# RECOMMENDER SYSTEM ON SOCIAL NETWORKING SITE WITH DOMAIN SPECIFIC AND SPARSE DATA

by

Chuy Chang Nian

B.Sc. University of Toronto, Toronto (ON), Canada, 2012

A thesis
presented to Ryerson University
in partial fulfillment of the
requirements for the degree of
Master of Science
in the program of
Computer Science

Toronto, Ontario, Canada, 2017

## Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my thesis may be made electronically available to the public for the purpose of scholarly research only.

# Abstract

## Recommender System on Social Networking Site with Domain Specific and Sparse Data

Chuy Chang Nian
Master of Science, Computer Science
Ryerson University, 2017

In many recent domain-specific social networking sites, posts are organized in chronological order, where later posts are shown first at the top, even though they might not be of everyone's interest. As a result, if users want to read posts that interest them, they will have to scroll down and sift through all the posts. To overcome this information overload problem and relieve users' burden, a recommender system is needed in social networking sites. In this thesis we propose a hybrid approach of Recommender System (RS) that combines both Collaborative Filtering and Content-based approach. Although each approach has their own weaknesses independently, by joining them together we can improve the accuracy of our recommendations. From our expriements, we noticed that using learning to rank algorithms in combining each recommender algorithm greatly enhances the system's performance.

# Acknowledgements

Foremost, I would like to express my immeasurable appreciation and deepest gratitude to my supervisor Dr. Cherie Ding, for her continuous guidance and support throughout my graduate studies. Throughout my research, she provided me with proper direction, feedback and helped me overcome all challenges and difficulties in my work. Her invaluable suggestions, knowledge and research skills helped me throughout my research work, and for the completion of this thesis.

I would like to thank the company for giving me this opportunity to work on this project. Also, I would like to thank the team members from the company for providing useful feedback and their time on explaining the data set.

I would like to thank Natural Sciences and Engineering Research Council of Canada (NSERC), Yeates School of Graduate Studies, Department of Computer Science at Ryerson, and Dr. Cherie Ding for providing additional funding that helped me through my research. I would also like to to thank all members of the Department of Computer Science at Ryerson University for their cooperation, help and for providing me with access to additional resources required for my research.

I would also like to thank Dr. Konstantinos Derpanis, Dr. Alireza Sadeghian and Dr. Andriy Miranskyy for taking the time to review my thesis and providing valuable feedback which enabled the improvement of the thesis.

Finally, I wish to express my deepest appreciation to my parents for their continuous encouragement and support throughout my life. I wish to also thank my brother, sisters and friends for their continuous encouragement throughout my studies.

# Table of Contents

# List of Appendices

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 Background

Any form of social interactions between two human being can be classified as social networking and with the development of World Wide Web, more and more users are socially interacting with each other over the Internet. According to [50], they defined social networking sites as “a web site that facilitates meeting people, finding like minds, communicating and sharing content, and building community”. In recent years, online social networking has been increasingly popular because modern technology had made sharing contents, collaborating with others, and connecting with each other easier, faster, and more accessible to a wider population than ever before. Initially, Social Networking Sites (SNS) were focused on interactions between friends but recently many celebrities and politicians have dived into social networking sites to share their views and reach out to public. As a result, social networking sites information has expanded exponentially, creating a massive load of online data.

With the explosion of Web 2.0 applications such as blogs, discussion forums, social and professional networks, the users’ online activities have changed. Web users are no longer the mere consumers of information, but the “producers of information”. Modern SNS such as Twitter and Facebook makes sharing information, collaborating with others, and connecting with each other as a community, faster and more accessible [14, 50]. The abundance and popularity of social networking sites has flood users with huge volumes of information. The amount of digital data that users produce daily had long surpassed our ability to process them, and finding useful information in this constant flow of data has become a major hassle for every users in the 21st century. In general, we can noticed two issues in preventing SNS from being sufficiently relevant and causing deterioration of user satisfaction in engagement [14]. When users are faced with large amount of User Generated Content (UGC) from their social peers that they could not process in an effective and efficient way, this leads to the problem of information overload for users. On the other hand, the information of a user is usually limited in scope to user's social connections, leading to the problem of information shortage. Therefore, to help users overcome these challenges, SNS have implemented Recommender System that analyze data about customers and products to predict and recommend items of interest based on user past preferences, behaviors and social connections.

In short, the recommendation problem can be reduced to the problem of estimating ratings of posts that have not been seen by users [2]. Recommender System is a family of methods that enable filtering through large information data space in order to provide recommendations in the information space that user has not observed or introduced. The information space contains all available posts that user could choose, select or forward. In another word, Recommender System works by suggesting items that are most appealing to users based on their past preferences [6].

User profile is the backbone of any social networking sites, it can include anything about the user's birthday, religion, ethnicity, and personal interests. A user's personal profile is a unique page where people express their inner thoughts and feelings, and post pictures of their daily adventures. With the massive quantities of UGC on SNS, recommender system could use these data to expand user profiles [50]. The Recommender System combines ideas from user profiling, information filtering and machine learning to deliver user proactive information [50]. However, in many existing social networking sites, new users and posts are continuously being added into the system, and existing users' preference and posts' popularity vary over time [22]. Therefore to cope with such changes, a dynamic recommender system is preferred to handle real-time updates and real-time response on recommended items.

With the prevalence of 3G/4G technology, people are now able to access their social accounts anywhere and anytime via mobile divices, and mobile internet service has become an indispensable part of people's daily life. Since mobile devices are usually Global Positioning System (GPS) enabled, a large portion of UGC via smart-phones, are associated with time-stamps and geographical annotations. The locations may be in the form of latitude and longitude coordinates, or venues with semantic meanings (e.g., Nathan Philips Square, and Harbourfront). These rich spatial-temporal-semantic information provide an exciting opportunity to develop many different context-aware applications, such as location-based and event-based applications [47].

## 1.2 Objectives

Recommender systems are widely used in e-commerce to aid customers in decision making process, convert browsers into buyers, and build customer loyalty. The success of social networking site highly depends on the level of ease in searching and discovering useful information. While interesting posts recommended to user will improve user's satisfaction, irrelevant posts will only hinder user's experience with the site.

In the age of Big Data, a large amount of information from social networking sites, digital libraries, news archives, and so on are available to us. Every one of us are constantly inundated with choices and information. We have more options and choices than we used to have and this will continue to increase in the future as well. The extremely large size of the Web has made it infeasible for any single user to browse through information without any filtering. If you just consider the available information space on a single social networking site such as Twitter or Facebook let alone the entire World Wide Web (WWW), you could find yourself in an immense decision domain, which may cripple your decision making.

Traditionally, users in communities seek peer recommendations such as word of mouth or expert advice such as reviews from domain experts as recommendations for new items. These methods are good and fairly accurate but are limited to the knowledge and preference of the recommenders. In this era of overload information, it is not possible for a single person or a group of people to know and understand every single detail of a product in this era. Due to these shortcomings, computer-based recommender system provides a much better alternative to users. Generally, recommender system helps people in retrieving information that matches their interests or needs by recommending products and services from piles of candidates [3]. Computer based recommender system could mine historical information and demographic information from users thanks to the advancement of technologies. This method will result in a more accurate and thorough recommendation than the traditional methods. Therefore, an efficient and effective Recommender System is essential for social networking sites to improve user's satisfaction and many e-commerce leaders have already made recommender system a salient part of their existing websites [21].

In industry, Recommender System has been widely used by many companies such as Amazon and Netflix, who analyze patterns of user interests and provide personalized recommendations that suit user's taste. For example, purchase history, browsing history and demographic information are mined and analyzed to provide recommendations of a wide variety selection of goods and products to users. In addition, there have been many recommender systems used in social networking sites in the past and they are continuously being improved to provide more accurate recommendations. Recent improvements on recommendation system are to utilize user's location and social relations. Researchers have been working on methods in combining these information into the traditional recommender system.

Recent Recommender Systems for social networking sites can be grouped into the following three types. First and the most common strategy is to incorporate social relations between users into the traditional Collaborative Filtering model. Secondly, a lot of Social Networking Sites have started to incorporate temporal interests in their recommender systems. Temporal interests includes breaking news, groundbreaking products and seasonal holidays. All these factors affect user's interests level during different period and it has to be reflected in the recommender model as well. Lastly, due to the advancement of hand-held devices, more and more recommender systems are including user's location as part of their input in the recommender system.

Unfortunately, there is no one-fit-for-all recommender systems that could satisfy every social networking site. Since there is no best Recommender System for all domains, we can only strive to improve recommender system for a specific domain with concentrated interests. By including different factors or inputs from user's information, it is possible to provide an adequate recommendation even with sparse data.

Since we are working on a social networking site that is focused on traveling and vehicles, our thesis has three main objectives. Firstly, we want to study user's interests based on their previous actions, friends, locations and type of vehicle owned. Instead of relying on user's explicit acquisition of rating posts; we focus on getting user's implicit acquisition where the system observes users' behaviors and infers users' interests from these interactions. Secondly, we want to show that integrating domain specific information, user location and social relations into the traditional hybrid model of Collaborative Filtering and Content-Based System, could improve the recommendation accuracy for this social networking site. Lastly, we want to illustrate that using learning to rank algorithm to combine multiple signals (e.g., content, ratings, social relations, temporal, location, etc.) yields better results than any single separate signal.

## 1.3 Proposed Methodology

There are many different social networking sites available nowadays. In order to better capture users' interests, Social Networking Sites are narrowing down on their contents into specific fields, such as tourist attractions, workers connection, and vehicles to stay ahead of their competitors. Even with the simmered down content, it is still impossible for user to sift through every piece of posts available on the social sites. Therefore, Recommender System is required to filter through all data and recommend those that are most interesting to users. In our research we focus on Recommender System that specializes in closed-domain with concentrated interests and sparse data.

In this thesis, we are not proposing to re-invent new methodologies to improve Recommender System. Instead, we are using proven known techniques and combining them to provide more precise recommendations. First, we group similar users using Collaborative Filtering technique by aggregating similar users' rating and predict items for users. Secondly, we rate posts using Content-Based technique of comparing posts with user based on keywords and locations. Thirdly, social relations between users are also taken into consideration in this model. Social strength or relations are calculated based on the frequency of interactions between users; intense interactions implies higher social strength. Moreover, since our social networking site is mobile friendly and geographically available, our model takes advantage of this information to provide a better prediction. Our model considers the distance between post's location and user's location to evaluate user's interest level on that particular post. Last but not least, we combine all these different features using learning to rank algorithm to personalize recommender system for each individual user.

## 1.4 Thesis Outline

The remaining of this thesis is organized as follows:

Chapter 2 reviews previous works on similar field. It includes some brief explanation on Recommender System and how Recommender Systems are used on social networking sites followed with a brief summary of existing models and similar works. Then we provide an explanations of learning to rank algorithms and its three approaches.

Chapter 3 explains algorithms and methods that are used in our model followed with an explanation of our hybrid model structure.

Chapter 4 provides a summary of experiment environment and discuss how it affects our experiment design. A description of evaluation metrics used is followed. It ends with a discussion of experiment results.

Lastly, Chapter 5 concludes the thesis and lists out future research directions.

# Chapter 2

# Related Works

In this chapter, we discuss the core concepts and logic of recommendation algorithms for social networking sites that are related to our thesis. In the first section, we provide a brief explanation on some existing approaches that were used in traditional recommendation system. Also, we included a short description of learning to rank algorithms that were used in our system. In the next section, we review some recent work done by other researchers under the same area of using recommender system for social networking sites.

## 2.1 Background

In this section, we discuss the basic concepts of recommender system and existing approaches that were used in our thesis. Recommender system combines ideas from user profiling, information filtering and machine learning to provide a more intelligent and proactive service by making service recommendations that match user preferences and needs. In general, recommender system can be categorized into three types: content-based model, collaborative model, and hybrid model.

### 2.1.1 Content-Based Method

Content-Based Filtering (CB) method recommends item that has high degree of similarity to an item that user has rated highly in the past. Content-based systems [2] are usually used to recommend text-based items and the content in these systems are usually represented with keywords. One advantage of this method is that it does not require creation of explicit user profiles, and any specific domain knowledge [9]. Some limitations of content-based recommender system are as follows: [2]

1 *Limited Content Analysis*: limited to features that could be mined from items

2 *Overspecialization*: recommending items of high similarity, limits the exploration of user's other interests

3 *New User Problem*: sufficient number of ratings is needed before the system can provide accurate recommendations

### 2.1.2 Collaborative Method

Collaborative Filtering (CF) uses the known preferences of a group of users to make recommendations or predictions of unknown preferences to other users [41]. Collaborative recommendations can be grouped into two general classes: [2, 6, 15, 24, 48]

1 *Memory-based*: uses the entire or a sample of user-item rating data to generate prediction

2 *Model-based*: uses the collection of ratings as training data to learn a prediction model

According to [24], model-based technique handles the data sparsity problem better than memory-based technique and scales well with large data sets.

Some advantages of CF recommender system are as follows: [6, 48, 49]

1 Domain free and not confined by content features - content features are hard to mine and profile, which could be avoided with this approach

2 User's interest may change over time - focusing on user's past interest may not perfectly capture user's current interests. Users' current similar interests could provide better up-to-date recommendations because similar users may shift their interests in similar direction

3 Capture latent relations between items - even though two items features might seem totally unrelated, if most similar interests users are interested in these two items then others who share the same taste are likely to be interested as well

Some limitations to CF recommender systems are as follows: [2, 8, 41]

1 *New User Problem*: system must first learn user's preferences from ratings before it can provide accurate recommendation

2 *New Item Problem*: a new item must be rated by a substantial number of users before it will be recommended to others

3 *Sparsity*: the number of ratings provided by users are usually very sparse

4 *Scalability*: does not scale well with large number of users and items

5 *Privacy*: not all users want their past history to be mined

6 *Context Information*: does not incorporate context information

Collaborative filtering has some advantages over content-based methods [9], when it is difficult to analyze the content of items. Semantic relatedness of items that cannot be detected by content-based methods can be easily inferred by using collaborative methods.

### 2.1.3 Hybrid Method

Hybrid methods combine both collaborative and content-based methods to overcome data sparsity issue. Hybrid recommendation system can be classified as follows: [2]

1 Implement both collaborative and content-based methods independently and combine their predictions

2 Incorporate some content-based characteristics into collaborative approach. This approach overcome some sparsity-related issues

3 Incorporate some collaborative characteristics to content-based approach

4 Construct a unifying model that incorporates both content-based and collaborative characteristics

Due to some limitations in Content-based and Collaborative method, the hybrid method is a more viable approach in designing our recommender system. Today's most sophisticated recommender system combines multiple relevant features to provide the best recommendation results, and a challenge had arise from it. How to tune or train these features (labeled data) becomes a major research topic recently. [40]

### 2.1.4 Learning to Rank

According to [21], the term "learning to rank" meant using machine learning algorithm to train ranking models. In recent years, learning to rank has become one of the most active research area in information retrieval field [21]. We can simplify the properties of learning-to-rank methods into the following [21] :

1. **Feature based** means that all documents or items under investigation are represented in feature vectors. Feature vector represents the relevance of the document or item to the search query.
2. **Discriminative training** means the automatic learning process based on training data. The learning process can be further described into four components namely: input space, output space, hypothesis space, and loss function.
   (a) *Input space*, which contains the objects under investigation and are usually represented in feature vectors.
   (b) *Output space*, which contains the learning target with respect to the input objects.
   (c) *Hypothesis space*, which contains class functions of mapping input space to output space.
   (d) *Loss function*, measures the degree of prediction generated by the hypothesis space in accordance to the training set.

Ideally, any learning algorithm would train a ranking model such that it could directly optimize the performance measures with respect to the training set [40]. The focus of learning to rank research is using machine learning algorithms such as classification and regression methods to assign scores to documents and then rank those documentss using the scores [11,40]. Two major components of constructing a learning to rank models are:

1 *Learning* - a ranking model with optimum model parameters is constructed using training data consisting of queries, corresponding documents and relevance levels

2 *Ranking* - a new given query and its' corresponding documents are sorted in descending order using the assigned scores from the trained ranking model

Learning to rank has become critical in serving users' information needs: ranking product based on previous purchases, ranking advertisements based on page content, and ranking articles based on users' interest [11].

In order to better understand different types of learning to rank algorithms, we can group them into three approaches: the pointwise approach, the pairwise approach, and the listwise approach [21].

