undoubtedly an arduous job and

Figure 1.5 – Heart of a recommendation system is the black-box that allows it to learn.

requires a competitive expertise in the field, even *choosing* the appropriate technology for the black box is very demanding. It is a result of the advancements in the machine learning field, opening a whole new world to the possibilities. Without loosing a sense of generality, these black box techniques largely depend on the type of recommendations desired in the output

and/or its related applications. Still there are many cases where more than one technology is suitable and hence leaves the designer in the dilemma. Each of these choices have their own pros and cons and therefore, it mostly becomes a choice dear to the designer. It is very difficult to answer this question of “How to learn? ".

Some of the most common learning techniques (and algorithms) for collaborative filtering recommendation systems include *Matrix Factorization*, *Singular Value Decomposition*, *K-Nearest Neighbors* and *non-negative matrix factorization* [7, 10, 123]. On a similar note, *classification* and *probabilistic learning methods* from machine learning are very popular for content-based recommendations [99]. *Clustering* and *Association Rules* can be used for both the recommendation techniques, depending on the type of data available [7, 97]. There are various parameters to keep in mind before choosing one or the other. Weather the computation for the recommendation process is performed online or not plays an important role in choosing the right category of the technique. Another significant factor is the *density* of data involved in the computations. Some of the techniques are more suitable for high density data as compared to sparse data and vice-versa. In addition to these, computation time, ease of implementation, recommendation quality and fault tolerance are other relevant parameters to be considered.

As pointed out earlier, the learning and modeling techniques like KNN, Matrix Factorization, Decision Trees, Support Vector Machines and others are well studied in literature. In fact each of the above technique is a full-fledged research area in itself. They have evolved progressively over the years, and still continue to do so. We can easily reason that it is not in the scope of one single work to focus on all these techniques. Their applications and areas that they serve in are not limited to recommendation systems. In fact, recommendation systems are only one of the prominent areas of their use.

In a nutshell, considering the depth of these modeling techniques, we chose one of them to study, analyze and improvise in our work. KNN is a very popular technique in the scope of recommendation systems. Even though it has been already investigated a lot, yet there is a significant room for improvement, more due to its widespread use, and ease of implementation. Therefore, in this work, we will mention and give a background for some of these techniques but will elaborate in depth only KNN, in-light of recommendation systems. Considering the numerous variations of each of these approaches in the last decade, we will restrict to the basic fundamentals of these.

1.1.3 PRIVACY

Recommendation systems have gained popularity with consumers and service providers alike. When designed efficiently, they can be extremely powerful tools for personalization and customization. Though, in addition to the efficient design, the input *data* plays a significant role. Personal details, accurate traits, exact opinions and preferences of users are among the most desired information to make these personalized recommendations. It has become a tendency of the service providers to often ignore the risks and *privacy* breaches involved in gathering and processing the users' personal data. Moreover, many users are not sufficiently aware of the privacy threats that they might be exposing themselves to. Two main factors responsible for this are: 1) Not being fully aware of all the data that is being collected. 2) Under-estimating the power of the collected data, and the learning models. For example, a user using Amazon to buy a camera might not be aware that his browsing history is being saved, even if she has not brought the product. Similarly, a user writing a review about a product she purchased might not be aware that, for example, the review combined with her personal details like gender and age, can reveal her city or any other unknown and undeclared information about her. The service providers, in turn, take advantage of the users' ignorance, by ignoring to explicitly mention

the kind of data they are collecting or the way the collected data is used. Also users' data can be sold to third parties while the user is not aware of that, though not all privacy breaches are deliberate. Some of them are accidental and unintentional. In addition to the privacy breaches of a user caused by data collection, there are other ways that their privacy can be intentionally or unintentionally compromised, without their knowledge. The manner of storing data is very crucial to safeguard the privacy of users. We often hear of data being stolen, publicly released and so on. These are some examples of users' privacy being compromised.

The word *privacy* has many meanings associated to it. In the domain of internet systems and applications, it is mostly associated to *information* privacy [84]. In the simplest of terms, it implies keeping the user information *confidential*, though it is not limited to confidentiality. In a more detailed definition, it focuses on the individual concerned by the information, and the effects that this information's disclosure can have on the individual, and also on her control and consent over these effects [81] Moving away from the e-commerce services, the term *information privacy* starts to mold into other forms. For example, concealing users' identity or hiding their religious, sexual, political or other such interests. Using social-networks, users share a lot of information, mostly personal, either knowingly or unknowingly. The information shared in terms of photos, statuses, opinions, and emotions can be all used against the users' privacy. Sometimes the users are not aware of how this sharing can harm them while on the other occasions, they do not have much of a choice, other than abstaining the use of the social-network completely. We will shed more light on this in our later sections, with examples.

Jeckmans *et al.* discuss various privacy concerns and their implications, in recommendation systems in details and also categorize them. They also mention some research advancements for combating these privacy threats. It becomes clear from their work that there are many ways of studying privacy issues in recommendation systems and it is not a viable option to study them all at once. Therefore, we chose to focus on the privacy in recommendation systems applied to online social networks. Recalling the classification of recommendation techniques from Figure 1.3, we can see that this comes under trust-aware recommendation techniques.

1.2 Contributions

We would like to mention that we do not follow a particular order for presenting the contributions in the subsequent chapters. To help the reader easily find the information; here is an organization map of our contributions.

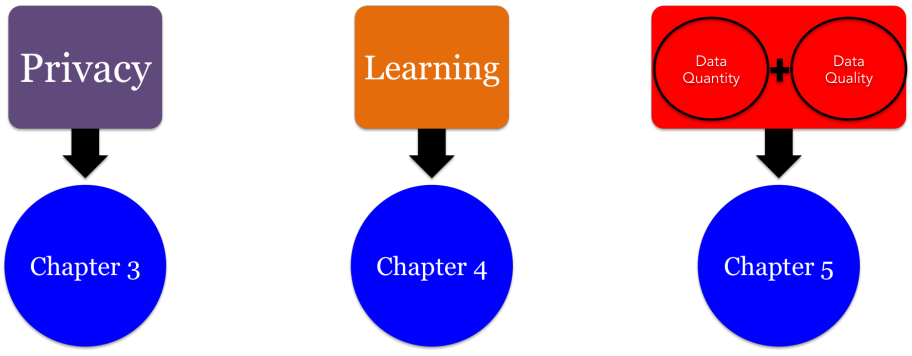


Figure 1.6 – Organization map of the contributions

1.2.1 PRIVACY OF USERS

RIPOSTE is our contribution to deal with one aspect of users' privacy in Online Social Networks, in a recommendation system setting. Recommendation systems play an important role in OSNs; to build the network, to spread the information in the network, and to enable users influence one another. From building the network to every step involved in spreading the information, showing trust on other users, influencing other users and choosing the people to get influenced from, every step involves privacy risks for the users. Spreading information in a social network is similar to proposing users with new items in a recommendation system. In a setting like this, it is of utmost importance to chose the information/item that the user will like. On the same note, it is also equally important to hide the information/items that the user will dislike so as to provide the users with a pleasant social networking/browsing or shopping experience. While ensuring to fulfill these recommendations related requirements in OSNs, it is very crucial to keep a control on the various privacy threats that the users can face. Among others, one of the common privacy threats (very common to microblogging services) is to be convicted for spreading ideas/information. Freely spreading information is the crux of social networks and microblogging services, like Twitter, LiveJournal, Facebook, Google+ etc. A lot of people are using these platforms as a medium to share their ideology, to disseminate news, political opinions, and their outlook on other pivotal topics. While the essence of these services is enabling free dissemination of information and ideas, yet it has been often noticed that several users fear and refrain from openly expressing their views on various subjects. This comes mostly from the fear of being prosecuted and convicted; a practice common in countries with authoritarian regimes, though existent even in countries with democratic regimes. Fundamentally, this obstruction in the dissemination of ideas and information in OSNs come from *privacy considerations*; a user in a favor of a particular idea chooses not to spread it fearing a breach of her privacy.

With RIPOSTE, we aim to fulfill these requirements of social networks (in a recommendation system setting) keeping in mind the privacy threats posed to the user. With this privacy preserving, information diffusion algorithm, the objective is to maintain the users' influence and let them freely spread information while preserving their privacy using the principal of *Plausible Deniability*. RIPOSTE is a distributed algorithm for disseminating information (ideas, news, opinions, or trends) in an Online Social Network. It also guarantees not to disturb the essence of the social networks. Hence, it makes sure sure that the information spreads widely if (and only if) a large fraction of users find it interesting. Also, if only a few users find the information interesting, RIPOSTE makes the information dissemination process die quickly, not to spread the information to a large fraction of network, and all this is done in a privacy preserving manner.

We provide our proofs to our theoretical foundations and support them with extensive experiments, taking snapshots of real-topological social networks like Facebook, Twitter, LiveJournal, Google+ among others.

1.2.2 LEARNING MODEL

In one of the techniques, namely *Collaborative Filtering*, recommendation systems analyze the patterns of users' interest in items and products to provide personalized recommendations of items that will suit a user's taste. In other words, in a collaborative filtering technique, the recommendation system is trying to predict a user's opinion on different items and to suggest her the *best* items based on her previous *likings* and on the opinions of other *like-minded* users. One of the challenging tasks here is to find *like-minded* users, or the users *similar* to each other. This is a classic problem in literature and is called *K-Nearest Neighbor* (KNN) graph construc-

tion [64, 146].

To understand what KNN is, we will draw parallelance from getting recommendations from friends. We choose people with same taste as ours and then ask them to recommend us something, lets say a book. If a majority of them suggests the same book, we will deduce that that we will also like this book. KNN is the algorithmic translation of this simple intuition. In the simplest terms, a KNN graph is a directed graph where each vertex is connected to its k most *similar* vertices. KNN algorithm finds its use in wide variety and range of applications and can be used in the black box of recommendation systems as seen in Section 1.1.2. It is used as a learning model by finding k most *similar* users (based on their likes) for each user in the dataset.

Although KNN is a widely used and studied algorithm, there are some challenges in constructing an efficient KNN algorithm. With all the data available from internet and its applications, it is not surprising to see datasets with more than billion users and/or items. In such a scenario, due to the intuition of comparing each pair of users, computing KNN quickly and efficiently is one of the crucial challenges. Other important issues include finding a *good* value of k. If the value of k is too small then the algorithm tend to include outliers whereas in case of a very large value, the algorithm risks to have irrelevant neighborhood. Finding a way to measure *similarity* between users is also a tedious but crucial task as the performance of KNN is highly dependent on the choice of this similarity measure. Although various measures can be used to compute the distance between two points, the most desirable similarity measure is one for which a greater similarity between two users implies a greater likelihood of being closer.

To tackle the above issues of KNN graph construction efficiently, we propose KIFF: a generic, fast and scalable KNN graph construction algorithm. KIFF directly exploits the bipartite nature of of most datasets to which KNN algorithms are applied. This simple but powerful strategy drastically limits the computational cost required to rapidly converge to an accurate KNN solution. We chose a wide range of datasets to perform experiments to support the results of KIFF being considerably faster and scalable than the other state-of-the art approaches. Our evaluation on a representative range of datasets show that KIFF provides, on average, a speed-up factor of 14 against recent state-of-the art solutions while improving the quality of the KNN approximation by 18%.

1.2.3 DATA COLLECTION, CLEANING AND COMPREHENSION

Data is a very important part of all web-based applications, and not only recommendation systems. We present an empirical study focusing on highlighting the power of user data. With this big picture in mind, we also show how combining information from various sources can reveal interesting unknown information about the users. This revelation is an important milestone for recommendation system applications, specially for the systems that often suffer with the problem of lack of user data. To investigate on the power of inference of various user attributes, we collect data from three different social networks. Designing crawlers and scrapers is itself a challenging task, mostly because of the limits imposed by these services. Bounded by these limitations, we only extract data enough for our research purposes. Due to privacy policy of these services, we cannot release the datasets publicly. Another challenging task is to cleanse the data, as many of the values are in a format that is not suitable for inference purposes. We detail the cleansing process used for each of the attributes, focusing on tweet processing and hashtag extraction. To conclude this work, we present the results of inference obtained by considering different combinations of attributes. Our empirical investigation shows that information in form of hashtags and trending topics from twitter can be a used a reliable source of inference, specially for predicting user's interest in movies, music and books.

In essence, our work provides a complete insight into the process of data collection and cleansing to use it to infer unknown user information, collected and combined from different information sources. We claim that they results can be easily extended when the sources of the information are changed.

1.3 Thesis Plan

The following chapters lay background and give details for the contributions made as a part of this thesis. Chapter 2 provides detailed background study and knowledge of the state-of-the art for each of the contributions made. In addition, it provides clear definitions of the notions used in the later chapters. Later in Chapter 3, we elaborate our contribution for the challenge of privacy issues in online social networks - RIPOSTE, from a perspective of recommendation systems. Chapter 4 explains the technical details and design idea of our contribution for learning model in recommendation systems, which we call KIFF. Our last contribution is presented in Chapter 5, dealing with various aspects of data in the field of recommendation systems. We conclude our work in Chapter 6 giving some insights into future work.

2
State-of-the Art

A successful person is the one who can lay a firm foundation with the bricks others have thrown at her.

David Brinkley

Recommendation Systems have attracted the attention of researchers across many domains like data mining, information retrieval, large scale social networks, distributed computing etc. One of the most significant facet for this attention can be attributed to their *multi-disciplinary* nature. Different components of recommendation systems, like data, learning model, prediction mechanisms and evaluation metrics, require understanding and knowledge from different fields. In addition, the ease of use and widespread scope of their applications gives them even more choices for the tools and techniques that can be used in their design and implementation. As a result of this varied nature of recommendation systems, the challenges in their design and implementation are also manifold. Depending on the type of recommendation technique used, its application, and availability of data and resources, there are numerous aspects to deal with, both in research and implementation. It is not surprising that within the scope of this work, we cannot cover all the aspects of recommendation systems. Therefore, we select three areas to study and propose solutions to the major problems in each of them.

In this chapter, we start by laying a foundation for a basic understanding and background for recommendation systems, their main components, and popular techniques used to design them. Afterwards, we discuss in details the three areas of recommendation systems that we have chosen, namely: *Privacy*, *Learning Model*, and *Data*. For each of these areas, we follow a systematic approach of starting with the basic introduction to the problem, and then moving towards more precise problem definition. In addition to this, we also highlight the remarkable contributions in each area from the rich literature of recommendation systems.

2.1 OVERVIEW OF RECOMMENDATION SYSTEMS

Recommendation systems are information filtering systems that deal with the problem of data explosion by using filtering important information from the overall data available to the system using user data, item data, user opinions, preferences, meta-data, demographic information, user behavior or combination of these [86, 127, 128]. Over the years, recommendation systems have evolved to extend beyond the concept of relying on similar user behavior to make predictions as opposed to the original idea of recommendation systems that surfaced in the mid-90s [78, 126, 133]. As we have mentioned earlier, recommendation systems witness a huge diversity in the tools, techniques and algorithms that they can use. The choice of these is guided by various factors, including the target application of the recommendation system, availability and type of data, computational resources etc. Before discussing the most popular recommendation system techniques in details, we use notations to mathematically define the idea behind recommendations.

2.1.1 Notations

In the present age of internet, personalization and commercialization, the idea of making a recommendation is simplified to making a prediction (in form of *ratings* about an unknown and unseen item by a user. [1] The data and techniques used to make these predictions are generally the leading factors in categorizing recommendation systems into different categories. Regardless of the categorization, here we present some basic notion about recommendation systems.

Let us consider a system S, that has a set of *users* $\mathbb{U}$ and a set of *items* $\mathbb{I}$. The users $\mathbb{U}$ of the system can have associations in $\mathbb{I}$. Such associations are represented by *ratings* and are usually a *quantitative* representation of a user $u's$ inclination for an item i in the system. In other words, *ratings* are a means to measure the utility of an item i for a user u.

$$r : \mathbb{U} \times \mathbb{I} \rightarrow \mathbb{R} \tag{2.1}$$

Here we consider that $\mathbb{R}$ is the domain or set of possible ratings and is restricted to non-negative integers only. In general, ratings can be either explicitly given by the users for various items, with the use of stars, non-negative numbers, and in some cases, they can also take adverbial form. If the ratings are not explicitly stated by the users, the system relies on implicit ratings, which can be users' navigation history, purchase record, interaction patterns etc., depending on the system.

Logically, the maximum value of the above utility function (represented as ratings) will mean that a user is very interested in the recommended item. In other words, the recommendation systems should recommend items $\hat{i}_u$ that a user u will be very interested in, for all the users in the system. This is mathematically represented as:

$$\forall u \in \mathbb{U}, \quad \hat{i}_u = \max_{u \in \mathbb{U}} r(u, i). \tag{2.2}$$

We use the notion of *profiles* to characterize the elements of set $\mathbb{U}$ and $\mathbb{I}$. The profile of a user consists of information related to her, for example, it can contain the items viewed and rated by the user, or it can consist of user's demographic information. Similarly the profile of an item is a representation of characteristics of an item. In this case, it can consist of users who have liked that item, or item features like price, size specifications etc. For example, in a music recommendation system, a user's profile can be composed of the songs that users has liked in

1. Although this is not the only idea behind recommendation systems but we will only use this in the scope of this thesis.

the past, and an item's profile (a song in this case) can consist of album name, singer, genre, release year, language etc.

A recommendation system predicts the *ratings* that a user is likely to give to an unknown item. To propose a user with a reasonably long list of items that she is estimated to like, a recommendation system postulates the ratings for several candidate items and only the items with best ratings are recommended. This postulation can be performed in several ways, for example using the *rating matrix* or using the similarities between items' characteristics, among several others. Before delving into the details of these techniques, we will underline the idea of a rating matrix. It is a matrix representation of a set that contains the values of ratings that the users have given to the items of a system. $r_{(u,i)}$ is a rating value given to item i by user u. One example of rating matrix is given in Figure 2.1.

Movies / Users	Fight Club	Inception	Chinatown	Maze runner
Alice	4	3	5	1
Bob			4	4
Carl	3	1	5	
Dave	5			

Figure 2.1 – Rating Matrix for a movie recommender system. The empty cells signify that the corresponding user has not seen the movie.

Another important aspect associated to recommendation systems is the notion of *similarity*. Depending on the type of recommendation system, this notion can vary in what it measures and the concept used behind that measurement. For example, the similarity between users is measures differently than the similarity between items. Also, there are several different ways of measuring similarity between users, depending on many factors, one of which is the application domain. We will discuss about the similarity measures with different types of recommendations in the following paragraphs.

2.1.2 Recommendation System Techniques

We will follow the classification of recommendation systems from Figure 1.3 present in Section 1.

2.1.2.1 Demographic Recommendation Systems

As the name suggests, demographic recommendation systems rely on a user's demographic information to find similarities between them for the purpose of making predictions. Such demographic information includes age, gender, location, education, marital status etc. and is mainly used to identify types of users. The underlying assumption of such systems is very simple. The existing users of the systems are first put into different groups based on the similarities of their demographic information. Later a new user in the system is directed to one of these existing groups depending on the demographic information of this new user. In a nutshell, the demographic information of users is used to identify the types of users that like a certain item. For example, in a system with people speaking varied languages, one of the groups consist of Spanish speaking males between the age range of $30-39$ years. It is also known from the existing data that all these users like "paella". Now if the system has to recommend a dish to a new spanish speaking male user, the system uses the previous data and suggests paella to this

new user. This is the simplistic idea behind demographic recommendation systems. Often in systems making use of demographic information of the users, one or more attributes are combined to make these groupings. It is difficult to find practical recommendation systems entirely relying on the demographic information but they are proved very influential in marketing and advertising domain [103].

We present a simplistic architecture of demographic recommendation systems in Figure 2.2. In such systems, the user profiles consist of the demographic attributes and it is assumed that users with similar values of these demographic attributes will rate the items of the items similarly. Therefore, clusters of users are formed for providing recommendations to a new user with known demographic attributes. Evidently, this solution of recommendations implicitly overcomes the problem of *cold-start* in recommendation systems.

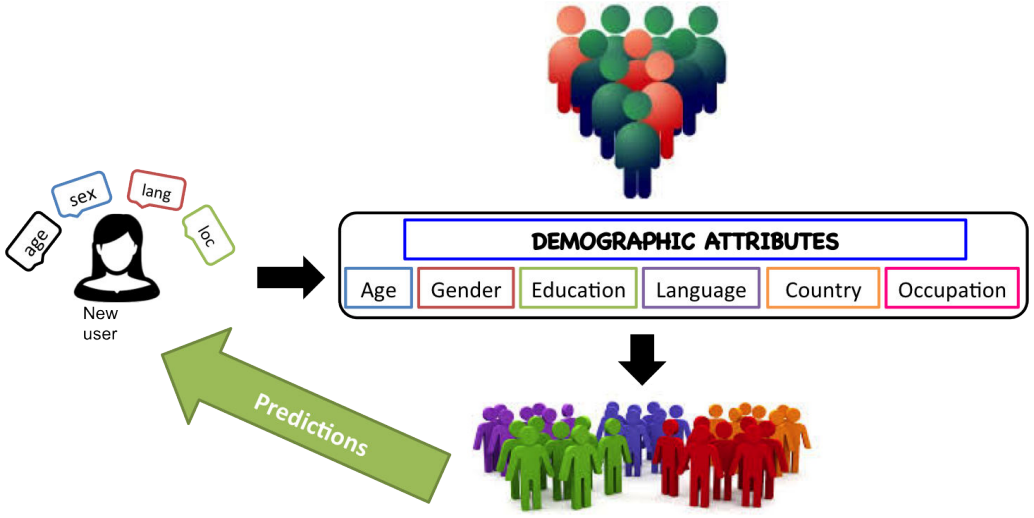


Figure 2.2 – Simple architecture of a demographic recommendation system

One of the major challenges involved in such a technique is difficulty in obtaining the demographic information about users. Users often tend to keep this information private over websites, internet platforms and applications or not disclose at all. At the same time, users might not like to be recommended items based on their gender, language, or ethnicity. For example: Geeta is offended as she is recommended the movie “Parched" based on her Indian origin, the language Hindi and that she is a female.

In conclusion, demographic recommendation systems do not provide for practical recommendations mostly attributed to the lack of access to the information they require. They are powerful means of predicting user’s interest and are often used in combination with other machine learning techniques like clustering to provide for more targeted recommendations and advertising.

2.1.2.2 Trust Aware Recommendation Systems

We have seen previously that the growth of internet has lead to the phenomenon of data explosion, which in turn, is one of the leading factors for the popularity of recommendation systems. This rapid growth of internet, as we have seen, has given birth to e-businesses, with a vast choice of products. Not only commercialization but also socialization has been revolutionized with the advent and growth of internet. People communicate with each other, they form social ties among themselves using internet, which can be different than the real social links that they have outside internet. This is a service generally provided by the so-called social networks :

linking people on internet. These networks are platforms for people to connect virtually, share and pursue each others' activities and in many cases share knowledge. Some famous examples of such platforms are MySpace, Facebook, Twitter, Google+, Snapchat etc.

Trust aware recommendation systems transform the classic recommendation systems by making use of *trust* that exists between users. Taking analogy from real life, out of all the acquaintances of a person, there are some that are trusted by her more than the others. In situations that require suggestions and advice, such a user generally tends to follow her *trusted* peers more than the rest. Trust aware recommendation systems are algorithmic transformation of this idea. In addition to the preferences of other users in the system, such recommendation systems also take into account the social ties that the user has, which we consider are a representative of her trust. Therefore, trust aware recommendation systems generate recommendations, which are personalized for each user, using the users' preferences, opinions and trust relationships.

In the scope of this thesis, we only focus on the social networks that can be seen as trust aware recommendation systems. In the simplest setting, a social network provides for directed (e.g., followers) or undirected (e.g., friends) relationships between users of the system. These relationships can represent numerous aspects of the link between two people, depending on the social network in question. For example, Linkedin is a representative of professional ties between individuals whereas Facebook is more a representative of real social connections. Trust is one of the foundations of online social networks and is a widely studied subject [6, 63, 106]. "Friends" on Facebook trust each other to share information like personal photos, comments and statuses. "Followers" on Twitter trust their followees' opinions, news items and other such information. Similarly "Connection" on Linkedin is a representation of trust between two professionals for sharing their professional insights and activities.
From the above discussion, we can claim that the users of the trust aware recommendation systems are aware of the reasons of particular recommendations made to them. They will know the reason for being chosen for a particular recommendation, mostly from their trust relationships, and they will be more confident about such recommendations, as claimed by Prasand et al. [122].

Figure 2.3 – Advantages of Trust aware recommendation systems

2.1.2.3 Content Based Recommendation Systems

From conceptualizing the idea of word-of-mouth into collaborative filtering, to extending it further, internet has been a major driving force behind the advent of all recommendation systems. For most of the recommendation systems, the users interact with the system and are recommended items that they are probable to like. The items can belong to a diverse range, depending on the application or system in question. For example, news items that a user would like, travel destinations that might attract a user, recipes that a user would want to either cook or eat, people that the user would want to be friends with, movies that she will prefer to watch, songs to listen, clothes to buy and so on. The list of applications and services over internet has increased manifold, almost exponentially over the past few years. As a consequence, the number of choices that a user has, or the items in our terminology, has gone beyond the track of a user. When a user is browsing for a movie to watch, it is not possible for the service provider to show all the available movie choices to the user, and is forced to crop the options. One way, among many others, of cropping the choices for each user is by keeping an account of the pattern of items that the user visits. This is the foundation of content based recommendation systems.

Content based recommendation systems use the description of items, available in diversified forms, to identify and select items that can be of interest to the user. In contrast to the collaborative filtering techniques, that keep a track of user's preference of items, content based techniques rely on the information available about items. Therefore, the base of these techniques is the content information available on the items present in the system. Afterwards, the correlation between items is searched to make recommendations using this information. For example, in a news recommendation system, news are the items and can have various information attached to them, like category of the news item, country of the news etc. Figure 2.4 shows a simplistic overview of a content based recommendation system. The source of items is generally the application or the service provider. For example, in a movie recommendation system, items are the movies present in the system. Depending on the item in consideration, either the *features* of the items are implicitly available or are extracted using various tools and techniques. It is possible that in the movie recommendation system, the movies, i.e., the items, already have descriptions available. In IMDB like systems, they are available in form of genres, actors, release year etc. In a news recommendation system, on the other hand, the features of news item may be extracted from the items themselves using complex NLP techniques. The extracted descriptions or features of items are used to identify the items in the system and are used in combination with the user profiles to provide recommendations.

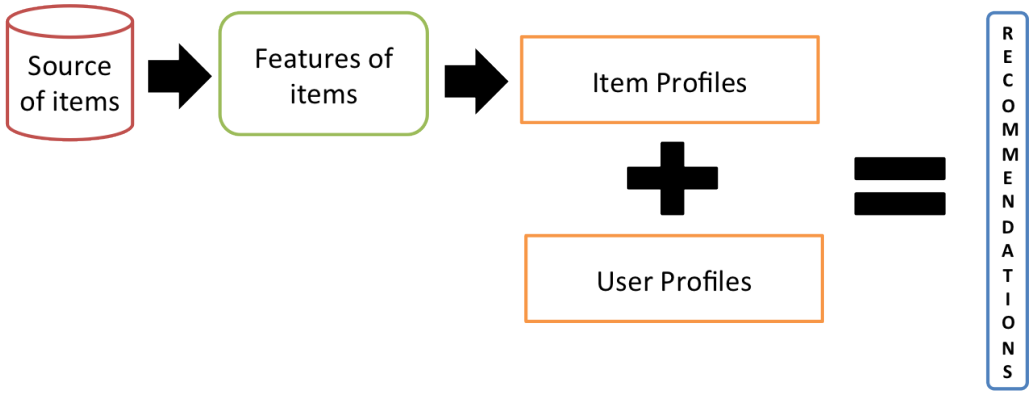


Figure 2.4 – Overview of a content based recommendation system

One of the important concepts of content based recommendation systems is to deter-

mine the importance of items for the users in the system. *Term-Frequency Inverse Document Frequency* is a very popular concept for the same. As the name suggests, TF-IDF is used to determine the *relative* importance of items based on the frequency count. This is the concept that originated in information retrieval systems and is easily applied and exported to content based recommendations systems. In this context of content based systems, for calculating TF-IDF, a *document* refers to an item in the system and *word* refers to the features used to describe the items. For example, in a movie recommendation system, movies become the documents, like God father, Rocky, Fight Club and Pulp Fiction. The words are the features that describe the movies, like comedy, drama, Jennifer Lawrence, Brad Pitt etc.

Term Frequency (TF) : is the number of occurrences of a *word* in a *document*.

Inverse Document Frequency (IDF) : is the inverse of the number of occurrences of *documents* among the whole document collection.

Therefore, the frequencies of item features and the corresponding tf-idf values are used to construct the user and item profiles here which are then used to make recommendations. We will not delve into the details of TF-IDF here as it can be a topic of research in itself. We conclude that information extraction is a very important paradigm in content based recommendation systems. The fineness of information extraction determines the quality of recommendations. As we can see that these system do not require a user's past experiences with other items, it is easy to recommend items to new users in the system, unlike their collaborative filtering counterparts. It is equally useful to recommend new items to the users in the system as the new items do not have ratings associated to them. Using their features, the content based recommendation systems render them more visible to the users who are expected to like them.

2.1.2.4 Collaborative Filtering Recommendation Systems

Collaborative filtering is the most studied techniques of recommendation that has advanced and expanded over the years. In its origin in the mid 1990s, it was the algorithmic conceptualization of the idea of word-of-mouth [78, 126, 133]. It has been widely used in practical recommendation systems, especially the ones in e-businesses, like that for Amazon, NetFlix and Grouplens. It allows to consider the opinions of hundreds and thousands of users on internet than just the acquaintances, while looking for a product or service. One of the main factors behind its popularity is the ease of implementation and its domain-independent nature. As the name suggests, these recommendation systems work in a collaborative manner. In general, the users of the system collaborate to share their inclination towards the items that they have seen in the system. In all it's forms, this inclination, as we have seen earlier, is represented in the form of ratings.
With the aim of recommending a new item to the user based on her past associations and the associations of other users like her, collaborative filtering can take various forms. The broad categorization of collaborative filtering based recommendations is shown in Figure 2.6 [25, 30].

Memory based Memory based collaborative filtering techniques are also called *neighborhood based* techniques [30]. The main motivation behind this kind of recommendation technique comes from the real life usefulness of neighbors for the purpose of making recommendations. As a result, it involves finding neighbors of the target user [2] who share similar interests as her in the items of the system. This similarity of interests is dependent on the ratings (both implicit and explicit) that the users have given to the items. In conclusion, the ratings of the target user and of the other users in the system are very important in finding her neighbors which are then

2. Target user is the user for which the predictions or recommendations are to be made.

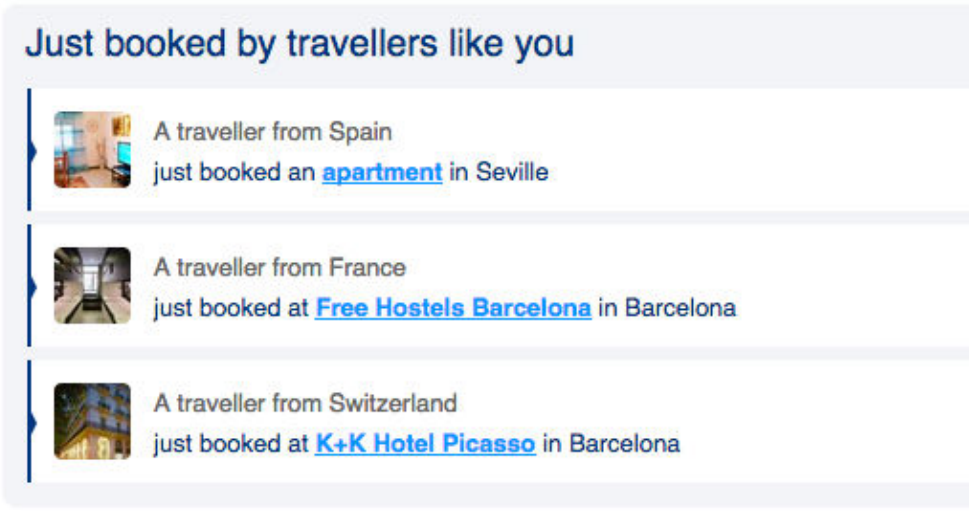


Figure 2.5 – Recommendations based on what other similar users have liked

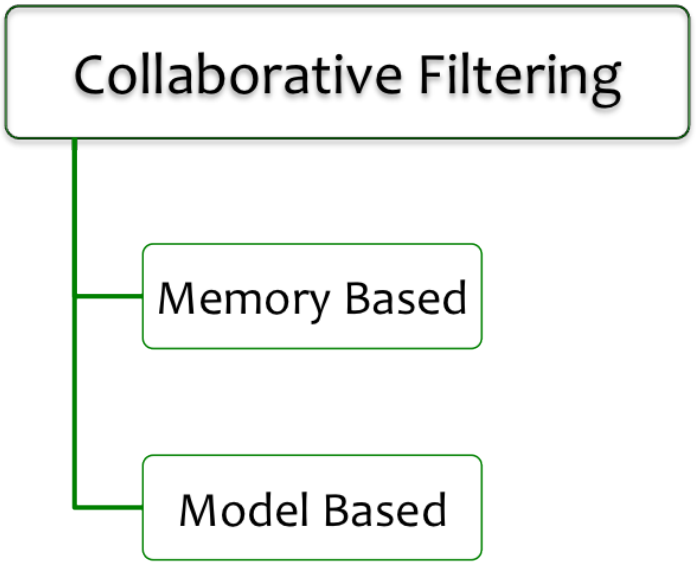


Figure 2.6 – Collaborative Filtering techniques can be further classified into Model based and Memory based techniques.

used for making recommendations. Once the neighbors have been identified, there ratings and associations can be combined in several different ways for making recommendations to the target user. Broadly, there are two such ways called *User-based* and *Item-based* techniques as displayed in Figure 2.7.

User-based collaborative filtering technique uses the similarity between the target user and the other users who have rated common items. The similarties between users can be computed using different similarity metrics available widely in the machine learning literature, cosine and jaccard being some popular ones. Item-based collaborative filtering uses correlation between all the items rated by the target user to find new items that the user will like. Similar to the user-based techniques, they also rely on similarity metrics to compute the correlations and

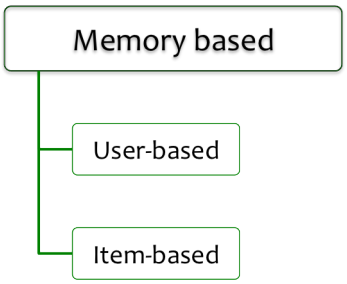


Figure 2.7 – Sub-classification of memory based collaborative filtering techniques.

similarities between items rated by the target user. One of the very popular similarity metric for such item-based techniques is Pearson correlation.

As we have seen, both of these techniques require finding neighbors, users in case of user-based approach and items in case of item-based approach. Systems using recommendation engines involve millions of users and items and therefore, it is only natural that there are many possible neighbors (that have items in common with the target user) for target user. Therefore in a setting like this, we often rely on k best neighbors of the target user, according to the measure of *similarity*. This is called finding k-nearest neighbors and is a classical problem of data mining, recommendation systems, classification systems or of machine learning and is addressed in this thesis also. Before we discuss the background details of the KNN approach in general in Section 2.3.1, we will mention some of the commonly used similarity measures in details here. We will use the notion of user (and item) profile to elaborate on the similarity measures as we have seen in Section 2.1.1.

Cosine Similarity : belongs to the family of inner product similarities as classified by Cha et al. [37] and uses the vector-space model. It is a very popular similarity measure in the field of information retrieval, mostly because of its efficiency in domain independent calculations. For two users u and v, with user-profiles UP_u and UP_v respectively, their cosine similarity is calculated by the inner product of their profiles:

Definition 1 (Cosine Similarity)

$$\text{cosine(u,v)} = \frac{UP_u . UP_v}{\|UP_u\| \|UP_v\|}$$

Jaccard Similarity : is based on the idea of intersection of profiles, and hence is simpler as compared to its counterparts. As it is based on counting of common items between two users, it can be easily used as a dissimilarity metric as opposed to cosine that has no such capabilities.

Definition 2 (Jaccard Similarity)

$$\text{jaccard(u,v)} = \frac{|UP_u \cap UP_v|}{|UP_u \cup UP_v|}$$

Person Correlation : is the measure of linear relationship between the profiles of two users and is based on covariance and standard deviation of the profiles.

Definition 3 (Pearson Correlation)

$$\text{pearson(u,v)} = \frac{covariance(UP_u, UP_v)}{\sigma_{UP_u} \times \sigma_{UP_v}}$$

We can claim that both user-based and item-based techniques are popular and useful in the recommendation system world. As pointed out by Ricci et al. [128], there are five main criteria before selecting a technique between user-based and item-based collaborative filtering; accuracy of predictions, memory and computational efficiency, stability in the number of users and items of the systems, justifiability of the end recommendations and the serendipity of results. Therefore, the type of technique is chosen mainly based on these parameters.

