UTILIZING USER FEEDBACK DATA FOR CONTENT RECOMMENDATIONS

by

Sneha

B. Tech, Rajasthan Technical University, 2012

M. Tech, The NorthCap University, 2014

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, 2019

AUTHOR'S DECLARATION FOR ELECTRONIC SUBMISSION OF A THESIS

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.

SNEHA

UTILIZING USER FEEDBACK DATA FOR CONTENT RECOMMENDATIONS

Sneha
Master of Science in Computer Science, 2019
Ryerson University, Toronto, Canada.

ABSTRACT

Nowadays, Internet contains massive amount of information. In this environment, people who seek specific information could be overwhelmed by the options that they can reach through the Internet. To help users filter the information and overcome the information overload problem, recommender systems play an important role. Here, we deal with a specific recommendation problem – recommending content to users in a content management system utilizing users' feedback data. We have tried both content-based and collaborative filtering approaches. In the content-based approach, once the content profile is built, user profile could be built based on different categories of user feedback data. We have explored the effect of these different feedback categories on the recommendation result. In the collaborative filtering approach, the feedback data is used for building the user-content rating matrix and matrix factorization is then applied. The experiment result shows that content-based approach outperforms collaborative filtering approach for this particular problem.

ACKNOWLEDGEMENTS

I am sure that it would not have been possible to complete this thesis and research work without the continuous support and guidance of the following people. I really appreciate their efforts and have my deepest regards for their help.

I would like to express my sincere and deep sense of gratitude to my supervisor **Dr. Cherie Ding,** Professor, Department of Computer Science for introducing me to the Data Science field. I am very grateful for her support, dedication, inspiring guidance, availability to help and continuous advice. She offered the environment to develop my ideas. Her feedback was very helpful to improve this thesis.

I would like to thank all the thesis committee members, especially **Dr. Abdolreza Abhari** and **Dr. Qinmin (Vivian) Hu** for taking the time to review my thesis and for their valuable feedback.

I would like to thank all the administrative and technical members of the Department of Computer Science for their cooperation, help and for giving me access to additional resources required for my research.

I would also like to thank our collaborating company for providing their CMS dataset to perform this experiment and the research work.

Finally, I would like to express my deepest appreciation to my family and my best friend, without their personal and emotional support this endeavor would not have been possible. I am grateful for their constant encouragement, unlimited patience and all that support they gave during the completion of this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF ABBREVIATIONS

RS	Recommender Systems
CB	Content-Based
CF	Collaborative Filtering
LDA	Latent Dirichlet Allocation
TF-IDF	Term Frequency-Inverse Document Frequency
MF	Matrix Factorization
CF-RS	Collaborative Filtering- Recommender Systems
CF-MF	Collaborative Filtering- Matrix Factorization

CHAPTER 1

INTRODUCTION

The invention of Internet and World Wide Web has made information publishing and sharing much easier. People can use computers, smart phones and laptops to access any kind of information published on the web to find books to read, best places to visit, movies to watch and songs to listen to. The immense volume of information accessible online cause the information overload problem. It becomes a challenge for people to efficiently and accurately find information they are truly interested in. Recommender systems provide a solution to alleviate this problem and facilitate the decision-making process involving online information search. Recommender systems can be defined as techniques and tools which are used to recommend items to the users, where item is the general term to describe things (such as news, article, movie, music) [4].

Recommender systems are powerful software tools which help people in a personalized way to find suitable items from huge number of alternatives [6]. As users need personalized recommendations to deal with the problem of information overload, there are many recommendation techniques which are developed for recommending items to users. The three main recommendation techniques which are used in the literature to generate the recommendations are content-based approach, collaborative filtering approach and knowledge-based approach. Hybrid approach is also used to generate recommendations which combines various different approaches to provide recommendations.

In order to generate these recommendations, the user interactions can be used by the recommender systems. The recommendation techniques use user's implicit or explicit

interactions for recommending relevant items to that user. Any form of interaction between user and the item can be defined as user/item interaction.

In the next sections, first we define the problem that we are going to solve in this research. Second, we explain our goals and objectives of this research. Third, we describe the overall methodology followed throughout this research. Lastly, we list the structure of this thesis.