### The Pointwise Approach

The pointwise approach treats each object under investigation as a single document and predicts the relevance degree of each document. It has strong correlation with the relevance feedback algorithms. The basic idea of feedback algorithm is to use explicit, implicit, or blind feedback to update original query. Then the new query is used to retrieve a new set of documents in an iterative manner. Thus, the original query will be updated into a closer version of the optimal query.

### The Pairwise Approach

The pairwise approach does not focus on predicting the relevance degree of each document; instead, it only considers the relative order between two documents. We can view this approach as classifications between document pairs, such as which document in a pair is preferred. According to [7], the pairwise approach is sensitive to noise due to the shapes of the loss functions. They proposed to replace hinge loss function with sigmoid loss which minimizes the impact of outliers. Although pairwise approach cannot determine the final ranked list, but one can make reasonable estimation on the ranking order by checking all pairs containing the documents. Since only the top pairs are important to users, one can compare all top pairs and estimate the ranking list.

### The Listwise Approach

The listwise approach takes the entire set of objects associated with a query and rank them using a ranking model returning a sorted list of objects. Intuitively speaking, the listwise approach is a more natural way than the pointwise and pairwise approaches [21]. The listwise approach takes the entire set of documents associated with a query as input and their ranked list as output. In turn, it has the potential of distinguishing documents from different queries and considering the positional information in the output ranked list. According to [38], the performances of listwise ranking algorithms are in general better than the pointwise or pairwise ranking algorithms.

### Analysis of the Approaches

As mentioned above, most of the three approaches' loss functions can upper bound measure-based ranking errors. Therefore, the minimization of these loss functions can directly lead to the optimization of the evaluation measures in certain situations. As explained above, the main differences between the approaches lies in their loss functions. However, the evaluation of the learned ranking models are based on the evaluation measures. Hence, an important issue to discuss is the relationship between the loss functions and the evaluation measures.

## 2.2 Related Work

In the following are some recent papers on Recommender System in Social Network that focused on multi-dimensional information recommendation and personalized recommendation. Multi-dimensional approach focuses on adding contextual information to handle data sparsity while personalized approach focuses on updating user's private model to handle user's uniqueness.

### 2.2.1 Social Recommender System

In this section, we review some traditional approaches for social recommender systems.

#### User Profiling

Adding on user rating data, user profiling contains rich semantic information and provides huge potential to obtain deeper knowledge about user-item relations. From user profile, it is possible to acquire users' opinions, perspectives, and interests towards items or other users. Many individual users are sharing their brand experiences and opinions, positive or negative, regarding to products and services online. These users voice can potentially influence the opinions of other consumers, and many companies are mining such feedback.

#### Blogs Mining

The term web-log, or blog refer to a simple web-page consisting of brief paragraphs of opinion, information, or links, called posts. Most blogs allow user to add comment below blog entry. People express their opinions, ideas, experiences, thoughts, wishes through these free-form writings. A blog can contain any format from text, links to image and video.

#### Tagging System

A tag is a keyword attached to a digital object (e.g. a picture or video clip) to describe it. Tags are freely chosen keywords by users who deem them to best describe the digital object. In addition, each tag serves as link to additional resources tagged the same way by others. The tag information is becoming an important information source of describing user's topic interests and classifying items.

One key advantage of tagging is that it is independent with content items, which made mining items such as photographs, videos and music possible. Also, tags are light weighted textural information which contain rich explicit topic information given by user explicitly and proactively. [13] conducted a study of social tagging and shown that navigable hierarchical taxonomy of tags could enhance user searches. The taxonomy of tags is used to help users broadening/narrowing the set of tags that best describe their interests.

Although, free-style tagging provides a lot of freedom for users to organize, search and explore resources. It also contain a log of noise such as semantic ambiguity which means same tag name has different meaning for different users, and tag synonym which means different tags that actually have the same meaning. A search for solving above challenges are still an ongoing topic in recommeder system for social networking site.

#### Trust Network

Trust is subjective, personal asymmetrical, and dynamic. People's trust to others is gradually built up and keeps changing over time. According to [12], trust has the property of transitivity, and it may propagate through the relationship network.

[17] has found that people's interest similarity is a strong predictor of interpersonal trust. This relationships between people's interest similarities and trust have been further confirmed by [51]. Their empirical analysis showed that people who have similar interests tend to be more trustful towards each other. In light of these studies, the trust model can act as supplement to current social networking recommender system. [26] claimed that this method can help alleviate the data sparseness problem and enhance the system scalability. In year 2016, [31] proposed semantic based trust recommender system which recommend trust companion based on high similarities in message sharing. It maintains virtual group of trust people having similar interest.

### 2.2.2 Contextual Information

Context is any information that can be used to characterize the situation of an entity. Contextual modeling [6] incorporates context information directly into the model used to estimate rating predictions. [6] found that recommender systems that involve contextual information such as location, time, activities, and user preferences provide more user trust in their item recommendation, which in thus, affect purchasing behavior.

User behavior on social media are influenced by both intrinsic interests and temporal context. Also, user's interest changes from time to time and new interests may arise. Moreover, user rating data is very sparse, so user's short-term interest is easily overestimated or underestimated. To overcome this challenge [42], social and temporal information can be used to enhance prior knowledge about user interest distribution.

### Temporal Information

Among contextual dimensions, time information is considered as one of the most useful information. The usage of time information as contextual information can be categorized into: [6]

1 *continuous time-aware approaches*: time is represented as continuous contextual variables

2 *categorical time-aware approaches*: time is represented as categorical contextual variables

3 *time adaptive approaches*: adjusting parameters or data dynamically according to changes of some data characteristics through time

The freshness of posts is considered the most important aspect, thus most Social Networking Sites (SNS)'s posts are shown in chronological order to users. Indication of whether a user is interested in a post is determined by many factors, such as quality of the post and the authority of the publisher [9]. Personal interest is also an important factor to decide whether a post is personally useful to user. Moreover, temporal affect such as breaking news and events has an impact on user's browsing behavior. According to [39, 47], exploitation of strong temporal patterns in data significantly improves the performance of recommender systems.

### Spatial Information

Location-aware recommender system [43] recommends users events with the consideration of both user interest and location preference. It enables users to share with their friends the places they want to go and who they wish to go with [4]. Given a user querying at a location, the system [43] will effectively compute ranking scores for all spatial items within certain distance and return top-k attractions.

Photos uploaded by users via mobile devices are usually geo-tagged, providing a wealth of geo-spatial data. These photos can be used for Point-of-Interest (POI) clusters, identifying location of photos from visual, textual, and temporal features, can help determine tourist attraction in that season and creating routes that are pleasing to users [5]. It allows them to discover interesting places in populous cities that are not easy to explore. From observation, users are more likely to visit locations close to their visited locations, and thus the locations visited form several spatial clusters [47].

However, the popularity of each POI varies greatly over time. Different POIs become popular at different time such as beach is popular during daylight but not so much after nightfall. The probability of check-in at a POI should reflect both its popularity at specific time and the distance to current user's location [45].

### Social Information

A quote from [50], "Trust is the outcome of observations leading to the belief that the actions of another may be relied upon". People's trust to others is gradually built up and keeps changing over time. Since, trust is a personal opinion, the personalization of trust means that a member could have different trust values with respect to different members [15, 43, 50].

Likewise, the essence of social recommendation approaches lies in the additional explicit and implicit social information from each user. Explicit relations suffer from sparseness, but are usually accurate. While implicit relations can overcome the sparseness problem, but in doing so may introduce 'noise' [10]. Therefore, a combination of these social relations can then be utilized to model a user's interest more accurately [25].

## 2.2.3 Multi-Dimensional Recommendation

In this sub-section, we will review some multi-dimensional recommender systems proposed for social networking sites by researchers in the past. To obtain accurate recommendations, it is important to capture users' contextual information (e.g., time and location) to understand users' intentions.

In [34], Sarwat et al. proposed an efficient and scalable location-aware recommender system. Their system produces location-based recommendations using taxonomy of three types of location-based ratings: spatial ratings for non-spatial items *(user, user_location, rating, time)*, non-spatial ratings for spatial items *(user, rating, item, item_location)*, and spatial ratings for spatial items *(user, user_location, rating, item, item_location).* The idea behind this scheme is preference locality and travel locality. Preference locality suggests users from different spatial region prefer items differently, whereas travel locality is the willingness of users' travel distance. This is a great observation and we have incorporated this idea in our system along with contextual and social relation information.

In [43], Sarwat et al. included content-awareness into their model. Their system consist of two components: offline modeling and online recommendation. The offline modeling portion is designed to learn user's interests and preferences. The online recommendation part automatically combines the learned interest and preference of individual user to provide recommendations. The authors focused on Event-based Social Networking Services (EBSN) such as Meetup (www.meetup.com) that provide platforms for users to establish social events held in physical places. Although their proposed location-content-aware recommender system exploits both location and content information, they still neglected temporal and social information from their system.

In [46], Yuan et al. proposed of using multi-dimensional contextual information to discover users' spatial-temporal topics. In their scheme, they use data from Location-based Social Network (LBSN) such as Foursquare and Facebook Places that allow users to share their current locations and activities. These information allow them to study the behaviors of individuals with respect to geographic location, time and activity. They focused on users' mobility based on location and time, if it is a weekday and user is at his or her work region, then his/her mobility is limited, as opposed to a weekend around home region.

In [47], Yuan et al. built on [46] and proposed using multiple contextual information with nonparametric Bayesian model to recommend context-aware recommendations. They use contextual information and user mobility behavior together to determine target users' interests and intentions. The authors claim that user's mobility centers at several personal geographical regions (e.g., home and work region), user's interest is influenced by the region, and user's vacation location is affected by region's activities and distance. Therefore, with the information of user's location and time, along with activities information on the region, they were able to improve recommendations' accuracy. However, they are neglecting one key feature social relations in their model.

In [5], Bhargava et al. proposed multi-dimensional collaborative recommendations for user, activity, time and location, using tensor factorization on sparse user-generated data. In their scheme, the first step is to infer various dimensions of information, such as location, activity and time. Geo-tags are hashed into discrete rectangular grid bins, activities are categorized into an activity hierarchy which consists of lifestyle, recreational and tourist activities, time-stamp are hashed into three type of distributions: monthly, weekly, and hourly. In order to address tensor sparsity, they formulated an objective function that simultaneously factorizes coupled tensors and matrices from heterogeneous data sources. The tensor setup is $user \times location \times activity \times time$, along with four matrices $location \times activity$, $location \times venue$, $location \times location$ and $activity \times activity$. Their main focus is using tensor factorization in improving model's accuracy, but we believe that personalized model would be a better approach in maximizing recommendation accuracy.

### 2.2.4 Personalized Recommendation

In this sub-section, we will review some personalized recommender systems proposed for social networking sites by researchers in the past.

In [16], Imran et al. proposed a personalized learning recommender system that supports learners by providing personalized recommendations based on learner's profile and other learners with similar profiles. Since most users have different interests and characteristics, a "one size fits all" approach does not support most users. Personalized model enables the system to uniquely address user's needs and interests. Although their model was used for e-learning sites, we have incorporated this idea of user profiling with additional contextual information to provide personalized recommendations on social networking site.

In [27], Majid et al. proposed a context-aware personalized travel recommendation based on geotagged information. Similarly to their multimedia data which are tagged with temporal context and spatial context, our data consists of additional contextual information. The authors focused their contextual information onto weather conditions and venue working hours which are less applicable to our social networking site. Additionally, their proposed personalized system only exploits users' traveling preferences without taking considerations of users' interests and behaviors.

In [30], Qian et al. proposed personalized recommendation combining user interest and social circle. In this paper, they used three social factors: personal interest, interpersonal interest similarity, and interpersonal influence, to fuse into a unified personalized recommendation model. Their focus was on social factors between user and his/her friends latent feature vectors, and used these information to enforce user's personalized model. In addition to social influence, our personalized model also incorporates spatial and temporal influences.

In [44], Yu et al. proposed to combine heterogeneous relationship information for each user to provide high-quality personalized recommendation results. Similarly, we agree with the researchers that the entity recommendation problem exists in a heterogeneous information network. They claimed that a personalized recommender system which combines user feedback (explicit ratings and implicit user's history) and additional information of users and items such as user demographic attributes, and social network information can achieve better recommendation results. Our model falls into the same category of such hybrid recommender systems. The difference between our work and theirs is that our model also includes temporal and spatial informations.

## 2.3 Summary

In this chapter, we have reviewed most of the concepts used in this thesis and also discussed related work done in the same area of recommender system for social networking sites. As we can see that most proposed recommender systems [5,6,10,15,43,45,47,50] take in many different input signals as complementing information. Most proposed models target on a one-fit-for-all recommender system that is used for all users across the social site. A strategy model may work well for some users but may not work for all users, in order to accommodate such challenges, we propose personalizing recommender system for each active user, a general good basic recommender system [2] consisting of both collaborative filtering and content-based filtering approach. In recent studies [6,43,50], they proposed incorporating various information such as temporal, spatial and social information as part of the recommender system inputs. From temporal information, the recommender system would be able to recommend posts that are recent and popular providing diversity in our recommendation outputs. The spatial information would increase relevancy of recommendation outputs whereas social information provide recent posts from close friends or colleagues. With all these overloading information, the recommender system needs to be tuned with ranking algorithms for proper filtering and accurate recommendations.

## Chapter 3

# Methodology

We propose a recommender system for a social networking site that is dedicated to vehicles enthusiasts. We extract related users' information from user's feedback and interactions throughout the social site. Our objective is to implement multiple scoring algorithms for recommending new posts and use learning to rank algorithm to combine those scores. In the following sections, we will discuss the system architecture, our proposed hybrid recommender model and the difference between generalized and personalized recommender system.

## 3.1 System Architecture

In this section, we will explain the overall architecture of our hybrid recommender system. The objective of our recommender system is to provide a list of recommended posts to users. We will be using multiple scoring algorithms in capturing users' interest, along with machine learning algorithm to combine all these scoring algorithms effectively. Here, we will discuss each component of our system in details along with reasoning on why such methods are chosen for our model.

The architectural model of our proposed system is shown in Figure 3.1, it consists of three parts: the profiling, score generator, and the storage.

Our model consists of two components, offline profiling and online recommendation. From 3.1, the profiling section is done during offline stage, and score generator section is used in real-time recommendation. User content profiling, user interaction profiling, social strength profiling and vehicle profiling can be run simultaneously. These four parts had to be run ahead of the score generator component to learn and capture user's interests and preferences. The score generator component will then use the learned result

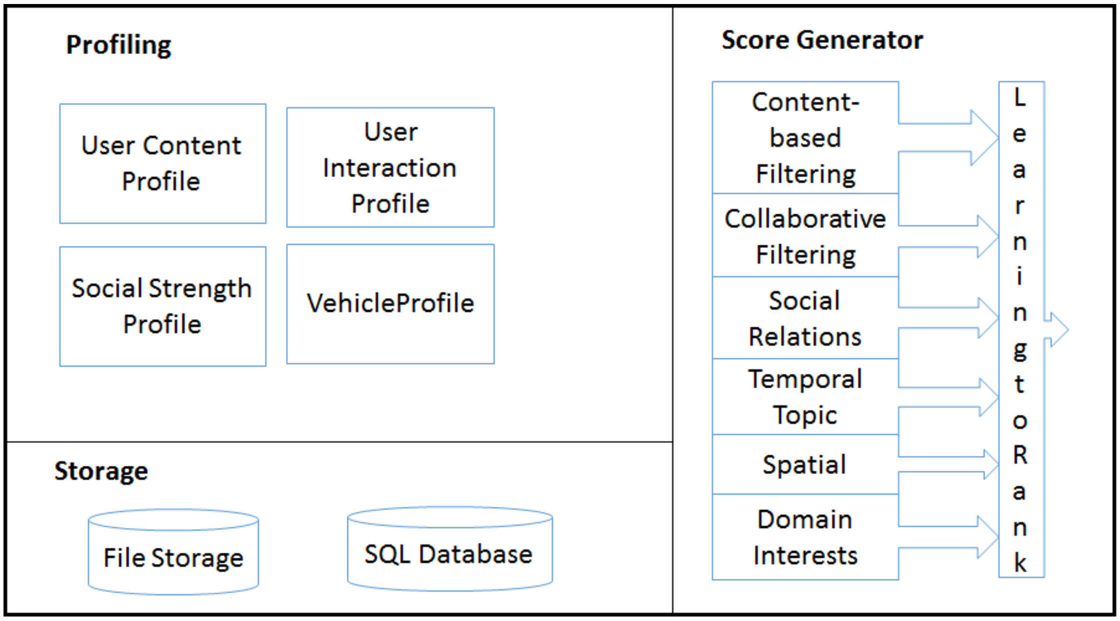

Figure 3.1: Architecture of the recommender system

from these profiles to predict top-N recommended posts to users.