Model Based As the name suggests, model based recommendation systems rely on finding patterns in data [90]. The past data is searched for patterns that can be used for training a model for finding patterns. Mostly the user-item rating patterns are used to train the model. To make recommendations, the model uses the patterns from the training data and apply them on the real data. It is clear from this description that such model based techniques use machine learning techniques to learn from data and then use the learned model to make real-time predictions and recommendations. As a result of using machine learning techniques to model a pattern which is pre-computed for the recommendations to be made, recommendation systems using model based techniques has very rapid performance. Some of the most popular techniques used to design model based recommendation systems are Singular Value Decomposition (SVD), Matrix Based techniques, Clustering based models, Regression techniques etc [87, 90]. It is out of the scope of this work to cover these techniques in details. Figure 2.9 shows a classification of these techniques under three main labels: classification, rule-based and clustering tecniques. One of the advantage of the model based techniques is dealing with the sparsity problem. Our contribution KIFF also exploits this feature of the model-based approaches.

2.1.2.5 Hybrid Recommendation Systems

Hybrid recommendation systems are an amalgam of the above mentioned recommendation approaches. In most of the cases, these systems combine features of collaborative filtering and content based techniques, to have the best of both worlds. The idea behind these systems is to overcome the limitations of various systems and to combine their power. As a result, these systems are inherently more complicated in their design and implementation but they have better prediction results. One of the most famous hybrid recommendation system is Google news recommender system [49]. There are several research works that document the use of two or more basic recommendation techniques to implement hybrid recommendation systems [7, 31, 33, 50]. The complexity involved in their design and implementation make them less favorable for practical recommendations as compared to their collaborative and content based counterparts.

2.2 Privacy

Recommendation systems have been a huge success in the field of commerce and research alike. As a result of their popularity e-business giants like amazon, netflix, youtube and imdb have devoted a huge share of their success to recommendation systems. Similarly, researchers

have spent years in the advancements of tools and techniques of recommendation systems and still continue to do so. Attributed to their success and usefulness, even online social networks have recognized and used recommendation systems in their basic operations. The OSNs rely on recommendation systems for many operations. One of the dominant feature of the online social networks is establishing social ties between its users. This includes both, constructing links that are obvious as well as predicting interesting and surprising social ties between users. As a result of this requirement, OSNs like Facebook and Twitter heavily rely on the predictions from a recommendation system. Seeing their success, with over 1 billion active users on Facebook [62], and users surpassing millions of followers on twitter [4], one can claim that recommendation systems play their role well in predicting social ties in OSNs.

The role of recommendation systems in OSNs is not limited to predicting social ties. In fact, they play a significant role in spreading information across the network, one of the core purposes of social networks. As we have seen in Section 2.1.2.2, social networks can be seen as trust aware recommendation systems where users place trust in each other in form of social relationships - friends in facebook, followers in twitter, connections in linkedin etc. to have suggestions from each other. These suggestions can take different forms in different social networks. For instance, users of twitter follow each other for information recommendation mostly in form of news items, whereas users of facebook establish friendships for photos, personal updates, location check-ins etc. A considerable research work is devoted to the study of trust and influence in social networks [43, 51, 68, 69] that is a significant factor in the origin and spread of information on these OSNs. It is clear from the above discussion that online social networks can be regarded as trust aware recommendation systems where user relationships are based on trust, reflected in form of social ties. These social ties are influential in spreading information over the network.

We consider the example of Digg [3], a voting based social network new dissemination system. In Digg, the social connection between users is called “friendship". Users follow their friends to see the news that the friends post and to see the votes that the friends have submitted. In short, the users follow people that they would like to receive the news from. Same reasoning applies to other social networks, like Google+, twitter and facebook. Once the user receives news from her friends, she spreads the news that she is most interested in to her friends. This spread of news received from friends is done by “voting" in Digg, “sharing" in Facebook and “retweeting" in twitter. As a result, the information originating from one source travels to different parts of the network because of the existence of social ties, somtimes resulting in large information cascades. In their work, Salganik et al. [131] talk about how some stories become much more popular than the others. A lot of research has been devoted to the study and analysis of spread of information in social networks, recognizing the factors responsible for the same. Till this point, we have seen how online social networks are a form of trust aware recommendation systems and their importance in spreading news over the network. Now we will detail how maintaining users’ privacy in an important challenge in such systems. Though there are many aspects related to users privacy in social networks and in recommendation systems, we will focus on preserving users’ privacy while disseminating information such that the information cascade patterns are also maintained.

2.2.1 Privacy threats in social networks

The concept of social networking is not new, in fact it dates back to the 60s [91] but they have seen a viral growth only since the advent of internet. Since the success of the first known social network “SixDegrees.com" in 1997, a lot of social networking sites have emerged with promis-

3. http://digg.com/

ing features. Online social networks, even though a recent phenomenon, have been used and adopted by a very large number of people. They provide near to an ideal platform for its users to publicly share their personal information.

NEW PLATFORM, OLD THREATS

In addition to sharing connections between people, they have revolutionized the process of information generation and sharing. As a consequence, they have become an indispensable tool for disseminating information to large sections of people rapidly. The popularity and these unique features of OSNs do not circumvent the privacy threats that they bring along. In fact, just like their popularity, the privacy threats associated to them have become more prevalent also, resulting in attracting the attention of both academia and industry.

Online social networks are subject to many privacy and security threats, some of them include: privacy threats [27, 109], identity thefts [23, 79], sybil attacks [34, 142], malwares [19, 149] etc. We give a brief introduction to some of the threats here:

Malwares: The wikipedia definition of a malware is a malicious software that is used to disrupt computer operations, gain access to private computer systems or to sensitive information or display unwanted advertising. Malware attacks are the most endemic and classic threats known to computer users, more dominating due to the ease of internet access. These attacks are prevailing attacks in online social networks also because they benefit from the well-connected structure of social networks to spread the malicious software even more rapidly. The high rate of interaction between the users of OSNs and third party applications only ease the process of infection. Koobface is known to be one of the most notorious malware successfully spread through numerous online social networks include myspace, facebook and twitter [19, 135, 148]. The attack has many variants where some of them target the user activity ion the social network. Other similar malwares, albeit not as prevalent and sophisticated as Koobface, has targeted other social networks.

Identity Theft: is identified as one of the most damaging threats in social network as the attacker gets access to a lot of personal information about the user, including username, passwords and therefore, the whole user-profile on the online social networks. Either the online social networks already have all the information related to the user or are capable of inferring it using the known information. Hence, these identity thefts can become even more dangerous when combined with inference attacks. Simple solutions to identity theft include the use of strong password, but such measures are not enough to counter identity thefts. Another hidden motive behind identity threats, commonly seen on recent social networks is stalking. Friends and connections are often a target of the attacker behind these identity thefts. Therefore cautious and prudent use of online social networks is a necessary measure not only to secure your own information but that of your connections also.

Sybil Attacks: is a term used to describe the forging of multiple identities for a malicious purpose [57].

> "ONE CAN HAVE, SOME CLAIM, AS MANY ELECTRONIC PERSONAS AS ONE HAS TIME AND ENERGY TO CREATE."
>
> – JUDITH S. DONATH

Sybil attacks are more profound in peer-to-peer systems and other distributed systems as their architecture gives way to an easier means of attacking as compared to their centralized counterparts. In terms of an online social network, an attacker claims multiple user identities to compromise as large part of the network as possible. Online social networks are based on the foundations of sharing personal information, photos, thoughts and opinions on subjects and emotions with a selected group of people, generally selected based on the trust we place in them. Even if the social ties are partially based on the trust between users, it is not always the case. In 2012, daily mail reported that *half of Facebook users accept 'friend requests' from strangers, while 13m U.S. users have NEVER opened their privacy settings* [138]. Mark Granovetter introduced the term "weak ties" in social networks to refer to social connections between users of different communities [71]. Weak ties are an indication of lesser interaction and intimacy between the users. Sometimes there is negligible interaction once the connection is established. Such associations and connections with strangers render Sybil attacks easily doable in social networks. Such attacks can be used for many harmful purposes, as they can easily modify the "voting" outcome, or poll results for example. Rupesh Gunturu [] gives an elaborate survey of the possible Sybil attacks in social networks, both centralized and decentralized. He also classifies Sybil attacks into various categories depending on how the attacker proceeds with the attack, or the kind of interaction that takes place between the attacker and the victim's profile, etc.

There are many other ways in which online social networks can pose privacy and security threats to its users. There are phishing scams, that can ultimately target to steal identities and compromise on very sensitive and personal user information. Other more sophisticated threats are of de-anonymization. Many a times users of online social networks chose to use pseudonyms or anonymize themselves for various reasons. In reidentification and de-anonymization attacks, the attacker reveals the true identity of the user, thereby breaching user's privacy.

Social networks combined with internet, i.e., online social networks, have become very powerful means of supporting freedom of speech. In their originality, they tend to break the barriers of censorship and have instead become a revolutionized form of activism. Social networking and microblogging services have transformed the face of political activities, coverage of information, spread of ideologies and expression of thoughts [48]. Social media platforms are increasingly used to organize and stage protests, manifestations and strikes, reaching a global participation, all due to the ease of connectivity on such platforms. While on one hand, these OSNs provide protesters and activists a platform to raise their voice, and attract a global audience, on the other hand, they make these activists vulnerable to spying eyes. Countries where the government wants to control people's freedom of speech and expression are specially targeting and spying activists on social networking platforms, to curb their activities. It is, therefore, the need of the hour to provide the users with tools to fight back the spying eyes. Our privacy-preserving dissemination algorithm RIPOSTE is an answer to the above problem. With

RIPOSTE, we aim to enable users to freely contribute to the spread of information. In our knowledge, this is one of the first steps taken in preserving users' privacy on social networks for the spread of information, without anonymizing the user.

The idea of RIPOSTE is based on the famous *Randomized Response Technique*. It is a common practice for individuals not to disclose their honest opinions and answers in surveys, questionnaires, and interviews. There can be many reasons for such confiding practice, including fear of rejection, humility, or a mere unwillingness to trust strangers with their honest answers. People with this tendency of either not answering the questions or intentionally answering them wrong lead to problems of refusal bias [46,52] and response bias respectively [76]. The answers to questions that are either too personal or involve sensitive subjects are often obtained with reluctance, specially when they tend to outreach a huge audience. Such is a case of online social networks, where the information up-voted or liked by an individual can reach a very large audience, revealing user's opinion about it. *Randomized Response Technique* is an algorithmic solution to incite individuals to reveal information, which they otherwise are resistant to, which they consider unfit to disclose. This technique has its origin mainly for the purpose of surveys and interviews, but as shown with RIPOSTE, we extend it to the idea of social networks. The basic idea behind this technique is to allow the individual to respond using a probability for some of the questions. In such a case, even the interviewer will have an estimation of the "original" answer of the individual, based on a probability. We will give more details about the technique applied to RIPOSTE in Section 3.3.

2.3 Learning in Recommendation Systems

Roughly speaking, the problem of recommendation is estimating a utility function that *automatically predicts* how a user will *like* an item, based on past behavior of users in the system, relation with other users of the system, *similarities* between different items, context and other information available in the system about the users and/or the items. On a general note, once the data (in a cleansed and processed form) has been fed to the recommender system, the learning process combined with a model or clustering technique helps to make predictions of the items in the system. Figure 1.5 in Section 1.1.2 shows a block representation of this idea of recommendations. In this section, we will focus on the Data Mining block of recommendation systems that provides for its *learning* ability through techniques like *classification*, *clustering* or *rule generation* as shown in Figure 2.8

Figure 2.8 – Learning block of the recommendation systems using techniques like classification, clustering or rule based associations.

We have seen previously that the most commonly used form of recommendations, i.e.,

collaborative filtering is traditionally based on either of the two techniques: User-User collaborative filtering or Item-Item collaborative filtering. Both, user-user and item-item collaborative filtering use the idea of *nearest neighbors* to select the most *relevant* candidates (users or items depending on the type of CF technique used) from the pool of all the users or items. This selection of candidates is generally performed based on the results of a *similarity* function. In this scenario, the technique of finding the nearest neighbors (users or items) based on the similarity function is the learning model for the underlying recommendation system. In addition to the user-user and item-item collaborative filtering, there are other collaborative filtering algorithms based on learning techniques like probabilistic learning, matrix factorization, clustering, rule-based approaches, classification, regression, latent dirichlet allocation etc. Some of these techniques, like classification techniques can be used in content based recommendations also. Figure 2.9 gives an overview of the most popular learning techniques used in recommendation systems along with their categorization.

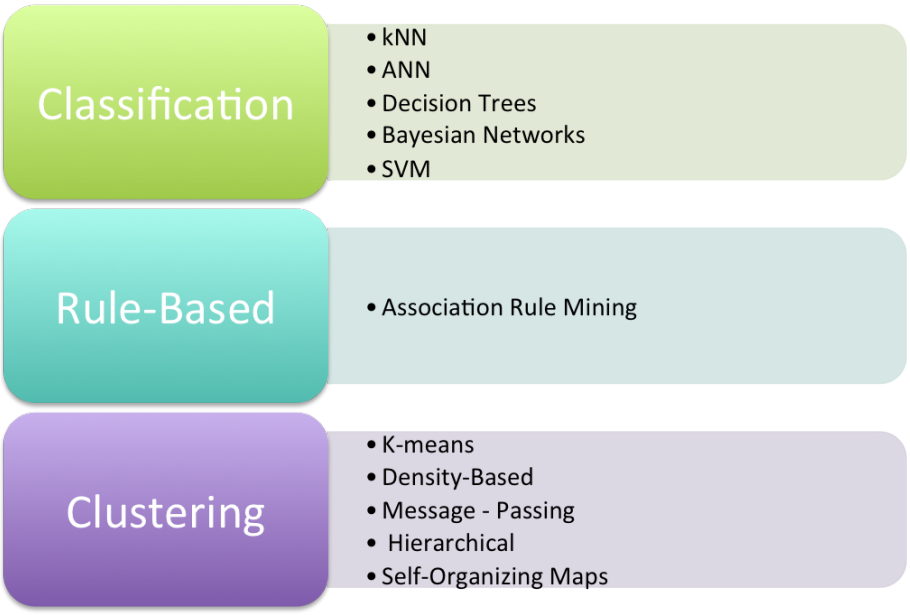


Figure 2.9 – A broad categorization of Learning techniques in recommendation systems

Each of these techniques have their own pros and cons. Their use in recommendation systems also depends on the target application. For example, TV program recommendations use association rule mining [40] where as movie recommendation systems are implemented mostly using KNN classification techniques [77, 125, 152].
Widely studied and analyzed, KNN is one of the most popular technique of recommendations. It can be used in recommendation systems for varied applications like book and document recommendations, image classification and recommendation systems, movie and music recommendations, e-commerce based recommendations etc. This simple technique of classification is easy to implement and is easily customizable with various parameters to tune, depending on the current need. Therefore, in the scope of this thesis, we focus on the *learning* in recommendation systems using *K-Nearest Neighbor* classification technique.

2.3.1 K-Nearest Neighbor

To explain the technical details of the K-Nearest Neighbor Algorithm, we will start with a small example. Alice wants to watch a movie, and she needs help to select one that she would like.

She decides to take help from her acquaintances who have interest in watching movies. Let's say that there are 100 people that Alice knows and are interested in movies, and have watched similar movies as Alice in the past. It is practically cumbersome to ask each of these people for their suggestions and then chose a movie, considering multiple suggestions from each of them. A more practical solution would be filter and shortlist, lets say, 10 people from 100, who are the most "similar" to Alice. Now with this new set of 10 most similar people to her, Alice can easily select a movie to watch. This example represents a basic example involving KNN computation. For the above scenario to work, here are some necessary components:

- movies
- people who have watched and shown *interest* in movies.
- similarity measure between Alice and other users

Collaborative Filtering based recommendation systems using neighborhood based techniques, such as KNN can be seen as the algorithmic translation of the principle of *word-of-mouth*. In this section, we will discuss the technical details of KNN algorithm. The details presented here use user-user collaborative filtering but can be easily extended to item-item collaborative filtering (by merely replacing users by items). Drawing this parallelance to collaborative filtering systems, using a KNN algorithm requires:

- Items
- Users
- Opinion of users in form of Ratings [4]
- Measure of similarity

Therefore, for a user u, the neighbors are the users v from the system who have similar tastes for the items seen by both u and v. The problem of K-Nearest Neighbors is finding k most similar neighbors for a given user. The problem of KNN can take different forms example, KNN search or KNN graph construction. While KNN search aims to find the k nearest neighbors of only a small number of users in the system, KNN graph construction does it for all the users in the system. Formally, a k-nearest neighbor (kNN) graph of a system is a directed graph where each node is connected to its k most similar neighbors where similarity is measured by a predefined metric. For example, the KNN representation of IMDB is a directed graph where all the users of IMDB are connected to their most similar counterparts. In this case the similarity is a result of like and dislike for the movies in the system that is explicitly stated by the users in form of ratings from 1 to 5. Albeit the simple idea, KNN graph construction is a computationally expensive algorithm. To find the most similar neighbors of user u, a trivial KNN algorithm requires to compute its *similarity* with all the other users in the system and then rank them based on the similarity measure. This is a brute-force approach in the literature of KNN and is evidently a computationally expensive task as it requires n comparisons for each user of the system where n is the number of users. Such an algorithm has a complexity of $O(n^2)$. When the size of users and items in the system increases, which is generally the case for the applications requiring recommendations like e-businesses, image classification etc, it becomes very difficult for a KNN to scale due to this complexity [111, 153].

The beginning of research in KNN graph construction mostly saw exact KNN graph construction algorithms [20,45,139]. This, in literature, was referred to as *all nearest neighbor* problem, where $k = 1$, i.e., only one nearest neighbor of each node in the system is computed. There

4. For the remainder of this section, we will consider the opinions in the form of ratings ranging from $1-5$, explicitly provided by the users for the items in the system, unless stated otherwise.

were many lines of attacks proposed for this problem of ANN. Bentley et al. [20] used a multidimensional divide and conquer method while Clarkson et al. proposed a randomized algorithm [45]. Though these algorithms claimed that they are easily extensible for $k > 1$ yet they were proved to be inefficient for large scale datasets [117].
The construction of a KNN graph is related to Nearest Neighbor (NN) search [110, 150, 151]. NN search addresses, however, a different problem: it seeks to find the k nearest neighbors of a small number of individual elements (the *queries*), rather than constructing a complete KNN graph as KIFF does. NN Search typically relies on complex pre-computed indexes, and aims to minimize both the size of these indexes, and the computation time of individual queries. NN Search has been extensively researched, in particular to address what is known as the *curse of dimensionality*, i.e., the observation that traditional space-partitioning indexes become too large in large dimensions, a situation commonly encountered in multimedia databases [82, 115]. Solutions to this problem rely on advanced hashing techniques such as *Locality-Sensitive Hashing* (LSH) [100, 115] or on compact metric approximations such as *product quantization* [82]. These approaches are, however, optimized for very dense data sets: all images of a media database are usually associated with non-zero values along all dimensions. On the contrary, the systems in the current era are sparse, where a user has a lot of unseen and unrated items in the system. Therefore, considering the sparsity of datasets is another challenge in finding a scalable and efficient *k-nearest neighbor* algorithm.

Also related to but different from the problem addressed in this paper, *clustering* [26] seeks to partition a dataset in k clusters while minimizing the graph cut between successive clusters. The bipartite graph underlying node-item datasets (e.g., containing users and items, or documents and words) has been exploited in this context to cluster such datasets simultaneously along their node and item dimensions, a technique known as *co-clustering* [53]. These works clearly underline the advantages of using bipartite graphs in problems involving users and items, which we use as motivation for our contribution in Chapter 4.

The idea of combining a coarse (in our case the number of overlapping items) and a finer metric (in our experiments cosine similarity) as KIFF does, has been proposed in the context of hybrid recommender systems [32] as a generic pattern to combine two similarity metrics. The approach is, however, different from ours in that the finer metrics are only used to resolve ties created by the coarse metric. In particular, it is not supposed to overturn a coarse ranking, which has higher priority. Further, and as far as we know, the approach has not been applied to KNN graph construction.

Because the construction of a KNN graph generally implies the computation of a large number of similarity measures, several works have proposed to use space decomposition techniques (similar a divide and conquer strategy) to reduce the complexity of the graph construction [39, 47, 113, 116]. This techniques are unfortunately limited to particular metric spaces, typically based on an Euclidean distance, and do not generalize to other similarity measures.

Originally designed for overlay construction in fully decentralized systems, a number of greedy approaches have been proposed to compute KNN graphs [28, 83, 143]. These methods leverage the transitivity of most similarity functions (if A is close to B, and B to C, A is usually close to C as well) to iteratively refine and converge to a good KNN approximation, or even produce the true KNN. NN-Descent [56] extends this strategy with several optimizations designed to fully exploit multi-core machines. NN-Descent has shown to deliver a better recall in a shorter computational time than a space partitioning technique using Recursive Lanczos Bisection (RLB) [39], or than an approach using Locality Sensitive Hashing (LSH) [100]. NN-Descent in particular performs well on very dense datasets such as image databases. As our evaluation shows, however, it is outperformed by KIFF sparse datasets, and its performance strongly depends on the value of k. Finally, HyRec [29] also exploits a similar greedy-based

approach to propose a generic recommendation framework based on user-based KNN graph. Recently, L2Knng [15] also adopts a two-phase approach and uses pruning to improve its KNN computation. L2Knng differs however markedly from KIFF in different dimensions. Firstly, L2Knng's approach is specific to the cosine similarity while KIFF can be applied to any similarity metric. Secondly, L2Knng exploits neighbors-of-neighbors relationships (as NN-Descent or HyRec) for its convergence phase while KIFF only exploits pre-computed candidate lists. Finally, the design and implementation choice of the candidate set of L2Knng renders it unsuitable for parallel execution. Particularly because its pruning mechanism of order n requires results from the remaining $n-1$ objects. By contrast, KIFF allows for a parallel implementation and execution, leading to full utilization of computing resources.

2.4 Data

> "WITHOUT DATA, YOU ARE JUST ANOTHER PERSON WITH AN OPINION"
> -- W. EDWARDS DEMING

Data is one of the prime and foremost requirement of any recommendation system, regardless of the technique it uses. Depending on their type, recommendation systems use different kind of data, for example, user-system interaction data, user demographic data, item description data, or data in form of users' opinion and views. We have seen earlier how the growth of internet and internet based applications has lead to the phenomenon of *information-explosion*. Data is constantly produced by the services and applications, while their tendency to consume it is much slower. However, *information-explosion* doesn't guarantee enough useful data for the purpose of making recommendations. The distribution of usable data is not uniform and as a result, many services still suffer, either with the problem of data insufficiency, or with the usability of data.

Online social networks are a rich source of user information. Most of these networks provide user demographic data (age, location, gender) as a part of users' public profile, while others give a free access to users' social connections, and her opinions. OSNs generally provide the users with ways to tune the privacy, and chose the attributes that are publicly visible but according to the *Consumer Report Survey*[5], around $14M$ U.S. users are not even aware of such privacy control settings. It also reports that in U.S. alone, around 28% people share their connections, and photos publicly. Users show similar behaviors across other social networks. Depending on the targeted application domain of the social network, users reveal different information on different networks.

This information, obtained from the public profiles of users from various sources, can indeed reduce the information gap for the recommendation based services that do not have access to enough user data for making recommendations. In the next section we give an overview of the literature work focusing on using social networks for *inferring* unknown user information. This inferred information about users (and the one directly available from various OSNs) can be used to feed and train recommendation system models, specially the ones that rely on user information. The user information, can also be combined with other available information like item attributes and meta data, to design hybrid recommendation systems, as discussed above in this chapter.

5. http://www.consumerreports.org/cro/magazine/2012/06/facebook-your-privacy/index.htm

2.4.1 Power of Data

Online social networks have gained unprecedented popularity in all domains. Initially created to reflect only the real social ties, they have now evolved to indulge in diverse purposes, like sharing photos, disseminating news, microblogging etc.; giving the users more reasons to join more than one social network. Every OSN requires its users to have a profile, which consists of some mandatory and other optional attributes, like age, gender, location, marital status, schools, music etc. The information in user's profile can vary depending on the OSN in consideration. The availability of data from online social networks has been already considered useful for personalization mechanisms and recommendation systems.

Accrediting to the usefulness of user based information in personalization services and recommendation systems, a lot of investigation has been done to extract maximum user information from the OSNs. While information retrieval based studies explains how the user-profiles can be crawled and scraped, there are other studies that talk about extracting the unknown information from a user's profile. This is a mechanism widely studied in the domain of online social networks and is called *inference*. Inference in OSNs can be used for wide range of purposes like inferring network structure, trust relations or user attributes. In the scope of this work, we only focus on user attribute inference in OSNs. It has claimed to offer promising results for various applications such as, constructing missing user's profile [8], investigating information theft and user privacy [42, 129], and guiding interest based recommendations [137]. Our work is an empirical study deriving its motivation from the work of Chen *et al.* [42]. As it is an investigation into the inference power of user attributes, specially focusing on the information derived from tweets, we do not aim to devise new inference algorithms. Instead, we rely on the previous works, specially from Chen *et al.* to use the Naive Bayes inference mechanism in our work. There are many other algorithms that can be used, and have shown to provide similar performances in the past studies. Below we will describe some of the most popularly used inference algorithms:

Naïve Bayes: classification is a form of supervised learning and is based on the famous *bayesian theorem*. Though naïve bayes is one of the simplest classification techniques, yet it is known to give a consistently good performance. This classification technique is suitable for both categorical and continuous data, and considers the variables (or features) independent of each other. In our case of user-attribute inference, we would like to treat each attribute independent of each other, even if there are strong correlations between them. The main reason behind treating the variables independently because the attributes extracted from OSNs depend on the method of information retrieval used. In technical terms, given a set of features (or attributes), $X = \{x_1, x_2, \cdots, x_n\}$, the goal is to predict the probability of an event E from a possible set of outcomes labeled as $\{e_1, e_2, \cdots, e_n\}$.

It is often observed that the variables used in Naïve bayes classifier are not always independent but this assumption simplfies the classification task dramatically. It allows the class conditional probabilities : $p(x_i|E)$ to be calculated individually for each variable, thereby decomposing a multi-dimensional task into several one-dimensional ones. Furthermore, the assumption does not seem to greatly affect the posterior probabilities, thus, leaving the classification task unaffected. Naïve bayes is particularly efficient and fast as compared to its counterparts when the number of feature values is high, which can be the case of values extracted from social media.

Logistic Regression: is a classification task that assumes a function in the form of $f : X \rightarrow C$. Here C has discrete values where as X can have discrete or continuous values or a

combination of both. It is a linear classifier in the sense that predictors can be written as:

$$C_i = \frac{1}{1 + e^{-\hat{\mu}}}$$

and μ is a linear function of X.

In its original form, logistic regression is a binary classifier but there are many variants available in the literature to deal with multi-valued attributes.

Support Vector Machines: are more complex in design and implementation than their counterparts. Also performance of support vector machine classifiers is dependent on the fine tuning of its parameters, which adds to the complexity of their design. Support vector machines focus only on the points that are the most difficult to tell apart, whereas other classifiers pay attention to all of the points. One of the major advantages of using SVM classifiers is that they do not suffer from the *curse of dimensionality*.

As we can see in the Figure2.10 below, there is more than one way of classifying the data points. In SVM, the focus is to maximize the margin around the separating hyperplane. Therefore as opposed to naïve bayes and logistic regression approaches, the goal of SVM is to use the 'hardest" to classify data points instead of using all of them. Therefore the choice of training dataset is very important to train a SVM classifier well.

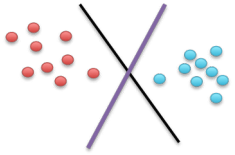


Figure 2.10 – More than one possible hyperplanes when separating classes in SVM

Due to the simple goal of our empirical study, we do not delve into the details of attribute processing. Instead, we focus on gathering maximum user information from three social networks and process the attribute values to have consistency and maintain a reasonable level of granularity. Our work highlights the importance of tweets (and other similar attributes like facebook statuses, messages etc.) in inferring user's interests in topics like movies, books and music. Our work reveals how information from one source can be used to infer the interests on a completely different source. For example, the study shows the importance of tweet related information to infer the *skills* that a user posses (mostly professional), available only on linkedin.

"You should have personalized genomics, personalized physiology, personalized medicine, where each person's different, and each body is an integrated whole"

-- George M. Church

3

Privacy of Users in Recommendation Systems applied to Online Social Networks

> The need is pressing and imperative if we are going to have any shot at privacy
>
> *Ari Schwartz*

In this chapter we will elaborate our contribution towards *preserving privacy* of a user while *disseminating* information in online social networks. In lines with our objective of providing users with a privacy mechanism that enables them to spread information in an OSN freely, we propose RIPOSTE. It is a distributed algorithm for disseminating information in form of ideas, news, trust, opinion or trends, in a social network. In addition to safeguarding users' privacy, RIPOSTE also ensures that the spread of information in a network follows some intuitive and well established norms. Without going into the technicalities and preciseness of its dissemination, it guarantees that information spreads widely in the network if and only if a large fraction of users find it interesting while if the information item is liked by a smaller fraction of users then the dissemination process dies quickly.

To support the ideas and intuitions presented as RIPOSTE, we perform an extensive set of experiments and evaluation using topologies and snapshots of real online social networks, including Twitter, Facebook and Google+.

To present and understand RIPOSTE, we will start by presenting an introduction to the idea and need of such an algorithm in Section 3.1. This is followed by Section 3.2 that presents the technical idea of the contribution made, with its overall view. Later in Section 3.4, we develop the RIPOSTE algorithm with all the details and parameters involved. We present our experiments and evaluation of RIPOSTE in Section 3.5, by finally concluding the work in Section 3.6.

3.1 Introduction

From the discussion in Section 1.1.3 it is comprehensible that recommendations can take various forms in an Online Social Network setting. Among others forms discussed before, following other users of the network to have access to information items that they spread/like

or are interested in, is one of the forms of recommendations in online social networks. This form of recommendation has become the heart of OSNs as it is one of the major forces behind spreading the information in the network.

Online Social Networking web sites have become an important medium for communicating and disseminating news, ideas, opinions, treads and behaviors. These online social networks have revolutionized the world of internet and that of the data industry by generating massive amounts of data. In addition to the generation of data, such online networks typically provide a *reposting* functionality also, e.g., *sharing* in Facebook or *retweeting* in Twitter, which allows users to share others' posts and information items with their own friends and followers. In essence, they serve as a bridge between the generation of vast amount of information and its spread from one user to another. As information is reposted from user to user, large cascades of reposts can develop, and an information item can potentially reach a large number of people, much larger than the number of users exposed to the information initially (e.g., the users who witness a news event, or learned about it from some local media). Since people tend to propagate information which they find interesting and worth sharing (rather than random content) [102], an information item may spread widely only if sufficiently many users find it interesting. Thus, the natural diffusion process facilitated by the reposting mechanism serves both as a filtering and dissemination mechanism. Ideally, the opposite direction should also hold: content that a sufficiently large fraction of users would find interesting and would propagate (if they knew about it), should be likely to spread widely. This is, however, not always the case.

In countries with authoritarian regimes, users may not propagate anti-government ideas for the fear of being prosecuted. Almost all of the popular social networks have faced the heat of this issue in the past years. There are in fact several examples of political activists (and others) that have been convicted for posting or just reposting anti-government opinions on social media [112, 136]. But even in democratic regimes, users may refrain from openly supporting their opinion on certain sensitive issues, from politics and religion to sexuality and criminal activity. For example, a user may not propagate a post supporting recreational drug use for the fear that it may have a negative impact on his career—as it is a common practice of employers to use social media for screening prospective employees [72]. Or more generally, users may refrain from reposting a (political or other) opinion when they believe it is not widely shared by their cycle—a well known principle in sociology known as the "spiral of silence" [114]. In all these cases, the dissemination of an idea in the social network is impeded by privacy considerations; even if many users support the idea, they may choose not to contribute to its propagation because they do not wish to reveal their own opinion (as reposting the idea would suggest the user is in favor of it).

Social media is powerful. Social media is information.
Let's choose wisely what messages we spread.

3.2 Contribution: Privacy-Conscious Diffusion

Inspired by the theory of branching processes [16], we investigate a dissemination algorithm that has, roughly speaking, the following properties: (1) information that a sufficiently large fraction of the population finds interesting is likely to spread widely; (2) information that not sufficiently many people find interesting does not spread far beyond the set of users ex-

posed to it initially; and (3) by observing the spreading process (in particular, the users' reposts), one cannot determine (with sufficient confidence) the opinion of any single user on the information that is disseminated. We assume that the diffusion mechanism does not know the opinion of users that have not received the information yet,

More specifically, we propose the following simple, local dissemination algorithm, which we call RIPOSTE. We show a high level view of RIPOSTE in Figure 3.1. We assume that a user comes across an information item on an online social network. She interacts with this OSN using her local machine through a browser over internet, as shown in the figure. We assume that for each information item, the user can either *like* it or *dislike* it. This like or dislike of the user for an item is the factor that decides weather the user wants to repost the information in the network to spread it further or not. In order to conceal her true opinion (of like or dislike), instead of directly acting on the OSN, she reveals her opinion to RIPOSTE, which is running on the user's system, which in turn adds some randomness to it before sending it to the OSN. In principle, this randomness makes it practically impossible for an adversary to guess the user's actual true opinion on the information item, thus avoiding any privacy breaches involved in the dissemination.

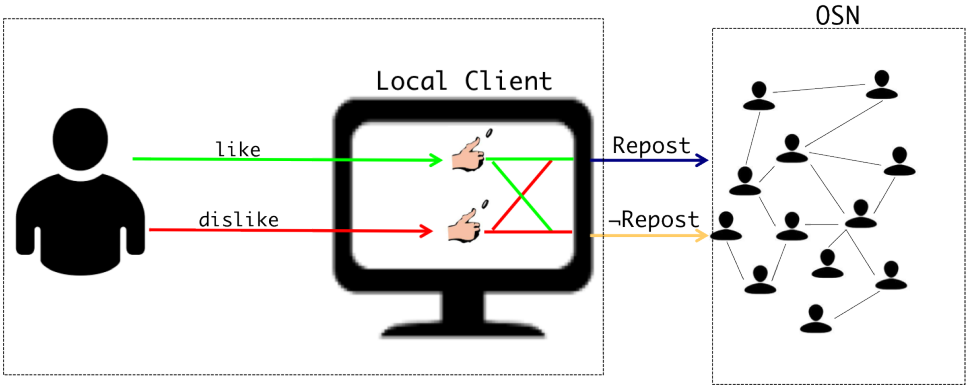


Figure 3.1 – Major components of RIPOSTE*: the local machine with the randomized algorithm and the online social network. In* RIPOSTE*, the user's opinion is locally randomized and only the output of* RIPOSTE *is exposed to the OSN. The* ¬RIPOSTE *arrow is represented for illustration purposes and is not an explicit action of* RIPOSTE.

Now that we know the overall idea behind the functioning and design of RIPOSTE, we will delve into a more technical discussion. Let G denote the (directed) graph modeling the social network, and n be the total number of users, and suppose that some (small) initial set of users learn an information item t. For each user u that learns t, RIPOSTE decides to either repost t, to all u's outgoing neighbors in G, or to not repost t, to anyone. The decision is randomized and depends on the user's (private) opinion on the information, and the number of the user's neighbors that have not received the information yet.

Precisely, if u likes t, then t is reposted with probability λ/s_u, and if u does not like t, then t is reposted with a (smaller) probability δ/s_u, where $0 < \delta < 1 < \lambda$ are global parameters of the dissemination mechanism, and s_u is an *upper bound* on the number of u's outgoing neighbors that have not received t yet. If the algorithm cannot have access to information about whether u's neighbors already know the information, then the *total* number of u's outgoing neighbors can be used as the upper bound s_u.We consider also a variant of this protocol, where in place of $\deg(u)$ we use instead the number of u's neighbors that have not already received the information (through some other neighbor). Intuitively, this variant results in larger dissemination. [1]

1. RIPOSTE can be viewed as a set of distributed pieces of software running at each user's machine connected

We argue that RIPOSTE achieves the property of *plausible deniability*: A user u can claim that, with reasonable probability, the act of reposting (or not) some information, does not reflect u's truthful opinion on the information, and is a result of the randomness in the decision mechanism. Intuitively, the closer the parameters λ and δ are to each other, the better the privacy. In the extreme case of $\lambda = \delta$, we have perfect privacy, but then the dissemination is independent of u's opinion (and thus of how interesting the information is). In the other extreme, if λ is the maximum degree and $\delta = 0$ (i.e., the user reposts the information iff it likes it), the act of reposting (or not) the information reveals with certainty u's opinion. We formally quantify the privacy properties of RIPOSTE in terms of *ϵ-differential privacy*. In particular, we argue that RIPOSTE is $\ln(\lambda/\delta)$-differentially private.

For the dissemination of information, we prove the following threshold behavior. Suppose that each user likes a given item t with probability p_t, independently of the other users (p_t is the same for all users and depends only on t). Thus p_t is a measure of how interesting item t is, and is equal to the expected fraction of users that like t. Let S denote the set of users who receive item t initially (e.g., these users receive the information from a news channel). We show that if $p_t < p^*$, for $p^* = (1-\delta)/(\lambda-\delta)$, then the expected number of users that learn the information is $O(|S|)$, i.e., at most a constant factor larger than the users exposed to the information initially. This is true for any graph G. On the other hand, we show that the following statement holds for a G from a family of random directed graphs with arbitrary out-degree distribution [41]. (Such a graph could, for example, model the Twitter network). If $p_t > p^*$, then for a random initial set S, information t spreads to $\Theta(n)$ users (i.e., at least some constant fraction of the network), with probability $1 - e^{\Omega(|S|/d)}$, where d denotes the average degree of G. In particular, this result says that information spreads to $\Theta(n)$ users with constant probability when $|S|$ is close to the average degree, and with high probability if $|S|$ is $\log n$ times larger than the average degree. The analysis draws from the theory of branching processes [16], and the intuition is simple: Basic computations yield that the expected number of users that a given user passes the information to, is less than 1 when $p_t < p^*$, and greater than 1 if $p_t > p^*$. The threshold phenomenon we observe follows then from a similar phenomenon in branching processes. We note that the result for $p_t > p^*$ does not hold for arbitrary graphs. However, we expect that it should hold for many graph families, of sufficiently high *expansion*.