1.1 Problem Statement

In this research, we are considering a company's CMS (Content Management System), which provides a central repository for their employees to publish, share and access content. In this CMS dataset, all the contents (posts) are saved, as well as the user interaction data. And our main focus is to recommend content to users according to their past interaction histories. CMS is a software or a set of related programs that facilitates creating, organizing, editing, sharing and publishing contents. CMS is typically used for two main purposes: enterprise content management and web content management. It has two main components: a content management application and content delivery application. The content management application is a graphical user interface which allows users to create, modify, publish, share or remove content from the website. The content delivery application offers backend functionality that supports delivery of the content once it is created in the content management application. The core features of CMS are indexing, searching, retrieving and publishing the contents.

In this thesis, we are considering a problem of this CMS recommending content to their users. The CMS faces the issue of developing intelligent and personalized content recommendations to the individual users, so that they can more effectively access and make use of relevant content. The current CMS offers a central place for users to search relevant content.

However, it does not take into consideration that users may have different preferences and patterns when accessing, consuming and sharing content.

Therefore, the challenge is to utilize these users' interactions in an effective way, find an optimal method to integrate them into a recommender system, and then recommend the most relevant content to individual users based on their historical interactions with the content. We will explore two ways of utilizing users' interaction data for recommendation purpose, one is to build user profile for individual users using their feedback scores and then use content-based filtering approach to generate recommendations, and the other is to use feedback scores as ratings and then use collaborative filtering approach for generating recommendations. Also, we need to investigate user interactions and recommendation approaches in detail for generating best possible recommendations to the individual user.

1.2 Goals and Research Objectives

The major goals for this research are: 1) to utilize user feedback data to model user profile and study methods for content-based filtering approach for recommendation; 2) study and generate recommendations based on collaborative filtering approach; 3) compare and evaluate results of recommendations from both approaches. To achieve these goals, we have defined five major research objectives as follows.

- RO1: Study different approaches to model content profiles

In this research objective, we will investigate what kind of data we can extract from the dataset about the content (in the form of posts in CMS) that is published, read and shared by users over a period of time. We will study two algorithms for content profile modelling and then we will implement these algorithms to model the posts. We will generate content profiles using

content of the posts. The algorithms will be implemented to design two content profiles and then tested to see which one is more accurate for recommender system.

- RO2: Study user interactions and utilize user feedback data from the dataset to generate user profiles

In this research objective, we will explore users, their previous actions and their interaction patterns with the posts. Individual users have their own preferences while interacting with the posts. We will calculate an implicit feedback score based on user's historic interaction data and then we will build individual user profile using the user interaction records with the posts.

- RO3: Generate content recommendations using content-based filtering approach

In this research objective, we will find similarity between the content profile and user profile to generate content recommendations. We build content profiles using approaches studied in RO1 and user profiles using user feedback score as explained in RO2 and then we generate content recommendations using a similarity algorithm.

- RO4: Study collaborative filtering approach and generate content recommendations

In this research objective, we will study collaborative filtering approach and then we will build a collaborative filtering model for content recommendations. For this model, we will use users' feedback score as the rating data and a collaborative filtering algorithm to generate content recommendations. These recommendations are based on feedback score only.

- RO5: Evaluate and compare content-based filtering recommendations with collaborative filtering recommendations.

In this research objective, we will investigate the effectiveness (accuracy) of these two approaches on the dataset using performance evaluation metrics and compare the content-based recommendations with collaborative filtering recommendations.

1.3 Proposed Approach

In this thesis, we propose to generate content recommendations using content-based filtering approach and collaborative filtering approach for effective content suggestion to the individual user in the CMS. The methodology we have followed involves five major stages which are described below.

The first stage is to model content profiles. The potential solution for this problem is to use bag-of-words models such as Term Frequency-Inverse Document Frequency (TF-IDF) and Latent Dirichlet Allocation (LDA) to build content profiles for the individual post. In order to do so, first we need to identify an effective way to extract and pre-process the data. For data extraction, the posts (raw contents) are extracted and imported as an input file for our model and then we apply data pre-processing techniques on it. Later we implement TF-IDF and LDA models on pre-processed data to build content profiles. This task is done for each individual post. The key elements delivered by this analysis are: discovering keywords and weighted term vectors for each post by using TF-IDF algorithm; discovering keywords and underlying topics presented in the content with their scores for each post by using LDA algorithm; content profiles.