User content profile is generated using vector space that represents user's interests where each dimension represents a topic or keyword and the value represents interest level. The profile is generated by aggregating all user's posts into a corpus in which a document includes all posts from a user. Then, we stripped all punctuation marks, spaces and new lines leaving only words. Additionally, we convert all words into lower case then stemming and stop-word removal are used to extract keywords from the corpus. Later, a document corresponding to one user's all posts is converted into a vector space representation and saved into the local file storage. So these profiles can be used later to compare with posts that might be of interest to user by using cosine similarity function. The cosine similarity scores are then saved in the SQL storage for later usage in score generator component. User interaction profile is generated using matrix factorization (Alternating Least Square which will be discussed in later section) where dimensions represent user and post, and value represents the rating of user to that post. In order to handle sparse input data from users' explicit feedback, we combine both implicit feedback and explicit feedback from users to create a denser input data set. Users' friend list, votes and comments on post are used as our implicit rating data. All these information are then compiled into a list and are used to generate the interaction profile. This profile model is then stored into local file storage system for later usage in score generator. Social strength profile is a module that calculates the social strength between each pair of users. It sums up the interactions such as private messages, up-votes and comments between users. These scores are then stored into the SQL storage system for later usage in score generator. Vehicle profile module has two parts, the first part is to extract vehicle models and name from a domain-specific vehicle-related web site and store those information into the SQL storage. Then, the second part is to filter posts that had mentioned any vehicle models and names that exist in the SQL table. Based on whether a vehicle name or model is mentioned in the post, we store a closeness score for post and vehicle in the SQL storage. We used closeness score because a range of users uses various naming convention representing the same model.

The score generator is the main component of our recommender system. It is here that the personalized and generalized model are trained using profile data from above. The best model structures are then saved into local file storage system for later usage. Each feature in the score generator is responsible to generate a list of ordered candidate posts independently with scores in descending order. The score represents the importance level of the recommended post based on different feature's recommendation strategies. A detailed explanation on each feature based strategy will be discussed in the next section. Although all features are treated equally at the beginning, we will use learning to rank algorithms to reassign importance of each feature based on user's preferences. The best learning to rank algorithm is then selected and stored in local file storage as the ranking model.

The final module of our system is the recommender model. It uses the score generator model to generate a list of candidate posts as input, and uses the best ranking model to rank those posts. The output would be a ranked list of candidate posts in descending order. The personalized model is generated from user's own historical activity data with multiple features and a learning to rank algorithm that personalizes the weight of each feature such that it is tailored to that single user. As for the generalized recommender model, we aggregate all users' historical activity along with all features and a learning to rank algorithm to generate the best model that suits the community as a whole.

## 3.2 Hybrid Recommendation Model

The Social Networking Sites (SNS) that we are working on consists of sparse input data with domain specific interests. In order to accommodate the sparse data challenge, we include implicit information from user such as user's online activities as part of our inputs. In contrast to traditional supervised approaches which rely explicitly on users' feedback, we focus on implicit learning with users' interactions. Unlike users' explicit feedback which is sparse and may place an increased cognitive burden on users, users interaction with the SNS is relatively easy to obtain and place little to no burden on users. Since our SNS is domain focused, we incorporated domain specific information into our model to create a specialized recommender system solely for this site. Another major factor on our approach is user profiling, it includes user's interests, interaction and demographic information. The more information we had of a user, the better our system understands the user and the better the result in our recommendations. Hence, we encourage users to complete their online profiles on the site. In addition, the SNS is also mobile friendly which induced posts with geographic information. Along with this information we could evaluate the post's interests level based on the distance with users.

For all users, we generate the following data as input features:

1. Collaborative Filtering where similar users are grouped and ranked. Users are grouped based on the following criteria:
    (a) List of users' friends list
    (b) List of users' votes on posts
    (c) List of users' comments on posts
2. Content Based where similar posts are grouped and ranked. Each post is rated based on similarities of keywords between users and posts.
3. List of followers where frequent interaction between users provides a higher rating.
4. List of locations where post closer to users provides a higher rating. A detailed explanation would be provided at subsection 3.2.3.
5. List of popular topics where hot topic provides a higher rating. A detailed explanation would be provided at subsection 3.2.3.
6. List of domain specific interests where similar interest provides a higher rating. A detailed explanation would be provided at subsection 3.2.3.

For domain interest similarity, we focus on finding vehicle model and brand name that users had owned or liked in the past. After all these features are generated, we combine them with learning to rank algorithms. The following learning to rank algorithms MART (gradient boosted regression tree), RankBoost, AdaRank, Coordinate Ascent, LambdaMART, ListNet, and Linear Regression are used in our proposed methods. The setup for each of the learning to rank algorithm is as follows:

1. RankNet: epoch = 100, layer = 1, node = 10 and rate = 0.00005
2. RankBoost: round = 300, and threshold = 10
3. AdaRank: round = 500, tolerance = 0.002, and times = 5
4. MART, LambdaMART: tree = 100, leaf = 10, shrinkage = 0.1, threshold = 256, minimum leaf support = 1, and estop = 100
5. Coordinate Ascent, ListNet, Linear Regression: no parameters

We run them on some training data and choose the best algorithm as our final ranking model.

### 3.2.1 Content-Based Filtering

In our approach we used vector-space model to extract keywords from posts. Since our social sites' domain is about vehicle and traveling, the most relevant and interesting keyword we extracted would be vehicle oriented.

Content-Based filtering creates user content profiles for each user to characterize their interests and recommend posts similar to what users had voted or commented in the past [20]. As we all know, posts are widely unstructured and to extract useful information from it is quite challenging.

Our system uses information retrieval techniques to perform text pre-processing. The algorithm we used is term frequency - inverse document frequency (TF-IDF), which is one of the most popular feature weighting methods used to describe text contents in vector space model. A term which is a keyword or interest in a post. The term score is given by dividing the term frequency in post by the sum of the frequency of the term over the whole collection of posts. The relevance of each post with user's interest is determined by computing the average relevance of all the words in the user document.

One major challenge when processing texts is that most posts contain a large set of words. If each of these words was to be converted into vector representation, the number of dimensions would be too high. Hence, text pre-processing is crucial in reducing the number of dimensions. Several pre-processing methods are performed on input posts before content profile generation including stemming and removal of stop words [37].

Stemming consists of converting each word to its stem (its neutral form). In order to get the stem of a word, the algorithm removes suffixes representing tag-of-speech and verbal/plural inflections [28]. Stop words are words that occur frequently in posts. Due to the fact that they appear in almost every post, they carry little information about the contents of posts they appeared in.

The structure of posts are generally unstructured or semi-structured. The most popular data representation in information retrieval is the bag-of-word model which represents each user as a vector [18]. Bag-of-word model keeps track of keywords that occurred in posts and treats each keyword from the post as an “attribute”. These keywords or attributes would be assigned with a value that represents the frequency of that word occurring in user’s posts. Each user is then represented in vectorial space as “profile”. The main advantage of this representation is its conceptual simplicity and computational efficiency.

A vectorial profile representation uses the idea of representing users and posts as multi-dimensional vectors. The equation for post profiling is as follows

$$V_d = (w_{1d}, w_{2d}, ..., w_{nd}), \tag{3.1}$$

where $w_{id}$ corresponds to the frequency of each keyword $k_i$ in post $d$, and value $i$ range from 1 to $n$. Each vector coordinate represents the TF-IDF value of a specific keyword that occurs in a post. A post vector normally has high dimension and most of those values take in the value of 0. The similarity of posts can be measured by the angle between the two vector representation. The smaller the angle, the more similar between those two posts. In short, by using vectorial representation we can evaluate posts relevance by measuring the value difference of each vector coordinate. [28]

In TF-IDF model, each keyword is assumed to have importance proportional to the number of times it occurred in a post [23]. The term frequency of a keyword $k_i$ in post $d$ is denoted by $TF(k_i, d)$, where the value represents the number of times that keyword $k_i$ has appeared in post $d$. The higher the $TF(k_i, d)$, the higher the chances of that keyword $k_i$ representing that post $d$. However, some common keywords tend to appear across many different posts, which have little discriminating power. In order to remedy this issue, IDF measure is used to identify keywords with high frequency across all posts which have little discriminating power.

While term frequency (TF) measures the occurrence of keyword $k_i$ in post $d$, inverse document frequency (IDF) measures the occurrence of keyword $k_i$ across all posts [36]. The value $|D|$ denotes the total number of posts and the document frequency of keyword $k_i$, is denoted by $DF(k_i)$, where the value represents the number of posts that the keyword $k_i$ had occurred in. The *inverse document frequency* of keyword $k_i$, is denoted by $IDF(k_i)$ with the following formula

$$IDF(k_i) = 1 + \log(|D|/DF(k_i)). \tag{3.2}$$

The intuition of IDF is that keywords with rare occurrence across posts are more valuable. The importance of each keyword is inversely proportional to the number of posts that contain the keyword [36]. Therefore, if $IDF(k_i)$ for a keyword $k_i$ is low, then this keyword must occurs in many posts, indicating this word has little discriminating power. On the other hand, if $IDF(k_i)$ is high, then the keyword $k_i$ only appeared in a few posts, indicating this word has great discriminating power.

So in short, we want keywords that have both high TF and IDF, and Salton et al. [33] demonstrated that the product of these two values would be a good indication of that keyword's performance. We can express this desire with the following equation

$$w_{id} = TF - IDF(k_i, d) = TF(k_i, d) * IDF(k_i), \tag{3.3}$$

where $w_{id}$ represents the weight of the keyword $k_i$ in post $d$.

### 3.2.2 Collaborative Filtering

Collaborative filtering algorithm recommends posts to user based on preferences from a collective group of users. Collaborative filtering approach is based on user's similarity; it analyzes relationship between users and inter-dependencies between posts to identify user-post associations [20]. The basic concept is people who liked similar posts in the past will probably like similar posts in the near future as well. In short, collaborative filtering is a subset of algorithms that exploit other users' ratings similarity and target user's own history to recommend posts that the target user has not rated.

#### Matrix Factorization

The standard approach of matrix factorization based collaborative filtering characterizes both users and posts by vectors inferred from rating patterns. It represents the input data in matrix form where one dimension represents users and the other dimension represents posts of interest. The most convenient inputs are explicit feedback given by users regarding their interest on posts. For example, user's rating or liking and disliking of posts, and high correspondence between post and user leads to recommendation. However, this approach does not suit well for social networking sites because users do not normally provide rating to posts they read. Moreover, even if users provide like or rating to posts they read, it is very likely that they have only liked or rated a small percentage of available posts resulting in small rating matrix.

Fortunately, one strength of matrix factorization is that it allows incorporation of additional inputs. When explicit input is sparse, we can include implicit feedback, which indirectly reflects opinion of user by observing user's past behavior, for example, browse history, search patterns or even user actions such as views, clicks, likes, shares and so on. These actions are related to the level of confidence in observing user preferences. Rather than relying solely on explicit ratings given by users. After the user-post matrix is built, the matrix factorization model then tries to find latent factors that can be used to predict the expected preference of user for each post.

Matrix factorization model maps both users and posts to a joint latent space of dimensionality $f$, such that user-post interactions are modeled as inner products in that space. The factor matrices are also known as latent feature models. The factor matrices represent hidden features which algorithm tries to discover. Each post $i$ is associated with a latent vector of $q_i \in R^f$, and each user $u$ is associated with latent vector $p_u \in R^f$. So, the elements in vector $q_i$ measure the extent of features that post possesses, while the elements in vector $p_u$ measure the extent of interest level that user has for those features. The resulting dot product, $q_i^T p_u$, captures the interaction between user $u$ and post $i$. The approximate rating $r_{ui}$ for user $u$ on post $i$ can be denoted by the following equation

$$r_{ui} = \vec{q_i}^T \vec{p_u}. \tag{3.4}$$

In [29,35], it is suggested that regularizing the model can and will avoid over-fitting. To learn the factor vectors ($p_u$ and $q_i$), the system minimizes the regularized squared error on the known training set ratings

$$\min_{p\dot{q}} \sum_{(u,i)\in f} ((s_{ui} - q_i^T p_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2), \tag{3.5}$$

where $f$ represents the dimension of joint latent space and $s_{ui}$ represents the score given by user $u$ to post $i$. The system learns the model by fitting the previously observed ratings. The goal is to generalize those previous ratings such that it predicts future recommendation without over-fitting. Thus the system avoids over-fitting by regularizing the learned parameters, whose magnitude is penalized. The constant $\lambda$ controls the extent of regularization, which is set to 1.3.

### Alternating Least Squares

Although the above equation 3.5 is not convex because both $q_i$ and $p_u$ are unknowns, if we fix one of the unknowns $q_i$ or $p_u$, then the optimization problem becomes quadratic and can be solved optimally [20]. Alternating Least Square (ALS) is a kind of matrix factorization approach that has been implemented in many machine learning libraries and widely studied in both academia and industry.

ALS is an iterative algorithm; in each iteration, it alternatively fixes one factor matrix $q_i$ or $p_u$ and solves for the other $p_u$ or $q_i$ respectively. When $p_u$ is fixed, the system recomputes the $q_i$ by solving a least-squares problem. This process continue until it converges. Then the unknown ratings between users and posts can be subsequently computed by multiplying these two matrices. With these decomposed matrices, the recommender system can recommend posts based on the predicted ratings to increase customer satisfaction.

One of the advantages of using ALS algorithm is parallelization. In ALS, the system computes each $q_i$ independent of the other post factors and computes each $p_u$ independent of the other user factors. This gives rise to potentially massive parallelization of algorithms. Also, as stated in [20], matrix factorization models are superior to most classic techniques for post recommendations.

**Adding Bias** As we all know, observed ratings are always affected by effects associated with users or posts, known as bias, and are unrelated to any interactions. For example, some users have a tendency of giving higher ratings than others; and some posts received higher ratings than others due to author's fame. Thus, it is unwise to interpret the full rating values as the form of $q_i^T p_u$. To address this issue, the system tries to identify the portion of individual user and post biases with the following equation

$$b_{ui} = \mu + b_i + b_u. \tag{3.6}$$

The bias rating for $r_{ui}$ is denoted by $b_{ui}$, which accounts for both user and post effects. The overall average rating is denoted with $\mu$, user bias is denoted by $b_u$, and post bias is denoted by $b_i$. By extending equation 3.6 we get the following equation

$$r_{ui} = \mu + b_i + b_u + q_i^T p_u. \tag{3.7}$$

Here, the observed rating is broken down into four components: global average, post bias, user bias, and user-post interaction. The system can learn the model by minimizing the following squared error function [19, 29]

$$\min_{p\dot{q}b} \sum_{(u,i)\in f} (r_{ui} - \mu \; b_u - b_i - p_u^T q_i)^2 + \lambda(\|p_u\|^2 + \|q_i\|^2 + b_u^2 + b_i^2). \tag{3.8}$$

Since bias tends to capture much of the observed signal, their existence is vital for an accurate model.

**Confidence Level** Since our model is built around implicit feedback, a user's exact preference level is hard to quantify. Therefore, it is valuable to attach confidence score for estimated preference. Confidence level can stem from frequency of actions and duration of actions. The matrix factorization model can readily accept varying confidence score by giving it less weight to less meaningful observation. Let's define $c_{ui}$ as the confidence level of observing $r_{ui}$, then the model cost function with confidence level is as follows

$$\min_{p\dot{q}b} \sum_{(u,i)\in f} c_{ui}(r_{ui} - \mu - b_u - b_i - p_u^T q_i)^2 + \lambda(\|p_u\|^2 + \|q_i\|^2 + b_u^2 + b_i^2). \tag{3.9}$$

In short, our focus is to use implicit feedback from users such as friends and follower-followee relations to accommodate explicit sparse user feedback. We gather users' information by monitoring users' actions and behaviors on the site such as up-votes, down-votes and comments. In our system, we provide rating to all posts posted by user's friend list such that posts related to user's friend list would receive rating 3. Next, we include user's vote on posts as part of user's rating system such that up-vote would get score 4 and down-vote would get score 2. Last but not least, we use sentiment analysis to rate posts that users have commented on. From user's comment, we clean the post by filtering positive words and negative words from it. For each positive word we give it a $+1$ and for each negative word we give it a $-1$, at the end we sum up all the scores. If the score is positive then we consider it as positive comment and give it a score of 5, on the other hand, if the score is negative then we assume it is a negative comment and give it a score of 0. We include all these information as input and use matrix factorization training model to learn user's and post's latent factors. Then we predict user's interest level on unread posts based on similar users' ratings.