We complement our analysis with extensive experimental results. We use a complete snapshot of the Twitter graph of roughly 40 million users from 2009, and smaller samples from other social networks, including Facebook and LiveJournal. The experiments demonstrate clearly the predicted threshold phenomenon, with very limited spread below the p^* threshold, and substantial spread above p^*. The latter suggests that our result for $p_t > p^*$ should qualitatively hold for a larger family of networks than the stylized model analyzed formally. We also experiment with non-uniform distributions of user opinions, where users closer to the source users are more likely to like the information, obtaining qualitatively similar results. Experiments suggest that reasonable values for RIPOSTE's parameters in the networks considered are $\delta = 0.75$ and $\lambda = 3$. For these values, the plausible deniability achieved ensures that, for example, if the prior probability for a user to like the information is 0.01 or 0.1, and the user reposts the information, then the probability increases to 0.04 and 0.3 respectively (see Sect. 3.4 for details).

We view the results of this paper as potentially useful for addressing some of the increasing concerns about users' privacy in social networking services. In particular, we think that RIPOSTE could be of interest as a tool for spreading information and petitions in Internet-based

to the social network. It is not the user who has the control of whether the item will be eventually reposted or not but this piece of software. It solicits the user's opinion on the item, and then flips a coin to determine whether the information will be reposted.

activism, a topic of considerable current interest [60, 66]. More generally, it is a tool that could be used for widespread dissemination of sensitive information, which people would care to be exposed to, but are not willing to disseminate themselves for the fear of being charged, stigmatized, or isolated. We believe that such a tool could be incorporated in existing social network services. Also our technique could find applications to other distributed problems, such as distributed polling algorithms.

3.3 Related Works

RIPOSTE as we have seen earlier, derives its motivation from a technique that is conceptually similar to the *randomized response technique (RRT)*. Roughly, the idea is to tell responder to *lie* with some fixed, predetermined, probability p_{lie} (e.g., roll a die and lie whenever the die shows one or two, in which case $p_{\text{lie}} = 1/3$).[2] Since p_{lie} is known to the interviewer, the distribution of responders' truthful answer can be easily estimated, and thus, accurate estimations of aggregate results can be extracted—but an individual's answer is always plausibly deniable. (See [38] for other variations of RRT, and [12] for a variant using cryptography to guarantee that the responder follows the RRT.) In our diffusion mechanism, the same probability of reposting could be achieved using the following RRT-like approach: User u is asked if she likes the post, but is instructed to lie with probability $p_{\text{lie}} = \delta/(\delta+\lambda)$; and if the answer is 'yes' then the post is reposted with probability $(\delta+\lambda)/s_u$.

We are not aware of other works that use randomized responses in a way similar to ours: to achieve dissemination that reflects user's aggregate opinion, while preserving the privacy of individual users' opinion. In a more standard use of RRT, Quercia et al. [124] proposed a method to aggregate location data of mobile phone users by having each user report several erroneous locations in addition to the correct one. Recently, Erlingsson et al. [61] presented an RRT-based algorithm for crowdsourcing of population statistics from end-user client software, deployed on Google's Chrome Web browser.

Another mechanism provided by social networking services, besides reposting, which has the potential to make interesting posts widely visible is that of *liking*. This mechanism has similar privacy issues as reposting. In [9], Alves et al. proposed a scheme to anonymize user's likes, which keeps the actual like count of a post without revealing the names of the users who like it. Unlike our approach, the scheme employs cryptographic techniques to achieve privacy, and requires a centralized server (but the server does not know the users' opinion).

We have said that our diffusion scheme could provide a tool for Internet-based activism [60, 66]. The use of pseudonyms, combined with methods for hiding the user's IP, has also been a common practice used by activists to hide their identity while spreading sensitive information [147]. Our scheme protects the users who contribute to the dissemination of information, but not the sources of the information. This is not a problem in some settings, for example, if we assume that anti-government information originates from a news channel (say, WikiLeaks) located in a different country. If this is not the case, then pseudonyms could be used to protect the privacy of the source.

We measure the privacy properties of our diffusion scheme in terms of *differential privacy* [58, 59]. Differential privacy was introduced in the context of privacy-preserving analysis of statistical databases. Roughly speaking, differential privacy ensures that (almost, and quantifiably) no risk is incurred by joining a statistical database. More relevant to our setting is the *local model* of differential privacy [85], also known as *fully distributed model*. In this model, there is no central database administrator of private data; each individual maintains their own

2. The closer is p_{lie} to 1/2 the better the privacy.

data element (a database of size 1), and answers questions about it only in a differentially private manner. The local privacy model was first introduced in the context of learning, where it was shown that private learning in the local model is equivalent to non-private learning in the statistical query model [74, 85].

3.4 RIPOSTE: The privacy preserving diffusion algorithm

In this section, we describe our diffusion mechanism for disseminating information in an online social network, and provide an analysis of its properties.

We model the social network as a directed graph $G = (V, E)$ with $|V| = n$ nodes. Each node $u \in V$ represents a user (from now on we will use the terms node and user interchangeably), and a directed edge from node u to v denotes that user u can send information to v. For example, for the case of the Twitter social network, an edge $(u, v) \in E$ in the underlying graph G denotes that user v is "following" u. Borrowing Twitter's parlance, in this paper, we will say that v is a *follower* of u if $(u, v) \in E$. The number of u's followers is thus the same as u's out-degree.

We assume that initially a set of users $S \subseteq V$ learns an information item (from a source external to the network). From each user that learns the information, this information can be *reposted* to all its followers. (So, information can either be sent to all followers of the user, or to none.) We propose a randomized distributed algorithm, running locally at each user (i.e., at the user's device connected to the social network service), which decides whether or not to repost the received information; we call this algorithm RIPOSTE.

RIPOSTE takes as input the opinion of the user on the information item, i.e., if the user *likes* or *does not like* the information, and the algorithm's effect is to either repost the information or not. RIPOSTE' decision depends on: (1) the user's opinion, (2) an upper bound on the number of the user's followers that have not received the information yet, and (3) two global parameters of the protocol (the same for all users), denoted δ and λ; both parameters are non-negative real numbers satisfying $\delta < 1$ and $\lambda > 1$. As explained later, these parameters control the privacy properties of the protocol, and influence the dissemination.

RIPOSTE Algorithm: For each new information item received by user u, if u has k followers and $s \leq k$ is an estimate bounding from above the number of u's followers that have not received the item yet, then:
if u likes the item, the algorithm reposts the item with probability

$$r_{\text{like}}(s) := \begin{cases} \lambda/s, & \text{if } s \geq \lambda + \delta, \\ 1 - \frac{\delta(s-\delta)}{\lambda s}, & \text{if } 0 < s < \lambda + \delta; \end{cases}$$

if u does not like the item, it is reposted with probability $r_{\text{dis}}(s) := \delta/s$ (if $s > 0$).

It is easy to verify that $r_{\text{dis}}(s) \leq r_{\text{like}}(s)$, for all s, i.e., the probability of reposting is larger when u likes the item. Also, the closer are δ and λ to each other, the closer are the two probabilities r_{dis} and r_{like}.

The definition of $r_{\text{like}}(s)$ for the case of $s < \lambda + \delta$ will be justified when we analyse the privacy of the protocol. Until then we can assume the following simpler definition for *all* $s > 0$: $r_{\text{like}}(s) := \min\{\lambda/s, 1\}$.

RIPOSTE needs to know an upper bound on the number of the user's followers who have not yet received the item. This information is readily available in some existing social network services, including Twitter, where the default setting is that a user can access the list of items each of its followers has received. If this information is not available, then the total number of followers k of the user can be used as the upper bound s. For the analysis and the experimental evaluation, we will make use also of that special variant of RIPOSTE, where $s = k$.

DB-RIPOSTE Algorithm (Degree-Based-Riposte): This algorithm is a special instance of RIPOSTE, where the total number of followers k of user u is used as the upper bound s on the number of u's followers who have not already received the information.

An attractive analytical property of DB-RIPOSTE is that the outcome of the dissemination does not depend on the order in which the algorithm is executed at different users, unlike in the general RIPOSTE algorithm. For our analysis of RIPOSTE we assume that the order can be arbitrary.

We stress that RIPOSTE does not reveal any information on the value of its input (the user's private opinion), other than the statistical information inferred by the outcome of the algorithm, to repost or not. Also, the user cannot *prevent* the algorithm from reposting the information, even if she does not like the information. In particular, if the user refuses to answer whether she likes an item or not, this is interpreted as a negative answer by the algorithm (the user has an incentive to answer positively if she likes the item, as this would potentially result in larger spread).

We now analyze the properties of RIPOSTE, regarding privacy and the spread of information.

3.4.1 Privacy

RIPOSTE achieves the property of *plausible deniability*: A user can claim that, with reasonable probability, the act of reposting (or not) an information, does not reflect the user's truthful opinion on the information, and is a result of the randomness in the algorithm.

The standard notion used to quantify plausible deniability is that of *differential privacy* [59]. We recall now the definition of an ε-differentially private algorithm. Let A be a randomized algorithm with input a collection of values, $x_1,\ldots,x_m$, that returns a value from some domain R. Since the algorithm is randomized, for a fixed input $x_1,\ldots,x_m$, its output $A(x_1,\ldots,x_m)$ is a random variable, with some distribution over R. Suppose that the input to A is not known to us (is private), and by observing the output of A we want to find out the value of some of the inputs. More generally, we may have some information about the input, i.e., a distribution over the possible combinations of input values, and we want, by observing A's output, to improve this information, i.e., obtain a distribution closer to the true input values. We can quantify the extent to which this is possible in terms of ε-differential privacy: algorithm A is ε-differentially private if changing exactly one of it inputs $x_1,\ldots,x_m$ changes the distribution of the output by at most an e^{ε} factor.

Definition 4 (ε-differential privacy)

A randomized algorithm A with inputs $x_1,\ldots,x_m$ from some finite domain and output $A(x_1,\ldots,x_m)$ on some domain R, is ε-differentially private if for any two sets of inputs $x_1,\ldots,x_m$ and $x'_1,\ldots,x'_m$ that differ in exactly one value, and for any set of outputs $Q \subseteq R$,

$$\Pr\big(A(x_1,\ldots,x_m) \in Q\big) \leq e^{\varepsilon} \cdot \Pr\big(A(x'_1,\ldots x'_m) \in Q\big).$$

In our setting, algorithm A is RIPOSTE, which takes a single binary input: the opinion of the user, and has a binary output: repost or not-repost.

Theorem 1

RIPOSTE is ε-differentially private for $\varepsilon = \ln(\lambda/\delta)$.

The proof is a straightforward application of the definitions, We must show that (i) the probability of reposting when the user likes the item is no larger than $e^{\varepsilon} = \lambda/\delta$ times the probability of reposting when the user does not like the item, i.e.,

$$r_{\text{like}}(s) \leq (\lambda/\delta) \cdot r_{\text{dis}}(s); \tag{3.1}$$

and (ii) the probability of not reposting when the user does not like the item is no larger than λ/δ times the probability of not reposting when the user likes the item, i.e.,

$$1 - r_{\text{dis}}(s) \leq (\lambda/\delta) \cdot (1 - r_{\text{like}}(s)). \tag{3.2}$$

For Ineq. (3.1), we have that $r_{\text{like}}(s) = \lambda/s$ if $s \geq \lambda + \delta$, and $r_{\text{like}}(s) = 1 - \frac{\delta(s-\delta)}{\lambda s} < \lambda/s$ if $0 < s < \lambda + \delta$; thus for any s, $r_{\text{like}}(s) \leq \lambda/s$. And since $r_{\text{dis}}(s) = \delta/s$, Ineq. (3.1) follows.

For Ineq. (3.2), if $0 < s < \lambda + \delta$, the right-hand side is

$$(\lambda/\delta) \cdot (1 - r_{\text{like}}(s)) = (\lambda/\delta) \cdot \tfrac{\delta(s-\delta)}{\lambda s} = \tfrac{s-\delta}{s} = 1 - r_{\text{dis}}(s);$$

while if $s \geq \lambda + \delta$,

$$\begin{aligned}
(3.2) \;&\Leftrightarrow\; 1 - \delta/s \leq (\lambda/\delta)(1 - \lambda/s) \\
&\Leftrightarrow\; \delta(1 - \delta/s) \leq \lambda(1 - \lambda/s) \\
&\Leftrightarrow\; (\lambda^2 - \delta^2)/s \leq \lambda - \delta \\
&\Leftrightarrow\; (\lambda + \delta)/s \leq 1 \\
&\Leftrightarrow\; \lambda + \delta \leq s,
\end{aligned}$$

and the last inequality holds. □

Theorem 1 implies that the closer is the ratio λ/δ to 1, the better the achieved privacy. In particular, if $\delta = \lambda$ we have perfect privacy, as the probability of reposting does not depend on the user's opinion—but this is not desirable from a dissemination point of view.

We discuss now what Theorem 1 implies about the information one can gain for the opinion of a user on some information item it receives, by observing whether or not the item was reposted from that user.

Let q be the (prior) probability that the user likes the information, capturing the knowledge of an observer about the user's opinion *before* the observer sees whether or not this information is reposted from the user. Then from Theorem 1 it follows that the probability $\hat{q}$ with which the observer believes that the user likes the information, after the observer learns whether or not there was a repost, satisfies the inequalities

$$\frac{q}{q + (1-q)(\lambda/\delta)} \leq \hat{q} \leq \frac{q}{q + (1-q)(\delta/\lambda)}. \tag{3.3}$$

The proof, by Bayes' Rule, is standard and is discussed here. Fig. 3.2 illustrates the above relationship between q and $\hat{q}$. E.g., for the typical parameter values $\delta = 3/4$ and $\lambda = 3$ we use later in the experimental evaluation, we have that if $q = 0.01$ then $0.0025 < \hat{q} < 0.039$; if $q = 0.1$ then $0.027 < \hat{q} < 0.31$; and if $q = 0.9$ then $0.69 < \hat{q} < 0.97$.

Above we have considered the amount of information leaked when observing the cascade of a single information item. However, if one can observe the cascades of a *sufficiently large* number of *sufficiently similar* items, possibly over a long period, then more information can potentially be leaked about the opinion of a user on this type of information. We leave as a future work the study of such correlation attacks.

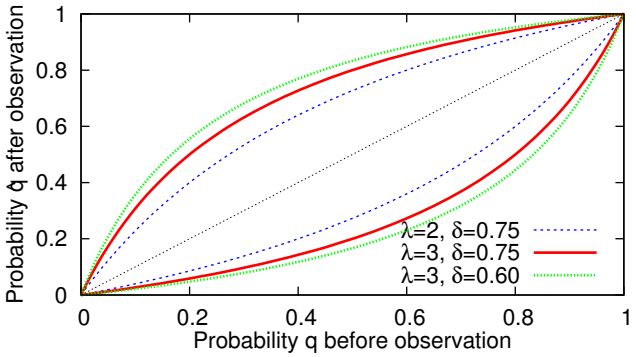


Figure 3.2 – Illustration of Ineq. (3.3) *that provides upper and lower bounds on the conditional probability $\hat{q}$ that the user likes the information, after observing whether the information was reposted from the user, in terms of the prior probability q before the observation. Probability $\hat{q}$ diverges more from q as λ increases or δ decreases.*

3.4.2 Dissemination

In terms of dissemination, the goal of RIPOSTE is that the fraction of users receiving an information item should reflect the users' overall opinion on the item. In particular, information that a large fraction of users like should, typically, be received by a lot of users, while less interesting information should not be received by many users. In the following, we quantify the notions of interesting/not-interesting information by defining a *popularity* threshold, and we provide bounds on the spread of *popular* items (with popularity above this threshold) and *unpopular* items (with popularity below the threshold).

For the analysis, we make the assumption that all users are equally likely to like a given item, independently of their position in the network and the opinion of other users.

Definition 5 (Popularity)

Each item t is associated with a probability p_t, called the *popularity* of t, and for each user u, the probability that u likes t is equal to p_t and independent of the other users' opinion about t. This is our setting of Uniform Opinion Model.

We note that popularity p_t is also equal to the expected fraction of users that like t. An item's popularity is not known in advance by the diffusion protocol.

We define the popularity threshold p^* as follows. Suppose that user u receives an item with popularity p. Since u has probability p of liking the item in the uniform model, the probability that RIPOSTE reposts the item, if $s > 0$, is $p \cdot r_{\text{like}}(s) + (1-p) \cdot r_{\text{dis}}(s)$. If $s \geq \lambda + \delta$, this probability is $p \cdot (\lambda/s) + (1-p) \cdot (\delta/s)$. Moreover, if s is the exact number of u's followers that have not received the item yet, then the expected number of new users that learn the item from u is s times that, i.e., $p\lambda + (1-p)\delta$. The popularity threshold p^* is then the probability p for which this expectation is equal to 1.

Definition 6 (Popular/Unpopular items)

For given λ and δ, we define the popularity threshold $p^* := \frac{1-\delta}{\lambda-\delta}$, and we call an information item t *popular* if its popularity is $p_t > p^*$, and *unpopular* if $p_t < p^*$. [a]

a. For the asymptotic bounds we show later, we assume for a popular item t that $p_t > p^* + \epsilon$, and for an unpopular item t that $p_t < p^* - \epsilon$, for some arbitrary small constant $\epsilon > 0$.

Next we establish an upper bound on the spread of unpopular items, and a lower bound on the spread of popular items.

We first argue that the expected number of users who receive a given unpopular item is by at most a constant factor larger that the number of user $|S|$ who receive the item initially (e.g., from a source external to the network). The constant factor depends on the popularity of the item and parameters δ and λ. This bound holds for any network G, assuming the uniform opinion model. Recall that an item is unpopular if its popularity is smaller than $p^* = (1-\delta)/(\lambda-\delta)$.

Theorem 2 (Spread of unpopular items)

For any G, and under the uniform opinion model, RIPOSTE guarantees that an item with popularity $p < p^*$ starting from any set S of users is received by an expected total number of at most $|S|/\beta$ users, where $\beta = (p^* - p)(\lambda - \delta)$.

The proof of Theorem 2, is based on the fact that the expected number of new users that learn the item from a given user that knows the item is smaller than one.

Recall that, in general, the dissemination achieved by RIPOSTE depends on the order in which users execute the algorithm. We consider the following representation of the random process underlying the dissemination of an item. At each point in time users are divided into three sets: (1) the set D of users who have received the item and RIPOSTE has been executed at those users (resulting in reposting or not reposting the item); (2) the set N of users who have received the item but RIPOSTE has not been executed yet at those user; and (3) the set U of the remaining users, who have not received the item yet. We assume that dissemination proceeds in steps: At each step, a single user u from set N is considered, and RIPOSTE is executed at that user. As a result, u is moved from set N to set D, and if RIPOSTE does repost the item then all u's followers from set U are moved to N. The dissemination is completed when set N becomes empty. For our analysis, the order in which users from N are considered can be arbitrary.

We denote by N_i the value of the set N defined above after the first i steps; thus $N_0 = S$. We also let $n_i = |N_i|$. The total number T of users that receive the item (including the source nodes from S) is then

$$T = \min\{i : n_i = 0\}.$$

Suppose that $i \le T$ (and thus $n_{i-1} > 0$), and consider the expected change on n_i in round i: If the user u considered in step i has at most $s > 0$ followers that have not received the item yet, then the probability that the item is reposted from u is

$$p \cdot r_{\text{like}}(s) + (1-p) \cdot r_{\text{dis}}(s) \le p \cdot (\lambda/s) + (1-p) \cdot (\delta/s) = \frac{1-\beta}{s},$$

where for the inequality we used that $r_{\text{like}}(s) \le \lambda/s$, for all $s > 0$. Thus, the expected number of *new* users that learn the item at step i is at most $1-\beta$, and the expected change in n_i is

$$\mathbf{E}[n_i - n_{i-1} \mid i \le T] \le (1-\beta) - 1 = -\beta, \tag{3.4}$$

where the '-1' in accounts for the removal of u from N_{i-1}.

It is now easy to understand the intuition behind the bound we must show: At each step, n_i drops *in expectation* by at least β. If, instead, the *actual* drop were at least β, then it would follow that the number of steps until n_i becomes zero would be at most $n_0/\beta = |S|/\beta$, which is equal to the bound we must show.

The formal proof is by applying Wald's theorem to the sequence of random variables $T_i = n_{i-1} - n_i$, for $i = 1, 2, \ldots$ From (3.4), it follows that $\mathbf{E}[T_i \mid i \le T] \ge \beta$, and thus from Wald's theorem it follows that $\mathbf{E}[\sum_i T_i] \ge \beta \cdot \mathbf{E}[T]$. And since $\sum_i T_i = n_0 = |S|$, we obtain $\mathbf{E}[T] \le |S|/\beta$. □

Observe that as p approaches the popularity threshold p^*, factor β decreases, and thus the bound on the expected spread increases. Further, substituting the definition of p^* gives $\beta = 1 - \delta - p(\lambda - \delta)$, which implies that increasing either λ or δ increases the expected spread. These observations are consistent with the intuition.

Next we consider the spread of popular items. We focus on a particular family of random directed graphs which is convenient for our analysis, but is also a reasonable model of some social network graphs, such as the Twitter graph, characterized by large variation in the nodes' out-degree (i.e., the number of followers) and small variation in the nodes' in-degree. This model is a simplification of one considered in [41], and has a single parameter, a distribution ϕ on the nodes' out-degree.

Definition 7 (Random graph G_ϕ)

For any probability distribution ϕ on the set $\{0, \ldots, n-1\}$, G_ϕ is an n-node random directed graph such that the out-degrees of nodes are independent random variables with the same distribution ϕ, and for each node u, if u has out-degree k, then the set of u's outgoing neighbors is a uniformly random set among all k-sets of nodes not containing u.

We establish a lower bound on the probability of a popular item to be received by a constant fraction of users in G_ϕ, for an arbitrary distribution ϕ (under a mild constraint on the min out-degree). The above probability and the fraction size grow respectively with the number $\sigma = |S|$ of source nodes, and the popularity p of the item. In particular, the probability converges to 1 for σ larger than the average node degree μ.

Theorem 3 (Spread of popular items)

Let ϕ be any probability distribution on the set $\{\lceil \lambda + \delta \rceil, \ldots, n-1\}$, let $\epsilon, \epsilon' > 0$ be arbitrary small constants, and $1 \le \sigma \le n$ be an integer. Any information item with popularity $p \ge p^* + \epsilon$, that starts from a random initial set of σ nodes and spreads in G_ϕ using RIPOSTE, is received by at least $(1-\epsilon') \cdot \frac{\beta n}{\beta+1}$ users, with probability at least $1 - e^{-\Omega(\sigma/\mu)}$, where $\beta = (p - p^*)(\lambda - \delta)$ and μ is the mean of distribution ϕ.

Observe that the same constant $\beta = |p - p^*| \cdot (\lambda - \delta)$ appears in both Theorems 2 and 3. Unlike the bound of Theorem 2, the bound of $(1-\epsilon') \cdot \frac{\beta n}{\beta+1}$ in Theorem 3 is independent of the number $\sigma = |S|$ of source nodes; substituting the definitions of β and p^*, yields $\frac{\beta}{\beta+1} = 1 - \frac{1}{p\lambda + (1-p)\delta}$, thus the bound above increases when any of λ, δ, or p increases. The independence from σ is intuitively justified, because as long as the item reaches a "critical mass" of users, it will almost surely spread to a constant fraction of the network. However, the probability with which such a critical mass will be reached does depend on σ. For σ close to the average degree μ, this probability is at least a constant, and quickly converges to 1 as σ/μ increases above 1.

The proof of Theorem 3 uses a coupling between the dissemination process and an appropriate branching process, to show that the probability of the event we are interested in, that at least a certain fraction of users receive the item, is lower-bounded by the survival probability of the branching process. Then we bound this survival probability using a basic result for branching processes.

Proof of Theorem 3. It suffices to prove the claim for DB-RIPOSTE. The reasons is that the reposting probabilities $r_{\text{like}}(s)$ and $r_{\text{dis}}(s)$ are minimized when s equals the number k of the user's followers, and thus a standard coupling argument shows that the number of users that receive the item if DB-RIPOSTE is used is dominated stochastically by the same quantity when RIPOSTE is used.

We couple the diffusion process in G_ϕ with an appropriate branching process. Recall that a (Galton-Watson) branching process is a random process starting with one or more individuals, and in each step of the process a single individual produces zero of more offsprings and then dies. The number of offsprings of an individual follows a fixed probability distribution, the same for all individuals. The process either finishes after a finite number of steps, when no individuals are left, or continues forever. The probabilities of these two complementary events are called *extinction* and *survival probability*, respectively.

First we compute the distribution of the number of new users that learn the item from a user u, at a point in time when *fewer than ℓ users in total have received the item*—we will fix the value of ℓ later. The probability that u has exactly i followers is $\phi(i)$, for $i \in \{\lceil \lambda + \delta \rceil, \ldots, n-1\}$ (and 0 for other i). Given that u has i followers, the probability that DB-RIPOSTE reposts the item from u is $(p\lambda + (1-p)\delta)/i = (\beta+1)/i$. Further, by the principle of deferred decision, we can assume that if the item is reposted from u, only then are u's i followers chosen. We can also assume that they are chosen sequentially, one after the other, and the item is sent to a follower before the next follower is chosen (this does not change the overall outcome of the dissemination). Then the probability that the j-th follower of u has not already received the item is at least $1 - \ell/n$, provided that at most ℓ users already know the item (including the first $j-1$ followers of u).

Consider now the branching process in which σ individuals exist initially, and the number X of offsprings of an individual is determined as follows. First an integer i is drawn from distribution ϕ; then with probability $1-(\beta+1)/i$ we have $X = 0$ offspring, and with the remaining probability, $(\beta+1)/i$, we draw X's value from the binomial distribution $B(i,q)$, for $q := 1 - \ell/n$ (this is the distribution of the number of successes among i independent identical trials with success probability q).

We use a simple coupling of the diffusion process with the branching process above, until the point when ℓ users have received the item or the dissemination has finished (whichever occurs first). We assume that the diffusion process evolves in steps, and each step involves the execution of the DB-RIPOSTE algorithm at a single node. Similarly a step in the branching process is that a single individual reproduces and then dies. Let N_t denote the number of new users that learn the item in step t of the diffusion process, and let X_t be the number of offsprings born in step t of the branching process. From our discussion above on the distribution of N_t and from the definition of the distribution of $X_t \sim X$, it follows that we can couple N_t and X_t such that $N_t \geq X_t$ if no more than ℓ users in total have received the item in the first t steps.

From this coupling, it is immediate that the probability at least ℓ users receive the item in total, is lower-bounded by the probability that the total progeny of the branching process (i.e., the total number of individuals that ever existed) is at least ℓ. Further, the latter probability is lower bounded by the *survival probability* of the branching process; we denote this survival

probability by ζ_σ. Thus to prove the theorem it suffices to show

$$\zeta_\sigma = 1 - e^{-\Omega(\sigma/\mu)},$$

for $\ell := (1-\epsilon') \cdot \beta n/(\beta+1)$. The remainder of the proof is devoted to that.

By the definition of the branching process, the expected number of offsprings of an individual is

$$\begin{aligned}\mathbf{E}[X] &= \sum_i \phi(i) \cdot \frac{\beta+1}{i} \cdot \mathbf{E}[B(i,q)] \\ &= \sum_i \phi(i) \cdot \frac{\beta+1}{i} \cdot iq = \sum_i \phi(i) \cdot (\beta+1) \cdot q = (\beta+1) \cdot q.\end{aligned}$$

We observe that $\mathbf{E}[X] > 1$, as

$$(\beta+1) \cdot q = (\beta+1) \cdot \left(1 - \frac{(1-\epsilon')\beta}{\beta+1}\right) = 1 + \epsilon'\beta. \tag{3.5}$$

Further,

$$\begin{aligned}\mathbf{E}[X^2] &= \sum_i \phi(i) \cdot \frac{\beta+1}{i} \cdot \mathbf{E}[(B(i,q))^2] = \sum_i \phi(i) \cdot \frac{\beta+1}{i} \cdot (i^2q^2 + iq(1-q)) \\ &= \sum_i \phi(i) \cdot (\beta+1) \cdot (iq^2 + q(1-q)) = (\beta+1) \cdot (\mu q^2 + q(1-q)),\end{aligned}$$

where $\mu = \sum_i \phi(i) \cdot i$ is the mean of ϕ. We will use the following standard lower bound on the survival probability ζ_1, when there is just one individual initially (see, e.g., [75, Sect. 5.6.1]),

$$\zeta_1 \geq \frac{2(\mathbf{E}[X]-1)}{\mathbf{E}[X^2] - \mathbf{E}[X]}.$$

Substituting the values for $\mathbf{E}[X]$ and $\mathbf{E}[X^2]$ computed above yields

$$\begin{aligned}\zeta_1 &\geq \frac{2(q(\beta+1)-1)}{(\beta+1)(\mu q^2 + q(1-q)) - q(\beta+1)} = \frac{2(q(\beta+1)-1)}{q^2(\beta+1)(\mu-1)} \\ &= \frac{2(q(\beta+1)-1)(\beta+1)}{q^2(\beta+1)^2(\mu-1)} \overset{(3.5)}{=} \frac{2\epsilon'\beta(\beta+1)}{(1+\epsilon'\beta)^2(\mu-1)} = \Omega(1/\mu),\end{aligned}$$

where the final equation holds because $\beta = (p-p^*)(\lambda-\delta) \geq \epsilon(\lambda-\delta) = \Omega(1)$.

We can now express ζ_σ in terms of ζ_1, by observing that a branching process starting with σ individuals can be viewed as σ independent copies of the branching process starting with a single individual each. [3] The former branching process survives if and only if at least one of the latter ones survives, thus,

$$\zeta_\sigma = 1 - (1-\zeta_1)^\sigma \geq 1 - e^{-\zeta_1\sigma} = 1 - e^{-\Omega(\sigma/\mu)}.$$

This completes the proof of Theorem 3. □

3.5 Experiments and Evaluation

In this section we provide experimental evaluation of the dissemination achieved by RIPOSTE on some real topologies of online social networks. The results are surprisingly consistent with our analysis, even though some of the analytical results were proven only for an ideal random graph model.

3. This is true for any branching process, and does not relate to the original diffusion process.

Datasets. We use the network topologies listed in Table 3.1. The Twitter dataset is a complete snapshot of the network from 2009 [88], while the other datasets are partial network samples. Twitter is a micro-blogging network service, LiveJournal is a blogging network site, while Facebook, Renren, and Google+ are online social networking sites. In each of these networks, every user maintains a list of friends, and/or a list of users she follows. The friendship relation is typically reciprocal, whereas the follower relation is not; the former is represented as an undirected edge, and the latter as a directed. In Twitter, LiveJournal and Google+ edges are directed, while in Renren and Facebook undirected.

Network	Nodes	Edges	Avg-deg	Source
Twitter	41.65M	1468M	35.2	[88]
LiveJournal	4.847M	68.99M	14.2	[17, 94]
Facebook	3.097M	23.66M	15.3	[145]
Renren	965.3K	57.56M	59.6	[54]
Google+	107.6K	13.67M	127	[95]

Table 3.1 – Network topologies used in the experiments. By avg-deg we denote the average degree of the network.

Setup. We consider the following protocols: (1) RIPOSTE, with exact information on the number of non-informed followers, i.e., s is the *actual* number of the user's followers that do not know the item yet—not just an upper bound; (2) DB-RIPOSTE, where no information about the followers status is available, and thus s is the total number of followers; (3) the basic non privacy-conscious protocol where a user reposts an item if she likes it and does not repost it if she does not like it; we refer to this as the STANDARD protocol.

While datasets on social network topologies are publicly available, access to user's activity, including the list of items they post, receive, like or repost, is severely restricted. Therefore, for our evaluation we rely on two synthetic models to generate users' opinions: (i) the *uniform opinion model*, where every item is assigned a popularity $p \in [0,1]$, and each user likes the item independently with probability p—this is the same model used in the analysis (see Definition 5); and (ii) the *distance-threshold opinion model*, where a user likes the item precisely if the (shortest-path) distance from a source to the user is at most some threshold h. The latter model is motivated by the principle that users close to each other tend to have similar opinions [107].

In all experiments, we choose the set S of users who know the item initially to be the followers of a random user, among all users with at least μ followers, where μ is the average degree. We think that this is more realistic than choosing an arbitrary or random set S: It is often the case that the source of the information (e.g., a news channel) is itself a node in the online social network; then the followers of that node constitute the set S of nodes exposed to the information initially. For each point in the plots we present, we average the results over 10,000 random independent experiments, with a new random set S each time. For the RIPOSTE algorithm, where the dissemination may depend on the order in which the protocol is executed at different users, we experimented with both breadth-first and depth-first orders, obtaining very similar results.

Results. Fig. 3.3 shows the average number of users that receive the item when using RIPOSTE, as a function of the item popularity, for all networks (for parameters $\lambda = 3$ and $\delta = 0.75$).