### 3.2.3 Context Information

In addition to the traditional hybrid model that uses only Content-based and Collaborative approach as explained above, our hybrid model also incorporates some new contextual features. Social relation is one of the most useful features in social recommender system as it is one of the fundamental information for social networking sites. Popular posts are used to increase diversity in recommended posts. Spatial information had been widely used in recent recommender system thanks to the advancement of mobile technologies. With the help of Global Positioning System (GPS), we can refine posts so that they are closer and more relevant to user's location. Domain interest is a unique feature for our recommender system as it is tailored to satisfy a closed group of users with specific field of interest.

### Social relations

Since our SNS has follower-followee relations, we include social relation between users as part of our input features in the recommender system. As we all know, users are more interested in news and posts from close friends. Thus, it is an important factor for our recommender system to include this feature as part of our input. We define social strength between users based on their interaction frequency. The more frequent two users interact the stronger the strength between that pair of users. For each user, we calculated their interactions such as comments, votes and private messages with their friends. Based on how frequently they interact with each other, we rank friends with frequent interactions to have higher priority when recommending posts.

We calculated the relation score $sc_{uj}$ of a user $u$ with friend $j$ by tallying up the number of up-votes, comments and messages with every friend and follower. Then we summed up the total number of up-votes, comments and messages that user $u$ had done on the social networking site as $ts_u$. Also to prevent a single user from dominating this aspect of recommendation, we normalized social strength between users. We use the following equation to get the normalize result.

$$ns_{uj} = \frac{sc_{uj}}{ts_u} \tag{3.10}$$

As mentioned above, $sc_{uj}$ represents interaction score between user $u$ and friend $j$, and $ts_u$ represents the sum between user $u$ and all his/her friends, and $ns_{uj}$ represents the normalized social relation score between 0 to 1 for user $u$ and friend $j$ with 0 representing the lowest interaction level between user $u$ and friend $j$ and 1 as being most active interaction level between user $u$ and friend $j$.

### Popular Topic

In reality, post perception and popularity constantly change as new selection of posts are being published. The system should account for temporal effect reflecting the dynamic, time-drifting nature of user-post interactions. Also, temporal interest is a big factor for many users in SNS. For instance, the publishing of a new post from user related to holiday event such as Christmas or New Year may be a trending topic for only a short period of time. Hence, it is essential to include this feature as part of our inputs in the system for an accurate prediction. Also, it helps provide diversity in our recommendation list and introduce users to some current trending posts. We evaluate posts from within 60 days and rank their popularity based on the number of votes and comments it received because older popular posts do not represent the current trend.

The popularity score $ps_i$ for each post $i$ is the sum of the number of votes $nv_i$ and number of comments $nc_i$ it gets using the following equation

$$ps_i = nv_i + nc_i. \tag{3.11}$$

These scores are then normalized using the following equation

$$ns_i = \frac{ps_i - ms}{xs}. \tag{3.12}$$

First, we find the maximum popularity score $xs$ and the minimum popularity score $ms$ among all posts. Then for each popularity score $ps_i$, we normalize the score into $ns_i$ using equation 3.12 where the normalized score would be between 0 and 1. Posts with higher vote counts and comments would receive a score closer to 1.

### Spatial

With the additional information of geographic location from some users and posts, we included this information as part of our input features for the recommender system. Through vast usage of smart-phones, more and more posts are tagged with geo-locations which indicate which region each post is being uploaded. Additionally, most users are generally concerned or interested in posts that are fairly close to them. Also, this is an important feature for our recommender system because most of our users are people who enjoy traveling through places.

We compare user's location and post's location to calculate the relative distance. A score is assigned to post based on the distance with user. We used geohash to determine the distance between user and post. Geohash is a spatial data structure which divides

Table 3.1: Table for grid size

| Length | width | height |
|---|---|---|
| 1 | $\leq$ 5000 km | $\times$ 5000 km |
| 2 | $\leq$ 1250 km | $\times$ 625 km |
| 3 | $\leq$ 156 km | $\times$ 156 km |
| 4 | $\leq$ 39 km | $\times$ 20 km |
| 5 | $\leq$ 5 km | $\times$ 5 km |
| 6 | $\leq$ 1.22 km | $\times$ 0.61 km |
| 7 | $\leq$ 153 m | $\times$ 153 m |
| 8 | $\leq$ 38 km | $\times$ 19 km |
| 9 | $\leq$ 5 m | $\times$ 477 m |

space into grids. Posts and users are assigned with a geohash of length 9, if their geolocations are provided to the system. Table 3.1 shows the grid size based on the length of a geohash. We compare posts with users geohash and rank them based on matching length. If a post's geohash fully matches user's geohash then we give that post a score of 1. If a user or post does not have geohash then we give it a score 0 or else for each mismatch of length would deduct 0.1 from the score. So an example of post with geohash *gbsuv7ztt* and user wtih geohash *gbsuv7zup* would result in a score of 0.8. Thus, closer posts would be given a higher rating in our system.

### Domain Interests

Since our SNS is domain specific, we incorporate domain specific information into our input. This is an unique feature that is specialized to our recommender system as we are working on a closed domain of interests where all users are vehicle enthusiasts. So vehicle brands and models are a huge indication of users' interest level. If a post is associated with some vehicle brands or models then a score is assigned to the post.

We pre-process each post with stemming and stop word removal. Then compare the content with existing vehicle model and brand, if it is a perfect match with a vehicle brand or model we give that post-vehicle a score of 1. We used a list of vehicle brands and models provided by the company, and compared them with each post $i$ to rank post-vehicle strength. String comparison is used in post-vehicle strength ranking where it compares vehicle model and post content, and returns number of different characters. The similarity score $ss_i$ for post $i$ is given based on the following equation

$$ss_i = 1 - \frac{dc}{tc}, \tag{3.13}$$

where $ss_i$ represents similarity score for post $i$, $dc$ represents the number of mismatching characters with the model and $tc$ is the total string length of the model or brand. We set a threshold to ignore any mismatching string with score lower than 0.7.

## 3.3 Generalized and Personalized Model

We are using multiple features such as content-based, collaborative, and context features. A learning to rank algorithm is used to rank each feature accordingly. From previous chapters, we would be using a range of learning to rank algorithms such as MART, RankNet, AdaRank, and ListNet to determine the best algorithm for each user. We categorize our models into two different categories, a personalized model for active users with ample feedback and a generalized model for new users or users with less activity.

A generalized model is designed for new users or less active users who do not have enough feedback or activity to create a personalized model. Due to the lack of information on these users, we design our recommender system to view users as a community by compiling all users' input together. By learning all users inputs and activities, our model would be able to determine the overall interest level of this community. Although some features such as content-based and collaborative filtering might seem odd to be including these features in the model, since we are using learning to rank algorithm to combine these features, if these features are truly insignificant they would have lower importance as opposed to features that rely less on user's information. In the next chapter, we would discuss a range of combination features we had tested to find the most optimal model.

With the advancement of technology, we are able to learn and store information faster and at a cheaper rate thus promoting personalized recommender system for user. A personalized model is designated to loyal customers who have been with the site for some time and are actively interacting with others. A generalized model could work well for most users but not all, thus we implemented the idea of creating a personalized model for each active user. All users are unique and different, so a personalized recommender model for each individual user would highly improve recommendations. Using learning to rank algorithms to join the above features, we would be able to modify the importance level of each feature differently depending on user's interest level and activity patterns. In the following chapter, we discuss a range of combination of features and learning to rank algorithms to determine the best personalized model for each user.

# Chapter 4

# Experiment

In this chapter, we discuss experiments we performed on our proposed recommender system to determine its effectiveness and accuracy. Firstly, we discuss our overall experiment design on how data are collected and used for testing our system. Secondly, we focus on what test data were used and how they were collected. Next, we further discuss the implementation of our proposed recommender system and packages used for enhancing our system. Lastly, we show the experiment results along with our analysis and evaluation on the improvement of our system brings to the site.

## 4.1 Experiment Design

Experiments were designed to validate the improvement of accuracy on recommended social posts to users with our proposed recommender system. We devised experiments which involved a closed group of active and not-so-active users who provided explicit feedback. We started with implementing the proposed recommender system to the social networking site and actively collecting user feedback on posts. We developed content-based module, collaborative filtering module and social relation module simultaneously. The content-based module focuses on configuring user's interested keywords based on their posts and messages. The collaborative filtering module mines user's interactions and rank posts based on his/her previous interactions. The social relation module, for each pair of users computes their relation strength based on their social interactions on the site. The rest of the features such as popularity and spatial features are retrieved during recommendation process. All these features are then inputted to a learning to rank model which ranks posts and outputs them in descending orders.

The data set that we are working with is an online social networking site about traveling experiences with vehicles. The data set consists of data that dates back to 2011 with over thirty thousands registered users and over seven thousands active users who had logged in within the last 60 days. We composed a list of closed group users based on their online activities with the social site and their activeness and willingness to provide feedback. We selected a group of seven highly active users for personalized recommendation and another group of twenty less active users for generalized recommendations.

Once we have sufficient number of users' rated posts, we use those data and split them into training and testing set for validating the accuracy of our proposed recommender system. We used the collected data to calculate scores using various recommendation strategies mentioned in Chapter 3. Lastly, we calculated our final recommendation score by combining various features using different learning to rank algorithms and compared our results.

### 4.1.1 Dataset

We used explicit and implicit feedback from users to generate recommendations in our experiment. Since our designed recommender system is for a domain specific social networking site, we focus more on input features that matter the most to users on this site. The dataset is categorized into two categories:

1 Complete data set: where all data are used to generate recommendations for less active or new users

2 Personalized data set: where individual user data are used to generate recommendations posts for more active users

Table 4.1 shows a list of content types classified in the database, and the number of posts in each category. The content types of posts can be categorized into four sections:

1. blog post such as Vehicle, News, Image, and Video where users share their traveling experiences
2. informative post such as HowTo, and Review where users share their thoughts on products
3. gathering post such as Event, and POI where users set up meeting location and time
4. private post anything that user had marked private

Table 4.1: Categories of posts

| Content Type | Number of posts |
|---|---|
| Blog Post | 134592 |
| Gathering Post | 4768 |
| Informative Post | 2393 |
| Private | 21756 |
| Total | 163500 |

We calculated the level of sparsity of our input data-set with the following equation

$$sp = \frac{n_f}{n_u \times n_i}, \tag{4.1}$$

where $sp$ represents the level of sparsity, $n_f$ represents the number of feedback from users, $n_u$ represents the number of available users, and $n_i$ represents the number of available items. Therefore the level of sparsity that we are working on is $6.425 \times 10^{-6}$.

Since we are working with a social networking site, our recommender system will not include or use any information or post if a user had classified them as private. We used the original copy of data set from the site. For accuracy improvement, we include all users' historical data in learning the recommender system but defined a threshold on recommended outputs to recommend only recent posts. As the ever changing topic in this era, human interest level change over time thus we are only recommending posts that are within 60 days period.

Our recommender system uses historical data to compose final recommendation. Our data set contains five full years of useful users' information. Since we are using a copy of an online social networking site's database, we collect their data in batches and store them in MySQL Database.

### 4.1.2 Implementation

Each of the module of our system is implemented using a combination of Java and Scala Programming Language. We used IntellijIDEA IDE as compiler for the source code and Windows 10 as our testing platform. We run our experiments with Intel Core i7 @ 2.70 Ghz, quad-cores and 16 GB RAM. We used both File Storage and SQL Database for storage.

Social relation strength is an important aspect in social networking sites and an important input feature for the Collaborative Filtering and Content-based module as well. Social relation strength module is written in Java programming language because we are using WEKA library to generate social relation graphs. For each user, we tallied up the number of interactions such as comments, up-votes and private messages with each of his or her friends and followers. Based on how frequently they communicate with each other, we rank their friends who have frequent interactions to have higher priority in recommending posts.

Next, Content-Based module was implemented along side with Collaborative module as they were independent modules. The Content-based module is written in Scala programming language which is known for its ease of partitioning and parallel programming. In Content-based module, we used a three part process where the first part is pre-processing posts, the second part is creating user's content profile vector TF-IDF scheme and the last part is comparing all posts with each user's profile using cosine similarities. For the first part, we did some pre-processing on collecting and standardizing vehicle brands, style and models. Since our site's domain is about vehicle, the most relevant and interesting keywords should be vehicle oriented.

For the second and third part, we used Apache Lucene an open source information retrieval software library, to implement the vector space model and cosine similarity calculation. First, we aggregate all user's posts into a single document then we did stemming and stop words removal from the document to define user's interest. We used the package “WordnetStemmer” as our stemming algorithms to find the stem of words. Stemming is a process of removing prefix and suffix from words, remove tenses from verbs, and strip “s” from plural nouns. A list of stop words were provided by the company and are removed from the corpus as well. After cleaning up the corpus, we use Lucene to store users' profile in form of vectors using local file storage system as separate profile. Each user's profile is named after their “user id” and within their profile it contains a list of keywords. When every user's profile is created, we then move on to the final part of the process.

Before comparing the user profile with each post, we filtered out all posts that have privacy flag on such as private messages, pictures and posts. Next, we stem and strip stop words from each post using the same criteria from above. We create a vector for each post and compare it to each user's profile vector. Then we use Lucene's scoring method to compare these vectors, for each post we return top 100 interested users in descending order along with similarity score between 0 and 1. These scores are then used for ranking user's interests level on each public post.

A third module Collaborative Filtering was also implemented using Scala programming language with Alternating Least Square (ALS) to provide recommendation based on similar user actions or interests. In this module we used both implicit and explicit feedback as measurement for similar users. All these ratings are then inputted into a learning model in list format such as the following:

```
<user id>, <post id>, <score>
1011, 102, 3
1012, 100, 4
1011, 111, 0
```

The above input is then used as input for the ALS model from Spark Machine Learning library to generate matrix factorization model. The trained model is then saved in local file storage and can be called to predict user's interest level on posts.

The next module would be the recommender system which takes all input and provides ratings for each post based on user's interest. In this module, we make use of social relation strength from the first module, Content-based ratings from the second module and Collaborative ratings from the third module. Each module's ratings are normalized into real value between 0 and 1, where 0 is the lowest value and 1 is the highest. In addition to the above mentioned features, we also use popularity, geographic information and vehicle interests.

In terms of popularity, we evaluate posts from within 60 days and rank their popularity based on votes and comments. Posts with higher vote counts and comments would receive a higher score.

Another signal that we include as input are the geo-tag from post. This is an important factor for our users because many of them are travelers who enjoy traveling to different places. For this signal, we use geohash to determine the distance between user and post. We rank them based on matching length of geohash.

As for the last signal, it is unique to our recommender system as we are working on a closed domain of interests where all users are vehicle enthusiasts. Each post that is associated with vehicle brand or model will be given a score.

At the end, we feed all the above mentioned features into the RankLib library [1]. RankLib is a library of learning to rank algorithms, it takes input data-set, and test data-set as text file and outputs ranking model. We can also set different evaluation metric for learning and testing the model. The input file to the learning to rank algorithms is as follows:

```
<relevancy> qid:<datasetId> <featureId_1>:<score>  <featureId_N>:<score>
1 qid:3 1:64.0 2:54.0 3:31.0 4:25.0 5:356.0 6:0.01954206542 7:0.020361286
5 qid:8 1:13.0 2:1.0 3:3.0 4:0.0 5:781.0 6:0.02467538727 7:0.0
```

In the above sample, the first line shows the column names; whereas the second and third lines are examples of the input data for the learning to rank algorithms. The first column represents the relevancy score, where score of 5 represents user is highly interested and the score of 1 means user is least interested. The relevancy score used in both testing and training are explicit feedback from users who had participated in evaluating our recommender system on their social networking site. The column "qid" represents the user identification number, for generalized model and as for the personalized model we use post identification number as "qid". Rest of the data for each line represents the set of feature ids and their corresponding scores. Since we are using six features as our recommendation strategies, we have 6 sets of feature ids and scores. All the features scores are normalized to a value between 0 and 1.