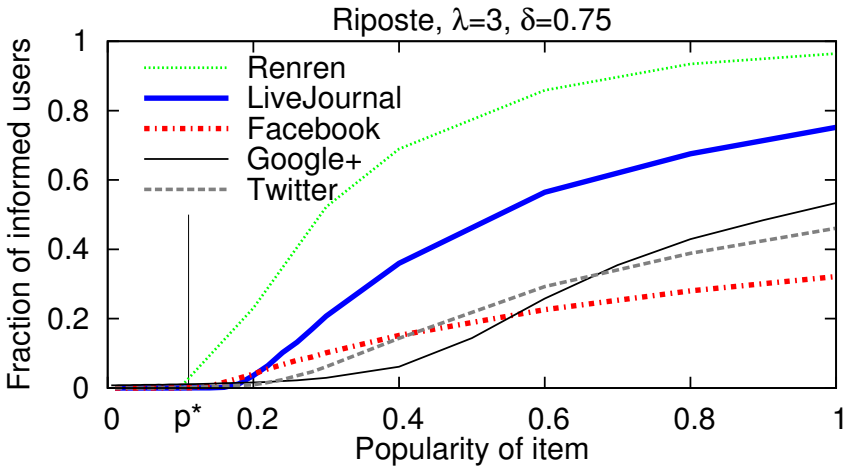
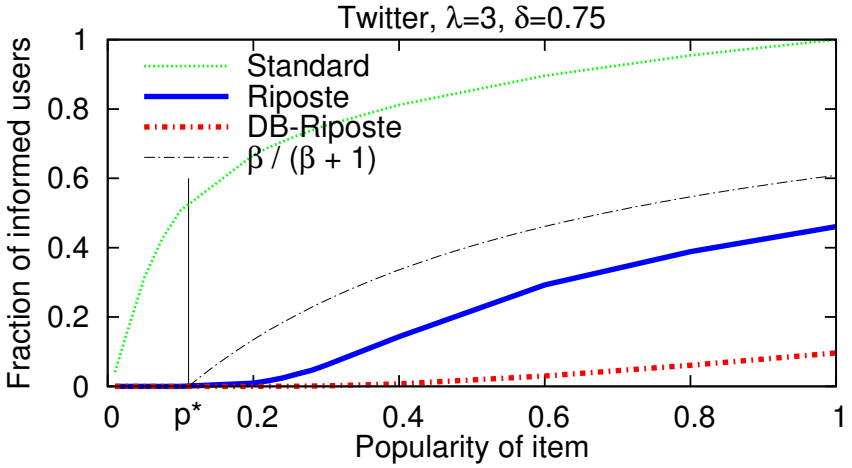
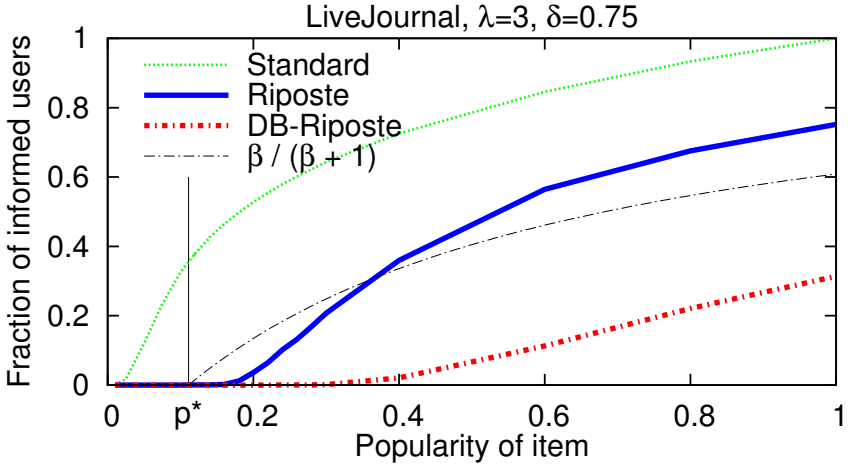


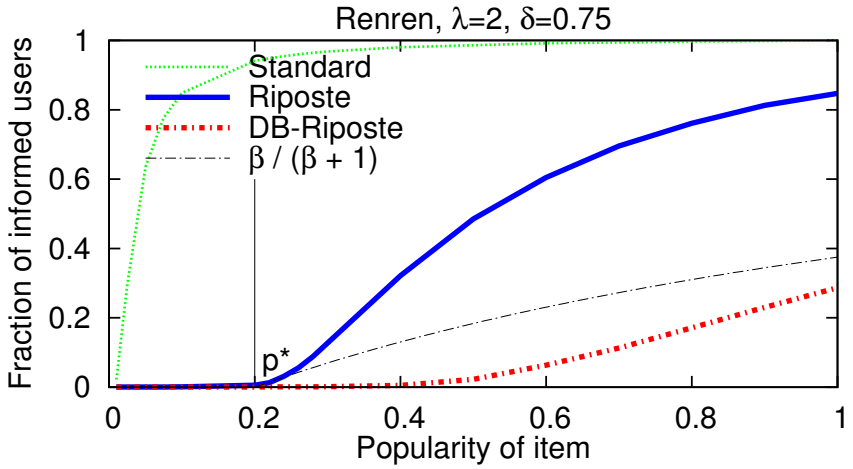
Figure 3.3 – Dissemination for RIPOSTE *as a function of the item popularity.*



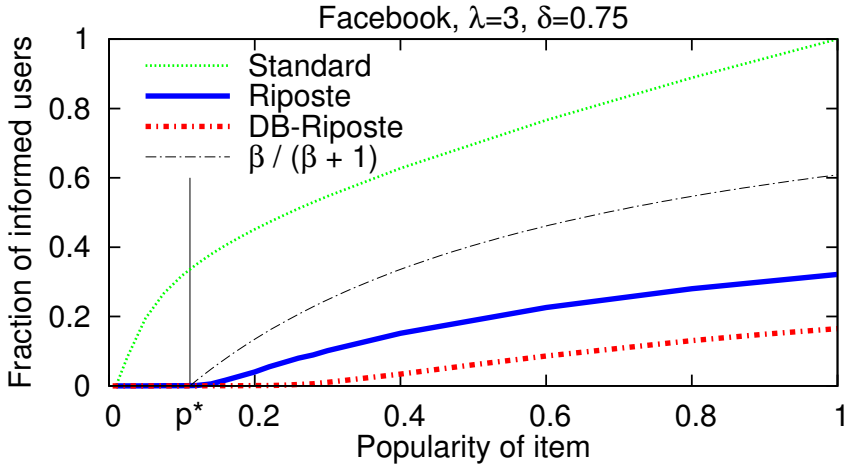
(a)



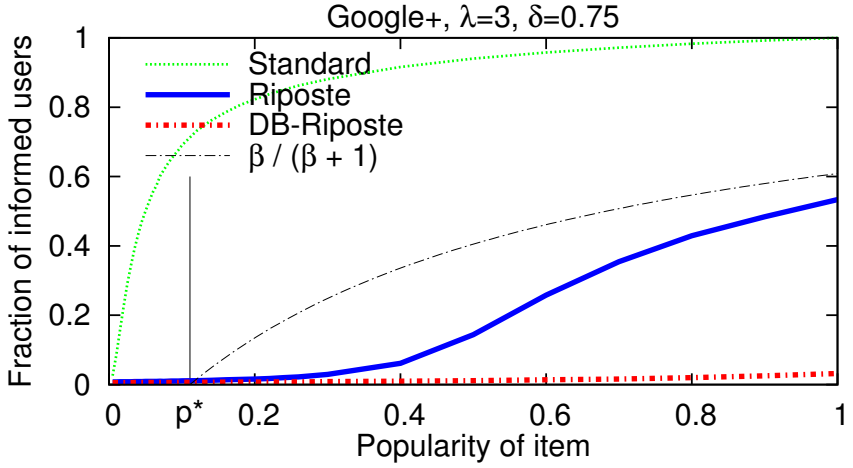
(b)



(c)



(d)

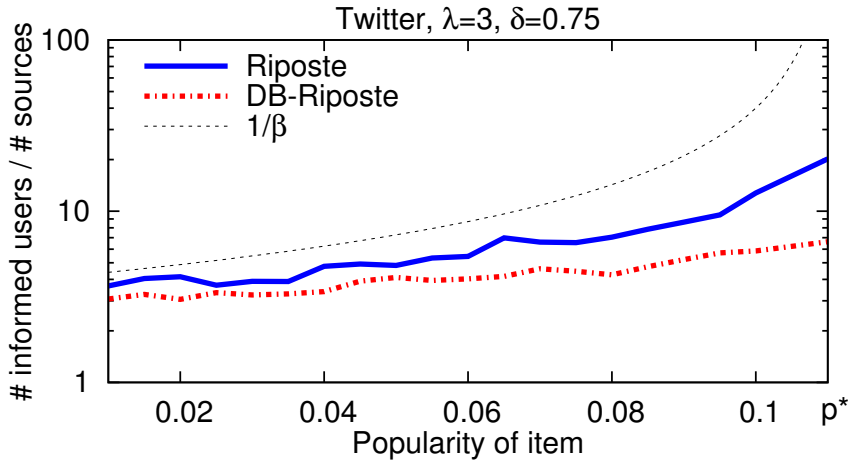


(e)

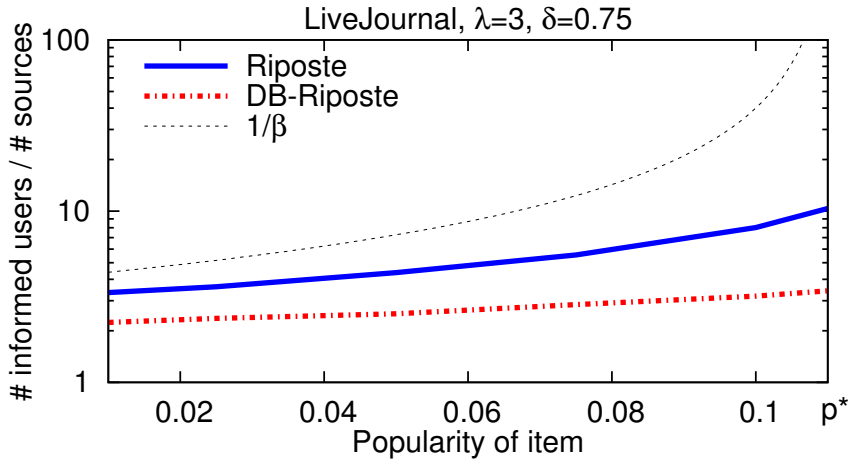
Figure 3.4 – Dissemination in Twitter, LiveJournal, RenRen, Facebook and Google+. Comparison with STANDARD *and the* $\beta/(\beta+1)$ *lower bound of Theorem 3.*

In all cases, unpopular items (with popularity p below the threshold p^* identified by our analysis) have very limited spread, while popular items (with $p > p^*$) spread to a fraction of the networks that grows quickly with p.
Fig. 3.4 compares the dissemination using RIPOSTE to that of DB-RIPOSTE and STANDARD, and also to the lower bound for the spread of popular items predicted by Theorem 3. As expected, DB-RIPOSTE informs fewer users than RIPOSTE but has overall qualitatively similar behavior. STANDARD achieves significantly wider dissemination, even for items with very low popularity, which may be undesirable. The $\beta/(1-\beta)$ bound of Theorem 3 is relatively close to the curve for RIPOSTE (slightly above it in the case of Twitter and intersecting it in the case of LiveJournal).

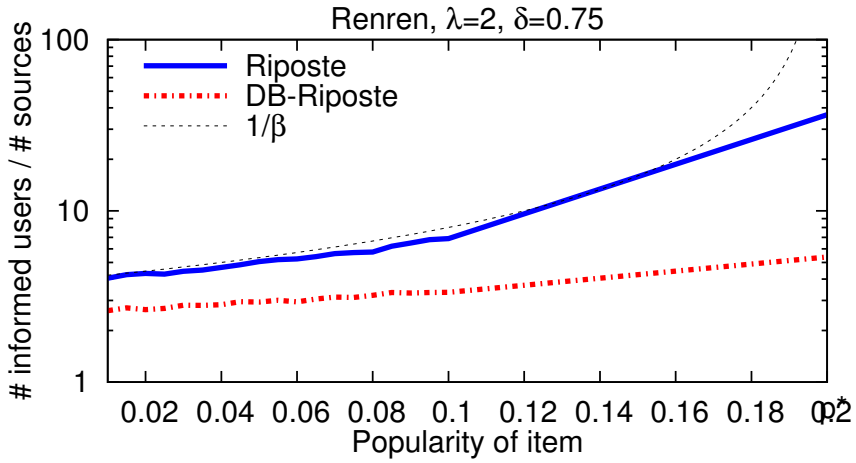
This lower bound was derived for an idealized random graph model, so it is reasonable that it does not apply exactly to the real topologies considered. On the other hand, the $1/\beta$ upper bound for unpopular items of Theorem 2 holds for *any* graph, and Fig. 3.5 shows that it indeed bounds the dissemination with RIPOSTE in all the datasets considered in our work.



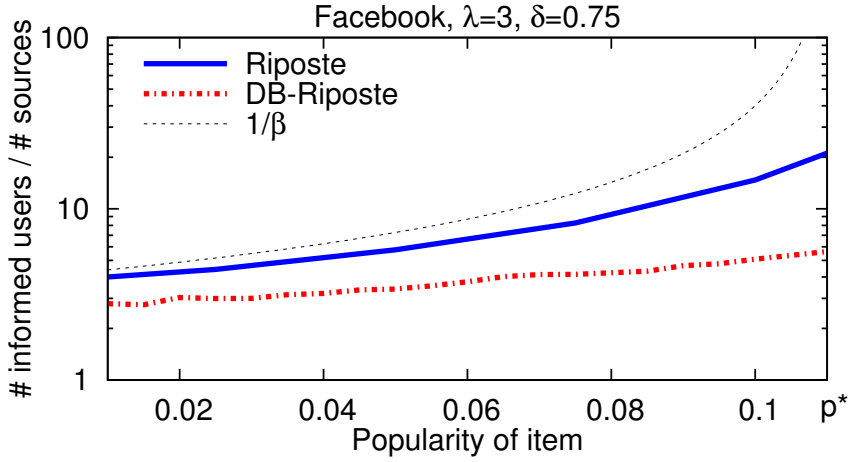
(a)



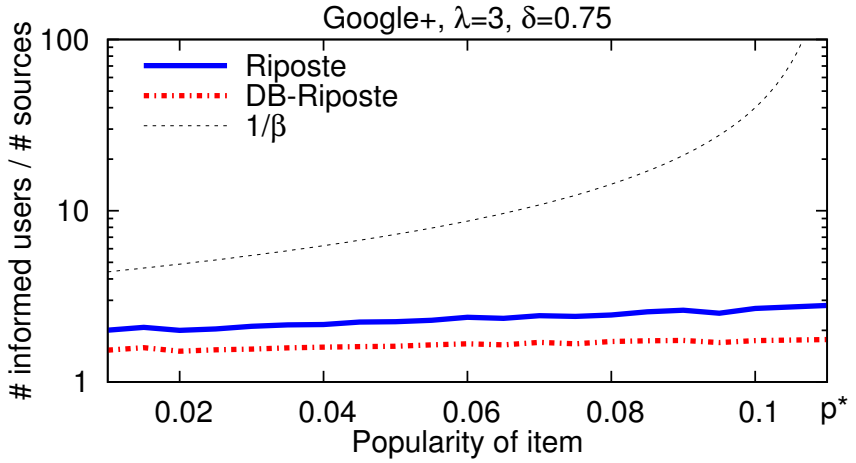
(b)



(c)



(d)

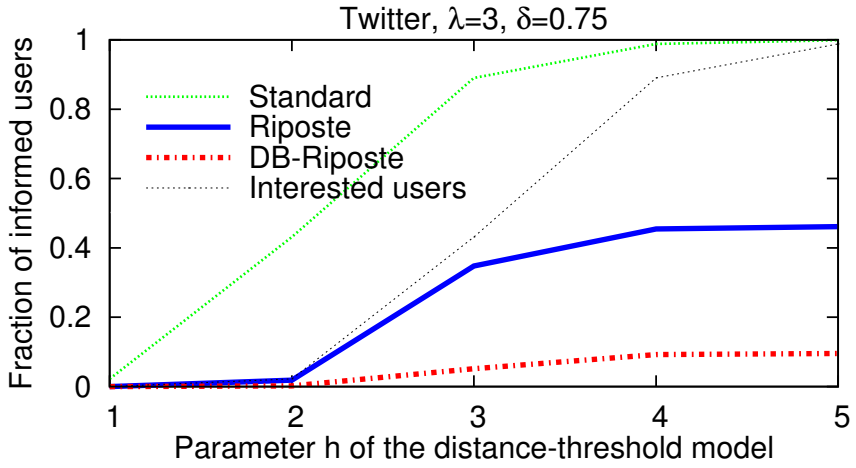


(e)

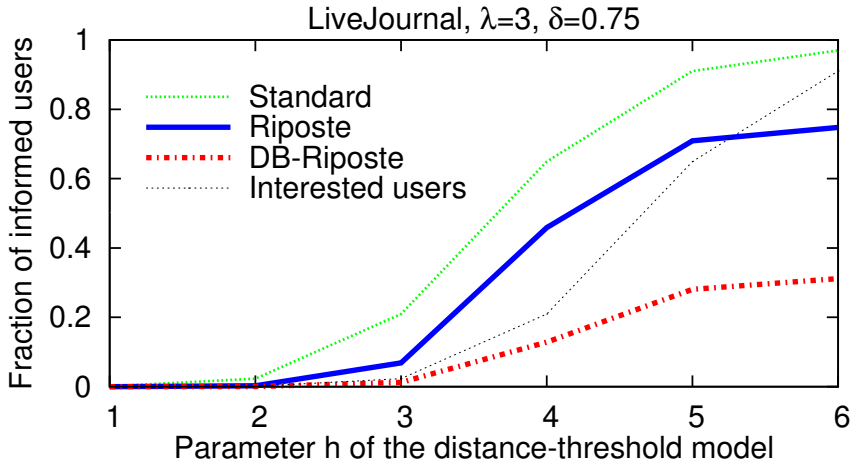
Figure 3.5 – Dissemination in Twitter, LiveJournal, RenRen, Facebook and Google+. Comparison with the $1/\beta$ upper bound of Theorem 2 for unpopular items.

Finally, Fig. 3.6 presents the same results as Fig. 3.4b but for the distance-threshold opinion model.

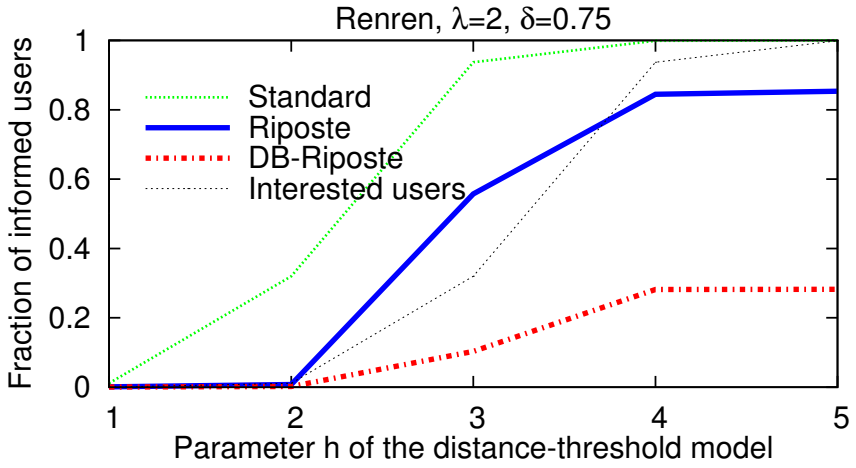
We observe that RIPOSTE achieves spread to a fraction of users that is relatively close to the fraction of users that like the item. As before, STANDARD may spread the item to a fraction significantly larger than the fraction that likes the item, in particular, for items that not many users like.



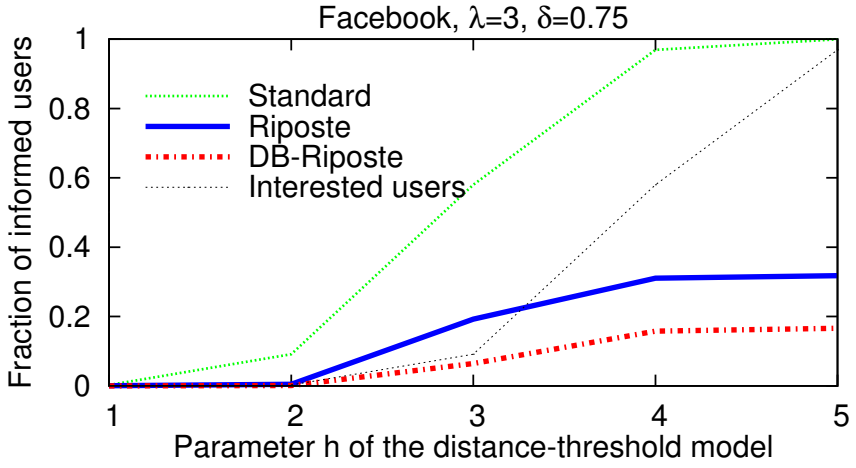
(a)



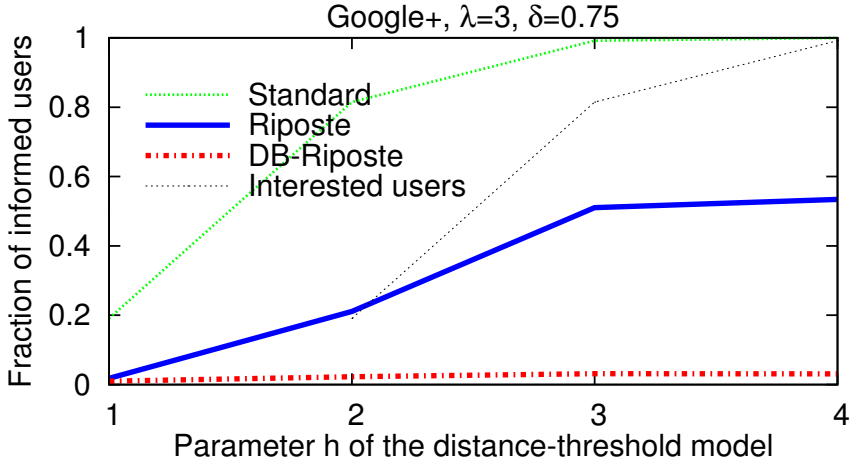
(b)



(c)



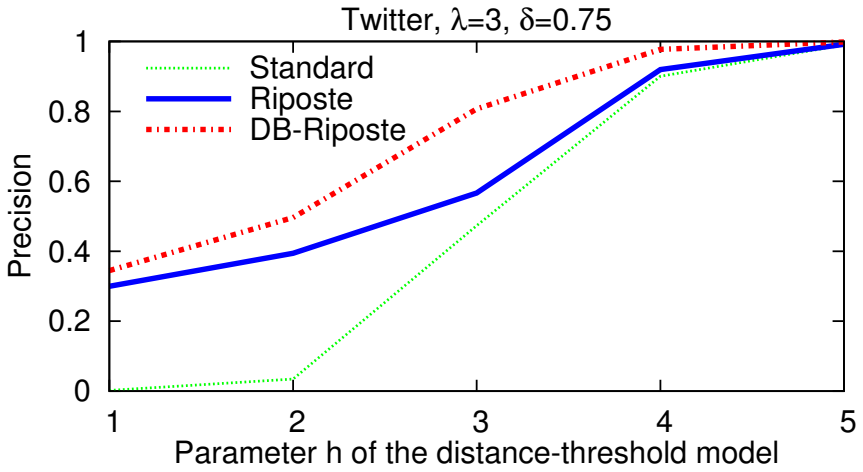
(d)

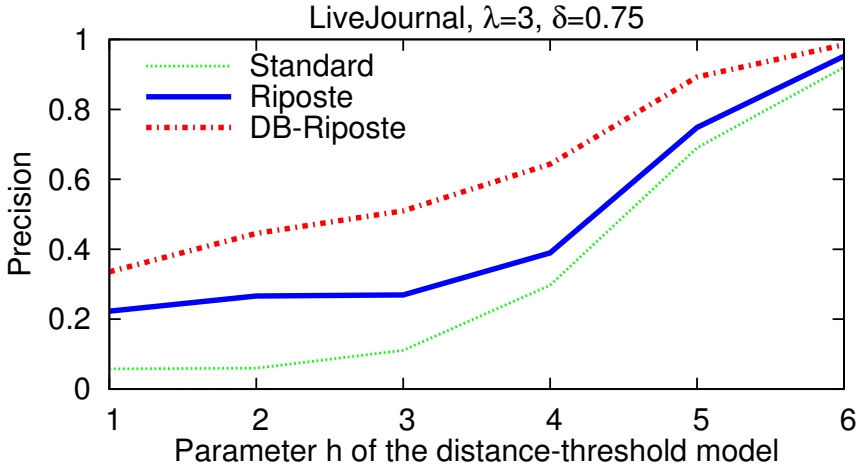


(e)

Figure 3.6 – Dissemination in Twitter, LiveJournal, RenRen, Facebook and Google+. Distance-threshold model (all users within distance h from the source, and only them, like the item).

Moreover, Fig. 3.7 shows that a large fraction of the nodes that receive the item are indeed ones that like it in case of RIPOSTE, and even more for DB-RIPOSTE.





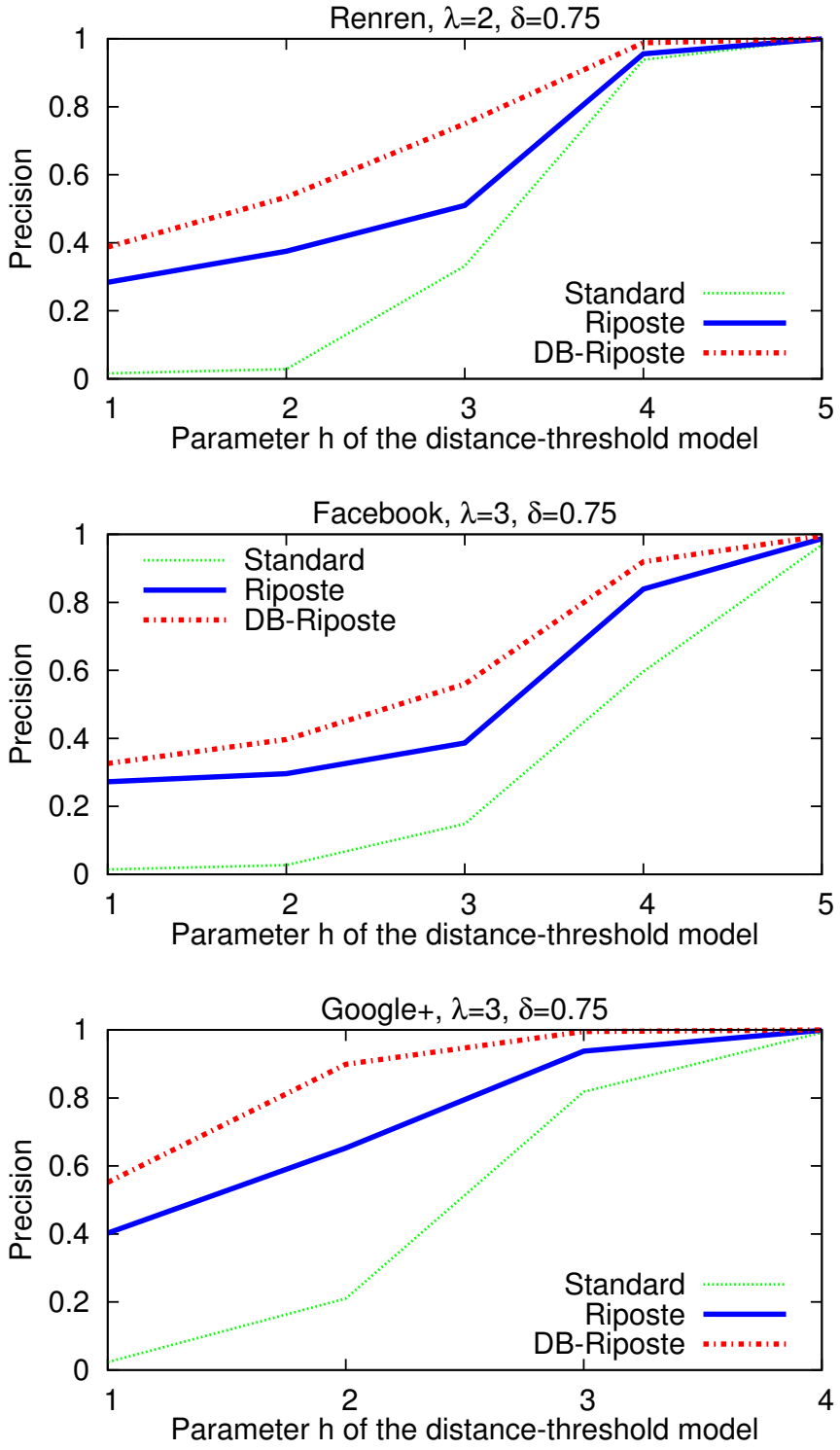


Figure 3.7 – Fraction of users that like the item among all uses that receive the item (a measure known as precision*), in the distance-threshold opinion model.* RIPOSTE *and* DB-RIPOSTE *achieve significantly better precision that* STANDARD *for smaller values of h.*

To conclude the section of experiments and evaluation, we present the effect on dissemination by varying our parameters, i.e., λ and δ. We illustrate this variation in Fig. 3.8 This graph (with more varied values) is also what we used to fix our parameters for both, theoretical and practical purposes.

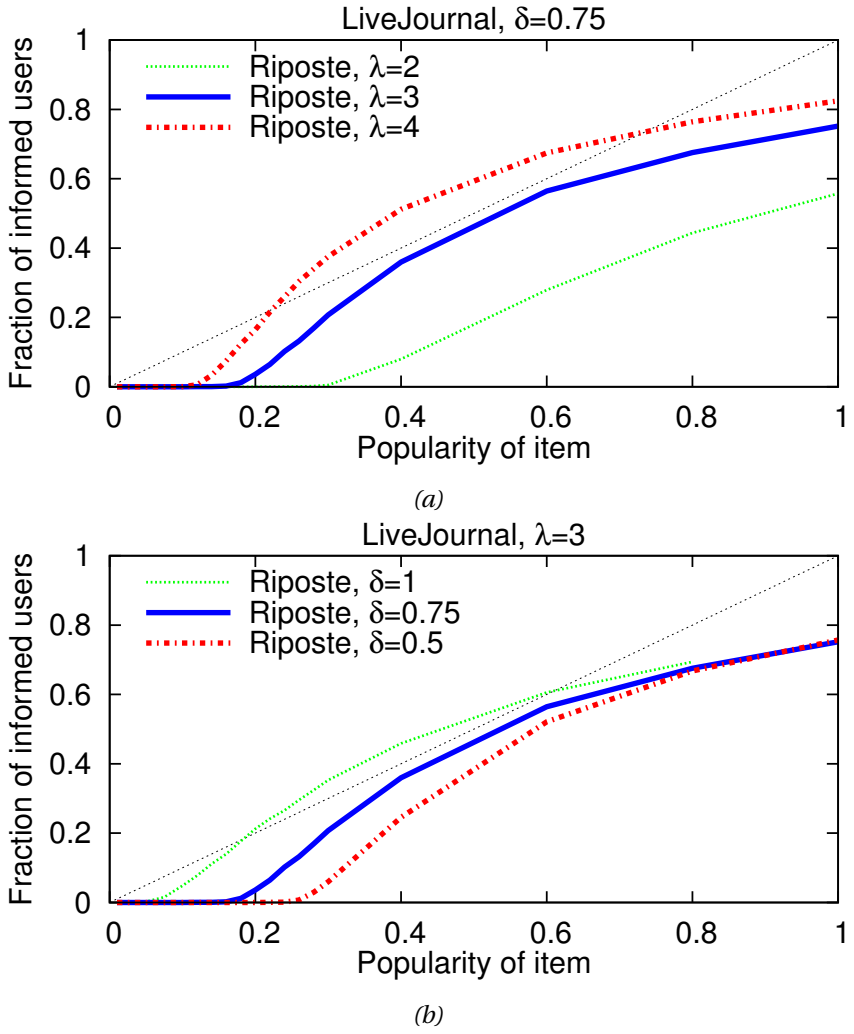


Figure 3.8 – Effect of variation in parameters λ and δ on the dissemination achieved with RIPOSTE *in LiveJournal. (a) Variation in λ for fixed $\delta = 0.75$. (b) Variation in δ for fixed $\lambda = 3$. Increasing either parameter decreases the popularity threshold and increases the spread.*

3.6 Conclusion

We have presented a simple and local diffusion mechanism for social networks, which guarantees widespread dissemination of interesting but possibly sensitive information, in a privacy-conscious manner. The mechanism randomizes the user's action of reposting (or not) the information, in a way reminiscent of the randomized response technique, and chooses the probabilities so that a branching-process-like phenomenon takes place: if more than a certain fraction of people like the information then a large cascade of reposts is formed, and if fewer people like it then the diffusion process dies quickly.

4

Fast and Efficient *k*-Nearest Neighbor Graph Construction

You are ... the people you surround yourself with

Matrin Jefferson Junior

Necessarily all recommendation systems rely on a learning model that can use the user-item data to learn and predict ranked preferences of users. There are innumerable ways of designing and implementing these learning models. These models are driven by a machine learning algorithm behind them. We have mentioned some of these most commonly used algorithms in Chapter 2.3 KNN is among one of the most popular algorithms, serving as a back bone of recommendation systems. It also finds its use in other applications like Information Retrieval, Image and Text Classification, Similarity search etc. Its simplicity of implementation and wide spread use in recommendation systems is one of the driving forces behind our contribution.

In this chapter, we present KIFF: a **generic**, **fast** and **scalable** KNN graph construction algorithm. Recommendation systems are based on user-item relationships, where a user can be interested in one or more items in the system, therefore, leading to a system that can be easily viewed as a bipartite graph. As a result, KIFF directly exploits the bipartite nature of most datasets to which KNN algorithms are applied. This simple but powerful strategy drastically limits the computational cost required to rapidly converge to an accurate KNN solution, especially for sparse datasets. We use vivid set of experiments and simulations to quantify the actual gains in terms of speed-up and reduction in computation costs that KIFF achieves as compared to its competitors : some of the most powerful state-of-the art approaches.

To systematically present KIFF, we present an elaborated Introduction to KIFF in Section 4.1. Section 4.2 gives a detailed schema of the technical contribution, starting with some background knowledge. Later we formally present KIFF in Section 4.3 by introducing a formal problem definition. The choice of parameters and data structures is also explained in this section. The experiments are presented in Section 4.4 and their evaluations and analysis in the subsequent Section 4.5. We conclude this contribution in Section 4.6, by finally giving some future perspectives into the work.

4.1 Introduction

Recommendation systems lead to relations between its users and items. In a typical setting, a user can be associated to one or more items in the system. To fulfill the purpose of providing a user with recommendations, a classic way is to rely on knowledge of neighbors and the items that they are interested in. This leads to the problem of finding neighbors for every user in the system : a problem quintessentially called the *knn graph construction problem*.

K-Nearest-Neighbor (KNN) graphs play a fundamental role in many web-based applications e.g., search [18,22], recommendation [29,96,98,119] and classification [113]. A KNN graph is a directed graph of entities (e.g., users, products, services, documents etc.), in which each entity (or *node*) is connected to its k most similar counterparts or *neighbors*, according to a given *similarity metric*. In contemplation of computing the similarity between entities, a well-chosen similarity measure is used, from a list of various available measures depending on various factors like the size and type of entities involved, the targeted application etc. Some examples of similarity measures from literature include Jaccard, Cosine, Pearson and Dice similarity. It turns out that in a large number of applications, this similarity metric is computed from a second set of entities (termed *items*) associated with each node in a bipartite graph (possibly extended with weights, such as ratings or frequencies). For instance, in a movie rating database, nodes are users, and each user is associated with the movies (items) she has already rated. **A simple recommendation service can be implemented by constructing the KNN graph of users using Jaccard's coefficient or Cosine similarity [140], two typical item-based similarity metrics, and using a collaborative filtering approach on the resulting neighborhoods.**

As data volumes continue to grow, constructing a KNN graph efficiently remains an ongoing and open challenge [1] A brute force computation of a KNN graph requires $O(n^2)$ similarity computations, which does not scale, in particular when nodes are associated with large item sets, with direct implication for a raft of concrete applications such as recommendations, feature extraction, Principle Component Analysis among many others.

To address this problem, most practical approaches approximate the actual KNN and deliver an Approximate-Nearest-Neighbor graph, or ANN. This approximation can take several forms, which can often be combined. One such line of attack uses dimension reduction techniques to lower the cost of computing individual similarities [24, 67, 100].

Another complementary strategy avoids an exhaustive search altogether and executes a greedy, localized search to converge rapidly towards a KNN or ANN graph [22, 29, 56, 83, 105, 143]. These greedy approaches start from a random k-degree graph and exploit the fact that similarity functions tend to behave transitively over neighbors-of-neighbors links: if A is similar to B, and B to C, then A is likely to be similar to C. This greedy procedure is often completed by a low-probability random exploration of arbitrary nodes to avoid local minima [22,29,143,144].

Some greedy KNN approaches have recently been found to perform particularly well [29, 56]. These approaches tend, however, to induce a substantial number of similarity computations, which directly impact their computing time. This point is illustrated in Figure 4.1 for two characteristic greedy KNN-graph algorithms, NNDescent [56], and HyRec [29]. The figure shows the breakdown, in seconds, of the computing time of both algorithms on a small Wikipedia dataset [93] containing 6,110 users (i.e. nodes), and 2,381 items, using the standard cosine similarity metric. On average, both approaches spend more than 90% of their total execution time repeatedly computing the similarities values (large checkered bars in the figure).

In this work, we propose to reduce this substantial cost by exploiting two observations that apply to most item-based similarity metrics: first, two nodes must usually share some items to

1. Note that the problem of computing a complete KNN graph is related but different from that of answering a sequence of KNN queries, a point we revisit when discussing related work.

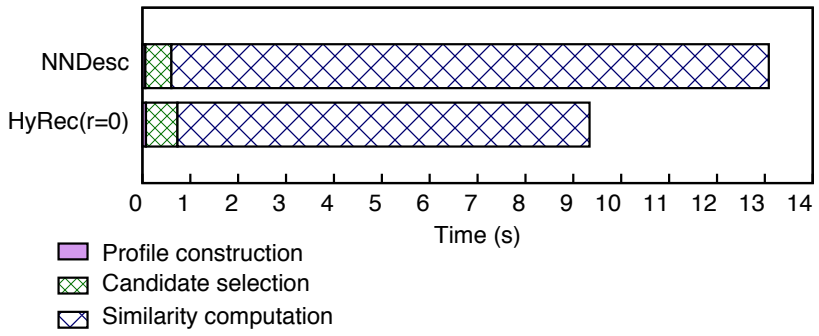


Figure 4.1 – State-of-the art greedy KNN-graph approaches spend over 90% *of their computation time computing similarity values (Wikipedia dataset).*

be KNN neighbors; second, the more items two nodes have in common, the more likely they are to appear in each other's KNN neighborhood. Based on these observations, our solution, KIFF (*K-nearest neighbor Impressively Fast and eFficient*) first uses the node-item bipartite graph to compute a coarse but very cheap approximation of the similarity between two users. KIFF then uses this rough approximation to orient and to prune the search for a good KNN approximation using a greedy procedure, rather than starting from a random graph as many greedy solutions do [22, 29, 143, 144]. It then refines this initial approximation in a greedy iterative procedure based on the information extracted from the bipartite graph to iteratively refine this initial solution.

The result is a novel, fast and scalable KNN graph construction algorithm that produces a very close approximation of the real KNN.

4.2 Contribution: Fast and Efficient KNN Graph Construction

Greedy KNN-graph approaches have been shown to offer a promising alternative to more traditional KNN-graph construction techniques, for instance based on Locality-Sensitive Hashing (LSH) or Recursive Lanczos Bisection (RLB) [39, 56, 67, 100]. Greedy approaches incrementally improve an approximation of the KNN graph, and thus avoid an exhaustive $O(n^2)$ search among potential KNN edges. They perform well for two main reasons: they are inherently parallel, and show a strong memory locality. As shown in Figure 4.1, they tend, however, to induce a substantial number of similarity computations, which directly impact their computing time.

We propose to reduce this time substantially by limiting the number of similarity values a greedy approach needs to compute. Our intuition is based on two simple observations: (i) almost all similarity metrics used to construct KNN graphs (cosine, Jaccard's coefficient, Adamic-Adar's coefficient) return zero when two users [2] do not share any items ; and (ii) these metrics usually contain a term that grows with the number of items two users have in common.

Counting common items is typically much cheaper computationally than computing full blown similarity metrics. This is because (i) common items are usually a first step of more advanced similarity metrics ; and (ii) most similarity metrics use floating point operations (divisions, square roots, logarithms) that are much more expensive than the simple integer arithmetic involved in counting items. Our approach, therefore, employs a simple but powerful strategy: we use common item counts as a first coarse approximation of the similarity between

2. For ease of exposition, we will assume for the rest of the paper that nodes are users who have rated items. Our approach can, however, be applied to any type of nodes, as long as node similarity is computed from items associated with these nodes.

two users to prune the pairs of users who have no items in common. We then refine this approximation by iterating over potential KNN neighbors in reverse order of this item count.

Directly comparing every pair of users to count common items would, however, scale poorly. We, therefore, use an indirect method: we exploit the bipartite nature of the user-item graph (users are linked to items, and items to users) as an efficient structure to construct lists of users sharing items.