Based on user feedback, we decided to choose 7 active users for personalized recommendation model and 20 less active users for generalized recommendation model. For each active user, we generate an unique ranking model individually. For less active users or new users we generate a one for all ranking model. We run through all available ranking algorithms such as *MART, RankNet, AdaRank, Coor-Ascent, LambdaRank, LambdaMART, ListNet, Linear-regression* and choose the best model accordingly.

## 4.2 Results and Analysis

We generated recommendations for our experiment using two approaches, generalized model and personalized model. We collected 60 explicit feedback from 7 active users for evaluating our personalized models and about 15 explicit feedback from 20 less active users for evaluating generalized model. For active users data set, we have 60 ratings feedback from each user, and we used 50 out of 60 ratings as training set and the remaining 10 as testing set.

Table 4.2, shows the composite of our training and testing cases for our less active users group. The number of users in testing is greater than training because one of the user only rated one post and we grouped this data into testing. The reason behind this experiment is to show that the generalized model should work well for new user without any prior information of that user.

Table 4.2: Generalized model inputs for learning to rank algorithms

| Number of users | Number of ratings | Input type |
|---|---|---|
| 19 | 270 | training |
| 20 | 61 | testing |

The secret to a successful and effective recommender system lies in the ranking of recommended posts. A good recommender system not only generates a list of recommended posts but also ranks those recommendations. From both system and user perspective, it is not wise for a recommender to generate a long list of recommended posts. A long list of recommended posts would only confuse users and waste resources in generating those posts. Thus, the recommender system should rank the recommended posts in descending order of importance and shows only the most important recommendations. In most cases, the recommender system sets a threshold and only identifies the top 10 to 15 most recommended posts and presents those selected posts to users.

Our recommender system generates a list of posts with associated scores from each recommendation strategy. Each post is associated with six scores from six different strategies. Also, for evaluation, each post is associated with a relevancy score given by the users for expressing their level of interest. A higher level of relevancy score indicates a higher level of user's interest on that post. Therefore, each post in our experiment is associated with a relevancy score and six attribute score which are predicted by six different strategies. So, a good indication of accurate prediction is when relevancy score and attribute scores are in sync. In the final phase of recommendation, we sort the list of posts in descending order using those six attribute scores. The post with the highest attribute scores is at the top and when the post has a relevancy score of 5, it proves that our ranking order is accurate. If most posts with high relevancy score are among the top posts, then it proves the accuracy of our recommmender system. If a low relevancy score is among our recommended list then our ranking process might be ineffective.

However, the ranking of six attribute is not an easy task. In order to provide the most accurate and efficient ranking list, we need to provide different weights to these six attribute scores. A simple linear combination with equal weight might not be sufficient for this experiment. Since users are unique and their preference to each individual post is different, our proposed personalized recommender model is trying to capture this uniqueness in user. A learning to rank algorithm is used to derive a ranking model for generating a ranked list of recommendations.

In this experiment, we used Normalized Discounted Cumulative Gain (NDCG) as our main evaluation metric. NDCG is often used in information retrieval to determine the effectiveness of various ranked list. Precision and Recall metrics were not used as our evaluation metric because binary relevance assessments are heavily influenced by outliers [32]. The NDCG metric captures the following two key factors: highly relevant posts are preferable at top ranking rather than mildly relevant ones, and relevant posts that appear at the end of ranking is less valuable.

The NDCG metric captures the following two key factors:

1 highly relevant posts are preferable at top ranking rather than mildly relevant ones

2 relevant posts that appear at the end of ranking are less valuable

For example, given a set of test queries evaluated by specialist and are graded on a scale 0-3, with 3 indicating strong relevance and 0 indicating post is non-relevant. Now with each query, our Recommender System returns top 10 results in descending order. Here we replace each post with their relevance score, which is known as $gainvectorG$. For example, the following are queries for $G_1, G_2$:

$$\begin{aligned} G_1 &= (1,0,1,0,0,3,0,0,0,2) \\ G_2 &= (0,0,2,0,0,0,0,1,0,0) \end{aligned} \tag{4.2}$$

By summing the above graded relevance scores up to any point of the ranking, we obtain a measure of Cumulated Gain (CG). For instance, for query 1, the cumulated gain at first position is 1, at second position is $1 + 0$, at third position is $1 + 0 + 1$, and so on. So, the *cumulated gain vector* for $q_1$ and $q_2$ are as follows:

$$\begin{aligned} CG_1 &= (1,1,2,2,2,5,5,5,5,7) \\ CG_2 &= (0,0,2,2,2,2,2,3,3,3) \end{aligned} \tag{4.3}$$

A formal definition for CG is as follow [32]:

$$CG_j[i] = \begin{cases} G_j[1] & \text{if i=1} \\ G_j[i] + CG_j[i-1] & \text{i} > 1 \end{cases} \tag{4.4}$$

where $CG_j[i]$ refers to the cumulated gain at the $i-th$ position of the ranking for query $q_j$.

In order to enforce the second rules, that relevant posts at the end of the ranking have lower value. We need to have a discount factor that reduces the impact of the gain as we move upper in the ranking. A straight forward discount factor would be the logarithm of the ranking position. Thus, the discount factor at position 2 would be $log_2 2$, at position 3 would be $log_2 3$, at position 4 would be $log_2 4$ and so on. Therefore, the Discounted Cumulated Gain (DCG) formula is as follow [32]:

$$DCG_j[i] = \begin{cases} G_j[1] & \text{if i=1} \\ \frac{g_j[i]}{log_2 i} + DCG_j[i-1] & \text{i} > 1 \end{cases} \tag{4.5}$$

where $DCG_j[i]$ refers to the discounted cumulated gain at the $i - th$ position of the ranking for query $q_j$.

Following the examples from above, the DCG vectors for queries $q_1$ and $q_2$ is as follow:

$$\begin{aligned} DCG_1 &= (1.0, 1.0, 1.6, 1.6, 1.6, 2.8, 2.8, 2.8, 2.8, 3.4, 3.4) \\ DCG_2 &= (0.0, 0.0, 1.3, .1.3, 1.3, 1.3, 1.6, 1.6, 1.6, 1.6, 1.6) \end{aligned} \tag{4.6}$$

For the above instance, we notice that the discounted cumulated gains are less affected by relevant posts at the end of the ranking which is precisely as intended.

To produce CG and DCG curves over a set of test queries, we need to average the overall queries, as follow [32] :

$$\begin{aligned} \overline{CG}[i] &= \frac{1}{N_q} \sum_{j=1}^{N_q} CG_j[i] \\ \overline{DCG}[i] &= \frac{1}{N_q} \sum_{j=1}^{N_q} DCG_j[i]um \end{aligned} \tag{4.7}$$

For instance, the example queries $q_1$ and $q_2$ their averages are as follow [32] :

$$\begin{aligned} \overline{CG} &= (0.5, 0.5, 2.0, 2.0, 2.0, 3.5, 3.5, 4.0, 4.0, 5.0) \\ \overline{DCG} &= (0.5, 0.5, 1.5, 1.5, 1.5, 2.1, 2.1, 2.2, 2.2, 2.5) \end{aligned} \tag{4.8}$$

As many will notice the CG and DCG defined above are not computed relative to any baseline; unlike precision and recall figures that are computed relatively to set of relevant posts. Therefore, to normalize CG and DCG metrics, we need to define a new baseline to be used for normalization. This new baseline would be the ideal CG and DCG metrics, where ranking order matches the relevancy scores, such as :

$$IG = (3, ..., 3, 2, ..., 2, 1, ..., 1, 0, ..., 0) \tag{4.9}$$

The ideal gain (IG) vector for queries $q_1$ and $q_2$ would be as follow:

$$\begin{aligned} IG_1 &= (3, 3, 2, 2, 2, 1, 1, 1, 1, 0) \\ IG_2 &= (2, 1, 0, 0, 0, 0, 0, 0, 0, 0) \end{aligned} \tag{4.10}$$

As a result, the ideal CG would be as follow:

$$\begin{aligned} ICG_1 &= (3, 6, 8, 10, 12, 13, 14, 15, 16, 16) \\ ICG_2 &= (2, 3, 3, 3, 3, 3, 3, 3, 3, 3) \end{aligned} \tag{4.11}$$

Also, the ideal DCG would be as follow:

$$\begin{aligned} IDCG_1 &= (3.0, 6.0, 7.3, 8.3, 9.1, 9.5, 9.9, 10.2, 10.5, 10.5) \\ IDCG_2 &= (2.0, 3.0, 3.0, 3.0, 3.0, 3.0, 3.0, 3.0, 3.0, 3.0) \end{aligned} \tag{4.12}$$

Further, the average $\overline{ICG}$ and average $\overline{IDCG}$ scores can be computed as follows:

$$\begin{aligned} \overline{ICG}[i] &= \frac{1}{N_q} \sum_{j=1}^{N_q} ICG_j[i] \\ \overline{IDCG}[i] &= \frac{1}{N_q} \sum_{j=1}^{N_q} IDCG_j[i] \end{aligned} \tag{4.13}$$

For instance, the example queries $q_1$ and $q_2$ are as follow:

$$\begin{aligned} \overline{ICG} &= (2.5, 4.5, 5.5, 6.5, 7.5, 8.0, 8.5, 9.0, 9.5, 9.5) \\ \overline{IDCG} &= (2.5, 4.5, 5.1, 5.6, 6.0, 6.2, 6.4, 6.6, 6.7, 6.7) \end{aligned} \tag{4.14}$$

By comparing the average CG and DCG curves with the average ideal curves, we gain insight on how much room for improvement, given that the ideal curve is the maximum retrieval quality attainable.

Given the ideal DCG curve, it is still difficult to compare directly DCG curve for two distinct ranking algorithms. However, this issue could be corrected by normalizing the DCG metric as follow [32] :

$$\begin{aligned} NCG[i] &= \frac{\overline{CG}[i]}{\overline{ICG}[i]} \\ NDCG[i] &= \frac{\overline{DCG}[i]}{\overline{IDCG}[i]} \end{aligned} \tag{4.15}$$

For instance, the example queries $q_1$ and $q_2$ are as follow :

$$\begin{aligned} NCG &= (0.2, 0.1, 0.4, 0.3, 0.3, 0.4, 0.4, 0.4, 0.4, 0.5) \\ NDCG &= (0.2, 0.1, 0.3, 0.3, 0.2, 0.3, 0.3, 0.3, 0.3, 0.4) \end{aligned} \tag{4.16}$$

Thus, the area under the NCG and NDCG curve represent the quality of the ranking algorithm. The higher the area, the better the result. NDCG is used in our experiment due to the following benefits [32] :

1 allow systematic combination between post rank and relevance score

2 cumulated gain provides retrieval quality at any position in the ranking

3 cumulated gain stress top ranking so it is more immune to outliers

4 discounted cumulated gain allows down weighting of relevant posts at lower ranking

Thus, we implement NDCG as our main evaluation metric in comparing the accuracy of our Recommender models. We divide our user models as generalized model and personalized model. The generalized model is mainly targeting at new and less active users who are new to the site and are still in the phase of discovering the site. We proposed a generalized model for this group of users to help promote the site and actively help users discover their interests with the site. On the other hand, the personalized model is for more active users who had tons of interactions with the site and lots of active feedback. Personalized model is suggested for this group of users because we have more understanding about the user, thus in turn could create a better personalized recommender system for them.

### 4.2.1 Generalized Model

Since not all users on the social sites are active with ample interactions and feedback to the site, we devised an alternate model for this group of users. A generalized model which incorporates all users' implicit and explicit feedback as input, then uses learning to rank algorithms to create the recommendation. Results shown in Figure 4.1 are the average NDCG scores for those 20 less active users in testing set with different learning to rank models and various combinations of input features.

In "all signal" model, we used all features such as Collaborative Filtering, Content-based Filtering, Social Relation Strength, Temporal, Location and vehicle features, as input in learning the model. The idea behind this proposed model is that any additional data or input would increase the performance and accuracy of the learned model. In "no CF signal" model, we used features such as Content-based Filtering, Follower, Temporal, Location and vehicle features. We did not use Collaborative Filtering as part of the input for this model because Collaborative Filtering normally does not work well for new or less active users. Therefore, we believe excluding it from our model might help improve the

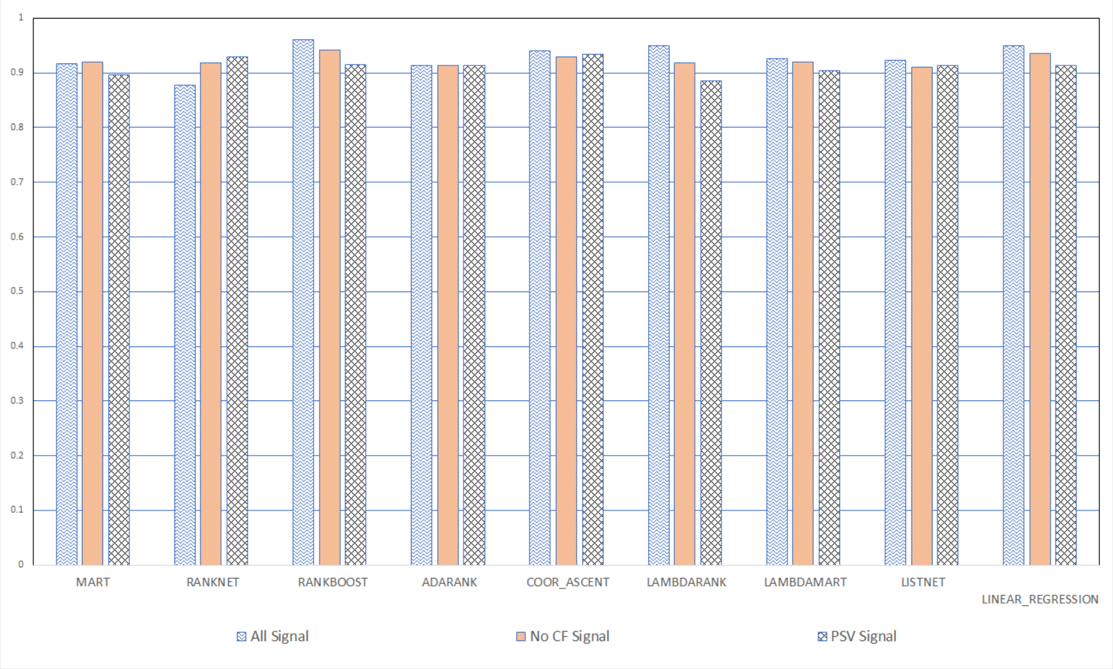

Figure 4.1: General model with NDCG scores

accuracy of our final model. In “PSV signal” model, we used only Temporal, Location and Vehicle features because these features do not require any prior knowledge from user. Content-based Filtering and Collaborative Filtering require substantial amount of user’s feedback to be accurate. We also removed follower signal from the input because we assume new users and less active users do not have interested users that they intently follow. The purpose of this model is to verify the performance of using only features that does not depend upon users’ feedback.

From Figure 4.1, we found out that “all signal” model has the best NDCG result combined with RankBoost learning to rank algorithm. The “no CF signal” model has the most consistent NDCG result, it has over 90% accuracy combined with any learning to rank algorithms. The “PSV signal” model has the lowest performance out of these three models because it uses the least input features.