In depth, our approach (called KIFF) works in two phases, called the *counting* and the *refinement* phase. In the counting phase, KIFF preprocesses the user-item bipartite graph and builds a *Ranked Candidate Set* (RCS) for each user. Ranked candidate sets are ordered weighted sets that bring together users who share items, while counting how many items these users have in common. Ranked candidate sets are used in the refinement phase to initiate, and then iteratively refine the KNN approximation of each user. We describe both phases informally in the following on a small example, before providing a more formal presentation in the next section.

4.2.1 Counting Phase

The *Users* column of Figure 4.2 represents a toy user-item dataset in which the users of an online social network are associated with the objects or activities that they have liked (the items). Alice likes books and coffee, Bob coffee and cheese, and Carl and Dave like shopping. The items a user likes make up this user's *profile*. Here, Alice's profile (UP_{Alice}) contains {book, coffee} (Label **1**).

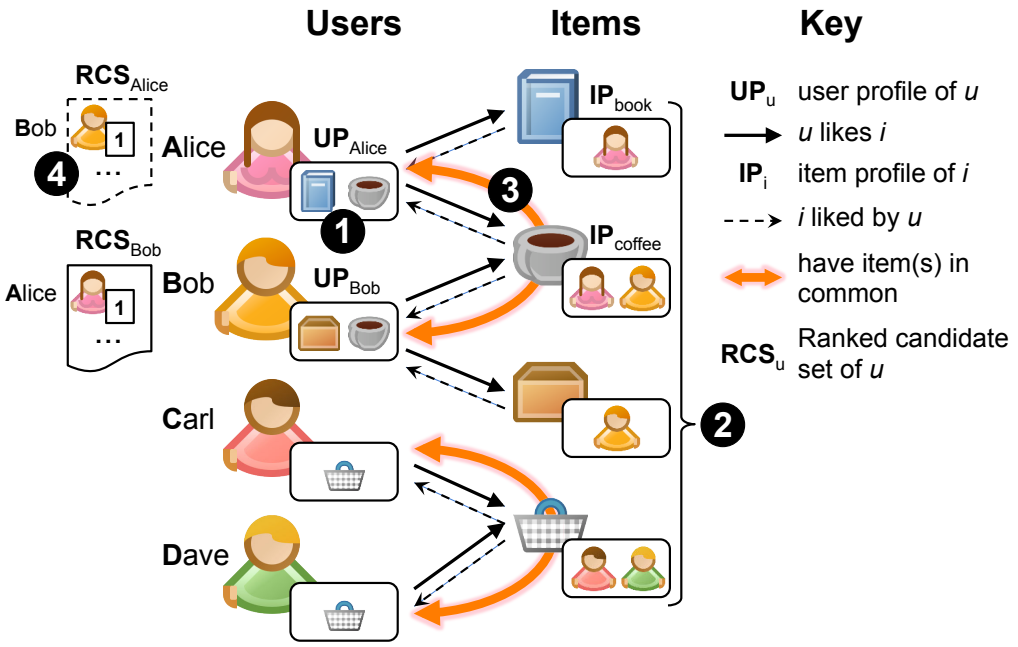


Figure 4.2 – KIFF *exploits the bipartite nature of user-item datasets to pre-compute* Ranked Candidate Sets *(noted* RCS_u*).*

The role of the counting phase is to compute the ranked candidate set (RCS) of each user. Because Alice and Bob share one common item (coffee), KIFF should add each to the RCS of the other with a count of 1 [3]. To do so, KIFF first computes the item profiles (IP_i) of each item

3. We explain later on how we avoid the redundancy caused by this symmetry.

(Label **2**), i.e., the set of users who have liked a particular item i. For instance, here, the item profile of coffee (IP_{coffee}) contains Alice and Bob, while the item profile of book (IP_{book}) contains only Alice. Item profiles reverse the node-to-item links (solid arrows) of the initial dataset into item-to-node edges (dashed arrows). Item profiles also provide a crude hashing procedure, in which users are binned into as many item profiles (which acts as buckets) as the items they possess. As in LSH, these "buckets" are used to find KNN candidates.

Once the item profiles have been computed, KIFF constructs Alice's ranked candidate set (RCS_{Alice}) by navigating the bipartite graph induced by user and item profiles (Label **3**) in order to collect all users who share an item with Alice: here Bob. The count associated with Bob in RCS_{Alice} (Label **4**) is the number of common items between Bob and Alice (1 for coffee), i.e., the number of paths from Alice to Bob in the user-item graph.

4.2.2 Refinement Phase

Let's now imagine that the dataset of Figure 4.2 has been enriched with new items, so that Alice's ranked candidate set ends up containing the following users:

$$RCS_{\text{Alice}} = \{(\text{Bob}, 10), (\text{Carl}, 9), (\text{Dave}, 8), (\text{Xavier}, 6), (\text{Yann}, 3), \ldots\}$$

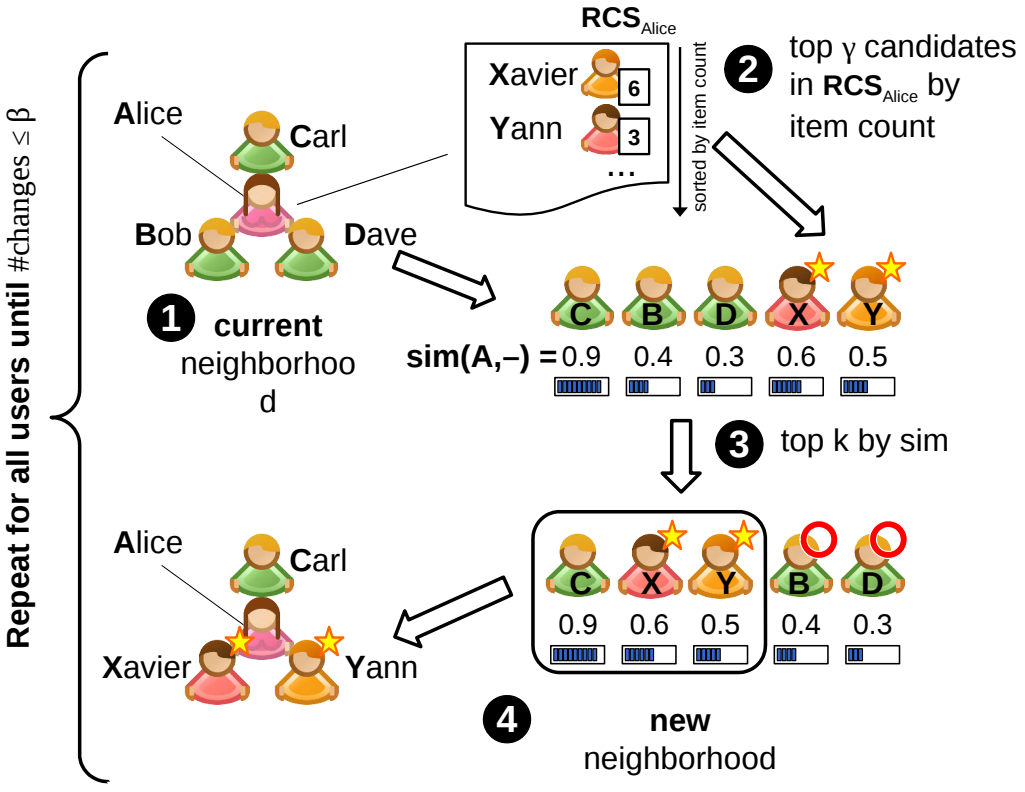


Figure 4.3 – The pre-computed Ranked Candidate Sets *are used in an iterative greedy process to construct an approximation of the KNN.*

KIFF first fills up Alice's neighborhood with the top k users of her RCS, i.e., the k users with whom Alice shares the most items: in our example Bob, Carl, and Dave (if we assume $k = 3$). These k users are removed from the RCS.

KIFF then repeatedly applies the procedure depicted in Figure 4.3 to iteratively construct an approximation of Alice's KNN, and works in 4 main steps (black circles in the figure). KIFF first merges the current neighborhood of Alice ({Carl, Bob, Dave}, Label **1**) with the current top γ neighbors found in Alice's RCS (RCS_{Alice}, Label **2**). In the example, $\gamma = 2$, but we generally use a γ twice the value of k.

Alice's current two top users in her RCS are Xavier with 6 items in common, and Yann with 3 items. Both Xavier and Yann are removed from RCS_{Alice} in the process, and added to Alice's current neighborhood.

The union of Alice's current neighborhood {Carl, Bob, Dave}, with the γ users extracted from RCS_{Alice} ({Xavier, Yann}) form the potential new neighborhood of Alice in this iteration. From then on, the procedure is very similar to that of other greedy approaches [22, 29, 56, 143, 144]: the potential new neighbors are ranked according to their similarity with Alice (Label **3**), and the top-k users (here Carl, Xavier, and Yann) are selected as Alice's new top-3 neighbors (Label **4**). Note in this example how Yann, with a common item count of 3, has a higher similarity to Alice than Bob, with a common item count of 10. This is because a similarity metric (e.g. cosine, Jaccard's coefficient) although it tends to be linked to the number shared items is not entirely determined by this count. This whole procedure is repeated for all users until the total number of neighbors changes during an iteration is smaller than a fixed termination parameter β.

4.2.3 Optimization and Discussion

In the basic procedure we have described, Ranked Candidate Sets are heavily redundant: if Bob is in Alice's RCS, then Alice is in Bob's. To limit the memory overhead of KIFF, and the cost of maintaining individual RCSs, we therefore use a pivot strategy: if Alice and Xavier have items in common, only the user with the lower ID (e.g., Alice) will store the other user in his/her RCS. The procedure of Figure 4.3 is adapted accordingly, so that, Alice is also considered as a potential neighbor for Xavier (and Yann).

In a second optimization, we simplify the initialization of neighborhoods at the start of the refinement phase: rather than handling this initialization as a special case, we treat it as a standard iteration that starts from empty neighborhoods that are filled up using the procedure of Figure 4.3.

In terms of computing time, KIFF trades off the cost of constructing item profiles and ranked candidate sets (Counting Phase, Figure 4.2), for the ability to limit similarity computations between users which (i) are guaranteed to have some items in common, and (ii) are considered in the reverse order of their shared item counts. In the example of Figure 4.3 for instance, Carl and Dave will never be considered as potential neighbors for Alice because they do not share any items. This trade-off pays off if item profiles (IP_{book}, IP_{coffee}) and ranked candidate sets (RCS_{Alice}, RCS_{Bob}) tend to be small.

This advantageous situation corresponds to datasets in which users rate few items from a large item set. These datasets are precisely those problematic for greedy KNN approaches such as HyRec and NN-Descent [29, 56]. That is because greedy approaches start from an initial random graph in which (i) initial neighbors are unlikely to be similar, and (ii) neighbors of neighbors will also tend to show a low similarity, resulting in a slow convergence.

4.3 The Algorithm: KIFF

We now present formally the working of KIFF, starting with some notations (Section 4.3.1) and a formal problem definition (Section 4.3.2), before moving on to the algorithm itself (Sec-

tions 4.3.3 and 4.3.4). As stated earlier, the presentation refers to users and items for the sake of readability, but KIFF is generally applicable to any kind of nodes associated with items, and can be applied to datasets containing ratings.

4.3.1 Notations

We consider a set of users $U = \{u_1, u_2, ..., u_m\}$ who have rated a set of items $I = \{i_1, i_2, ..., i_n\}$, as captured by a rating function, $\rho : U \times I \rightarrow R$ where R is the set of possible rating values.

A special case is when ratings are singled-valued, as in Figures 4.2 and 4.3. In this case, R is a singleton (e.g., $R = \{1\}$), and the function ρ simply captures whether or not a user u is associated to an item i (i.e., whether i is in her profile): $\rho(u,i)$ is defined if u is associated with i, undefined otherwise.

We denote by UP_u, the *user profile* of a user u. UP_u is a dictionary that associates the items rated by u to their corresponding rating.

The rating function ρ defines a *Labeled Bipartite Graph* $G = (V, E, \rho)$. The vertices of this graph are the users and items of the dataset $V = (U \cup I)$. A rating by a user u of an item i is represented in G by an edge $(u,i) \in U \times I$ labeled by $\rho(u,i)$: $E = \{(u,i) | i \in UP_u\}$.

4.3.2 Problem definition

Our objective is to approximate a KNN graph over U (noted G_{KNN}) according to some similarity functions computed over user profiles. There are many choices available for similarity functions over dictionaries. We use the cosine similarity in the rest of the paper [140]. Formally, G_{KNN} is a directed graph that connects each user $u \in U$ with a set knn_u of k other users that maximize the similarity function $sim(u,-)$:

$$knn_u \in \underset{v \in U \setminus \{u\}}{\text{top}_k} f_{sim}(UP_u, UP_v) \equiv \text{KNN}_u \tag{4.1}$$

where f_{sim} can be any similarity function over dictionaries and top_k returns the set of k-tuples of U excluding u that maximize the similarity function $sim(u,-)$

Because computing an exact KNN graph over a very large dataset can be computationally expensive, many scalable approaches seek to construct an approximate KNN graph $\widehat{G}_{\text{KNN}}$, i.e., to find for each user u a set $\widehat{knn}_u$ that is as close as possible to an exact KNN set. The quality of this approximation is typically measured in terms of *recall*, i.e., the ratio of exact KNN neighbors in the current KNN approximation. If the exact KNN neighborhood of each user knn_u is unique, the recall for a user u is simply defined as:

$$recall_{\widehat{G}_{\text{KNN}}}(u) = \frac{|\widehat{knn}_u \cap knn_u|}{k} \tag{4.2}$$

In the more common case where knn_u is not unique, $recall_{\widehat{G}_{\text{KNN}}}(u)$ becomes

$$recall_{\widehat{G}_{\text{KNN}}}(u) = \max_{knn_u \in \text{KNN}_u} \left(\frac{|\widehat{knn}_u \cap knn_u|}{k} \right) \tag{4.3}$$

where KNN_u denotes the top_k expression of Equation (4.1). (Because $|\widehat{knn}_u| = |knn_u| = k$ in this particular case, *precision*, a second metric commonly considered in retrieval and recommendation problems is equal to recall here and not discussed further.)

This leads us to define the overall recall of an approximate KNN graph $\widehat{G}_{\text{KNN}}$ as the average recall over all users:

$$recall(\widehat{G}_{\text{KNN}}) = \mathop{\mathbb{E}}_{u \in U} recall_{\widehat{G}_{\text{KNN}}}(u) \tag{4.4}$$

where $\mathbb{E}$ represents the expectation operator, i.e., the mean of an expression over a given domain.

Given a dataset $D = (U, I, \rho)$ and an item-based similarity function f_{sim}, our goal is to construct $\widehat{G}_{\text{KNN}}$ for D and f_{sim} in the shortest time with the highest recall.

4.3.3 KIFF

The pseudo code of KIFF is shown in Algorithm 20, which describes both the *counting* (lines 1-5) and *refinement* phases (lines 7-20). The counting phase constructs the item profiles $(IP_i)_{i \in I}$ and the ranked candidate sets $(RCS_u)_{u \in U}$ of the dataset. The item profiles reverse the user-item edges found in the bipartite graph G:

$$IP_i \equiv \{u \in U | i \in UP_u\}, \forall\, i \in I$$

(dashed arrows in Figure 4.2) and are computed while the dataset is being loaded (lines 1-2).

The ranked candidate sets are computed using the multiset union $\uplus$ (lines 4- 5). RCS_u only contains users whose ids are higher than that of u (constraint $v > u$ at line 5), as discussed in Section 4.2.3.

In terms of implementation, the ranked candidate sets RCS_u are maintained as explicit multisets only while they are being constructed (line 5). Once they are completed, they are sorted according to the multiplicity of their elements, and transformed into plain ordered lists (i.e., without multiplicity information), since only this order is used in the refinement phase (line 11). This stripping procedure lowers KIFF's memory overhead, and improves the performance of the refinement phase.

The refinement phase (lines 7-20) exploits the ranked candidate sets RCS_u to iteratively construct an approximation of the dataset's KNN. In each iteration (**repeat**-**until** loop, lines 8-14), KIFF loops through each user u (line 10), and removes the top γ users from RCS_u (operation $top\text{-}pop$, line 11). The returned users are stored in the variable cs. They correspond to the γ users with higher ids than u, who have the most shared items with u among those not yet considered in earlier iterations.

In the inner **for** loop (lines 12-14), each of these top users v is considered as a potential neighbor for u, and reciprocally, u is considered as a potential neighbor for v. This symmetry is required because we have truncated ranked candidate sets to users with higher ids at line 5.

Neighborhoods are updated in the function UPDATENN (lines 18-20). The current approximation $\widehat{knn}_u$ of each user u's neighborhood is stored as a heap of maximum size k, with the similarity between u and its neighbors used as priority.

The number of changes made over an iteration are tracked in the variable c (lines 9 and 14). The algorithm stops when the average number of changes per user during an iteration has dropped below a predefined threshold β (line 14).

Algorithm 1: KIFF

Input: dataset U, user profiles $(UP_u)_{u \in U}$, similarity $f_{sim}()$, parameter k, termination β, parameter γ

Output: $(\widehat{knn}_u)_{u \in U}$ an approximated KNN

▷ *Counting Phase: building the Ranked Candidate Sets*

1 **for** $u \in U \wedge i \in UP_u$ **do** »*Executed at loading time*
2 $\quad IP_i \leftarrow IP_i \cup \{u\}$ »IP_i *initialized to* $\varnothing$
3 **end**
4 **for** $u \in U$ **do**
5 $\quad RCS_u \leftarrow \biguplus_{i \in UP_u} \{v \in IP_i | v > u\}$ »*Multiset union*
6 **end**

▷ *Refinement Phase: greedy convergence*

7 $\widehat{knn}_u \leftarrow \varnothing, \forall u \in U$ »*Initialize the KNN heaps (size k)*
8 **repeat**
9 $\quad c \leftarrow 0$ »*Counts changes during one iteration*
10 $\quad$ **for** $u \in U$ **do**
11 $\quad\quad cs \leftarrow top\text{-}pop(RCS_u, \gamma)$ »*Top* γ *users from* RCS_u
12 $\quad\quad$ **for** $v \in cs$ **do** »*By construction* $v > u$ *(l. 5)*
13 $\quad\quad\quad s \leftarrow f_{sim}(UP_u, UP_v)$
14 $\quad\quad\quad c \leftarrow c +$ UPDATENN$(u, v, s) +$ UPDATENN(v, u, s)
15 $\quad\quad$ **end**
16 $\quad$ **end**
17 **until** $\frac{c}{|U|} < \beta$

18 **Function** UPDATENN(user u, user v, similarity s) **is**
19 $\quad$ update $\widehat{knn}_u$ heap with (v, s)
20 $\quad$ **return** 1 if changed, or 0 if not

4.3.4 Convergence and discussion

In the refinement phase, the size of the ranked candidate sets (RCS_u) decreases strictly until the sets possibly become empty. This behavior guarantees that the number of similarity computations is limited by the sizes of the the ranked candidate sets, and cannot exceed $\sum_{u \in U} |RCS_u|$.

The parameter γ and β control how aggressively KIFF approximates an optimal KNN graph. One extreme case is when γ is chosen as ∞. In this case, all ranked candidate sets are exhausted during the first iteration of the **repeat**-**until** loop (lines 8-14). When this happens, the resulting KNN approximation $(\hat{knn}_u)_{u \in U}$ is optimal, assuming the similarity function f_{sim} fulfills the following two properties:

$$\forall A, B \in (I \nrightarrow R) : A \cap B = \varnothing \Rightarrow f_{sim}(A, B) = 0 \tag{4.5}$$

$$\forall A, B \in (I \nrightarrow R) : A \cap B \neq \varnothing \Rightarrow f_{sim}(A, B) \geq 0 \tag{4.6}$$

These two properties, which are fulfilled by most item-based similarity metrics, such as cosine

similarity applied to positive ratings or Jaccard's coefficient, ensure that only users with at least one shared item can improve each other's KNN neighborhood. As a result KIFF does not miss out any potential good candidate by limiting and pruning the search to the ranked candidate sets $(RCS_u)_{u \in U}$, whose purpose is precisely to exclude pairs of users with no items in common from being considered as potential neighbors.

4.4 Experimental Setup

We evaluate KIFF by comparing it with two other state-of-the art approaches on a number of representative datasets. As we have mentioned before, there are many state-of-the art techniques and algorithms but we chose these, i.e., NN-Descent and HyRec because of two main reasons. 1) Their approaches are conceptually comparable to KIFFand 2) They already make a vivid comparison with many other state-of-the art approaches and outperform them. Thus, comparing to NN-Descent and HyRec and having a performance better than them implicitly implies that KIFFperforms better than others as well.
We have implemented KIFF and its competitors in Java [4] and execute them on a single machine. All implementations are multi-threaded to parallelize the treatment of individual users. Both the source code of all implementations and the datasets, are publicly available [1] on-line. Our experiments were conducted on a commodity server running a 64-bit Linux with an Intel Xeon E3-1246 v3, a 3.5GHz quad core processor totalling 8 hardware threads, with 32 GB of memory and a 256Gb PCI Express SSD.

4.4.1 Datasets

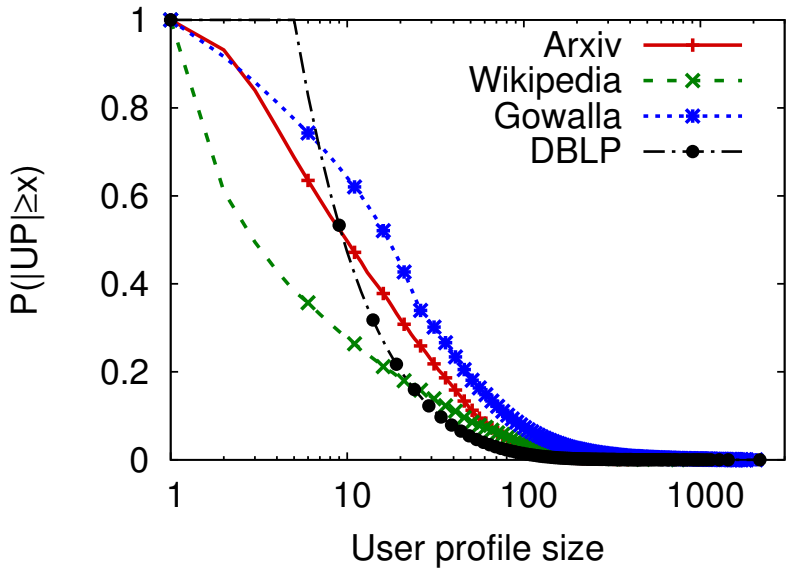
We use four publicly available user-item datasets [93] for our evaluation. These datasets are representative of a wide range of domains and applications, including bibliographic collections (Arxiv and DBLP), voting systems (Wikipedia) and on-line social networks with geo-location information (Gowalla). Some key properties of these four datasets are presented in Table 4.1, including the number of users ($|U|$), number of items ($|I|$), and number of ratings ($|E|$, i.e., the number of weighted edges in the user-item bipartite graph). The density is the percentage of edges present in the dataset over a complete bipartite graph, i.e., a graph in which each user would have rated every item : $density = |E| \div (|U| \times |I|)$.

Dataset	**#Users** $\|U\|$	**#Items** $\|I\|$	**#Ratings** $\|E\|$	**Density**	**Avg.** $\|UP_u\|$	**Avg.** $\|IP_i\|$
Wikipedia	6,110	2,381	103,689	0.7127%	16.9	43.5
Arxiv	18,772	18,772	396,160	0.1124%	21.1	21.1
Gowalla	107,092	1,280,969	3,981,334	0.0029%	37.1	3.1
DBLP	715,610	1,401,494	11,755,605	0.0011%	16.4	8.3

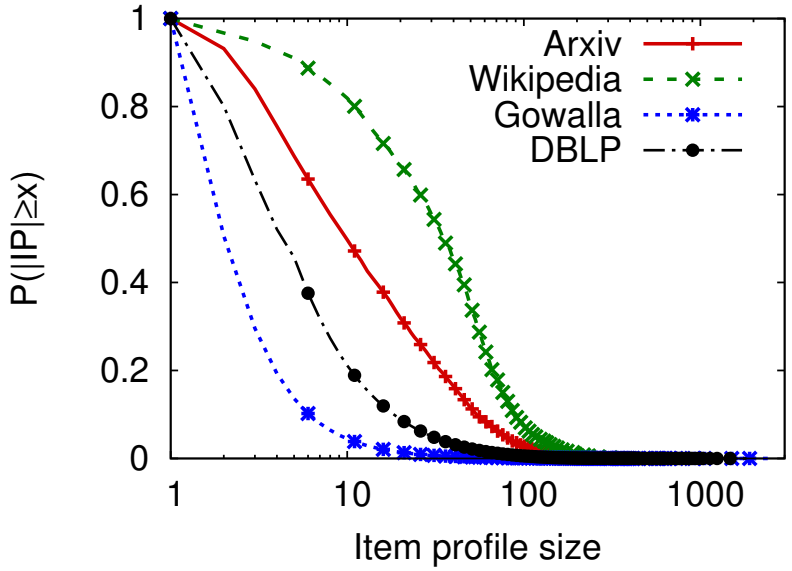
Table 4.1 – Dataset description (All datasets are available on-line [1])

As stated earlier, Table 4.1 summarizes the characteristics of the datasets we used in our experiments, and allow us to evaluate KIFF with different scalability and sparsity properties. Density is an important characteristic of a dataset that might significantly impact the time required to compute the KNN.

4. The public sources of NN-Descent [56] are only available in C++. We have therefore, re-implemented the published algorithm in Java.



(a) Size of the users profile



(b) Size of the items profile

Figure 4.4 – CCDF of the size of the user and item profiles, top and bottom respectively. Long-tailed curves show that most of the users have few ratings.

Figure 4.4a and 4.4b show the Complementary Cumulative Distribution Functions (CCDF) of the sizes of user and item profiles respectively, in these datasets. The long-tailed curves are consistent with earlier observation [35,36,108] and show that most users have very few ratings. For instance, 50% of users in the Arxiv dataset have less than 10 ratings. In the following we briefly present each dataset.

4.4.1.1 Wikipedia

Wikipedia is the well-known collaborative and free encyclopedia edited by volunteers. The promotion of a user to the role of administrator involves a consultation of other Wikipedia edi-

tors, who indicate whether they support a candidate or not. (This consultation is not, according to Wikipedia, exactly a vote.) Our dataset contains the complete Wikipedia consultation data from the origin of Wikipedia till January 2008 [92]. Nodes in the network represent wikipedia users and a directed edge from node i to node j represents that user i voted on user j. A positive vote of a Wikipedia editor (a user) for a candidate (an item) is encoded as a rating of 1, thereby resulting in binary ratings. User profiles consist of binary ratings associated to this voting network.

4.4.1.2 Gowalla

Gowalla was a location-based social networking website, acquired by Facebook in 2011, in which users could share their current location. This dataset consists of the check-ins of a subset of Gowalla users from February 2009 to October 2010 [44]. The profile of each user contains the list of locations (items) associated to her check-ins with a rating reflecting the number of time the location was visited.

4.4.1.3 DBLP

The DBLP computer science bibliography provides a comprehensive list of research papers in computer science. This dataset consists of the co-authorship network where two authors are connected if they have published a paper together. Our dataset is a snapshot of DBLP collected in September 2015 and contains information about users with at least five co-publications [5]. The profiles of the authors contain the list of co-authors, where the ratings reflect the number of publications they co-authored.

4.4.2 Competitors

KIFF relies on a preprocessing phase (the counting phase) that exploits the user-item bipartite graph to provide a first approximation of the KNN graph. This unique design places KIFF in a category of its own, which makes it difficult to choose obvious competitors to compare KIFF against. Since the refinement phase of KIFF relies on techniques similar to those used in greedy-based KNN construction algorithms, we picked two such recent state-of-the art approaches as comparison baselines: NN-Descent [56] (shown to be more efficient than LSH [100] and RLB [39] in [56]), and Hyrec [28].

As we have mentioned in the Section 4.1, the greedy-based approaches (for example, NN-Descent and HyRec) rely on a simple heuristic: a neighbor of a neighbor is also likely to be a neighbor. More precisely, if we have an approximation of the KNN for each node at any point of time, we can improve that approximation through a greedy search by exploring the neighborhood of its direct neighbors while keeping the most similar ones, repeating this process until we converge to the ideal neighbors. Though KIFF also uses a greedy-based approach to refine the KNN of a node, it completely differs from these state-of-the art approaches as it uses a well-designed initial neighborhood construction phase as compared to randomly chosen neighborhood. The combination of this preprocessing phase, exploring the bipartite nature of a dataset, with the incremental greedy-based KNN refinement makes KIFF, a new revolutionary approach for rapid and efficient KNN graph construction. Though based on the same fundamental idea of starting from a random neighborhood, the approaches differ in the way the approximation converges to the ideal KNN and as a consequence, they also differ in the recall and other computational costs.

5. Note that this snapshot is different from that used in NN-Descent's original evaluation, which we were not able to reconstruct from the information provided in [56], in spite of our best efforts.

NN-Descent [56] favors *node locality* to iteratively converge to a close approximation of an exact KNN. Starting from a random graph, NN-Descent iteratively refines the neighborhood of a user by considering at each iteration a candidate set composed of the direct neighborhood of the current bidirectional neighbors (both in-coming and out-going neighbors). To avoid repeated similarity computations, NN-Descent uses a system of flags to only consider new neighbors-of-neighbors during each iteration. Since such a candidate set may still be very large, only a sample might be considered to speed-up the process. NN-Descent also uses a pivot strategy similar to ours to ensure that the similarity computation between a pair of users is performed only once, by iterating on both the in-coming and out-going neighbors of the current pivot user.

HyRec [28] is a recommendation framework which relies on an approach similar to NN-Descent but distributes the KNN graph construction onto the browsers of users to improve the scalability of the KNN computation. Similar to NN-Descent, HyRec relies on node locality to iteratively converge to an accurate KNN from a random graph. During each iteration, HyRec considers the neighbors of neighbors of each user, as well as a set of few random users to avoid a local minimum as in [144]. In this approach, a parameter r is used to define the number of random users considered in the candidate set. For a fair comparison, although not mentioned in [28], we implement the same pivot mechanism as in NN-Descent and the early termination of KIFF.

4.4.3 Metrics

We assess the quality of the produced KNN graphs by measuring the recall of the graphs returned by each approach, as defined in Section 4.3.2. For each dataset, an ideal KNN is constructed using a brute force approach. The recall is then obtained by comparing the similarity values of the ideal neighborhoods and those of the approximated ones.

We measure the computational cost of each solution using two metrics: the *scan rate*, and the *computation time*. The scan rate measures the number of similarity evaluations required to produce the final approximated KNN graph, normalized and expressed as a percentage of all possible KNN edges.

Definition 8 (Scan Rate)

$$scanrate = \frac{\#(similarity\ evaluations)}{|U| \times (|U|-1)/2}$$

The computational time is the *wall-clock* time required to construct the KNN graph, excluding the JVM's start-up time, but including the loading time of the graph from persistent storage (i.e., measured from the JVM's entry into the `main` method). We also separately measure the time spent by each solution for different activities of each approach, namely *preprocessing* (loading the dataset, building user profiles, and in the case of KIFF building item profiles and performing the counting phase), *candidate selection* (constructing candidate neighborhoods, using the RCS sets in KIFF and using neighbor-of-neighbor links in Hyrec and NN-Descent) and *similarity computation*.

4.4.4 Default parameters

In our evalations, for the sake of simplicity we use the cosine similarity metric [140] but KIFF is designed to perform consistently regardless of the similarity metric considered. For a

fair comparison, we set the parameters of each approach to their default values [29, 56]. More precisely, we set $k = 20$ (except for DBLP where we use $k = 50$), and use an early termination parameter β at 0.001 for KIFF. For NN-Descent, we report results without sampling (as in the original publication), and we consider $\gamma = 2k$ for KIFF. For HyRec, by default, we consider no random nodes in the candidate set ($r = 0$). Though not shown in the evaluation for space reasons, random nodes cause random memory accesses and drastically increase the wall-time (three times longer on average, with $r = 5$) while only slightly improving the recall (4% on average).

4.5 Evaluation

We now present the results obtained by KIFF, HyRec and NN-Descent under the experimental setup of the previous section. Our results show that KIFF efficiently converges to an approximate KNN graph, while outperforming its competitors both in terms of recall and computation time. We also assess the sensitivity of the three approaches to k and to the *density* of the datasets. We also evaluate the impact of γ on KIFF.

4.5.1 Performance Comparison

Table 4.2 shows the recall, wall-time, scan rate and number of iterations of NN-Descent, HyRec and KIFF on the four datasets considered for evaluation. The best recall and wall time values are highlighted in bold. The lines marked "KIFF's Gain" show, for each dataset, the improvement in recall and the speed up achieved by KIFF over its competitors (averaged over NN-Descent and HyRec). These gains, averaged over all datasets, are summarized in Table 4.3.

datasets	Approach	recall	wall-time (s)	scan rate	#iter.
Arxiv	**NN-Descent**	0.95	41.8	17.6%	9
	HyRec	0.90	38.6	16.0%	12
	KIFF	**0.99**	**10.7**	2.5%	36
	KIFF's Gain	*+0.06*	*×3.7*		
Wikipedia	**NN-Descent**	0.97	13.1	51.69%	7
	HyRec	0.95	9.4	44.64%	8
	KIFF	**0.99**	**4.4**	7.37%	22
	KIFF's Gain	*+0.03*	*×2.5*		
Gowalla	**NN-Descent**	0.69	307.9	3.67%	16
	HyRec	0.56	253.2	2.69%	22
	KIFF	**0.99**	**146.6**	0.84%	115
	KIFF's Gain	*+0.36*	*×1.9*		
DBLP	**NN-Descent**	0.78	10,890.2	3.08%	19
	HyRec	0.63	8,829.9	2.37%	26
	KIFF	**0.99**	**568.0**	0.07%	33
	KIFF's Gain	*+0.28*	*×17.3*		

Table 4.2 – Overall perf. of NN-Descent,HyRec, & KIFF

These results show that KIFF consistently outperforms NN-Descent and HyRec in terms of recall and wall-time across all datasets. While KIFF consistently achieves a recall of 0.99, the recall of NN-Descent and HyRec ranges from 0.56 (HyRec on Gowalla) to 0.97 (NN-Descent on

Wikipedia), with an average of 0.85 for NN-Descent and 0.76 for HyRec. The higher recall values obtained by KIFF validate the use of ranked candidate sets which provide KIFF with a clear advantage in terms of KNN quality over the initial random graph used by HyRec and NN-Descent.

	KIFF's gain	
Competitor	**speed-up**	**Δrecall**
NN-Descent	×15.42	+0.14
HyRec	×12.51	+0.23
Average	*×13.97*	*+0.19*

Table 4.3 – Average speed-up and recall gain of KIFF

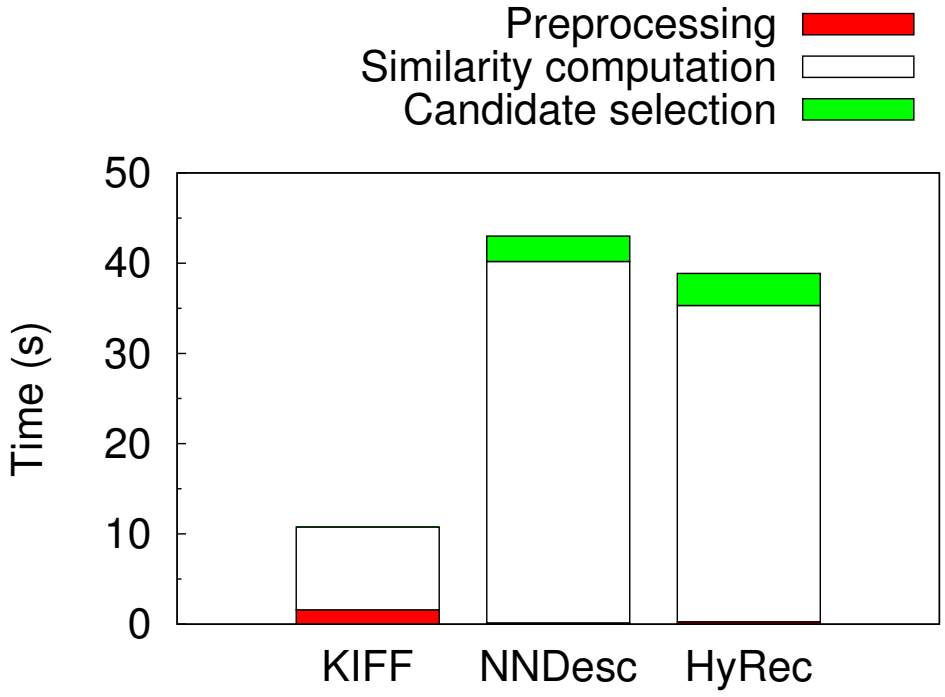
Dataset	Wall-time (ms)		Δ(ms)	%age total time
	(UP_u)	$(UP_u)\&(IP_i)$		
Arxiv	135	185	50	0.5%
Wikipedia	59	69	10	0.2%
Gowalla	2,354	5,136	2,782	1.9%
DBLP	7,492	12,996	5,504	1.0%

Table 4.4 – Overhead of item profile construction in KIFF

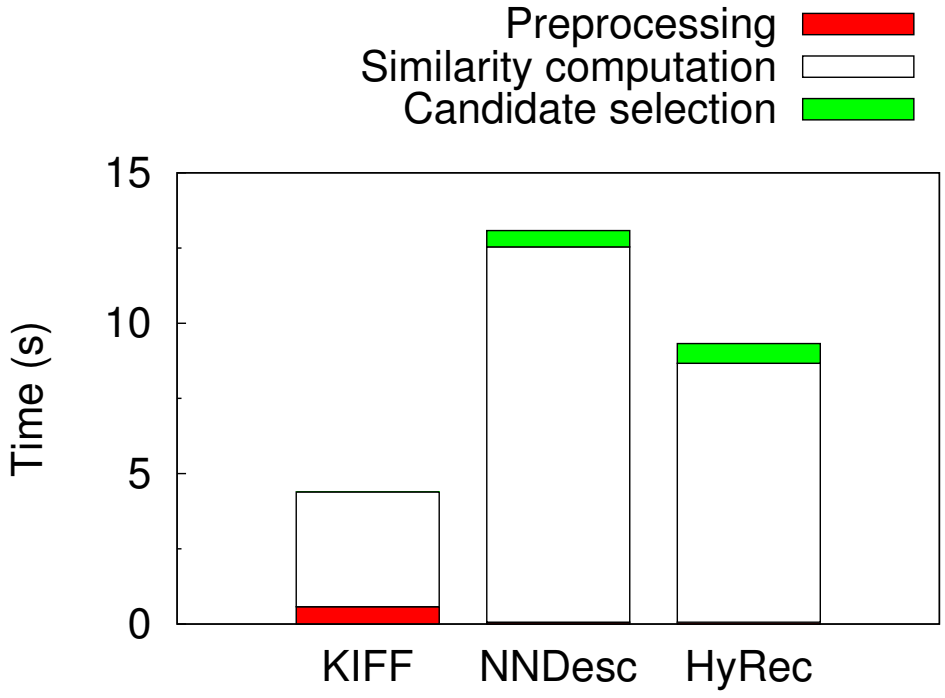
The results show that the scan rate (number of similarity computations required to converge) for KIFF is 7 and 6 times lower than that of NN-Descent and HyRec respectively. As KIFF requires lesser similarity computations, it is also faster than its competitors. The wall-time values of KIFF confirm our intuition that the counting phase of KIFF, by preprocessing the bipartite graph of the dataset, results in relatively faster convergence to an approximate knn graph. The speed-up achieved can be substantial: for instance, KIFF is 17 times faster than Hyrec and NN-Descent on the DBLP dataset, and 14 faster on average (Table 4.3).