Table 4.3: Best generalized model output for each approach

| Strategy | Best NDCG |
|---|---|
| All features | 0.932 |
| Without Collaborative Filtering | 0.933 |
| Temporal, Location, Vehicle | 0.920 |

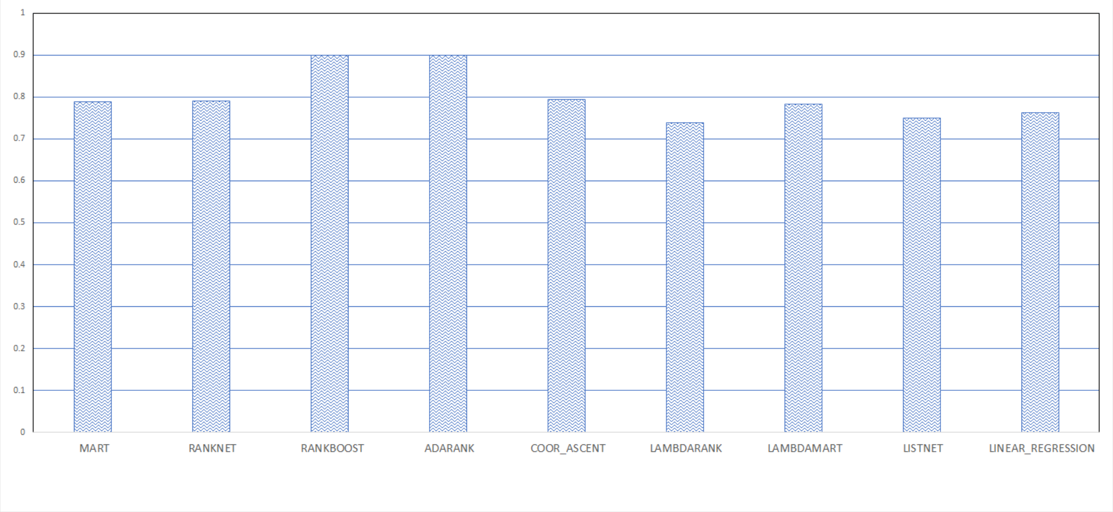

Figure 4.2: NDCG average for 7 users

Table 4.3, show the best averaged NDCG result on all 20 users and we noticed that all proposed methods are fairly accurate with average NDCG scores over 90%. The best averaged accuracy is the second model which does not include Collaborative Filtering signal. This demonstrates that the removal of insignificant signal might actually improve the overall accuracy of the output model, although the improvement between the first and second approach is not significant. The last approach on using only non-user-dependent features has the lowest accuracy. It is reasonable that the last model would have the lowest accuracy as it has the fewest input or data to learn from. However, the close proximity of result between models do indicate that generalized model still perform well without any user dependent features. Therefore, in our generalized model we propose to use all features for accuracy and non-user dependent features for efficiency.

### 4.2.2 Personalized Model

For active users with ample explicit feedback to the site, we implemented personalized recommender model for each of them. Based on the number of feedback from users, we set a threshold for learning personalized model. For those users with sufficient feedback, we generate all features such as: Collaborative Filtering, Content-based Filtering, Social Relations, Location, Temporal, and Vehicle feature as inputs. Then all these inputs are passed on to a list of learning to rank algorithms and the best fit model is chosen as their personalized model.

In Figure 4.2, we averaged the NDCG score from 7 users for each learning to rank

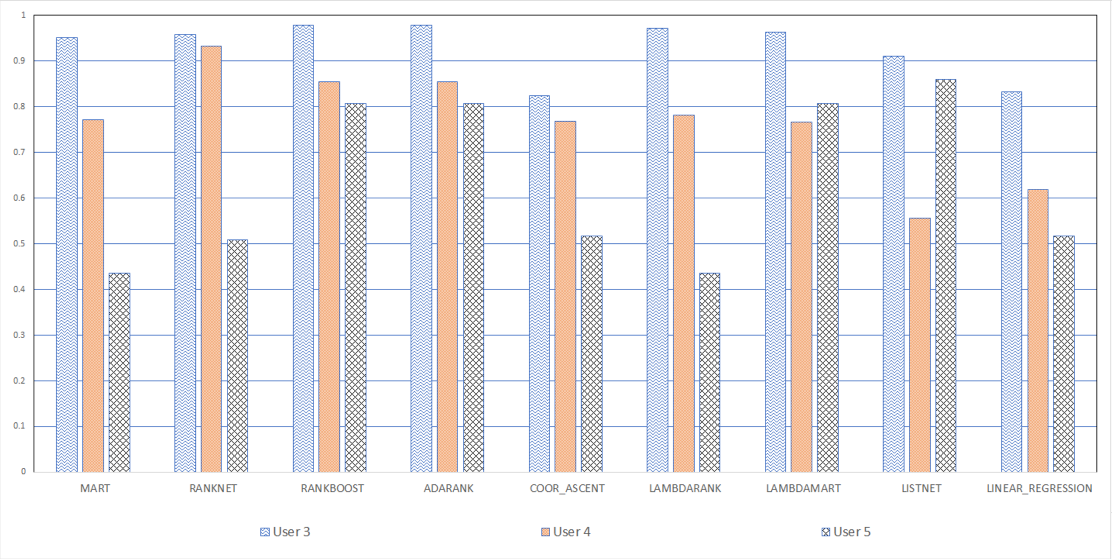

Figure 4.3: NDCG score of 3 active users

algorithm, and from this figure we learned that algorithm RankBoost and AdaRank have the best average result and LambdaRank has the worst average result. In Figure 4.3, we compared 3 different users with 8 different learning to rank algorithms to show how different users react to each learning to rank algorithm. From it, we learned that there does not exist one learning to rank algorithm that yields the best result for all active users. In fact, ListNet algorithm works best for user 6 but is the worst algorithm for user 5. Thus, we propose an independent learning to rank model for each active user, so the best recommender model would be chosen for each user. Table 4.4 summarizes the best learning model for each user along with their NDCG scores.

Table 4.4: Best personalized model output for each user

| User | Algorithm | score |
|---|---|---|
| 1 | Ranknet | 0.850 |
| 2 | Rankboost | 0.955 |
| 3 | Adarank | 0.978 |
| 4 | Ranknet | 0.932 |
| 5 | Lambdarank | 0.806 |
| 6 | Rankboost | 0.918 |
| 7 | Adarank | 0.964 |

From Table 4.4 we notice that each user may be suitable with different learning algorithm and there is not a single learning algorithm that proves to be the best for all

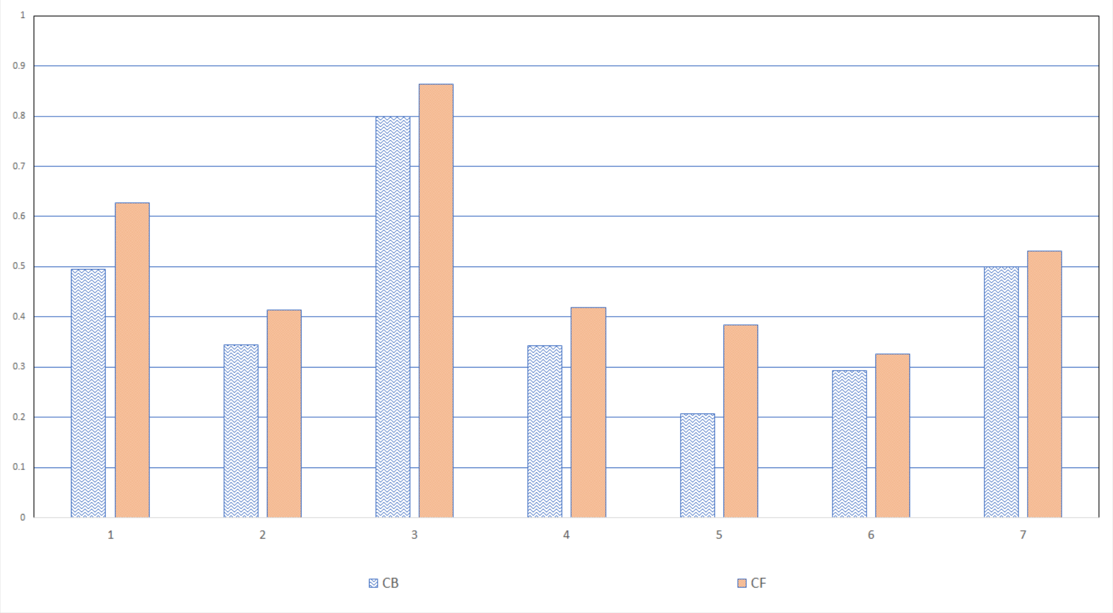

Figure 4.4: NDCG scores with one feature

active users. Therefore, in our personalized model we choose the best learning model that for each individual and store those models for future recommendations. An individualized model might be proven to be space consuming but in order to reward a loyal and active customer, we think it is a fair trade-off. Due to advancement in technologies, space resources are more affordable in recent times whereas feedback from loyal customers are priceless.

In the following results, we try to demonstrate the importance of different features such as Content-based filtering, Collaborative filtering, Social relation strength, Spatial, Temporal and Domain interest. We use various combination of features to find the perfect combination to be used in our recommender system. We used the feedback from those 7 active users to show the NDCG score. In the following graphs x-axis represents the user and y-axis represent the score.

Based on Figure 4.2, we established that RankBoost has the best average for learning to rank algorithm. In Figure 4.4, we compared RankBoost ranking algorithm with only Content-based input feature against Collaborative input feature. These test cases is to verify the importance of Content-based Filtering and Collaborative Filtering feature. Although the accuracy with only one feature is fairly low for both individual feature, we noticed that Collaborative Filtering feature model works slightly better than Content-based Filtering feature model.

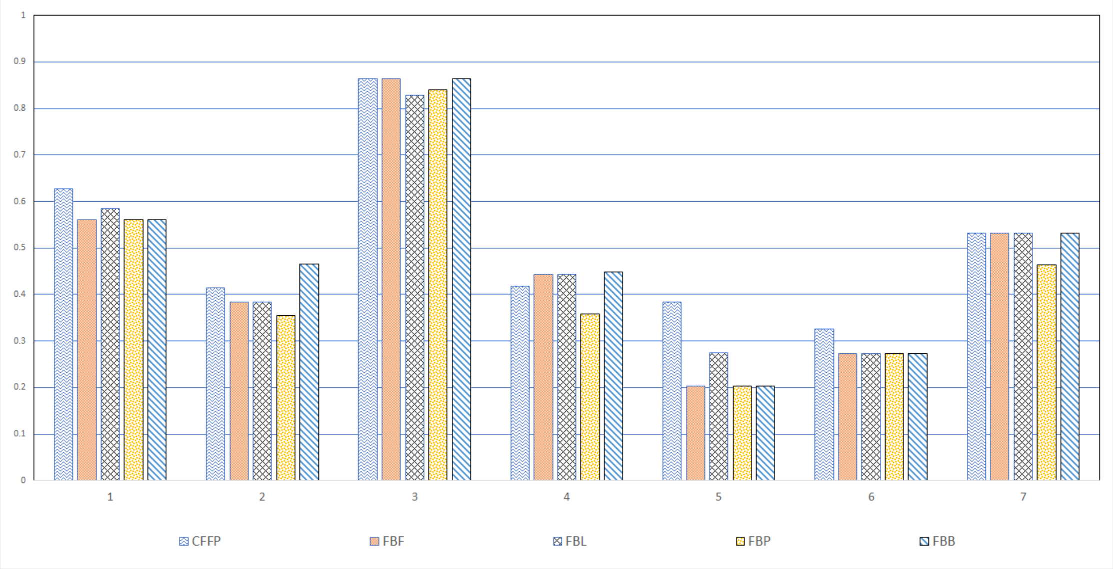

Figure 4.5: NDCG scores with various combination of features

In Figure 4.5, we used various combination of features and grouped them by user to identify which combination of features works best for different users. In "CFFP" model, we uses only Collaborative filtering and Social relation strength as our input features. This combination is to test user's interest level on friends and similar users' posts. According to the result, we can verify that 3 out of 7 users find this model that best describing their interest.

In Figure 4.5, we also tried out different feature along with a basic hybrid model of Content-based and Collaborative feature. The "FBF" model contains the basic hybrid model and Social relation feature, "FBL" model contains the basic hybrid model and Spatial feature, "FBP" model contains the basic hybrid model and Popularity feature, and "FBB" model contains the basic hybrid model and Vehicle feature. From these results, we noticed that any of these combinations features perform better than the previous single feature models. Also we noticed that each user correspond to each feature differently.

In Figure 4.6, we try to pinpoint the effect of each feature has on different users. From these results we could confirm that each user has different preferences, and each feature has different impact on accuracy. From this test group, the most significant feature would be Collaborative and the least important feature would be Content-based. Even though the accuracy increased is small with every added feature, at the end when we included all features the accuracy is satisfying.

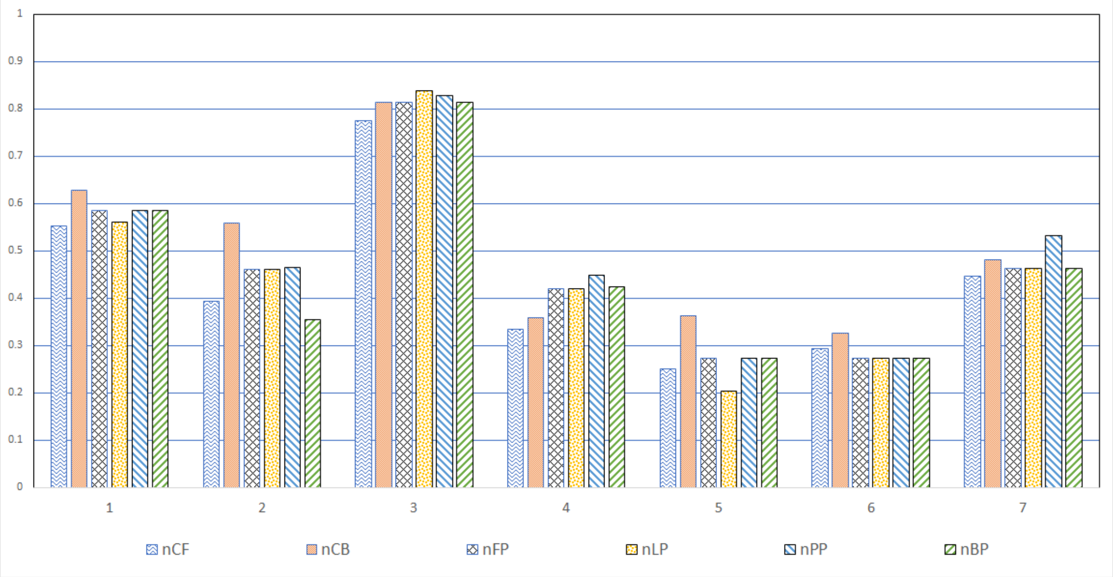

Figure 4.6: NDCG scores with leaving out one feature

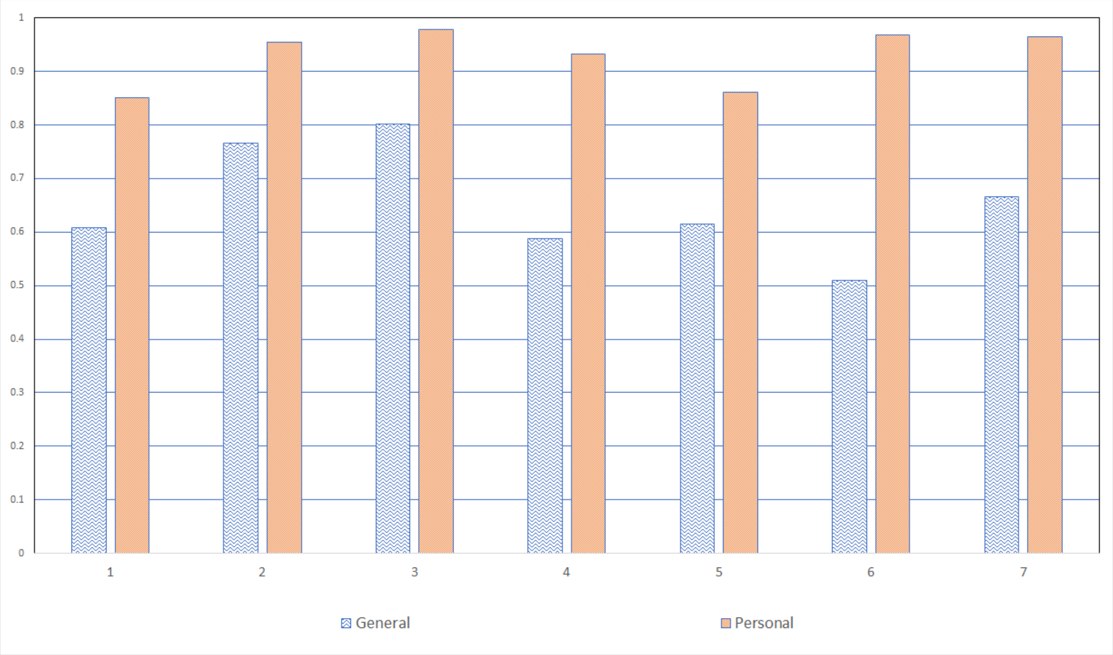

Figure 4.7: Comparing General Model with Personal model

In Figure 4.7, we compared generalized model against personalized model with 7 active users. From the figure, we determine that personalized model out performed generalized model for all 7 active users. It comes at no surprise that personalized model are more accurate because we are able to tweak this model such that it tailors to its' user's preference. Therefore, we recommend using personalized model for active users and generalized model for new users.