The reason why KIFF outperforms its competitors in terms of wall-time is visible in Figure 4.5, which charts the breakdown of the computation time of KIFF, NN-Descent, and HyRec into the three types of activities introduced in Section 4.4.3: (1) preprocessing (which includes KIFF's counting phase); (2) candidate selection; and (3) similarity computations. The figure shows that the counting phase of KNN indeed introduces some overhead (ranging from 10.01% to 14.74% of KIFF's overall computation time), but that this overhead is largely compensated by a much faster convergence, with fewer similarity computations, and lesser time spent on candidate selection.

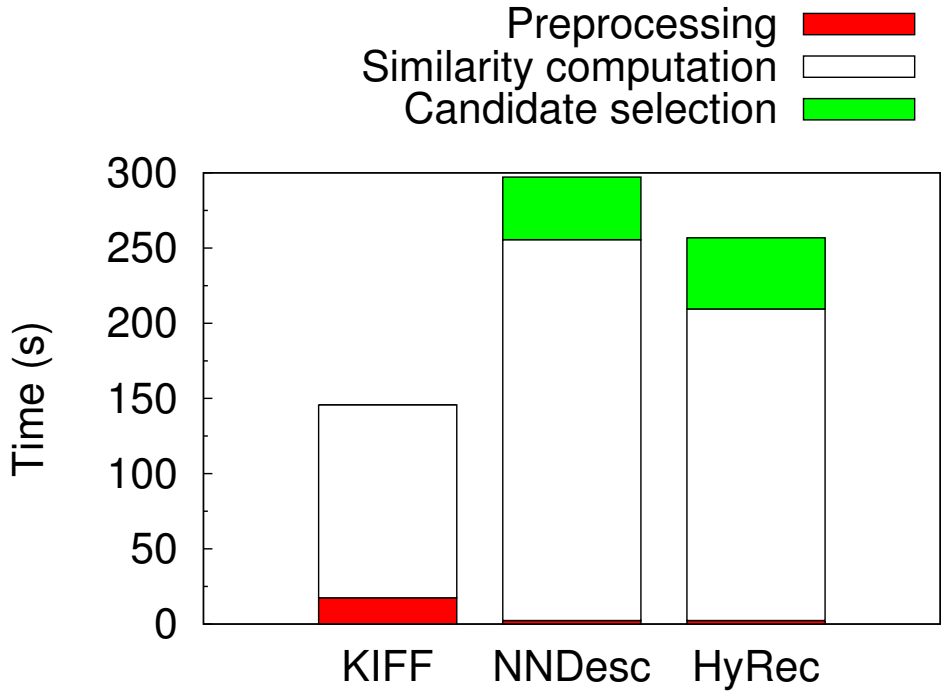
In the following, we investigate in more detail how KIFF exploits ranked candidate sets to converge faster: we first analyze the overhead of KIFF's preprocessing and the cost of computing RCSs; we then discuss the properties of the resulting RCSs; and we finally compare the convergence of KIFF's refinement phase with the convergence of NN-Descent and HyRec.



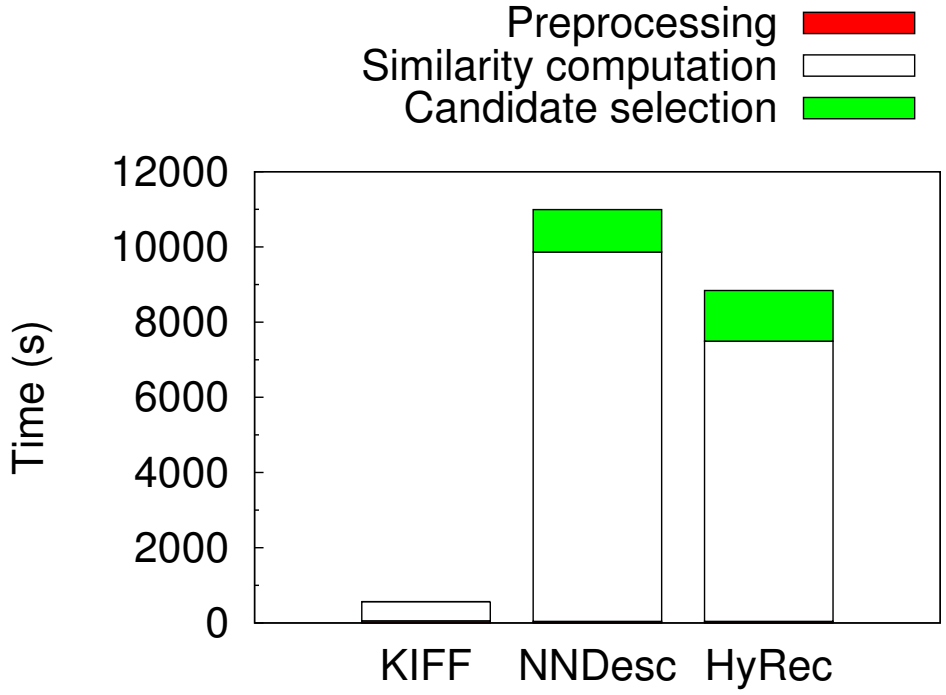
(a) Arxiv



(b) Wikipedia



(c) Gowalla



(d) DBLP

Figure 4.5 – Although KIFF *must pay higher preprocessing costs to constructs its Ranked Candidate Sets, this overhead is largely balanced out by a smaller number of similarity computations due to a much faster convergence compared to Hyrec and NN-Descent.*

4.5.1.1 Overhead of KIFF's preprocessing

KIFF's preprocessing essentially comprises its *counting phase*, which computes *item profiles*, and *Ranked Candidate Sets* (RCSs). In KIFF, item profiles are built when the user profiles (required by all approaches) are created from the input data stream. Table 4.4 compares the time taken to construct user profiles only ((UP) column) to that taken to construct both user and item profiles simultaneously, and contrasts the difference between the two (Δ column) with the overall running time of KIFF (last column), for all data sets. The table shows that the overhead caused by item profiles only represents a very small fraction of the total running time of KIFF across all datasets (at most 1.9% on Gowalla).

The time taken to compute ranked candidate sets ($RCSs$) is shown in Table 4.5, along with some additional statistics (which we return to below). Constructing $RCSs$ takes substantially more time than constructing item profiles, and makes up, in effect, most of KIFF's preprocessing overhead, amounting to up to 13.5% of KIFF's overall running time (with Arxiv).

Dataset	RCS const. (ms)	%age total time	avg. |RCS|	max RCS scan rate
Arxiv	1,404	13.1%	247.0	2.63%
Wikipedia	465	10.6%	228.7	7.48%
Gowalla	12,255	8.4%	458.1	0.85%
DBLP	42,829	7.5%	267.8	0.07%

*Table 4.5 – Overhead of RCS construction & statistics (*KIFF*)*

4.5.1.2 Properties of the resulting RCSs

The extent to which KIFF is able to recoup this overhead strongly depends on the capacity of the resulting RCSs to limit similarity computations by pruning pairs of users that have no items in common. This *pruning effect* is directly linked to the size of RCSs: short RCSs (in comparison to the total number of users in the dataset) eliminate many similarity computations and help accelerate KIFF's refinement phase. To shed light on this aspect, Table 4.5 shows the average lengths of the RCSs computed for each dataset and the corresponding maximum scan rate induced by these RCSs

$$max\_scan = \frac{|U| \times |RCS|}{\frac{1}{2} \times |U| \times (|U|-1)} = 2 \times \frac{|RCS|}{|U|-1}$$

i.e., the scan rate one would obtain by iterating through the entire RCSs in the refinement phase ($\gamma = \infty$ in Algorithm 20). We note that this maximum scan rate is in fact very close to the actual scan rate reported in Table 4.2: the value of the termination parameter β chosen for KIFF (0.001) causes most RCSs to be fully iterated through, which explains the high recall values obtained by KIFF.

This phenomenon is confirmed by Table 4.6, and Figures 4.6 and 4.7. Table 4.6 repeats the number of iterations executed by KIFF from Table 4.2. This number corresponds to a maximum number of entries considered by KIFF from each RCS, which we note $|RCS|_{\text{cut}}$ (second column, this is simply #iters $\times \gamma$): RCSs which are smaller that this size are fully iterated through, while larger RCSs are truncated. The percentage of users with truncated RCSs for each dataset is shown in the last column, and illustrated graphically using vertical bars in Figure 4.6 (which shows the CCDF of RCS sizes).

Only a minority of users experience truncated RCSs, but this minority can be important, as in the case of Wikipedia, where 16.24% of users are not compared with all their candidate KNN neighbors. This truncation could potentially hurt KIFF's recall if too many ideal KNN neighbors are found towards the tail of the truncated RCSs. Figure 4.7 shows that this is not the case. (We only show the Wikipedia dataset due to space reasons.) Each point in the figure is a RCS with a size larger that Wikipedia's truncation value (440), which is plotted according to its size and Spearman's correlation between the RCS (ordered according to common item counts) and the same list of nodes sorted following either Jaccard's coefficient or the cosine similarity metrics. The figure shows that Spearman's correlation coefficient is generally high (on average 0.60 for Jaccard, and 0.63 for cosine) for truncated RCSs, and increases with RCSs sizes, suggesting that the approximation provided by the counting phase does not generally exclude good candidates from being considered as potential neighbors.

Dataset	#iters	$\|RCS\|_{\text{cut}}$	%user $\|RCS_u\|$ $> \|RCS\|_{\text{cut}}$
Arxiv	36	720	9.57%
Wikipedia	22	440	16.24%
Gowalla	115	2300	4.82%
DBLP	33	660	10.32%

Table 4.6 – Impact of KIFF*'s termination mechanism*

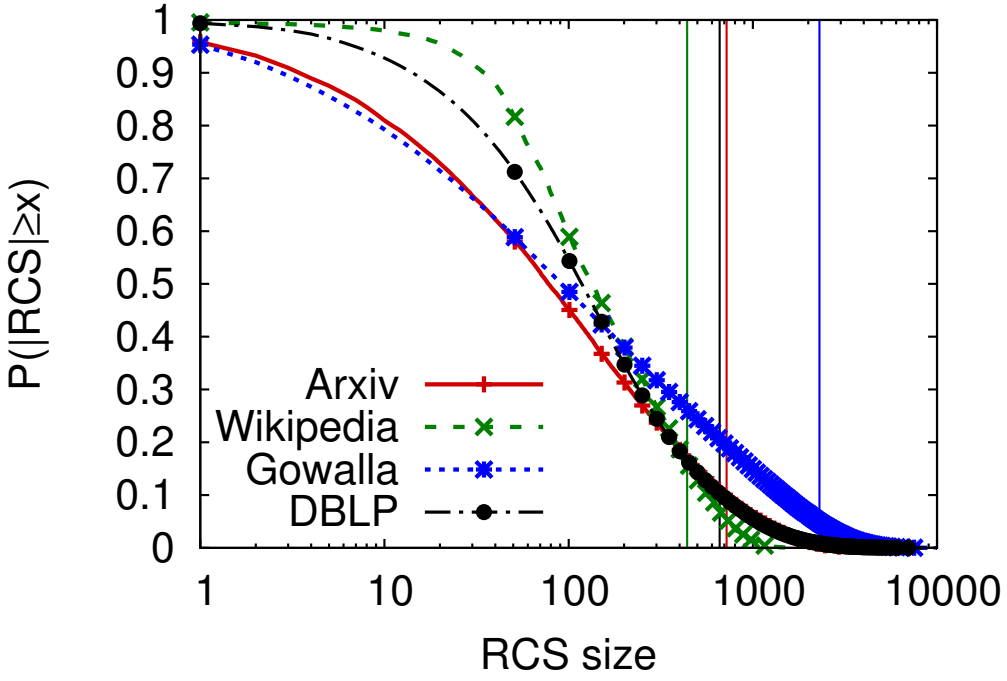


Figure 4.6 – CCDF of $|RCS|$. The vertical bars shows the cut-off sizes enforced by KIFF*'s termination mechanism (also shown in Table 4.6)*

The quality of the RCSs is finally further illustrated by Table 4.7, which shows the recall obtained by KIFF without any convergence ($\beta = \infty$ in Algorithm 20), i.e., when using the top k users of each RCS as a KNN graph approximation. Although the resulting recall varies widely from 0.54 (Wikipedia) to 0.82 (Arxiv), these values are much larger than those obtained through

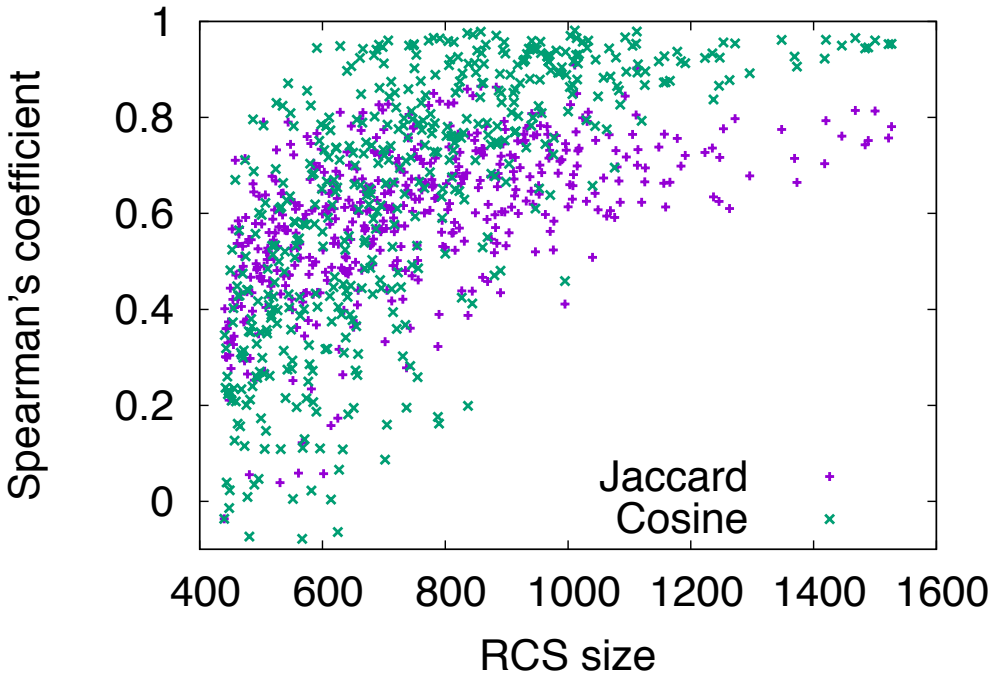


Figure 4.7 – Correlation between the rank of nodes in RCSs and the cosine and Jaccard metrics for users with $|RCS_u|$ *above* $|RCS|_{\text{cut}} = 440$ *for Wikipedia.*

a random initial graph (which peaks at 0.15 for Gowalla), as used by traditional greedy approaches, and illustrates the immediate benefit obtained by KIFF from its counting phase.

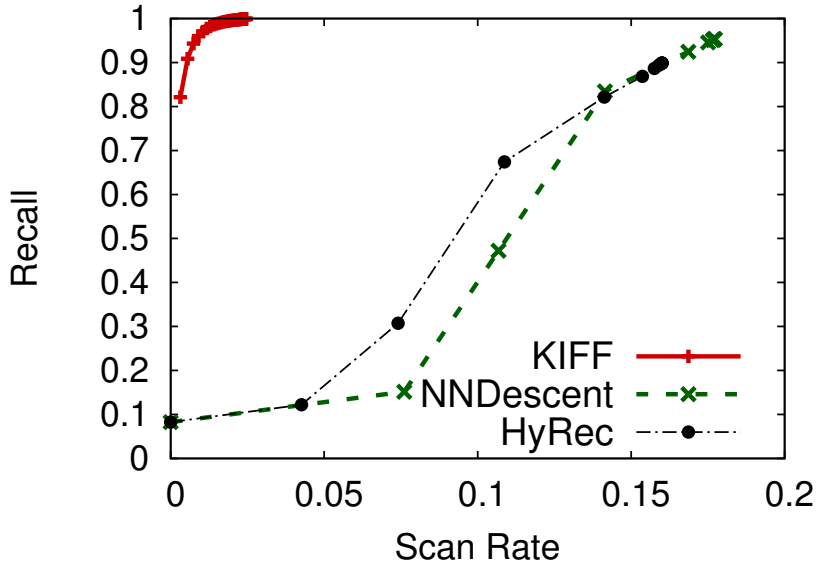
	Recall	
Dataset	Top k from RCS	Random
Arxiv	0.82	0.08
Wikipedia	0.54	0.01
Gowalla	0.55	0.15
DBLP	0.79	0.09

Table 4.7 – Impact of initialization method on initial recall

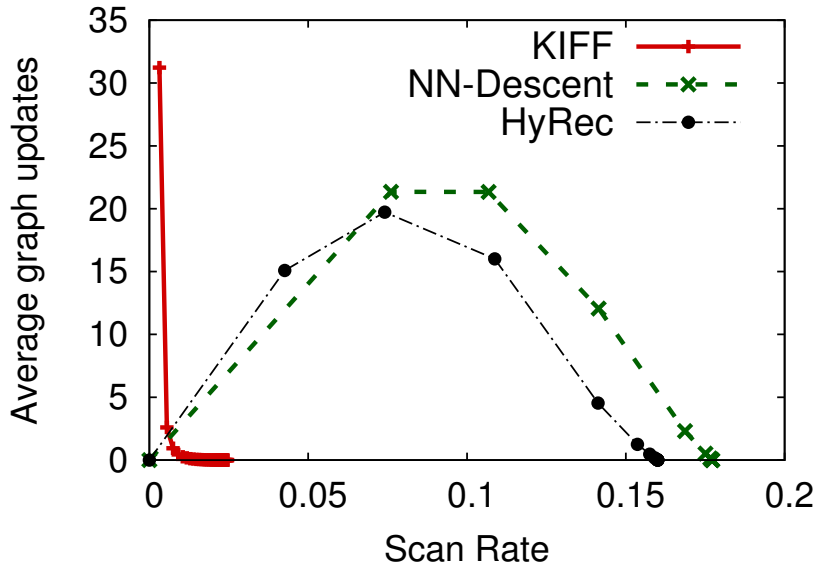
4.5.1.3 Convergence

Although the incremental convergence of KIFF derives its motivation from greedy-based approaches, its convergence pattern is clearly different from that of its greedy-based competitors, NN-Descent and HyRec, as shown in Figure 4.8. Figure 4.8a charts the quality of the KNN graph under construction as a function of scan rate for the three approaches on the Arxiv dataset. As indicated in Tables 4.2 and 4.7, KIFF immediately starts with a high recall value (0.82), and terminates after a very small scanrate (2.5%). By contrast, HyRec and NN-Descent start from a low recall value (0.08) induced by their random initial graph, and only converge at a much higher scan rate (resp. 16.0% and 17.6%). (KIFF shows similar behavior for the remaining datasets, which are not included here due to space reasons.) The rapid convergence of KIFF can be attributed to the effective pruning induced by *RCSs* that leads to a reduction in similarity

computations. The slow convergence of NN-Descent and HyRec by contrast can be credited to their random initial start and their transitive convergence mechanisms (exploiting neighbors-of-neighbors links), which requires more computational resources to converge to a good KNN graph approximation.



(a) scan rate vs recall



(b) scan rate vs updates

Figure 4.8 – KIFF quickly provides an accurate KNN approximation by using relevant candidates compared to competitor approaches (Arxiv dataset).

The effect of the convergence procedure (RCSs vs. neighbors-of-neighbors links) on the convergence speed is visible in Figure 4.8b, which shows the average number of updates performed on the KNN graph during each iteration. We observe that the use of RCSs to select potential candidates allows KIFF to perform many relevant updates to the KNN graph during its first iteration. The number of updates performed in subsequent iterations decreases rapidly in the case of KIFF, reflecting the fact that RCSs are ordered according to decreasing common item counts. This property in turn leads to a much faster convergence. In contrast, NN-Descent and HyRec present a three-step convergence pattern. First, the random nature of the initial graph significantly increases the number of beneficial updates during the first few iterations. Recall only improves slightly, however, during this first step (Figure 4.8a). During the second step, more and more potential KNN candidates are encountered, leading to an improved approximation with a better recall. Finally, during the third phase, the last iterations before convergence, fewer updates are performed as good potential candidates become harder to discover in the graph, a fact reflected in a slowly increasing recall till convergence.

4.5.2 Sensitivity analysis: k, γ and density

4.5.2.1 Impact of parameter k

The number of similarity computations NN-Descent and HyRec execute is closely related to the size of the neighborhoods they maintain (parameter k). This is because both approaches leverage neighbors-of-neighbors relationships to improve their KNN approximation. Large neighborhoods facilitates this process by offering more options for convergence, at the price of longer computation times, due to the overheads of managing larger data structures, and computing more similarities. Smaller neighborhoods constrain their search for potential neighbors. As a result, smaller values of k should reduce the number of similarity values both approaches need to compute, and lead to better computation times. There is however, a trade-off, as too small neighborhoods might disrupt the greedy convergence of NN-Descent and HyRec and yield degraded KNN results.

Contrary to NN-Descent and HyRec, KIFF does not use its intermediate neighborhoods (the variables $k\hat{n}n_u$ of Algorithm 20) to drive its convergence, but relies instead on precomputed ranked candidate sets (RCS_u). One might therefore expect its computation time and recall to be less impacted by lower k values.

To better understand how KIFF and its two competitors behave for smaller values of k, Table 4.8 presents their performance in terms of computation time, recall, and scan rate when we lower k from 20 to 10 (from 50 to 20 for DBLP) on the same datasets as Table 4.2. These results confirm that a smaller k does lead to substantial gains in computation times for NN-Descent and HyRec (speed-up ratios ranging from 2.35 to 4.07), with a less pronounced improvement for KIFF (speed-up varying from 1.09 to 1.36). Simultaneously, however, the recall values of NN-Descent and HyRec degrade substantially (with losses varying from 11% to 35%), while those of KIFF remain stable. Across the board, KIFF continues to outperform its competitors in terms of computation times and recall on 3 datasets out of 4 (Arxiv, Wikipedia, and DBLP). On Gowalla, NN-Descent and HyRec have become faster, but with much degraded recalls (0.35 and 0.26), that can no longer compete with that of KIFF (0.99).

It is worth noting that the time spent in similarity computations depicted in Figure 4.5 depends linearly on the complexity of the similarity metric used to compute the KNN graph. Therefore, using a metric requiring twice as long to evaluate the similarity between two profiles will double the global time of similarity computation, and will further increase the benefits of our solution compared to NN-Descent and HyRec.

datasets	Approach	recall		wall-time (s)		scan rate	
Arxiv	**NN-Descent**	0.74	(-0.20)	17.7	(÷2.36)	5.49%	(-12.1%)
	HyRec	0.55	(-0.34)	16.4	(÷2.35)	4.66%	(-11.3%)
	KIFF	**0.99**	(=)	**7.8**	(÷1.36)	1.97%	(-0.4%)
	KIFF's Gain	*+0.34*	(+0.28)	*×2.18*	(÷1.70)		
Wikipedia	**NN-Descent**	0.86	(-0.10)	5.3	(÷2.46)	16.39%	(-35.3%)
	HyRec	0.74	(-0.20)	3.6	(÷2.55)	13.98%	(-30.6%)
	KIFF	**0.99**	(=)	**3.2**	(÷1.36)	6.86%	(-0.5%)
	KIFF's Gain	*+0.19*	(+0.16)	*×1.39*	(÷1.80)		
Gowalla	**NN-Descent**	0.35	(-0.33)	117.8	(÷2.61)	0.89%	(-2.7%)
	HyRec	0.26	(-0.29)	**98.7**	(÷2.70)	0.61%	(-2.0%)
	KIFF	**0.99**	(=)	120.4	(÷1.21)	0.73%	(-0.1%)
	KIFF's Gain	*+0.68*	(+0.32)	*×0.89*	(÷2.13)		
DBLP	**NN-Descent**	0.20	(-0.57)	2,673.4	(÷4.07)	0.43%	(-2.6%)
	HyRec	0.11	(-0.52)	2,272.5	(÷3.88)	0.26%	(-2.1%)
	KIFF	**0.99**	(=)	**516.6**	(÷1.09)	0.07%	(=)
	KIFF's Gain	*+0.83*	(+0.55)	*×4.78*	(÷3.62)		

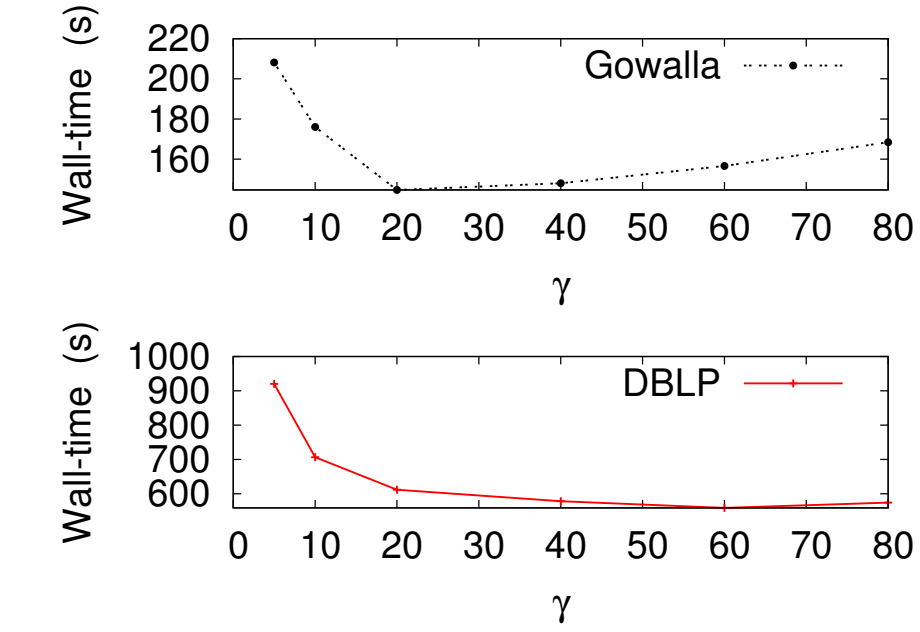
Evolution of wall-time, recall, and scan rate when reducing k from 20 to 10 (DBLP: 50 to 20). The numbers in brackets show the change w.r.t to the results with k = 10 (DBLP:k = 20) presented in Tab. 4.2. KIFF shows little sensitivity to the value of k, with slightly lower computation times, and identical recall values. By contrast NN-Descent and HyRec are strongly impacted, with much reduced wall-times accompanied by degraded recall values. KIFF remains the best performing solution across the board.

Table 4.8 – Impact of k on recall and wall-time (k=10, DBLP: k=20)

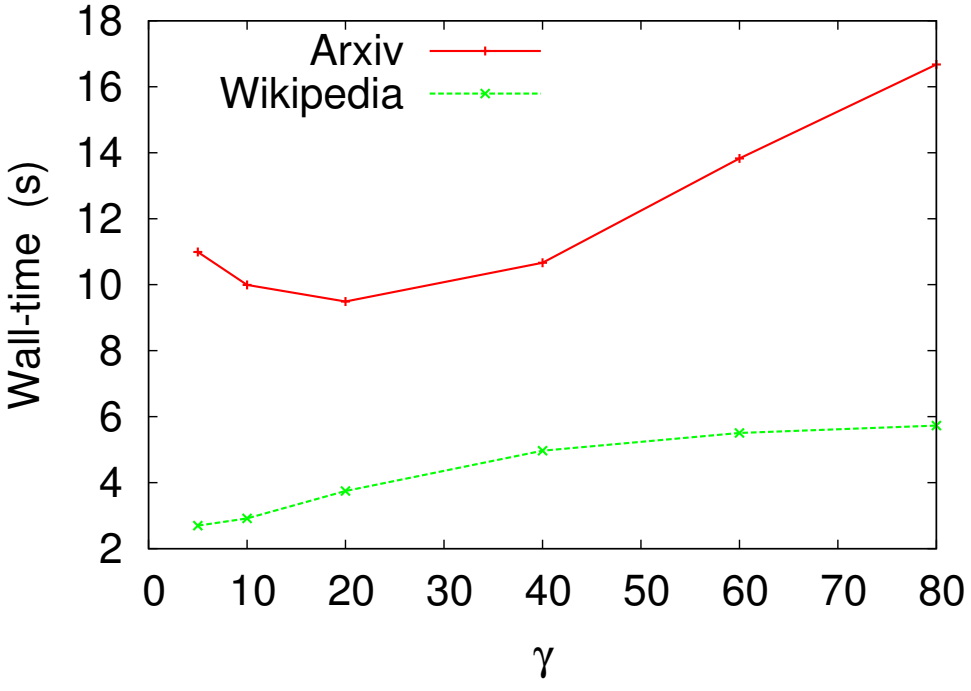
4.5.2.2 Impact of parameter γ

The parameter γ in KIFF defines the number of elements removed from the ranked candidate sets of each user and considered as potential neighbors at each iteration. If the ranked candidate set of user u, RCS_u, contains less than γ elements, only these elements are considered and RCS_u becomes empty. Consequently, γ impacts the number of similarity computations performed at each iteration which is bound by $|u|\gamma$. The number of considered neighbors which become actual neighbors over an iteration are tracked (variable c in Algorithm 1) and KIFF stops if the average number of changes is bellow a predefined termination threshold β.

On the one hand, the parameter γ impacts the number of iterations that KIFF needs to converge. A larger γ will reduce the number of iterations, in contrast a smaller γ will force KIFF to perform more iterations to converge. While the number of iterations does not impact the scan rate which depends on the global number of similarity computations actually performed, a too small γ will tend to increase the wall-time due to iteration overheads as shown in Figure 4.9 which depicts the wall-time of KIFF according to the value of γ. On the other hand, at the last iteration where $\frac{c}{|u|} < \beta$, the probability to perform more similarity computations than actually required to reach the termination threshold is a function of γ. As a consequence, a too large γ tends to increase the scan rate. However, as depicted in Figure 4.9, the impact of γ on the wall-time remains low.



(a)



(b)

Figure 4.9 – The impact of γ on the wall-time of KIFF *remains low.*

Although we cannot fully investigate its impact for space reasons, let us note for completeness that the termination threshold β also has an important influence on the performance of KIFF. The value of β represents a trade-off between recall and scan rate (and hence wall-time). For instance, on the Arxiv dataset, increasing β hundredfold to 0.1 (from 0.001) causes KIFF to take 36% less time to converge by halving its scan rate to convergence. Recall is mildly impacted, being reduced by 0.01, down to 0.98.

4.5.2.3 Impact of density

KIFF is designed to work well on datasets with small user and item profiles, relatively to the total number of items and users. This situation typically corresponds to a low density of the associated bipartite graph. The datasets we have considered (Table 4.1) have this property, with density values remaining below 1% (our highest density is 0.71% for the Wikipedi dataset). For completeness' sake, we investigate in this last evaluation, how KIFF behaves on denser datasets, and contrast its performance to that of NN-Descent. For this comparison, we derive from the same underlying dataset, a family of bipartite graphs with different density characteristics.

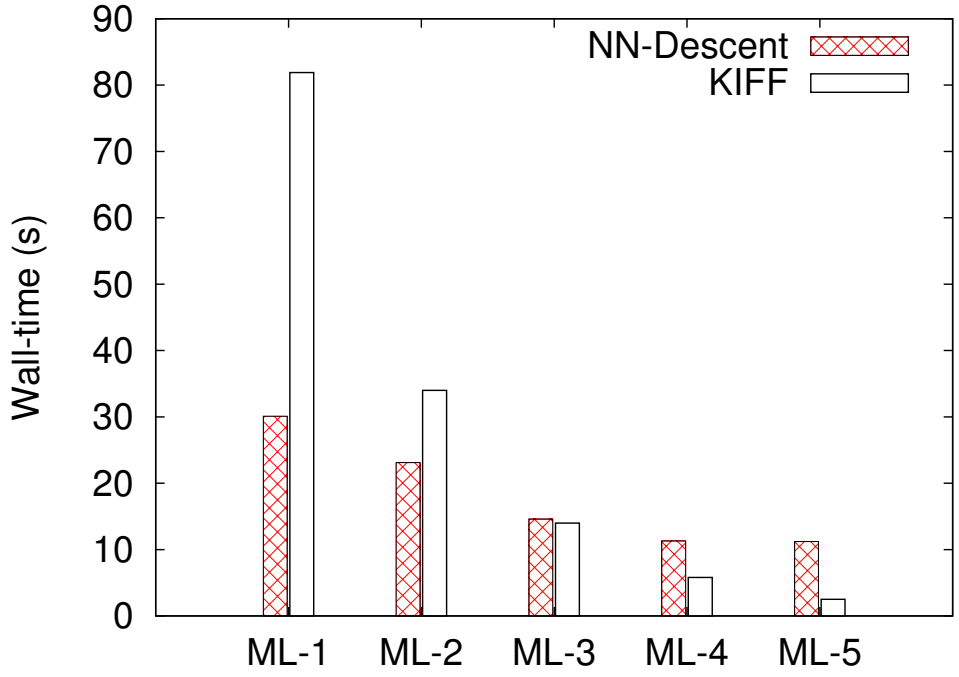
We start from a dataset from the MovieLens (ML) repository [13]. MovieLens provides movie-rating data collected on the ML recommender website over a 7-month period. ML ratings range over a 5-star scale, with half-star increments. The ML dataset we use (called ML-1 in the following) contains 6,040 users and 3,706 items (movies), in which each user has at least made 20 ratings, with an average of 165.1 ratings per user. ML-1 is substantially denser than the datasets we have considered so far, with a density of 4.47%. Starting from ML-1, we progressively remove randomly chosen ratings and obtain four additional datasets (numbered ML-2 to ML-5) showing decreasing density values (ranging from 2.23% to 0.30%, Table 4.9).

Dataset	**Ratings**	**Density**	**average** $\lvert RCS \rvert$
ML-1	1,000,209	4.47%	2,892.7
ML-2	500,009	2.23%	2,060.6
ML-3	255,188	1.14%	1,125.4
ML-4	131,668	0.59%	510.8
ML-5	68,415	0.30%	202.5

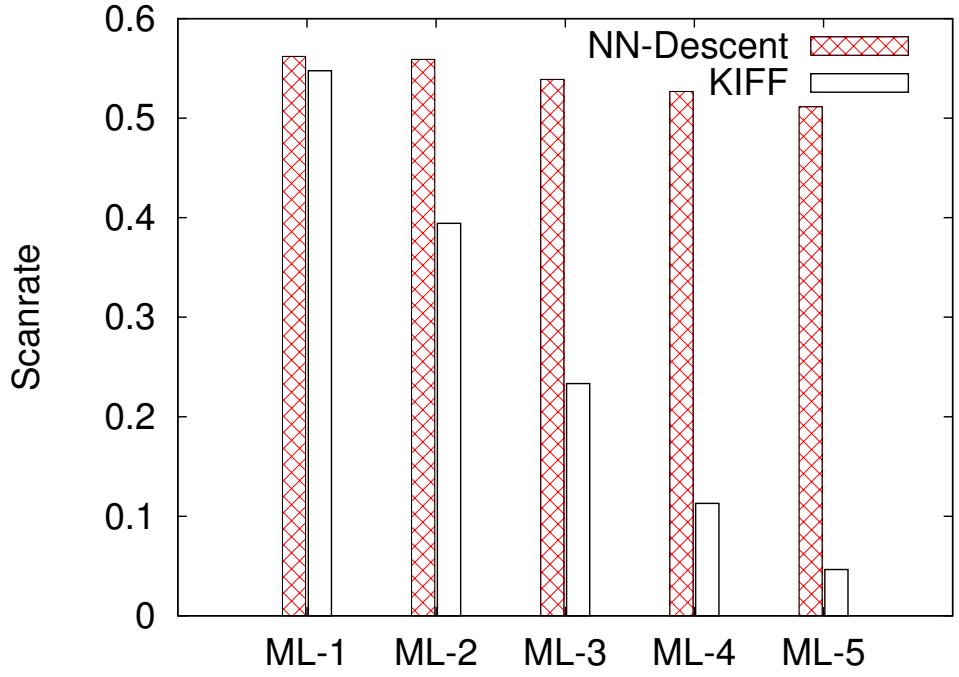
Table 4.9 – MovieLens datasets with different density.

For a fair comparison between KIFF and NN-Descent we first measure the recall obtained by NN-Descent on ML1-5 with the default parameters mentioned in Section 4.4.4. NN-Descent yields a consistent recall value of 0.93 for ML1, ML2, ML3, and ML4, and a recall of 0.97 for ML-5. We then set the β parameter of KIFF for each dataset so as to obtain the same recalls as NN-Descent.

The overall execution time (wall-time) and scan rate obtained by the two competitors when applying this procedure are reported in Figures 4.10a and 4.10b. These results show that KIFF is slower than NN-Descent for dense datasets (ML-1 and ML-2), but that the situation reverses itself for sparser versions (ML-4 and ML-5), with the two approaches being close on ML-3 (density of 1.14%). The evolution of KIFF's computation time can be directly related to its scan rate (Fig. 4.10b) and the average size of the ranked candidate sets KIFF constructs (Table 4.9). While the scan rate of NN-Descent remains roughly stable in the 5% – 6% range across all datasets, that of KIFF decreases sharply with dataset density, highlighting the importance of low density values (the common case) for KIFF to perform well.



(a) wall-time



(b) scan rate

Figure 4.10 – a) KIFF *is faster than NN-Descent for sparse datasets ; b) the scan rate for* KIFF *decreases according to the dataset density compared to NN-Descent which is not correlated to density.*

4.6 Conclusion

We have presented the design and evaluation of KIFF, a novel, generic, scalable, and efficient algorithm to construct KNN graphs. KIFF leverages the bipartite node-item graph found in many datasets to build ranked candidate sets. These ranked sets, in turn, drastically reduce the number of similarity computations performed, and hence the convergence time.

We have provided an extensive evaluation of KIFF over a variety of datasets. Our evaluation demonstrates that the proposed solution achieves a substantial speed-up in comparison to the state-of-the art approaches while improving the quality of the KNN graph. Moreover, we have also assessed the sensitivity of KIFF and show that its performance is independent of the value of k. We can also clearly conclude than KIFF outperforms its competitors even more on sparse datasets.

Furthermore, leveraging a heuristic to limit the insertion of elements in the ranked candidate sets is an interesting perspective. Preliminary works considering a naive threshold on multiple-ratings to insert, in the ranked candidate sets, only those users who have positively rated items, reduces the $RCSs'$ size and improves the performance of KIFF.

5

On the power of data: the specifics of data quality and quantity

> The goal is to turn data into information, and information into insight
>
> *Carly Fiorina*

In the recent years, ***personalization*** has become a buzz word in all machine learning domains and applications, more percussive in case of e-businesses. In the context of recommendation systems, personalization is the back-bone of their working and a significant factor for their increasing popularity. The global idea of personalization, as the name suggests, is to tailor the results to suit the preferences of each customer or a group of similar customers. The former CEO of Amazon said "If 48% of consumers spend more when their experience is personalized, it's no wonder that 90% of marketers believe personalization is **the future** and are making it a strategic priority over years."

The experience of *personalization*, in turn, is driven by *data*.

Although data is a prime requirement for personalization, it is not the only requirement to have a good experience of personalization. The quantity, and the quality of data are additional factors governing personalization, in machine learning, and specifically in recommendation systems.

In this chapter, we focus on the ***data***. To understand the importance of data in the process of recommendation, we use datasets from diverse sources, containing a variety of user attributes and items information. With the overall goal of analyzing the effects of data *quantity* and *quality* and its inference and prediction capabilities, we start by obtaining data from different sources. Later, the data is cleansed and processed to have consistent and unambiguous attributes. As we will see in the rest of the chapter, the preprocessing of the datasets is highly dependent on the attributes into consideration. For example, the techniques used for finding

missing gender values in a dataset will differ considerably from the ones for finding missing age values. Once the data is collected and cleansed, we will use various machine learning algorithms to show the inference capabilities of the data.

Our results show that the attribute (or information) to be predicted depends on the choice of the attributed used to train the learning model. With the help of these datasets, we present an extensive study, in the form of survey, highlighting the importance and effects of data quantity and quality on the process of inference/prediction in recommendations.

5.1 Introduction

The growth of businesses over internet : *e-businesses* is one of the major factors contributing to the surge in demand for recommendation systems. This can be easily verified by the success of ventures like Amazon, Netflix, Lastfm and others. In 2006, Amazon reported that over 35% of their sales are generated from the underlying recommendation system [2]. Furthermore, around 75% of the users select movies based on Netflix's recommendations [3].

One of the most popular technique for implementing recommendation systems : *collaborative filtering*, uses data related to the preferences of users from the past, to predict a user's new interests. This **on-site data**, predominantly in the form of ratings given by the users to various items of the system, customer preferences, click-streams and purchase and browsing history, is used to construct *user-profiles*. User-profile, constructed for each user of the system, is a representation of her affinities for one or more items in the system. For example, in a system like IMDB, Alice's user-profile can be assumed as : {`Pulp Fiction: 4; 500 Days of Summer: 2; Forrest Gump: 5; Inception: 3; Fight Club: 4; Crossover: 1`}, where each movie *seen* and *rated* (here in a range of 1-5) by Alice is stored. In case of collaborative filtering, these profiles, constructed using users' preference data, are the foundation of recommendations. Similarly in case of other techniques of recommendations, like content based and hybrid recommendations, it is the *data* that plays a pivotal role in the process of predictions. As we have seen in chapter 2, these techniques deviate from using only traditional on-site content like most of the collaborative filtering techniques. Instead, they rely on more complex form of data: **contextual data**, with or without combining it with customer preference data (or the on-site data). Mobile usage data, social and economic interactions, geolocation data, email responses and conversations etc. are some of the forms that contextual data can take.

Even though the generation and the consumption of data on internet has increased manifold over the years yet it has been a problem for some internet services and businesses to acquire enough data to make good recommendations and provide their customers with a satisfying level of personalization. According to a survey conducted by a major e-commerce consultancy firm: FitForCommerce [1] in Figure 5.1, there are around 33% of businesses on internet that lack access to basic data to make recommendations.

Just like the lack of data is a challenge in making good recommendations, Derrick Harris, in his work [3], points out that excess of data can also be a challenge in making relevant predictions and providing a good customization experience. The flooding of internet with data has lead to increasing number of research works to analyze the pros and cons with the volume of data, targeting recommendations and personalization [5, 65, 80]. With the above statistics we conclude that data *quantity* is an important parameter, and must be tuned accordingly to make relevant and desirable recommendations.

The survey in Figure 5.1 clearly shows that there are more challenges related to data in recommendation systems than its availability. 35% of the total e-businesses surveyed report

1. http://www.fitforcommerce.com/

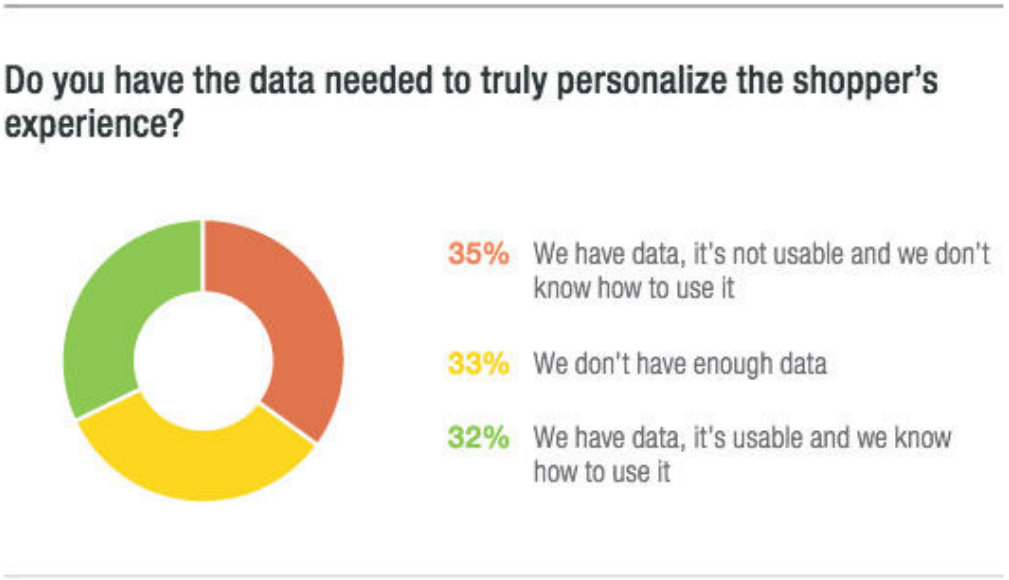


Figure 5.1 – Survey from Fitforcommerce showing that 33% of businesses on internet do not have access to enough data for personalization.

that there is data but not usable for recommendations/personalization. Another 32% of e-businesses have data which is usable but they do not have enough knowledge to use it. With these numbers, one can easily conclude that data quality is another challenge in rendering it suitable for the purpose of recommendations and personalization. Shalom *et al.* [132] also elaborate the challenge of treating data to improve its quality to be able to use it for recommendations. Their main focus is the problem of data sparsity and redundancy, which hampers the quality of data, even if it is present in huge quantities. Data quality is a well studied and document problem but little has been done in its application to recommendation systems. In this work, we deviate from the approach of Shalom *et al.* [132] and provide more generalized analysis of data quality in light of recommendation systems.

As we have seen in chapter 2, online social networks have also become significantly important for recommendation systems, in addition to being social connection and sharing platforms. While on one hand, these OSNs can themselves be considered as a form of trust-aware recommendation systems, on the other hand, they are more and more used to extract user information. Demographic recommendations, for example, rely on the user's demographic information, which is not always easily accessible, or not in quantities enough to make good recommendations. In such a scenario, OSNs come handy and can provide with a lot of user information, most of which is publicly available. Online social networks require their users to create a profile, which consists of several user attributes, for example, name, gender, age, profession etc. The user-profile attribute vary as per the social network into consideration. The application domain that the OSNs serve has also expanded quickly over the past years. They cover important areas like social relationships, professional ties, news recommendations, photo sharing etc. Therefore, it is only natural for a user to be a member of several of these OSNs instead of just one, to access all these application areas. As a result of users present in multiple social networks, their personal information and attributes are also naturally spread over these different OSNs. Our work is based on the assumption that combining user information from different platforms can lead to more powerful information. We exploit the combination of user's public profiles from different OSNs to infer new information about the user.

After treating the two aspects of data: the quality and quantity, we also provide statistical results to show how combining data from various sources can render it more powerful. We measure this power in terms of inference or prediction capabilities of the combined datasets, and compare the results to that of individual datasets. Our experiments and results are based on unique datasets (that cannot be publicly released due to company policies) collected from various online social networks, and are explained in detail later in this chapter.

5.2 Data Collection

In this section, we will explain the process of data collection. In our earlier discussion, we have highlighted the fact that a user is likely to be present on more than one social network. She has a user profile on each of the online social network that she uses, and these profiles in turn, can each have several different attributes. Though there are several attributes that are repeated over different social networks, yet there can be attributes unique to each of these networks. When the attribute information is redundant over networks, sometimes it can be inconsistent and ambiguous, as we will see in details in the next Section 5.3. Therefore, we collect and store information from a set of four different social networks, namely *Facebook*[2], *Google+*[3], *Twitter*[4] *and LinkedIn*[5]. Before we develop our motivation for the choice of these networks, it is important to note that we collect user-profiles only for the users who have public profiles on all four of these networks, to fulfill the aim of combining their profile information. The choice of these online social networks is dependent on some of the factors below:

- Facebook, Twitter and LinkedIn cover distinct application domains and hence can provide with very varied information and interests of users.
- The user-base is very large for each of these four online social networks, thereby, enabling us to have a bigger dataset.
- These are among the most famous OSNs and therefore it is easy to find overlapping users.

We use the Google+ profiles extracted by Perito *et al.* [120] as our base for further data collection. The original dataset consists of 3.5 million Google+ public profiles, and each user has a unique username. To collect the data from three other social networks, we assume that there are many users who use the same username over several platforms. There can be several reasons for keeping the same username, sometimes a username is a part of person's identity like her name, age or combination of such attributes, or for the convenience of remembering only one username for using different platforms, or due to lack of awareness about the security risks etc. From the 3.5 million usernames, we obtained 9,465 distinct usernames whose users have *active* and *public* profiles on each of the three other online social networks, Facebook, Twitter and LinkedIn. The users were matched with their Google+ usernames. The following paragraphs give details of the data collection from each of the social network considered in this work.

Twitter: is a microblogging platform where users have *follower-followee* relationships with each other. These relationships are uni-directional, meaning if A follows B without B having to follow A. In its originality, a user follows another user to receive *tweets* from her. Tweets are maximum 140 character long messages, which are mostly news or opinionated messages. Twitter provides the developers and researchers with REST API[6], that allows col-

2. http://www.facebook.com
3. http://plus.google.com/
4. http://twitter.com/
5. https://www.linkedin.com/
6. https://dev.twitter.com/rest/public

lecting data of twitter's users, for the purpose of research and study. This REST API allows access to user information like:

— UserId
— Name
— Screen name
— Followers count
— Followers List
— Maximum of 3200 latest tweets (including retweets)
— Textual description of the user, if she has provided any
— Location
— Timezone

In the information mentioned above, only the userId, name and screen names are the required fields, and the remaining are optional. We used the Java library Twitter4J [7] to access the Twitter REST API, to collect the user information from using the user ids. The table 5.1 shows the number of missing values for each attribute from Twitter dataset, from a total of 9,465 users. We do not include userid, name and screen name as these are present for all the users, being mandatory fields.

Attributes	FollowerCount	FollowerList	Tweets	Description	Location	TimeZone
# Missing Values	0	0	0	7,234	1,256	5,234

Table 5.1 – Number of missing values for each attribute in Twitter dataset for a total of 9,465 users

Facebook: is a social networking platform where the main connections between the users exist in form of "friendships", which are bidirectional. The idea of Facebook is to mimic the real life relations in the digital environment, by sharing *statuses*, photos and sending direct messages to other users. The statuses, unlike the tweets in Twitter, are not limited in number of characters. As clear from the functionality and features of Facebook, it originated for the purpose of social interactions. Though Facebook has its own Graph API [8] to enable reading and writing to the Facebook network, it is only accessible by apps, and not individuals. Therefore, to get user attributes from the public profiles, we use a crawler designed in Ruby. Facebook allows users to select the attributes that they want to be displayed as a part of their public profile. Below we give the list of attributes that were displayed in at least one of the profiles crawled during the collection procedure.

— Gender(G)
— Age
— Location (Loc)
— Hometown (Home)
— Language (Ln)
— Music
— Movies
— Books
— TV
— Friend list (FrLst)

Table 5.2 shows the number of missing values for each of these attributes, from a total of 9,465.

7. twitter4j.org/en/
8. https://developers.facebook.com/docs/graph-api

Attributes	G	Age	Loc	Home	Ln	Music	Movies	Books	FrLst
# Missing Values	1,563	1,098	1,753	2,034	6,762	2,346	2,454	2,968	0

Table 5.2 – Number of missing values for each attribute in Facebook dataset for a total of 9,465 users

LinkedIn: LinkedIn is a business oriented networking platform where the relationships exist in form of bidirectional *connections*. It is a network of professionals, with connections based on educational and career domains, like classmates, colleagues, faculties, professors, etc. Clearly, it is a more formal network than its two counterparts. LinkedIn also provides a REST API that can be easily programmed with some Java libraries but the fields returned by the API are not enough for our purpose, so we use a scraper written in Ruby, scraping the user information from their linkedin profiles. We collect the following information from the users' profiles, though there are other fields such as profile summary, and languages but they require advanced text processing to extract useful information from them. We consider that out of the scope of our goal, to perform a simple empirical survey and analysis on data and its inference capabilities.
— Birthday
— Job title
— Location
— Country
— Skills
— Schools
In the attributes extracted from linkedin, there is no missing information, except birthdays that is present for 4,458 users out of a total of 9,465 as shown in table 5.3.

Attributes	Birthday	Job	Location	Country	Skills	Schools
# Missing Values	4,458	0	0	0	0	0

Table 5.3 – Number of missing values for each attribute in LinkedIn dataset for a total of 9,465 users

5.3 Data Cleansing and Preprocessing

In this section, we see how the missing values were handled. Data preprocessing and cleansing is a subject that is well focused on by the research community. As a result, there are existing methods well suiting our requirements, and hence we restrict ourselves to using the same.

5.3.1 Missing Values

Gender: As we see in the previous tables, gender attribute of the user is available only from the Facebook dataset, which has around 1.5K missing values. Literature provides interesting methods for handling various kinds of missing attributes, including categorical and numerical values [55]. In this particular case, we make use of other user-attributes to impute the gender of the users with missing values. More specifically, *names* of users are considered as good associations to impute the missing gender fields. As we have seen above, *names* are available, both from the twitter and the linkedin dataset. Clearly, linkedin being a more professional platform is more unlikely to have fake user profiles or attribute values. The reason is obviously in the application area of the network, i.e., for professional purposes. Therefore, we chose the *names* attribute values

coming from linkedin, in combination with an API specially designed for guessing gender values from first names. Though we chose linkedin to take the values of the “name" attribute but we observe that 93.8% users have same values for “name" on both twitter and linkedin. “Genderize API" [9] is a well-tested API with promising results. The database used for training the API comprises of millions of user profiles coming from many different social networks. Another viable solution could be to build a simple learning model, with a database of “name-gender" associations. Though practically feasible, it requires a large database of names and associated genders for the training purpose. Also, as our datasets from twitter, facebook and linkedin are not location specific, it model in consideration might not work with only english-names database. Hence, using an existing API is one of the best method of imputing gender, given the first names of the users. As the Genderize API has a rate limit of 1000 names per day, it took our application around 2 days to get the gender for all the missing values.

Age: attribute can be availed from facebook as well as linkedin dataset. From the facebook dataset, the attribute is directly available whereas on linkedin, its the birthday of the user that is available. Before imputing the missing age values, we combine the values of age from the two datasets. Following the same reasoning as before and considering linkedin information more reliable, we keep the age values obtained from the linkedin dataset in case of ambiguity. After combining the age values from two sources, there are a total of 3,697 missing values. Instead of using the exact age values, we use age range with an interval of 10.
From linkedin dataset, we also have information about the “school" attribute of the users, which in 97.5% of cases is accompanied by the value of *passing year*. We use this value to infer the age group of the users. For example, a user with passing year value of 2011 is likely to be 17 years old in the year 2011, and hence, 25 or 26 years old in 2015. In either case, the user would belong to the age range group of 20-29 years. Using the above information, our dataset consists of 0 missing age values.

5.3.2 Ambiguous Data

As we have seen in case of “age" attribute above, there is an inconsistency in values coming from various data sources. We follow, in order, the two following rules, to deal with inconsistent data values:

1. Majority of values over three networks. For example location can be obtained from either of the three datasets. In a case like this, we chose the value that is prevalent in the majority of the datasets.
2. In case of no clear majority, or when there are values coming from only one dataset, we consider values from linkedin as our ground truth. This is attributed to the professional nature of the linkedin network.

5.3.3 Text processing

The datasets collected above have many attributes containing textual information. Movies, music and book attributes from facebook dataset, tweets and description from twitter dataset, and skills attribute from linkedin dataset are attributes containing data in form of text, mostly unprocessed natural language text. Movies, music and books are generally available as nouns corresponding to existing movie, music and book names. We process these attributes to obtain categorical values, for example, instead of movie names, we obtain movie genres for each user,

9. https://genderize.io/

for the values present in the datasets. This is an important step to reduce the complexity of using and training the inference algorithms. Granularity of information to be inferred plays a significant role in its inference. There are experimental studies in this direction but it can be understood as an obvious phenomenon also. Pontes *et al.* [121] show how finer-grained inference is more difficult as compared to coarser information chunks. In their work, they specially talk about “geo-location" data, but it is easy to bring the parallelance to other attributes also. Roughly speaking, it is easier to impute what kind of movies a user would like as compared to guessing the “movie names".

Movie, Music and Books Data

Categorizing the values from movies, music and books is done with the help of corresponding datasets, and APIs publicly available. To determine the genres of the movies, we use the 1M movielens dataset [10]. The genres available for the movies are shown in Table 5.4.

action	adventure	animation	biography	children	comedy
crime	drama	fantasy	film-noir	horror	mystery
romance	sci-fi	sport	thriller	war	western

Table 5.4 – Movie genres derived from Movielens dataset

For the “music" attribute, the values are mixed, in the sense that users use different ways of writing their music preference. They use names of artists, song titles, albums, genres of music etc. For the purpose of homogeneity and to have coarse grained information for the reasons mentioned above, we use the API from Last.fm [11] to replace these values by music-tags (which are music genres) given by the API. There are around 12% music values for which no tags were suggested. We used some music websites to annotate these values manually. The extracted music genres are given in table 5.5.

metal	pop	folk	country	rock	indi rock	contemporary
hip hop	jazz	classical	blues	bollywood	children’s music	choral
punk	opera	indie	rap	broadway	dance	new age
reggae	anime	beats	asian Pop	electronic	jingles	exercise
grunge	inspirational	soul	trance	dubstep	swing	gospel

Table 5.5 – Music genres derived (mostly) from Last.fm API

We used a Java program to extract the book genres from the Goodreads.com website [12]. The book genres are extracted and presented in table 5.6.

romance	fantasy	comics	crime	mystery
poetry	sci-fi	psychology	horror	history

Table 5.6 – Book genres extracted from Goodread.com

10. http://grouplens.org/datasets/movielens/1m/
11. http://www.last.fm/api
12. http://www.goodreads.com/genres

Tweet processing

Twitter provides its users with the functionality of “hashtags" : words preceded with ‘#’, which are used annotate keywords in tweets, that can be 140 characters long. Generally hashtags are words that give the essence of the whole tweet or idea behind it. However, sometimes it is not easy to guess the meaning and context of the information that the hashtags want to convey. In other cases, there are hashtags that occur together, and their co-occurrence has a different meaning than each of the hashtags alone. These and other such problems make hashtag and tweet processing a very tedious and sophisticated task, involving intricate natural language processing. To classify the extracted hashtags under pre-defined categories, that we will discuss shortly, we build an API to query tagdef.com [13] and Hashtags.org [14] and create associations between hashtags and trending categories. As a result of this, we obtain a set of 800 trending topics for the 623,234 unique hashtags of our dataset. We further reduce the trending topics by manual annotation into 35 categories. Topic modeling, as discussed in Chapter 2, is a very popular and well researched subject since the advent of social networks and microblogging services. This work uses a simplified process of topic modeling, by using the above mentioned websites in combination with a supervised naive bayes classifier, from weka machine learning library [15].

5.4 Inference Results

We would like to stress that the focus of this contribution is not to devise new inference mechanisms, but to study the inference capabilities of user data, using the existing inference techniques. Therefore, we only show the results using naive bayes inference approach, which is also used in the state-of-the art approaches [42], where we derive our motivation from. We experimented with other inference approaches like SVM and Logistic Regression and obtained similar results.

To investigate the inference capabilities of attributes and their combination, we start by using individual attributes to infer target attributes. We use “accuracy" to measure the quality of inference. It is done using a 10-fold cross validation technique, where average of accuracy values obtained at each of the 10 fold is computed. Table 5.7 shows the results of inference when only one source attribute is used. We see that some attributes are more powerful for inference as compared to others. For example, twitter trends in general are more powerful than any other attribute for inference, regardless of the target value.

		Target Attributes						
		Age	Books	Gender	Movies	Music	Skills	Twitter Trends
Source Attributes	Age	-	0.212	0.333	0.102	0.110	0.312	0.224
	Books	0.240	-	0.315	0.118	0.254	0.185	0.354
	Movies	0.196	0.224	0.310	-	0.179	0.268	0.168
	Music	0.312	0.225	0.357	0.228	-	0.219	0.262
	Skills	0.334	0.175	0.398	0.199	0.220	-	0.312
	Twitter Trends	0.394	0.299	0.357	0.286	0.255	0.366	-

Table 5.7 – Values of inference accuracy for using individual source attributes for each target attribute.

To understand the power of aggregation, we now present the values of inference accuracy for various combinations of source attributes. With the choice of source attributes that we have,

13. https://tagdef.com/
14. https://www.hashtags.org/
15. http://www.cs.waikato.ac.nz/ml/weka/

there can be a lot of possible combinations, and due to obvious reasons, we not present all of them here. Our main focus is on the importance of Twitter data, which also makes us different from our state-of-the art approaches. Therefore, the rest of the results are presented with the twitter trend attribute., combined with different source attributes.

	Target Attributes				
	Books	Gender	Movies	Music	Skills
Age + Twitter Trends	0.357	0.574	0.440	0.553	0.397
Age + Gender + Twitter Trends	0.567	-	0.601	0.597	0.531
Age + Music + Twitter Trends	0.559	0.606	0.554	-	0.528
Age +Gender + Music + Twitter Trends	**0.572**	-	**0.677**	-	**0.702**

Table 5.8 – Values of inference accuracy obtained by using age and twitter trends as source attributed, for all the target attributes.

From Table 5.8, we notice that twitter trends is a strong attribute in predicting user's interests in other fields like movies, books and music. We observe that the combination of twitter trends with other attributes like gender and age is very successful in predicting the *skills* of users, with an average accuracy of around 70.2%. We would like to mention that these results are obtained using Naive Bayes classifier, but similar results are obtained by using other inference techniques like SVM and Logistic Regression. Other experiments yield the same conclusion: aggregation of attributes to infer a target attribute is more powerful than individual attributes.

5.5 Conclusion

With the help of this empirical study, we show how various services can make use of users' data, from various sources, specially online social networks, in predicting users' interests in other fields, like movies, music, books and skills. In the light of this work, we also discuss the process of user data collection from three different online social networks, that is publicly available. Furthermore, without going into details, we give a brief introduction about the data cleansing methods, which are different than the contemporary methods highlighted in the literature. These methods for data preprocessing are attribute and dataset dependent, though our results show their effectiveness in the task of new interest prediction.
Our work lays important foundations for the two following studies:

1. This is an study highlighting the importance of user attributes to predict her unknown interests. In particular, the opinions expressed in form of posts, tweets etc on online social networks, is a very helpful attribute in inferring her interests in movies, music and predicting her skill sets. We show how the importance of this kind of opinionated data can be enhanced by combining it with other available attributes. This type of inference can facilitate users' interests to train a learning model of recommendation systems, specially the ones that suffer with the problem of lack of data.
2. While on one hand the aggregation of users' attributes reveals more information about the users and is very helpful for the service providers, on the other hand, this extra revelation is a threat to users' privacy. It might reveal the information that the user doesn't necessarily want to. Such attacks, as we have discussed in Chapter 2 are called inference attacks and breach users' privacy. Empirical analysis like the one we perform can help in understanding the inference attacks better, by understanding the role of each attribute

separately and in combination with others. They can help in measuring the information leak in quantitative terms to devise anti-inference attack mechanisms.

6
Conclusion and Perspective

A conclusion is a place where you get tired of thinking.

Arthur Bloch

We have underlined the importance of recommendation systems in the current age of internet dominated marketplaces and services. Recommendation systems help the users sift through the data overloaded services and applications to find their way to the content that they would be interested in. Modern recommendation systems are designed keeping in mind the *personalization* needs of each user. In this thesis, we have recognized five types of recommendation systems: collaborative filtering, content based, demographic, trust aware and hybrid. Each of these systems is unique in the conceptualization of the idea of providing recommendations, and its implementation. This uniqueness leads to a large choice of tools and algorithms that can be used for each of the above recommendation system types. Moreover, combined with diverse range of applications that the recommendation systems serve, this choice becomes even larger, thus making recommendation systems a *multi-disciplinary* field. Clearly, each candidate from the choice of the available techniques to construct recommendation systems comes with its own challenges, and more the field, more the challenges.

WITHOUT RECOMMENDATION SYSTEMS, COMPANIES AND CUSTOMERS ARE BLIND AND DEAF, WANDERING OUT ONTO THE WEB LIKE DEER ON A FREEWAY.

In this work, we have identified three different areas associated to the design and implementation of recommendation systems that we study in details and provide answers to some of the challenges associated to them. In light of this, we see recommendation as a three step process, starting with acquiring the *data*, using it to *learn a model* and then finally making recommendations using the learnt model. In addition to this, we also concern ourselves with another trending concern in the field of recommendations in particular, and internet in general: *privacy*.

In the rest of this chapter, we will summarize each contribution separately and before giving an insight into the future prospective.

RIPOSTE

Privacy of users has become one of the most challenging concerns for many services and applications available on the world wide web. It becomes even more serious when it comes to the online social networks, as the user privacy can be compromised not only by the user herself but also by her connections, friends, followers etc. There are many ways in which a user's privacy can be threatened. With RIPOSTE we provide users with a mechanism to be able to disseminate information privately in online social networks. RIPOSTE assures that the properties of dissemination are preserved, and the users are able to make their dissemination *differentially private*. We use experimental evaluations to show the effectiveness and working of RIPOSTE on a diverse range of datasets.

KIFF

KIFF is a new *k-nearest neighbor* graph construction algorithm, which finds its application in many areas like search, information retrieval, classification and recommendation systems. It is a fast and scalable solution for constructing a KNN graph, leveraging the bipartite structure of the datasets involved in these applications. In addition to its scalability, KIFF is also a generic solution, meaning that it performs equally well for all the choices of similarity metrics, which is generally not the case of most of the KNN graph construction algorithms. We evaluate the performance of KIFF using datasets from different domains, with varied characteristics. We also give a detailed account on how sparsity of the underlying datasets effects the performance of all KNN algorithms in general, and KIFF in particular.

Understanding the power of data

In our empirical study of data collection, cleansing and inference, we show how different attributes contribute to the inference of different information. We chose three different social network to collect data, which is a unique dataset in itself, comprising of various user attributes. Different data cleansing and preprocessing tools are used to obtain the data in the desired format, and to make it more consistent and as least ambiguous as possible. We also categorize the user attributes, obtained from three different sources, into various genres and comment on the inference power of each of these genres. Our study highlights the power that aggregation of user attributes can have, specially for the purpose of inferring user interests in targeted domains.

Future Prospects

We believe that RIPOSTE lays very strong foundation for preserving users' privacy in internet-based activism in particular, and disseminating opinions and ideas on an online social network in general. In our theoretical and practical setup of RIPOSTE, we make certain assumptions regarding the opinion of users of OSN for a each item in the system.

— In our current setting, we assume that all the users are equally likely to like a given item, independent of its position in the network and independent of the opinion of other users in the network. It would be an interesting extension to evaluate RIPOSTE using algorithms that model the probability for each user for each item in the system. This work can be combined with the existing studies of information diffusion models and techniques in online social networks.

— RIPOSTE randomizes a user's decision of propagating (or not) an information item in the network. This may lead to cases where user likes an item and it is not disseminated in the network due to the randomized decision taken by RIPOSTE. In light of such cases, one can examine the effect it will have on the *user-influence* in the network. *Influence of a user* is a widely studied phenomenon which roughly gives an idea of a user's power of influencing other users in the network. Depending on the social network into consideration, this influence can be measured as a factor of several parameters, like number of followers, number of reposts of the information occurring from this user etc.

— As RIPOSTE derives its motivation from randomized response techniques, and we test it on various social networks, we understand that it can be used for other applications working on similar principles, like that of distributed polling.

Privacy, as we underline in our work, is an issue of significant concern. We provide RIPOSTE as a direct measure to counter the privacy issues involved in disseminating information in social networks.

— In one of our contributions, we demonstrate the *power* of user data, specially coming from the OSNs. While on one hand this data is blessing for personalization based services like recommendation systems, on the other hand, it also leads way to threats like information leaking an inference attacks. While we mention these threats in our work, we do not explore how public posts from social networks like twitter and facebook can pose danger to user's privacy and reveal information that the user doesn't reveal *directly*. Taking inspiration from works done to obfuscate data to conceal user's undisclosed attributes, this study can be extended to measure exact quantities of information leak from each attribute and their possible combinations, focusing on *reviews*, *opinions* and similar information sources.

— *Homophily* is another cause of revealing users' private and personal details. In case of *homophily* related attacks, users' connections are exploited to learn new details about the users. Influential studies show the strength of homophily in exposing users' hidden information, but as per our knowledge, it is done for individual networks. We can extend our work to take into consideration the connections of users to infer more information about them. However, one of the challenging issues here is to have access to such experimental datasets, even with small number of users.

In the coming paragraph, we highlight some research areas that are related to the field of recommendation systems in general, not only with the focus of extending our contributions. Recommendation systems, as we have seen, are a multi-disciplinary field. They are a result of careful mix of many subjects and domains, each with it's own challenges. We have identified and reviewed some of them in this thesis. From the various discussions during this work, we have seen that *privacy* has become one of the most alarming issues, for internet based applications in general and recommendation systems in particular. Privacy of users can be compromised at various stages of the recommendation process, right from the collection of data, then while designing the learning model and even during recommendations. We have also highlighted some of the major ways in which the users' privacy can be undermined. Taking inspiration from the importance of users' privacy, this work can be extended to make recommendations in a complete privacy preserving manner. This would imply the use of privacy-preserving techniques at various levels, for example, calculating the exact privacy that is leaked while getting users' data from their browsing and navigation history, and through inference. Once the information leak is quantitatively measured, the next step is to minimize this leak while still being able to access useful user information.

Online social networks play a very important role in all kinds of recommendation systems. Mostly to learn about the users, and acquire user demographic information which is otherwise not available on recommendation system platforms. History has witnessed a lot of privacy thefts on social networks, directly inflicted on the services that make use of such social networks. Therefore, combining OSNs with recommendation systems make an interesting area of research, for making better quality recommendations but it also presents a challenge of doing it in a privacy preserving manner.

As closing remarks, we believe that the quality of recommendation and associated users' privacy are to be traded-off. As seen in the literature, most of the services (businesses and websites) make assumptions on users' privacy and only focus on high end recommendations. It would be interesting, not only in theoretical world, but also practically to see recommendation systems being aware of users' privacy, with least compromise on recommendation quality.

A

Résumé en Français

La croissance et la popularité du web nous ont conduits à une nouvelle ère de données. Elles ont fourni une plate-forme pratique et facile à utiliser pour les consommateurs et les fournisseurs de services. Avec l'avènement d'Internet, les e-entreprises ont connu une croissance exponentielle, non seulement de leurs services et produits, mais également de leur base de consommateurs. De même, Internet est également responsable du développement rapide des réseaux sociaux en ligne. Alors qu'Internet a ouvert la voie aux e-entreprises et aux réseaux sociaux en ligne, il a également conduit à une explosion de la quantité informations produites à partir de tous les services en ligne, qui est devenue beaucoup plus grande que ce que toute entreprise, organisation ou personne peut raisonnablement consommer. Des milliers de nouveaux produits sont ajoutés chaque jour sur les sites d'e-commerce. Environ de 40 milliards de photos ont été postées sur Instagram [1] depuis sa création en 2010. Des millions de tweets sont générés chaque heure, et environ 62Une telle quantité d'information confronte les utilisateurs à la tâche difficile de les filtrer pour trouver les plus pertinentes et intéressantes. Ce large choix ainsi que le temps nécessaire pour les parcourir peuvent souvent être déroutant et frustrant pour les utilisateurs. Beaucoup d'attention a été porté au problème de surcharge d'informations au cours de ces dernières années. Les moteurs de recherche ont fait partie des premières solutions apportées au problème de surcharge d'informations, basés sur le concept de popularité du contenu. Plus le contenu est vu, plus le produit est acheté, ou plus les nouvelles sont partagées, plus il est considéré comme populaire et plus il apparaît dans les résultats de recherche. Cela signifie également que les mêmes résultats sont retournés à tous les utilisateurs d'Internet (qui sont plus d'un milliard), indépendamment de leurs goûts individuels. Puis est née l'idée de fournir des résultats dépendant des intérêts et des goûts des individus. Ainsi sont apparus les systèmes de recommandation.

Chaque individu fait usage de recommandations dans la vie quotidienne, par le bouche-à-oreille, les guides de voyages, etc. A l'origine, les systèmes de recommandations sont l'implémentation algorithmique de l'idée de se baser sur l'opinion d'un groupe de personne dans le but de sélectionner du contenu à partir d'une grande variété de choix. Ils ont évolué avec le temps et on inclus d'autres idées dans leur conception, toujours dans le but de personnaliser les résultats pour chaque utilisateur, en utilisant au maximum l'information disponible. En conséquence, les systèmes de recommandations sont maintenant un important domaine de recherche, com-

1. https://www.instagram.com/

prenant diverses techniques de recommandation comme le filtrage collaboratif, les recommandations basées sur le contenu, sur la confiance, sur les données démographiques, ainsi que des méthodes hybrides.