## 4.3 Summary

In this chapter we explained the dataset used in our experiment and our experiment design with detailed explanation of our recommender system's implementation. After analyzing and evaluating our results, we showed that our proposed recommender system performs better than the conventional system. Based on the figures shown above, our approach is better than the conventional approach. We can see that our Normalized Discounted Cumulative Gain (NDCG) score obtained through learning to rank algorithm is better than the NDCG score using individual feature alone. In short, a composite of many strategies is better than using a single strategy, because every strategy has their own strengths and weaknesses. By combining them, we could minimize the flaws of one strategy by introducing other strategy's strengths that make up for those losses. Thus, in this paper we show that using historical data to learn separate recommender models and then combining them with learning to rank algorithm would generate a better recommender system. Also, active users would have personalized model that tailors to his or her changing interests; whereas new users or less active users would have a generalized model that sums up the community interest.

## 4.4 Threats to Validity

As mentioned in previous chapter, the level of sparcity that we are working on is $6.425 \times 10^{-6}$. Due to the limitation of explicit feedback from users, we are only able to test our system against 7 active users for personalized models. However, based on the limited resources available, we are able to produce promising results. For our training and testing data split, we did 4 to 1 split. Using 80% of our data for training and the remaining 20% data for testing. Unfortunately, this test-train split is the best approach we could provide because of the small data-set we are working on. The model we proposed is tailored to the Social Networking Sites (SNS) that we are working with, so this approach may not work well with other SNS.

# Chapter 5

# Conclusion

## 5.1 Conclusions

In this thesis, we proposed a generalized recommender system for novice users and a personalized recommender system for active users to recommend posts to read on an online Social Networking Sites (SNS). Our system uses user's historical data and feedback to extract user's interest and create user profile. The recommender system uses six input features to compute scores that represent the importance of recommended post to user.

We use six input features including content-based, collaborative, social relation strength, spatial information, popularity and vehicle information (model and name). Each input feature provides candidate recommendation list with associated score for each post in descending order. Since each feature calculates posts' score based on their individual algorithm, each of them produces different order of posts ranking. Thus, in order to join all features together and re-order posts, we used learning to rank algorithms. From the results, we choose the best learning algorithm for both generalized and personalized model independently.

We proposed to have both generalized and personalized recommender model for the site because not all users have enough information to create their own personalized model. We use generalized model for novice users or users who have less activities on the site. For this type of users, we recommend posts based on the collective information of all users. When training for generalized model we use the traditional approach of recommender system of viewing the community as a whole and develop a recommender system that suits the majority of users. This model would recommend the community's collected interests to them.

A generalized model may work well for some users, but the same model may not work well for all users. Thus, we propose in introducing a personalized recommender model for all active users. With enough user's feedback and activity history, we could generate an accurate user profile and in turn personalize a recommender system for that particular user. For each personalized model, we first create user's content profile, interaction profile, social strength profile and vehicle profile. After all these profiles are created, we combine them with other input features such as spatial and popularity and use learning to rank algorithms to generate their personalized model.

The effectiveness of our proposed system is illustrated by the results in our experiment. We have shown how various input features and learning to rank algorithms have impacted the model. For both generalized and personalized model we have shown that all six input features are essential for our recommender system. Also, we tried a list of popular learning to rank algorithms to combine these six input feature and choose the best ranking algorithm to generate the optimal recommender system.

The major contributions of our research are:

1. We proposed a hybrid recommender system that combines Collaborative Filtering algorithm, Content-based algorithm, Social Relation Strength, Spatial, Popularity and Vehicle information to create an efficient and accurate recommendation model. None of the prior recommender system has combined all the above input features.

2. We proposed to use learning to rank algorithms to build a generalized model for novice users and a personalized model for each active user. With the inclusion of both model our system could help alleviate the cold-start problem.

## 5.2 Future Work

We would like to continue working on our recommender system to implement incremental learning to increase efficiency in updating the model. We would like to also implement incremental learning in content-based model while creating user's content profile; also incremental learning in collaborative model. With implementation of incremental learning, we could shorten the time needed to update user profile thus enhancing its response time.

Another direction is to implement more input features into our model. Currently, we are using six input features and our recommender model is flexible to add more input features in the future. In addition of new features, we would also like to introduce new ranking algorithms that could improve the ranking order of candidate posts. Therefore, we would like to find other learning to rank algorithms that may increase accuracy and speed in ranking.

# Appendix A

# Personalized Model Result

In Figures A.1, A.2, A.3, A.4, A.5, A.6, A.7, we show how different learning to rank algorithms fare with each user.

In Figure A.1, the best learning algorithm is Ranknet and the worst learning algorithm is Listnet for user 1.

In Figure A.2, the best learning algorithm is Rankboost and the worst learning algorithm is Listnet for user 2.

In Figure A.3, the best learning algorithm are Rankboost and Adarank. The worst learning algorithm is CoorAscent for user 3. In case of similar Normalized Discounted Cumulative Gain (NDCG) scores, we pick a model at random to provide diversity and for this experiment we chose Adarank.

In Figure A.4, the best learning algorithm is Ranknet and the worst learning algorithm is Listnet for user 4.

In Figure A.5, the best learning algorithm is Rankboost, Adarank, Lambdamart and listnet. The worst learning algorithm is Mart for user 5.

In Figure A.6, the best learning algorithm is Rankboost, and Adarank. The worst learning algorithm is Lambdarank for user 6.

In Figure A.7, the best learning algorithm is Rankboost, and Adarank. The worst learning algorithm is lambdamart for user 7.