Dans un cas classique, le calcul de recommandation par un système de recommandation peut être considéré comme un processus en trois étapes. D'abord, les données sont collectées et traitées afin d'obtenir les propriétés souhaitées pour être utilisé par un modèle d'apprentissage. Ensuite, le modèle est entraîné sur les données d'entrées. Enfin, celui-ci est utilisé pour faire des prédictions ou des recommandations.

Les systèmes de recommandation sont un domaine multidisciplinaire, impliquant des outils et des techniques de divers domaines pour accomplir ces trois étapes. Des méthodes de recherche d'informations sont souvent utilisées pour collecter des données provenant de diverses sources. L'utilisation de techniques d'exploration de données telles que la classification, les réseaux de neurones et les machines à vecteurs de support sont très répandues pour entraîner le modèle à apprendre des données et faire des prédictions. Pour finaliser des prédictions et/ou des recommandations, ces systèmes utilisent diverses métriques et mesures pour assigner des scores aux produits pour chaque utilisateur. Enfin, différentes métriques d'évaluations provenant du domaine des statistiques sont utilisées dans le but d'évaluer la performance des systèmes de recommandation sur différents critères. Par conséquent, les défis des systèmes de recommandation sont multiples, et proviennent de nombreux domaines. Nous ne pouvons pas tous les couvrir dans le cadre de ce travail, nous nous concentrerons donc sur trois domaines principaux : les données, les modèles d'apprentissage, et la vie privée.

Data

Challenge

Les données constituent la base des systèmes de recommandation, quel que soit leur type. Les techniques de filtrage collaboratif utilisent l'opinion et les préférences antérieures des utilisateurs, alors que les techniques basées sur le contenu font appel à des données descriptives associées aux produits. Dans des techniques de recommandations basées sur la confiance, les liens sociaux et les relations entre les utilisateurs jouent un rôle prépondérant dans le processus de recommandations. Fondamentalement, sans données il n'y a pas d'apprentissage, et sans apprentissage il n'y a pas de recommandations.

Avec l'immense quantité de données provenant de diverses sources sur Internet, il est surprenant que les données soient toujours un défi dans les systèmes de recommandation. Néanmoins ce n'est plus nécessairement l'absence de données qui est un défi. Les problèmes associés aux données peuvent être classés en deux groupes: ceux liés à la qualité des données, et ceux liés a leur quantité. Entre autres, voici des problèmes communs liés aux données:

— Il y a trop de données à filtrer.
— Il n'y a pas assez de données utiles pour faire des recommandations.
— Les données ne sont pas disponibles dans un format utilisable.
— Il n'y a pas suffisamment de connaissances, d'outils ou de techniques pour utiliser les données.
— Les données sur les utilisateurs et les produits peuvent inclure des données périmées, car il est probable que leurs préférences changent au fil du temps. Détecter ces changement peut être difficile.

Contribution

Les données sont un important sujet de recherche, en particulier avec l'avènement d'Internet et sa croissance rapide. À notre connaissance, il n'y a que très peu d'études s'intéressant en particulier aux systèmes de recommandation. Dans notre contribution, nous fournissons un aperçu de l'importance de la qualité et de la quantité de données afin d'élaborer des recommandations. Comme objectif principal de cette contribution, nous mettons en évidence la puissance cachée des données, sous forme d'inférence. Nous présentons notre contribution sous forme d'une étude empirique, en utilisant les techniques d'inférence provenant du domaine de l'apprentissage automatique. Dans notre travail, nous montrons comment des données d'utilisateur collectées sur trois réseaux sociaux différents peuvent être combinées et traitées pour en déduire l'intérêt des utilisateurs dans des domaines ciblés, fournissant ainsi un profil d'utilisateur plus fort. Ceci est un besoin indispensable des systèmes de recommandation. Par nos résultats expérimentaux, nous montrons que différents attributs de l'utilisateur ont différentes capacités d'inférence. Nous présentons aussi la différence entre les capacités d'inférence des attributs de l'utilisateur provenant d'une source unique et les capacités d'inférence d'attributs de l'utilisateur provenant de différentes sources combinées. En conclusion, notre travail constitue une base importante pour la recherche dans l'agrégation d'informations de multiples sources et d'inférence d'attribut de l'utilisateur.

Apprentissage

Challenge

Les progrès dans le domaine de l'exploration de données apportent un large choix de techniques d'apprentissage aux concepteurs de systèmes de recommandation. Cependant, le choix d'une technique d'apprentissage dépend de nombreux facteurs: de la disponibilité des données, des ressources de calcul, des techniques de recommandation utilisées, et du domaine d'application des recommandations proposées. Chacune de ces techniques, telles que la factorisation de matrice, les plus proches voisins, les arbres de décision, les machines à vecteurs de support, ou les règles d'association, sont très étudiées dans la littérature et il en existe de nombreuses variantes en fonction des besoins de l'application. Parmi d'autres, les méthodes basées sur le voisinage sont très populaires. Leur facilité de mise en œuvre et leur simplicité les rendent très attirantes, en particulier dans les systèmes de filtrage collaboratif. Toutefois, ces techniques font face à un défi: le passage à l'échelle quand le nombre d'utilisateurs et de produits augmente. Un autre problème est la faible densité des données. Par exemple pour le e-commerce, en raison de la gamme de produits très large, un utilisateur n'est souvent associé qu'à une petite partie d'entre eux. Par conséquent, les associations entre les utilisateurs et les produits, qui sont la base du filtrage collaboratif, ne suffisent pas à entraîner un modèle convenablement et à fournir de bonnes recommandations. Nous avons choisi de mettre l'accent sur la méthode des K plus proches voisins, qui est l'une des techniques d'apprentissage la plus répandue. Nous nous intéressons en particulier aux problèmes de densité des donnés et de passage à l'échelle.

Contribution

Bien que le KNN (K plus proches voisins) soit un algorithme très utilisé et très étudié, des défis existent dans la construction d'un algorithme KNN efficace. Avec toutes les données disponibles à partir d'Internet et de ses applications, il n'est pas rare de voir des jeux de données contenant plus d'un milliard d'utilisateurs et/ou d'articles. Dans un tel scenario, en raison

du besoin de comparer chaque paire d'utilisateurs, le calcul rapide et efficace d'un KNN est un défi crucial. Un autre problème est la recherche d'une bonne valeur pour le paramètre k. Si la valeur de k est trop basse, alors l'algorithme à tendance à n'inclure que des « outliers », alors que si elle est trop élevée, l'algorithme risque de sélectionner des voisins non pertinents. Trouver un moyen de calculer la similarité entre utilisateurs est également crucial, car la performance d'un KNN dépend fortement du choix de cette mesure de similarité. Pour résoudre ces problèmes, nous proposons KIFF, un algorithme de construction de graphe générique, rapide et capable de passer à l'échelle. KIFF exploite la nature bipartie de la plupart des jeux de données sur lesquels les algorithmes de KNN sont appliqués. Cette stratégie simple mais efficace limite drastiquement le temps de calcul nécessaire pour converger vers un graphe de plus proches voisins précis. Pour effectuer nos expériences, nous avons choisi un large panel de jeux de données permettant de montrer que KIFF est significativement plus rapide et passe mieux à l'échelle que les approches existantes. Notre évaluation montre que KIFF fournit en moyenne un facteur d'accélération de 14 comparé aux solutions existantes tout en améliorant la qualité de l'approximation du KNN de 18

Privacy

Challenge

Le terme de vie privée regroupe plusieurs sens. Dans le domaine des applications et services d'internet, il est principalement associé aux données personnelles. En termes simples, la protection de la vie privée implique de garder confidentielles les informations des utilisateurs. Une définition plus précise se concentre sur les personnes concernées par l'information, et les effets que la divulgation de ces information peuvent avoir sur elles, ainsi que leur contrôle et leur consentement sur ces effets. Si l'on s'écarte des services d'e-commerce, le terme de données personnelles prend d'autres sens. Par exemple dissimuler l'identité d'une personne, ou masquer leur préférences religieuses, sexuelles, politiques ou autres. En utilisant les réseaux sociaux, les utilisateurs partagent beaucoup d'information, principalement personnelles, sans forcément s'en rendre compte. Les information partagées, comme par exemple des photos, des status, des opinions, ou des émotions, peuvent toutes être utilisées contre la vie privée de l'utilisateur. Parfois, les utilisateurs ne sont pas conscients de la façon dont ces partages peuvent être nocif dans d'autres occasions, et ils n'ont pas d'alternative autre que de s'abstenir complètement d'utiliser les réseaux sociaux s'ils veulent s'en prémunir. Beaucoup de recherches étudient différents types de risques pour la vie privée et leurs implications, entre autres dans les systèmes de recommandations, et proposent des solutions permettant de combattre ces menaces. Il devient clair par ces travaux qu'il y a de nombreuses façons de s'attaquer aux problèmes de protection de la vie privée, mais il n'est pas raisonnable de tous les étudier. Par conséquent, nous avons choisi de nous concentrer sur les systèmes de recommandations appliqués aux réseaux sociaux.

Contribution

RIPOSTE est notre contribution s'attaquant à un des aspects des problèmes de vie privée sur les réseaux sociaux, dans le contexte d'un système de recommandations. Les systèmes de recommandations jouent un rôle important dans les réseaux sociaux: la construction du réseau, la diffusion d'information sur ce réseau, et la possibilité pour les utilisateurs de s'exprimer . Toutes les étapes, de la construction de son réseau à la diffusion d'information, implique un risque pour la vie privée des utilisateurs. Diffuser des informations sur un réseau social est similaire au fait de proposer des articles à un utilisateur dans un système de recommandation.

Dans un contexte comme celui-ci, il est très important de proposer à l'utilisateur du contenu pertinent. De même, il est aussi très important de masquer à l'utilisateur les contenus susceptibles de lui déplaire, afin de lui fournir une bonne expérience de navigation. Tout en s'assurant de répondre à ces besoins, il est crucial de garder sous contrôle les différentes menaces sur la vie privée des utilisateurs. Entre autres, une menace commune, en particulier avec les services de micro-blogues, est le fait d'être poursuivi et condamné pour avoir partagé des idées ou des informations. La diffusion libre de l'information est le cœur des réseaux sociaux et des services de micro-blogues, comme par exemple Twitter, LiveJournal, Facebook, Google+, etc. Un très grand nombre d'utilisateurs se servent de ces plateformes comme moyen de partager des nouvelles, leur pensée politique ou leur opinion sur tout autre sujet important. Bien que le cœur de ces services soit la diffusion libre des informations et des idées, il à souvent été remarqué que certains utilisateurs ont peur et se retiennent d'exprimer ouvertement leur opinion. Cela vient principalement de la peur d'être poursuivi et condamné; une pratique courante dans les pays à régime autoritaire bien qu'existant également dans les pays démocratiques. Fondamentalement, les obstacles à la dissémination des idées et des informations dans les réseaux sociaux proviennent de la peur des utilisateurs d'exposer leur vie privée.

Avec RIPOSTE, nous avons pour but de remplir ces objectifs des réseaux sociaux tout en gardant à l'esprit les menaces sur la vie privée des utilisateurs. Avec cet algorithme de diffusion d'information respectueux de la vie privée, l'objectif est de permettre aux utilisateurs de s'exprimer librement tout en les protégeant avec le principe de déni plausible.

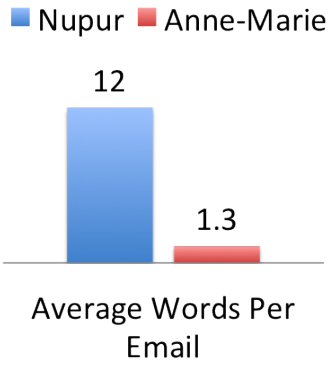
B

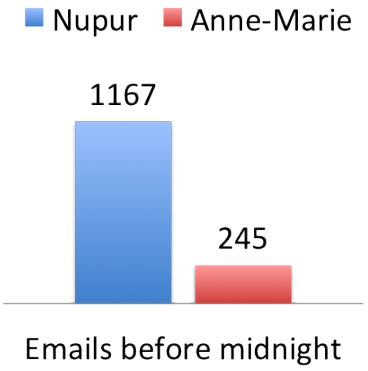
List of Publications

1. Giakkoupis, G., Guerraoui, R., Jégou, A., Kermarrec, A. M., & Mittal, N. (2015, October). Privacy-conscious information diffusion in social networks. In International Symposium on Distributed Computing (pp. 480-496). Springer Berlin Heidelberg.
2. Boutet, A., Kermarrec, A. M., Mittal, N., & Taïani, F. (2016, May). Being prepared in a sparse world: the case of KNN graph construction. In International Conference on Data Engineering (ICDE).

C

Extension of Acknowledgments





References

[1] KIFF: Knn Impressively Fast and eFficient java library, https://gitlab.com/antoine.boutet/KIFF.

[2] Aggregate knowledge raises $5*m* from kleiner, on a roll. `http://venturebeat.com/2006/12/10/aggregate-knowledge-raises-5m-from-kleiner-on-a-roll/`.

[3] Netflix analyzes a lot of data about your viewing habits.

[4] Twitter usage statistics. `http://www.internetlivestats.com/twitter-statistics/`.

[5] Why too much data is stressing us out.

[6] Alfarez Abdul-Rahman and Stephen Hailes. Supporting trust in virtual communities. In *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on*, pages 9–pp. IEEE, 2000.

[7] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, 17(6):734–749, 2005.

[8] Faiyaz Al Zamal, Wendy Liu, and Derek Ruths. Homophily and latent attribute inference: Inferring latent attributes of twitter users from neighbors. *ICWSM*, 270, 2012.

[9] Pedro Alves and Paulo Ferreira. AnonyLikes: Anonymous quantitative feedback on social networks. In *14th International Middleware Conference (Middleware)*, pages 466–484, 2013.

[10] Xavier Amatriain, Alejandro Jaimes, Nuria Oliver, and Josep M Pujol. Data mining methods for recommender systems. In *Recommender Systems Handbook*, pages 39–71. Springer, 2011.

[11] Xavier Amatriain, Josep M Pujol, Nava Tintarev, and Nuria Oliver. Rate it again: increasing recommendation accuracy by user re-rating. In *Proceedings of the third ACM conference on Recommender systems*, pages 173–180. ACM, 2009.

[12] Andris Ambainis, Markus Jakobsson, and Helger Lipmaa. Cryptographic randomized response techniques. In *7th International Workshop on Theory and Practice in Public Key Cryptography (PKC)*, pages 425–438, 2004.

[13] B. Amento, L. Terveen, and W. Hill. Does "authority" mean quality? predicting expert quality ratings of web documents. In *SIGIR*, pages 296–303, 2000.

[14] Sarabjot Singh Anand and Bamshad Mobasher. Intelligent techniques for web personalization. In *Proceedings of the 2003 international conference on Intelligent Techniques for Web Personalization*, pages 1–36. Springer-Verlag, 2003.

[15] D. C. Anastasiu and G. Karypis. L2knng: Fast exact k-nearest neighbor graph construction with l2-norm pruning. In *CIKM*, pages 791–800, 2015.

[16] Krishna B. Athreya and Peter E. Ney. *Branching processes*. Springer-Verlag, 1972.

[17] Lars Backstrom, Dan Huttenlocher, Jon Kleinberg, and Xiangyang Lan. Group formation in large social networks: Membership, growth, and evolution. In *12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 44–54, 2006.

[18] Xiao Bai, Marin Bertier, Rachid Guerraoui, Anne-Marie Kermarrec, and Vincent Leroy. Gossiping personalized queries. In *EDBT*, pages 87–98, 2010.

[19] Jonell Baltazar, Joey Costoya, and Ryan Flores. The real face of koobface: The largest web 2.0 botnet explained. *Trend Micro Research*, 5(9):10, 2009.

[20] Jon Louis Bentley. Multidimensional divide-and-conquer. *Communications of the ACM*, 23(4):214–229, 1980.

[21] Shlomo Berkovsky, Tsvi Kuflik, and Francesco Ricci. The impact of data obfuscation on the accuracy of collaborative filtering. *Expert Systems with Applications*, 39(5):5033–5042, 2012.

[22] M. Bertier, D. Frey, R. Guerraoui, A.M. Kermarrec, and V. Leroy. The gossple anonymous social network. In *Middleware*, pages 191–211, 2010.

[23] Leyla Bilge, Thorsten Strufe, Davide Balzarotti, and Engin Kirda. All your contacts are belong to us: automated identity theft attacks on social networks. In *Proceedings of the 18th international conference on World wide web*, pages 551–560. ACM, 2009.

[24] E. Bingham and M. Heikki. Random projection in dimensionality reduction: Applications to image and text data. In *KDD*, pages 245–250, 2001.

[25] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. Recommender systems survey. *Knowledge-Based Systems*, 46:109–132, 2013.

[26] D. Bortner and J. Han. Progressive clustering of networks using structure-connected order of traversal. In *ICDE*, pages 653–656, 2010.

[27] Yazan Boshmaf, Ildar Muslukhov, Konstantin Beznosov, and Matei Ripeanu. The socialbot network: when bots socialize for fame and money. In *Proceedings of the 27th Annual Computer Security Applications Conference*, pages 93–102. ACM, 2011.

[28] A. Boutet, D. Frey, R. Guerraoui, A. Jégou, and A.-M. Kermarrec. WhatsUp Decentralized Instant News Recommender. In *IPDPS*, pages 741–752, 2013.

[29] A. Boutet, D. Frey, R. Guerraoui, A.-M. Kermarrec, and R. Patra. HyRec: Leveraging Browsers for Scalable Recommenders. In *Middleware*, pages 85–96, 2014.

[30] John S Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 43–52. Morgan Kaufmann Publishers Inc., 1998.

[31] Robin Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370, 2002.

[32] Robin Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.

[33] Robin Burke. Hybrid web recommender systems. In *The adaptive web*, pages 377–408. Springer, 2007.

[34] Qiang Cao, Michael Sirivianos, Xiaowei Yang, and Tiago Pregueiro. Aiding the detection of fake accounts in large scale social online services. In *Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*, pages 197–210, 2012.

[35] M. Cha, H. Haddadi, F. Benevenuto, and K. P. Gummadi. Measuring user influence in twitter: The million follower fallacy. In *ICWSM*, 2010.

[36] M. Cha, A. Mislove, and K. P. Gummadi. A measurement-driven analysis of information propagation in the flickr social network. In *WWW*, pages 721–730, 2009.

[37] Sung-Hyuk Cha. Comprehensive survey on distance/similarity measures between probability density functions. *City*, 1(2):1, 2007.

[38] Arijit Chaudhuri. *Randomized response and indirect questioning techniques in surveys*. CRC Press, 2010.

[39] J. Chen, H. Fang, and Y. Saad. Fast approximate knn graph construction for high dimensional data via recursive lanczos bisection. *Journal of Machine Learning Research*, 10:1989–2012, 2009.

[40] Li Chen and Pearl Pu. Experiments on the preference-based organization interface in recommender systems. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 17(1):5, 2010.

[41] Ningyuan Chen and Mariana Olvera-Cravioto. Directed random graphs with given degree distributions. *Stochastic Systems*, 3(1):147–186, 2013.

[42] Terence Chen, Roksana Boreli, Mohamed-Ali Kaafar, and Arik Friedman. On the effectiveness of obfuscation techniques in online social networks. In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 42–62. Springer, 2014.

[43] Wei Chen, Yajun Wang, and Siyu Yang. Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 199–208. ACM, 2009.

[44] E. Cho, S. A. Myers, and J. Leskovec. Friendship and mobility: User movement in location-based social networks. In *KDD*, 2011.

[45] Kenneth L Clarkson. Fast algorithms for the all nearest neighbors problem. In *Foundations of Computer Science, 1983., 24th Annual Symposium on*, pages 226–232. IEEE, 1983.

[46] William G Cochran. *Sampling techniques*. John Wiley & Sons, 2007.

[47] M. Connor and P. Kumar. Fast construction of k-nearest neighbor graphs for point clouds. *Transactions on Visualization and Computer Graphics*, 2010.

[48] Nick Couldry and James Curran. *Contesting media power: Alternative media in a networked world*. Rowman & Littlefield Publishers, 2003.

[49] 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*, pages 271–280. ACM, 2007.

[50] Luis M De Campos, Juan M Fernández-Luna, Juan F Huete, and Miguel A Rueda-Morales. Combining content-based and collaborative recommendations: A hybrid approach based on bayesian networks. *International Journal of Approximate Reasoning*, 51(7):785–799, 2010.

[51] Pasquale De Meo, Emilio Ferrara, Giacomo Fiumara, and Alessandro Provetti. On facebook, most ties are weak. *Communications of the ACM*, 57(11):78–84, 2014.

[52] William Edwards Deming. *Some theory of sampling*. Courier Corporation, 1966.

[53] I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *KDD*, pages 269–274, 2001.

[54] Cong Ding, Yang Chen, and Xiaoming Fu. Crowd crawling: Towards collaborative data collection for large-scale online social networks. In *1st ACM Conference on Online Social Networks (COSN)*, pages 183–188, 2013.

[55] A Rogier T Donders, Geert JMG van der Heijden, Theo Stijnen, and Karel GM Moons. Review: a gentle introduction to imputation of missing values. *Journal of clinical epidemiology*, 59(10):1087–1091, 2006.

[56] W. Dong, C. Moses, and K. Li. Efficient k-nearest neighbor graph construction for generic similarity measures. In *WWW*, pages 577–586, 2011.

[57] John R Douceur. The sybil attack. In *International Workshop on Peer-to-Peer Systems*, pages 251–260. Springer, 2002.

[58] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *3rd Theory of Cryptography Conference (TCC)*, pages 265–284, 2006.

[59] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014.

[60] Jennifer Earl. The dynamics of protest-related diffusion on the web. *Information, Communication & Society*, 13(2):209–225, 2010.

[61] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. RAPPOR: Randomized aggregatable privacy-preserving ordinal response. In *ACM Conference on Computer and Communications Security (CCS)*, pages 1054–1067, 2014.

[62] Facebook. Facebook news room. `http://newsroom.fb.com/company-info/`.

[63] Rino Falcone and Cristiano Castelfranchi. Social trust: A cognitive approach. In *Trust and deception in virtual societies*, pages 55–90. Springer, 2001.

[64] Evelyn Fix and Joseph L Hodges Jr. Discriminatory analysis-nonparametric discrimination: consistency properties. Technical report, DTIC Document, 1951.

[65] Ron Friedman. Why too much data disables your decision making. `https://www.fastcompany.com/3000676/why-too-much-data-disables-your-decision-making`, August 2012.

[66] Kelly Garrett. Protest in an information society: A review of literature on social movements and new icts. *Information, Communication & Society*, 9(02):202–224, 2006.

[67] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *VLDB*, pages 518–529, 1999.

[68] Jennifer Golbeck. Generating predictive movie recommendations from trust in social networks. In *International Conference on Trust Management*, pages 93–104. Springer, 2006.

[69] Jennifer Golbeck, James Hendler, et al. Filmtrust: Movie recommendations using trust in web-based social networks. In *Proceedings of the IEEE Consumer communications and networking conference*, volume 96, pages 282–286. Citeseer, 2006.

[70] David Goldberg, David Nichols, Brian M Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

[71] Mark S Granovetter. The strength of weak ties. *American journal of sociology*, pages 1360–1380, 1973.

[72] Jennifer Grasz. Forty-five percent of employers use social networking sites to research job candidates, CareerBuilder survey finds. *CareerBuilder Press Releases*, August 2009. http://www.careerbuilder.com/share/aboutus/pressreleasesdetail.aspx?id=pr519&sd=8%2f19%2f2009&ed=12%2f31%2f2009&siteid=cbpr&sc_cmp1=cb_pr519_.

[73] Paul Grey. How many products does amazon sell? https://export-x.com/2015/12/11/how-many-products-does-amazon-sell-2015/.

[74] Anupam Gupta, Moritz Hardt, Aaron Roth, and Jonathan Ullman. Privately releasing conjunctions and the statistical query barrier. *SIAM Journal on Computing*, 42(4):1494–1520, 2013.

[75] Patsy Haccou, Peter Jagers, and Vladimir A Vatutin. *Branching processes: Variation, growth, and extinction of populations*. Cambridge Univ. Press, 2005.

[76] Morris H Hansen, William N Hurwitz, and William G Madow. Sample survey methods and theor. 1953.

[77] Jonathan L Herlocker, Joseph A Konstan, and John Riedl. Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, pages 241–250. ACM, 2000.

[78] Will Hill, Larry Stead, Mark Rosenstein, and George Furnas. Recommending and evaluating choices in a virtual community of use. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 194–201. ACM Press, Addison-Wesley Publishing Co., 1995.

[79] Tom N Jagatic, Nathaniel A Johnson, Markus Jakobsson, and Filippo Menczer. Social phishing. *Communications of the ACM*, 50(10):94–100, 2007.

[80] Fatima EL Jamiy, Abderrahmane Daif, Mohamed Azouazi, and Abdelaziz Marzak. The potential and challenges of big data-recommendation systems next level application. *arXiv preprint arXiv:1501.03424*, 2015.

[81] Arjan JP Jeckmans, Michael Beye, Zekeriya Erkin, Pieter Hartel, Reginald L Lagendijk, and Qiang Tang. Privacy in recommender systems. In *Social media retrieval*, pages 263–281. Springer, 2013.

[82] H. Jegou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011.

[83] Márk Jelasity, Alberto Montresor, and Ozalp Babaoglu. T-man: Gossip-based fast overlay topology construction. *Computer Networks*, 53(13):2321–2339, 2009.

[84] Jerry Kang. Information privacy in cyberspace transactions. *Stanford Law Review*, pages 1193–1294, 1998.

[85] Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? *SIAM Journal of Computing*, 40(3):793–826, 2011.

[86] Joseph A Konstan and John Riedl. Recommender systems: from algorithms to user experience. *User Modeling and User-Adapted Interaction*, 22(1-2):101–123, 2012.

[87] 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*, pages 426–434. ACM, 2008.

[88] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. What is Twitter, a social network or a news media? In *19th International Conference on World Wide Web (WWW)*, pages 591–600, 2010.

[89] Soanpet Sree Lakshmi and T Adi Lakshmi. Recommendation systems: Issues and challenges. *IJCSIT) International Journal of Computer Science and Information Technologies*, 5(4):5771–5772, 2014.

[90] Joonseok Lee, Mingxuan Sun, and Guy Lebanon. A comparative study of collaborative filtering algorithms. *arXiv preprint arXiv:1205.3193*, 2012.

[91] Andrew Leonard. You are who you know. *Salon. com, June*, 15, 2004.

[92] J. Leskovec, D. Huttenlocher, and J. Kleinberg. Signed networks in social media. In *CHI*, pages 1361–1370, 2010.

[93] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[94] Jure Leskovec, Kevin J Lang, Anirban Dasgupta, and Michael W Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123, 2009.

[95] Jure Leskovec and Julian J Mcauley. Learning to discover social circles in ego networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 539–547, 2012.

[96] J. J. Levandoski, M. Sarwat, A. Eldawy, and M. F. Mokbel. Lars: A location-aware recommender system. In *ICDE*, 2012.

[97] Weiyang Lin, Sergio A Alvarez, and Carolina Ruiz. Efficient adaptive-support association rule mining for recommender systems. *Data mining and knowledge discovery*, 6(1):83–105, 2002.

[98] G. Linden, B. Smith, and J. York. Amazon.com recommendations: item-to-item collaborative filtering. *Internet Computing*, 7(1):76–80, 2003.

[99] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*, pages 73–105. Springer, 2011.

[100] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li. Multi-probe lsh: Efficient indexing for high-dimensional similarity search. In *VLDB*, pages 950–961, 2007.

[101] Walid Maalej and Anil Kumar Thurimella. Towards a research agenda for recommendation systems in requirements engineering. In *Proceedings of the 2009 Second International Workshop on Managing Requirements Knowledge*, pages 32–39. IEEE Computer Society, 2009.

[102] Sofus A. Macskassy and Matthew Michelson. Why do people retweet? Anti-homophily wins the day! In *5th International Conference on Weblogs and Social Media (ICWSM)*, 2011.

[103] Tariq Mahmood and Francesco Ricci. Towards learning user-adaptive state models in a conversational recommender system. In *LWA*, pages 373–378. Citeseer, 2007.

[104] Tariq Mahmood and Francesco Ricci. Improving recommender systems with adaptive conversational strategies. In *Proceedings of the 20th ACM conference on Hypertext and hypermedia*, pages 73–82. ACM, 2009.

[105] Y. Malkov, A. Ponomarenko, A. Logvinov, and V. Krylov. Approximate nearest neighbor algorithm based on navigable small world graphs. *Information Systems*, 45:61–68, 2014.

[106] Stephen Paul Marsh. Formalising trust as a computational concept. 1994.

[107] Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27:415–444, 2001.

[108] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *IMC*, pages 29–42, 2007.

[109] Alan Mislove, Bimal Viswanath, Krishna P Gummadi, and Peter Druschel. You are who you know: inferring user profiles in online social networks. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 251–260. ACM, 2010.

[110] M. Muja and D. G. Lowe. Scalable nearest neighbor algorithms for high dimensional data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36, 2014.

[111] Marius Muja and David G Lowe. Scalable nearest neighbor algorithms for high dimensional data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2227–2240, 2014.

[112] NPR news. In south korea, old law leads to new crackdown. http://www.npr.org/2011/12/01/142998183/in-south-korea-old-law-leads-to-new-crackdown, December 2011.

[113] N. Nodarakis, S. Sioutas, D. Tsoumakos, G. Tzimas, and E. Pitoura. Rapid aknn query processing for fast classification of multidimensional data in the cloud. *CoRR*, abs/1402.7063, 2014.

[114] Elisabeth Noelle-Neumann. The spiral of silence a theory of public opinion. *Journal of Communication*, 24(2):43–51, 1974.

[115] M. Norouzi, A. Punjani, and D.J. Fleet. Fast search in hamming space with multi-index hashing. In *CVPR*, pages 3108–3115, 2012.

[116] M. Otair. Approximate k-nearest neighbour based spatial clustering using k-d tree. *IJDMS*, 5(1), 2013.

[117] Rodrigo Paredes, Edgar Chávez, Karina Figueroa, and Gonzalo Navarro. Practical construction of k-nearest neighbor graphs in metric spaces. In *International Workshop on Experimental and Efficient Algorithms*, pages 85–97. Springer, 2006.

[118] Deuk Hee Park, Hyea Kyeong Kim, Il Young Choi, and Jae Kyeong Kim. A literature review and classification of recommender systems research. *Expert Systems with Applications*, 39(11):10059–10072, 2012.

[119] Y. Park, S. Park, W. Jung, and S. Lee. Reversed cf: A fast collaborative filtering algorithm using a k-nearest neighbor graph. *Expert Systems with Applications*, 42(8):4022 – 4028, 2015.

[120] Daniele Perito, Claude Castelluccia, Mohamed Ali Kaafar, and Pere Manils. How unique and traceable are usernames? In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 1–17. Springer, 2011.

[121] Tatiana Pontes, Gabriel Magno, Marisa Vasconcelos, Aditi Gupta, Jussara Almeida, Ponnurangam Kumaraguru, and Virgilio Almeida. Beware of what you share: Inferring home location in social networks. In *2012 IEEE 12th International Conference on Data Mining Workshops*, pages 571–578. IEEE, 2012.

[122] RVVSV Prasad and V Valli Kumari. A categorical review of recommender systems. *International Journal of Distributed and Parallel Systems*, 3(5):73, 2012.

[123] BUILDING CUSTOMER PROFILES. Using data mining methods to build customer profiles. *Computer*, 2001.

[124] Daniele Quercia, Ilias Leontiadis, Liam McNamara, Cecilia Mascolo, and Jon Crowcroft. SpotME if you can: Randomized responses for location obfuscation on mobile phones.

In *31st IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 363–372, 2011.

[125] Naren Ramakrishnan, Benjamin J Keller, Batul J Mirza, Ananth Y Grama, and George Karypis. Privacy risks in recommender systems. *IEEE Internet Computing*, 5(6):54, 2001.

[126] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *CSCW*, 1994.

[127] Paul Resnick and Hal R. Varian. Recommender systems. *Commun. ACM*, 40(3):56–58, March 1997.

[128] Francesco Ricci, Lior Rokach, and Bracha Shapira. *Introduction to recommender systems handbook*. Springer, 2011.

[129] Eunsu Ryu, Yao Rong, Jie Li, and Ashwin Machanavajjhala. Curso: protect yourself from curse of attribute inference: a social network privacy-analyzer. In *Proceedings of the ACM SIGMOD Workshop on Databases and Social Networks*, pages 13–18. ACM, 2013.

[130] Alan Said, Brijnesh J Jain, Sascha Narr, and Till Plumbaum. Users and noise: The magic barrier of recommender systems. In *International Conference on User Modeling, Adaptation, and Personalization*, pages 237–248. Springer, 2012.

[131] Matthew J Salganik, Peter Sheridan Dodds, and Duncan J Watts. Experimental study of inequality and unpredictability in an artificial cultural market. *science*, 311(5762):854–856, 2006.

[132] Oren Sar Shalom, Shlomo Berkovsky, Royi Ronen, Elad Ziklik, and Amir Amihood. Data quality matters in recommender systems. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 257–260. ACM, 2015.

[133] 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*, pages 210–217. ACM Press, Addison-Wesley Publishing Co., 1995.

[134] Lalita Sharma and Anju Gera. A survey of recommendation system: Research challenges. *International Journal of Engineering Trends and Technology (IJETT)*, 4(5):1989–1992, 2013.

[135] Kurt Thomas and David M Nicol. The koobface botnet and the rise of social malware. In *Malicious and Unwanted Software (MALWARE), 2010 5th International Conference on*, pages 63–70. IEEE, 2010.

[136] TIME magazine. Indian women arrested over facebook post. `http://newsfeed.time.com/2012/11/19/indian-woman-arrested-over-facebook-like/`, November 2012.

[137] Amit Tiroshi, Shlomo Berkovsky, Mohamed Ali Kaafar, Terence Chen, and Tsvi Kuflik. Cross social networks interests predictions based ongraph features. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 319–322. ACM, 2013.

[138] Daily Mail U.K. Half of facebook users accept friend requests from strangers, while 13m u.s. users have never opened their privacy settings. http://www.dailymail.co.uk/sciencetech/article-2139424/Half-Facebook-users-accept-friend-requests-strangers-13m-U-S-users-NEVER-opened-privacy-settings.html, May 2012.

[139] Pravin M Vaidya. Ano (n logn) algorithm for the all-nearest-neighbors problem. *Discrete & Computational Geometry*, 4(2):101–115, 1989.

[140] C. J. van Rijsbergen. *Information retrieval*. Butterworth, 1979.

[141] Katrien Verbert, Nikos Manouselis, Xavier Ochoa, Martin Wolpers, Hendrik Drachsler, Ivana Bosnic, and Erik Duval. Context-aware recommender systems for learning: a survey and future challenges. *IEEE Transactions on Learning Technologies*, 5(4):318–335, 2012.

[142] Bimal Viswanath, Ansley Post, Krishna P Gummadi, and Alan Mislove. An analysis of social network-based sybil defenses. *ACM SIGCOMM Computer Communication Review*, 40(4):363–374, 2010.

[143] S. Voulgaris and M. v. Steen. Epidemic-style management of semantic overlays for content-based searching. In *Euro-Par*, pages 1143–1152, 2005.

[144] S. Voulgaris and M. van Steen. Vicinity: A pinch of randomness brings out the structure. In *Middleware*, pages 21–40, 2013.

[145] Christo Wilson, Bryce Boe, Alessandra Sala, Krishna P.N. Puttaswamy, and Ben Y. Zhao. User interactions in social networks and their implications. In *EuroSys*, pages 205–218, 2009.

[146] Xindong Wu, Vipin Kumar, J Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J McLachlan, Angus Ng, Bing Liu, S Yu Philip, et al. Top 10 algorithms in data mining. *Knowledge and information systems*, 14(1):1–37, 2008.

[147] Volker Wulf, Kaoru Misaki, Meryem Atam, David Randall, and Markus Rohde. 'On the ground' in Sidi Bouzid: Investigating social media use during the tunisian revolution. In *16th ACM Conference on Computer Supported Cooperative Work (CSCW)*, pages 1409–1418, 2013.

[148] Wei Xu, Fangfang Zhang, and Sencun Zhu. Toward worm detection in online social networks. In *Proceedings of the 26th Annual Computer Security Applications Conference*, pages 11–20. ACM, 2010.

[149] Guanhua Yan, Guanling Chen, Stephan Eidenbenz, and Nan Li. Malware propagation in online social networks: nature, dynamics, and defense implications. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, pages 196–206. ACM, 2011.

[150] S. Yang, M.A. Cheema, X. Lin, and Y. Zhang. Slice: Reviving regions-based pruning for reverse k nearest neighbors queries. In *ICDE*, pages 760–771, 2014.

[151] Cui Yu, Beng Chin Ooi, Kian-Lee Tan, and H. V. Jagadish. Indexing the distance: An efficient method to knn processing. In *VLDB*, pages 421–430, 2001.

[152] Chun Zeng, Chun-Xiao Xing, and Li-Zhu Zhou. Similarity measure and instance selection for collaborative filtering. In *Proceedings of the 12th international conference on World Wide Web*, pages 652–658. ACM, 2003.

[153] Yan-Ming Zhang, Kaizhu Huang, and Cheng-Lin Liu. Fast and robust graph-based transductive learning via minimum tree cut. In *2011 IEEE 11th International Conference on Data Mining*, pages 952–961. IEEE, 2011.