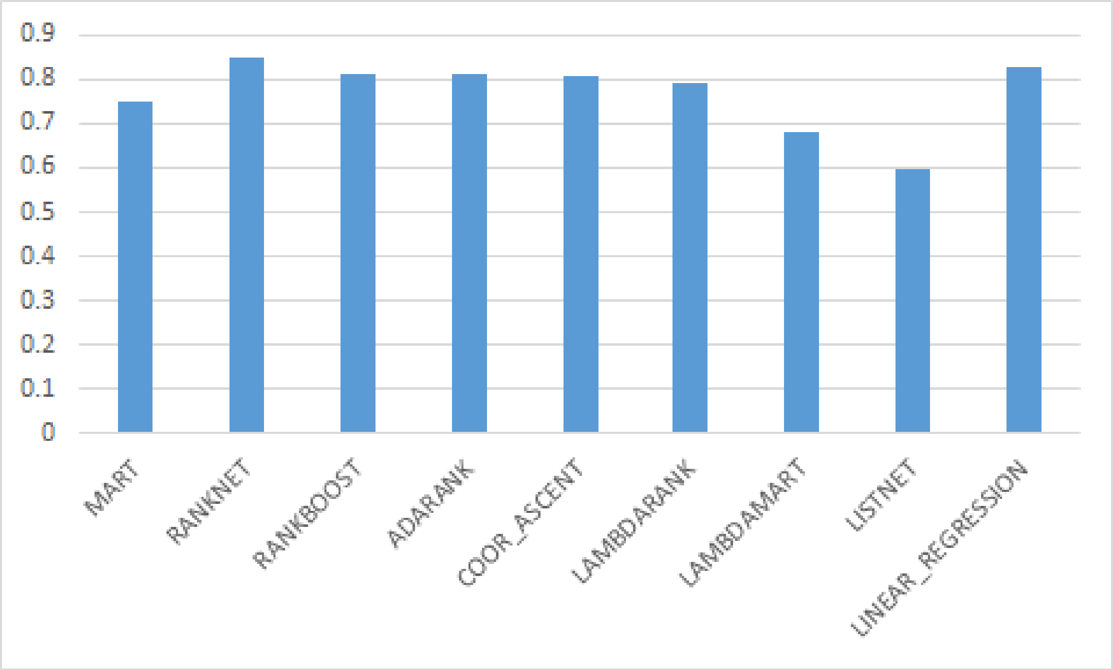

Figure A.1: NDCG scores for user 1

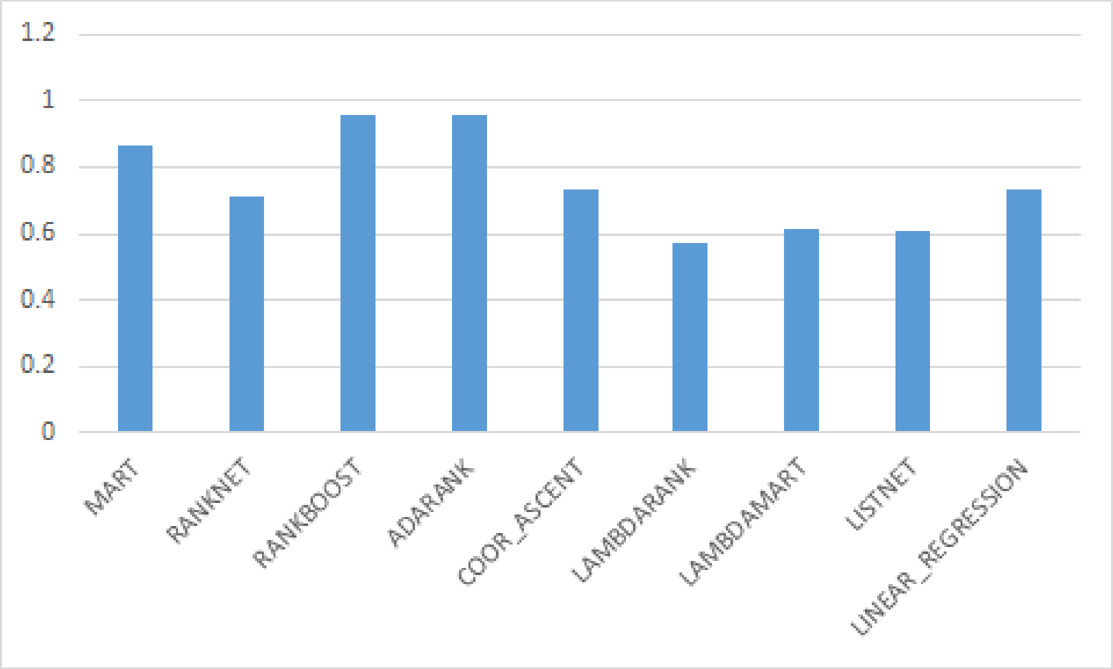

Figure A.2: NDCG scores for user 2

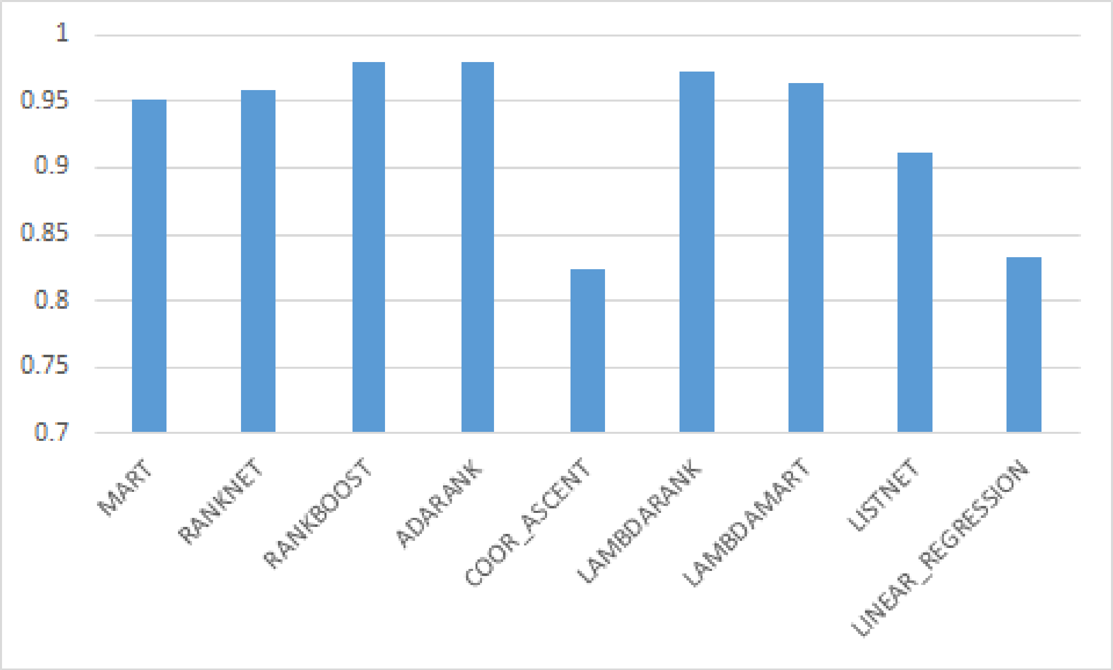

Figure A.3: NDCG scores for user 3

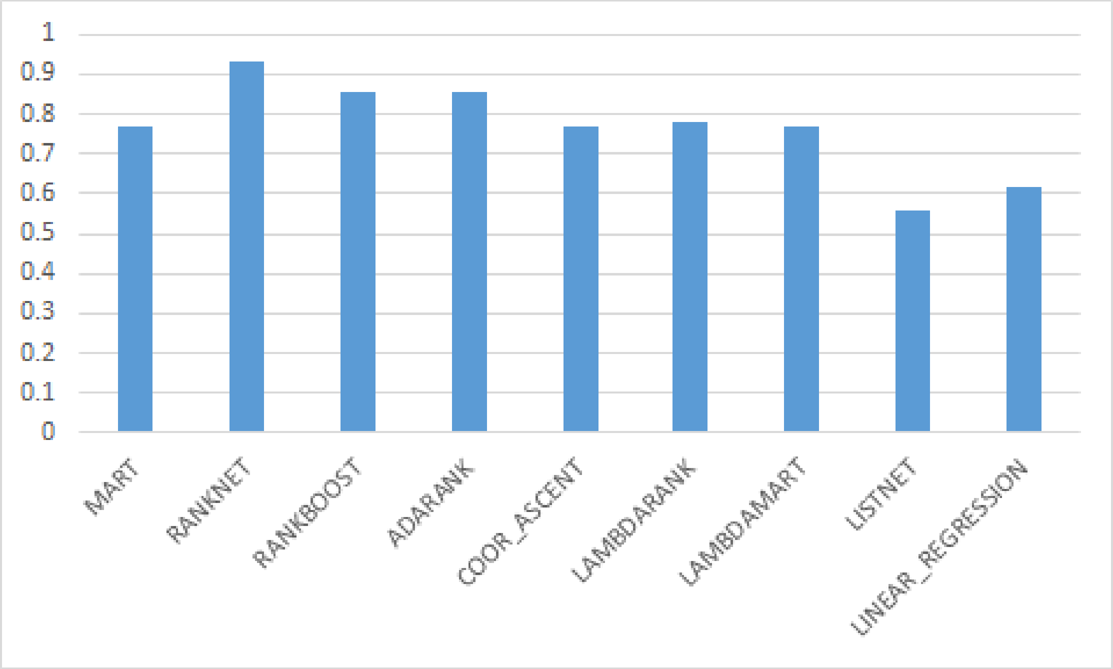

Figure A.4: NDCG scores for user 4

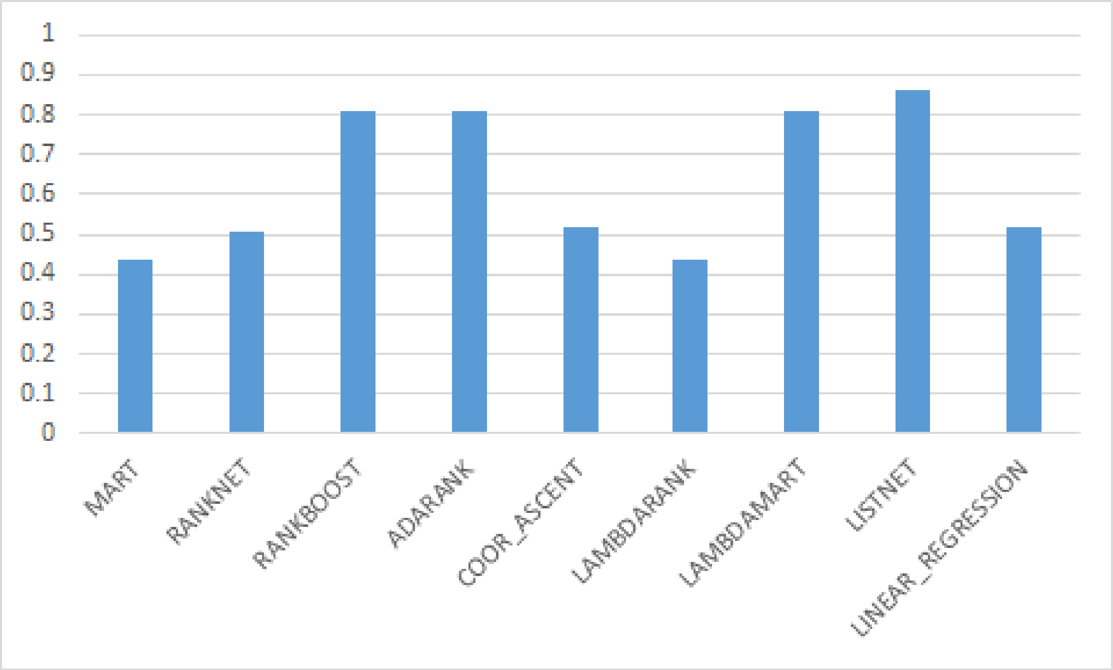

Figure A.5: NDCG scores for user 5

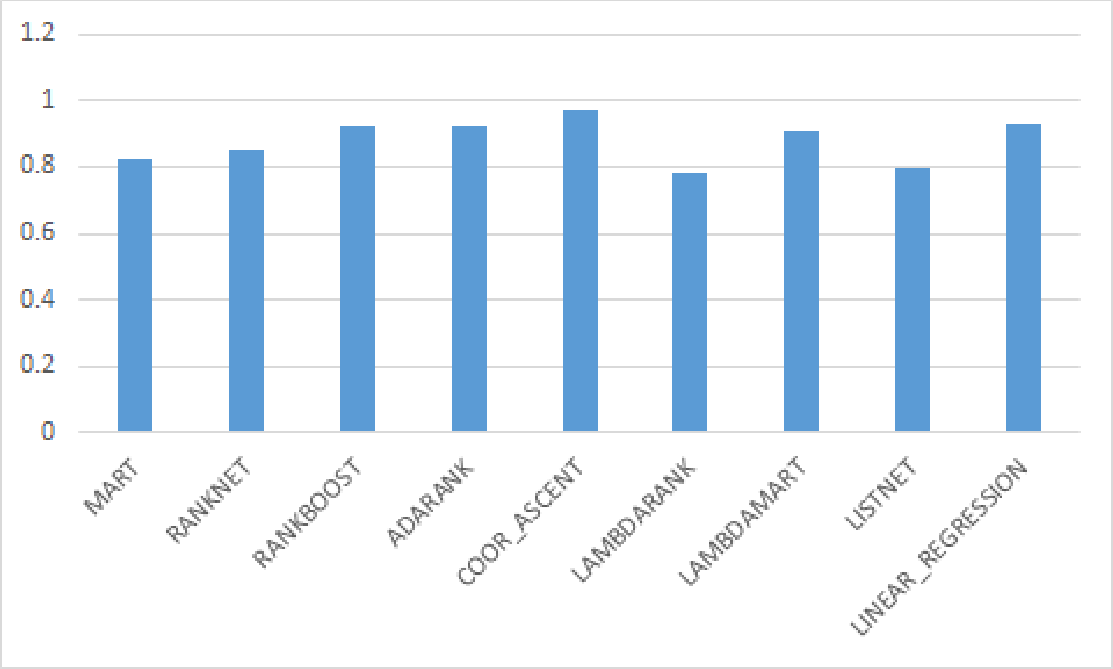

Figure A.6: NDCG scores for user 6

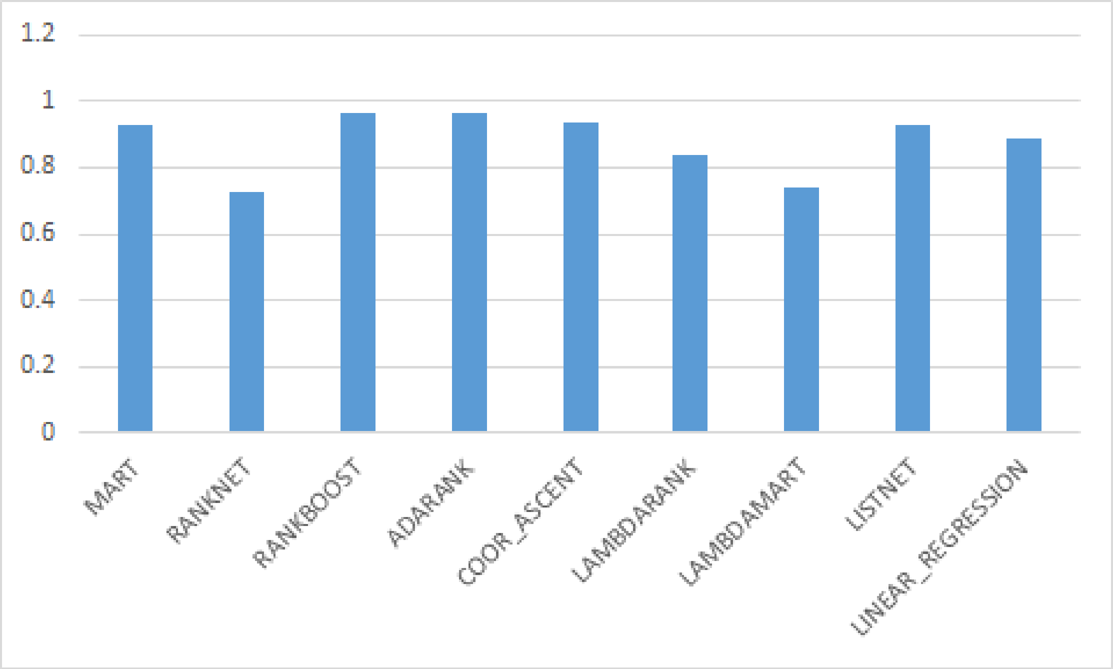

Figure A.7: NDCG scores for user 7

# Bibliography

[1] The lemur Project wiki - ranklib. `https://sourceforge.net/p/lemur/wiki/RankLib/`.

[2] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734–749, June 2005.

[3] F. Amato, V. Moscato, A. Picariello, and G. Sperl. Recommendation in social media networks. In *Proceedings of 2017 IEEE Third International Conference on Multimedia Big Data (BigMM)*, pages 213–216, April 2017.

[4] F. Ayala-Gómez, B. Daróczy, M. Mathioudakis, A. Benczúr, and A. Gionis. Where could we go?: Recommendations for groups in location-based social networks. In *Proceedings of the 2017 ACM on Web Science Conference*, WebSci '17, pages 93–102, New York, NY, USA, 2017. ACM.

[5] P. Bhargava, T. Phan, J. Zhou, and J. Lee. Who, what, when, and where: Multi-dimensional collaborative recommendations using tensor factorization on sparse user-generated data. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 130–140, Republic and Canton of Geneva, Switzerland, 2015. International World Wide Web Conferences Steering Committee.

[6] P. G. Campos, F. Díez, and I. Cantador. Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols. *User Model. User-Adapt. Interact.*, 24(1-2):67–119, 2014.

[7] V. R. Carvalho, J. L. Elsas, W. W. Cohen, and J. G. Carbonell. A meta-learning approach for robust rank learning. In *SIGIR 2008 workshop on learning to rank for information retrieval*, volume 1, 2008.

[8] P. Chakraborty. A scalable collaborative filtering based recommender system using incremental clustering. In *Proceedings of Advance Computing Conference, 2009. IACC 2009. IEEE International*, pages 1526–1529, March 2009.

[9] K. Chen, T. Chen, G. Zheng, O. Jin, E. Yao, and Y. Yu. Collaborative personalized tweet recommendation. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, pages 661–670, New York, NY, USA, 2012. ACM.

[10] W.-Y. Chen, J.-C. Chu, J. Luan, H. Bai, Y. Wang, and E. Y. Chang. Collaborative filtering for orkut communities: Discovery of user latent behavior. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 681–690, New York, NY, USA, 2009. ACM.

[11] X. Dong, X. Chen, Y. Guan, Z. Yu, and S. Li. An overview of learning to rank for information retrieval. In *2009 WRI World Congress on Computer Science and Information Engineering*, volume 3, pages 600–606, March 2009.

[12] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins. Propagation of trust and distrust. In *Proceedings of the 13th International Conference on World Wide Web*, WWW '04, pages 403–412, New York, NY, USA, 2004. ACM.

[13] P. Heymann, G. Koutrika, and H. Garcia-Molina. Can social bookmarking improve web search? In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, WSDM '08, pages 195–206, New York, NY, USA, 2008. ACM.

[14] L. Hong, A. S. Doumith, and B. D. Davison. Co-factorization machines: Modeling user interests and predicting individual decisions in twitter. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, WSDM '13, pages 557–566, New York, NY, USA, 2013. ACM.

[15] C. Hsiao, Z. Wang, and W. Teng. An incremental scheme for large-scale social-based recommender systems. In *Proceedings of International Conference on Data Science and Advanced Analytics, DSAA 2014, Shanghai, China, October 30 - November 1, 2014*, pages 128–134, 2014.

[16] H. Imran, M. Belghis-Zadeh, T.-W. Chang, Kinshuk, and S. Graf. Plors: a personalized learning object recommender system. *Vietnam Journal of Computer Science*, 3(1):3–13, Feb 2016.

[17] C. Jensen, J. Davis, and S. Farnham. Finding others online: Reputation systems for social online spaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '02, pages 447–454, New York, NY, USA, 2002. ACM.

[18] T. Joachims. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. Technical report, DTIC Document, 1996.

[19] Y. 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.

[20] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, Aug. 2009.

[21] T.-Y. Liu. *Learning to rank for information retrieval.* Springer Berlin Heidelberg, 2011.

[22] X. Liu and K. Aberer. Towards a dynamic top-n recommendation framework. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 217–224, New York, NY, USA, 2014. ACM.

[23] H. P. Luhn. A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of research and development*, 1(4):309–317, 1957.

[24] X. Luo, Y. Xia, and Q. Zhu. Incremental collaborative filtering recommender based on regularized matrix factorization. *Knowledge-Based Systems*, 27(0):271 – 280, 2012.

[25] H. Ma. An experimental study on implicit social recommendation. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '13, pages 73–82, New York, NY, USA, 2013. ACM.

[26] H. Ma, I. King, and M. R. Lyu. Learning to recommend with social trust ensemble. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 203–210, New York, NY, USA, 2009. ACM.

[27] A. Majid, L. Chen, G. Chen, H. T. Mirza, I. Hussain, and J. Woodward. A context-aware personalized travel recommendation system based on geotagged social media data mining. *International Journal of Geographical Information Science*, 27(4):662–684, 2013.

[28] J. L. Neto, A. D. Santos, C. A. Kaestner, N. Alexandre, D. Santos, C. A. A, K. Alex, A. A. Freitas, and C. Parana. Document clustering and text summarization, 2000.

[29] A. Paterek. Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD cup and workshop*, volume 2007, pages 5–8, 2007.

[30] X. Qian, H. Feng, G. Zhao, and T. Mei. Personalized recommendation combining user interest and social circle. *IEEE Transactions on Knowledge and Data Engineering*, 26(7):1763–1777, July 2014.

[31] M. Reshma and R. R. Pillai. Semantic based trust recommendation system for social networks using virtual groups. In *Proceedings of 2016 International Conference on Next Generation Intelligent Systems (ICNGIS)*, pages 1–6, Sept 2016.

[32] B. R.-N. Ricardo Baeza-Yates. *Modern Information Retrieva - the concepts and technology behind search*. Addison Wesley, 2011.

[33] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

[34] M. Sarwat, J. J. Levandoski, A. Eldawy, and M. F. Mokbel. Lars: An efficient and scalable location-aware recommender system. *IEEE Transactions on Knowledge and Data Engineering*, 26(6):1384–1399, June 2014.

[35] G. Takács, I. Pilászy, B. Németh, and D. Tikk. Major components of the gravity recommendation system. *ACM SIGKDD Explorations Newsletter*, 9(2):80–83, 2007.

[36] T. Tokunaga and I. Makoto. Text categorization based on weighted inverse document frequency. In *Special Interest Groups and Information Process Society of Japan (SIG-IPSJ*, pages 33–39, 1994.

[37] I. H. Witten, A. Moffat, and T. C. Bell. *Managing gigabytes: compressing and indexing documents and images*. Morgan Kaufmann Publishers, 2nd edition, 1999.

[38] F. Xia, T.-Y. Liu, J. Wang, W. Zhang, and H. Li. Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th international conference on Machine learning*, pages 1192–1199. ACM, 2008.

[39] L. Xiong, X. Chen, T. Huang, J. G. Schneider, and J. G. Carbonell. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *Proceedings of the SIAM International Conference on Data Mining, SDM 2010, April 29 - May 1, 2010, Columbus, Ohio, USA*, pages 211–222, 2010.

[40] J. Xu and H. Li. Adarank: A boosting algorithm for information retrieval. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 391–398, New York, NY, USA, 2007. ACM.

[41] X. Yang, Z. Zhang, and K. Wang. Scalable collaborative filtering using incremental update and local link prediction. In *Proceedings of 21st ACM International Conference on Information and Knowledge Management, CIKM'12, Maui, HI, USA, October 29 - November 02, 2012*, pages 2371–2374, 2012.

[42] H. Yin, B. Cui, L. Chen, Z. Hu, and X. Zhou. Dynamic user modeling in social media systems. *ACM Trans. Inf. Syst.*, 33(3):10:1–10:44, Mar. 2015.

[43] H. Yin, B. Cui, Y. Sun, Z. Hu, and L. Chen. Lcars: A spatial item recommender system. *ACM Trans. Inf. Syst.*, 32(3):11:1–11:37, July 2014.

[44] X. Yu, X. Ren, Y. Sun, Q. Gu, B. Sturt, U. Khandelwal, B. Norick, and J. Han. Personalized entity recommendation: A heterogeneous information network approach. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, WSDM '14, pages 283–292, New York, NY, USA, 2014. ACM.

[45] Q. Yuan, G. Cong, Z. Ma, A. Sun, and N. M. Thalmann. Time-aware point-of-interest recommendation. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '13, pages 363–372, New York, NY, USA, 2013. ACM.

[46] Q. Yuan, G. Cong, Z. Ma, A. Sun, and N. M. Thalmann. Who, where, when and what: Discover spatio-temporal topics for twitter users. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 605–613, New York, NY, USA, 2013. ACM.

[47] Q. Yuan, G. Cong, K. Zhao, Z. Ma, and A. Sun. Who, where, when, and what: A nonparametric bayesian approach to context-aware recommendation and search for twitter users. *ACM Trans. Inf. Syst.*, 33(1):2:1–2:33, Feb. 2015.

[48] W. Zhang, J. Wang, B. Chen, and X. Zhao. To personalize or not: A risk management perspective. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pages 229–236, New York, NY, USA, 2013. ACM.

[49] X. Zhou, J. He, G. Huang, and Y. Zhang. Svd-based incremental approaches for recommender systems. *J. Comput. Syst. Sci.*, 81(4):717–733, 2015.

[50] X. Zhou, Y. Xu, Y. Li, A. Josang, and C. Cox. The state-of-the-art in personalized recommender systems for social networking. *Artificial Intelligence Review*, 37(2):119–132, 2012.

[51] C.-N. Zieglera and J. Golbeck. Investigating correlations of trust and interest similarity-do birds of a feather really flock together? *Decision Support Systems*, 2006.

# Glossary

**ALS** Alternating Least Square

**CB** Content-Based Filtering

**CF** Collaborative Filtering

**CG** Cumulated Gain

**DCG** Discounted Cumulated Gain

**EBSN** Event-based Social Networking Services

**LBSN** Location-based Social Network

**NDCG** Normalized Discounted Cumulative Gain

**RS** Recommender System.

**SNS** Social Networking Sites

**UGC** User Generated Content

**WWW** World Wide Web