The second stage is to generate individual user profile based on user interactions with the posts. In this stage, we extract users' interactions from the dataset. The user interactions are interactions made by users with the posts, such as like, share, comment, tweet and reply. After extracting user interactions, we classify these interactions into three major categories based on the nature of interaction. First category is *Direct Interaction.* It means that user/post interaction

happened in the CMS itself. Second category is user *Social Share*. It means that user shared posts from the CMS system to their social networking websites. Third category is *Reading Statistics*. It refers to the feedback score user provided to the content by taking a certain action, such as reading progress (reading action) and clickthrough (clicking action). All the interactions have already been assigned a weight by the CMS. We will use those weight values and we will calculate user feedback score for each user. Furthermore, we will compare results from each category with overall feedback score to evaluate which category influences the user behavior the most.

The third stage is to generate content recommendations using content-based filtering approach. In this stage, we use a similarity algorithm for profile matching. Once we have content profiles containing a vector score for each post which are generated in stage one and user profiles which are generated in stage two, we use cosine similarity algorithm to find similarity between these content profiles and user profiles for generating content recommendations based on their similarity scores.

The fourth stage is to generate content recommendations using collaborative filtering algorithm. In this stage, we will use collaborative filtering approach and user implicit feedback scores as rating data. For collaborative filtering model, we use matrix factorization algorithm to generate recommendations for individual users.

The fifth stage is to evaluate and compare content recommendations for individual users. In this stage, we evaluate recommendation results from both approaches: content-based filtering approaches (TF-IDF and LDA) and collaborative filtering approach (Matrix Factorization) using performance metrics including Precision, Recall and F1-score. Since our target is to provide best recommendations to the individual user, we compare our recommendation results from all

algorithms to identify the best recommendation for the user. Furthermore, we compare results by using different interaction categories which are defined in stage two with content-based filtering algorithms and collaborative filtering algorithm.

1.4 Structure of the Thesis

This section briefly summarizes all the chapters in this thesis. It provides an outline to the whole thesis.

CHAPTER 1: This chapter aimed to present a brief introduction to this thesis. It described problem statement, goals, research objectives and research methodology.

CHAPTER 2: This chapter reviews the work previously done by researchers in the field of recommender systems in general. Then it critically reviewed comparative analysis of available literature in this area. This constitutes the work previously done in generating user profile, recommender systems using content-based approach and collaborative filtering approach. A number of different recommendation strategies are reviewed.

CHAPTER 3: The main objective of this chapter is to explain the overall system architecture, the content profile generation, the user profile generation, collaborative filtering-based model and recommendation strategies used in our thesis in details.

CHAPTER 4: This chapter describes the experiment to evaluate our proposed recommendations. It gives the details of data collection, data preparation, algorithms used, score calculation and result analysis.

CHAPTER 5: Finally, this chapter concludes our thesis with a summary of our experimental results, contributions of this research and our future research directions.

CHAPTER 2

BACKGROUND AND RELATED WORK

This chapter describes the core concept and algorithms for recommendations that will form the background of this thesis. First, the formal introduction about the recommender systems, input data types and user profile modelling approaches are given. Second, the recommendation techniques like content-based filtering and collaborative filtering are explained. Third, algorithms for content profile modelling that have been used in this thesis are given. Finally, we review some of the recent work done in the area of recommender systems, which are closely related to our work.

2.1 Background

This section covers research related to recommender systems and the approaches that we have used in the thesis. We describe recommender systems in general, input data types for recommender systems, recommender systems and application domains, user modelling and different recommendation techniques.

2.1.1 Overview of Recommender Systems

Recommender systems have been introduced in early 1990s. They are software applications to provide suggestions for the user to solve a specific problem [1]. They help users to deal with intense information and have become an important research area of information retrieval [2].

Recommender systems provide a list of items according to the user profile. The items in this context refer to what recommender system recommends to the individual user. For example,

recommender systems can help user to decide what movies to watch, which music to listen to, where to eat, where to go for shopping and what items to buy. These recommendations are commonly made based on knowledge about the user and the item. The user profile saves the information about the users that could include features such as gender, age, income, address and marital status [3]. The item profile saves the information about the item that often include domain-specific features, for example, if the items are books, the profile of a book could include the title, genre, writer and release year of the books [3].

User's satisfaction degree on an item can be measured by the rating score given by the user. Initially, users provide the rating scores for the items which are already known to them. The recommender systems then use those previous rating scores of the seen items of the users to predict the unseen items for the users.

The output data type for a recommender system is a recommendations list or a list of scores of predictions [7]. The input data types of a recommender system [4, 5] are items, users and interactions. Their details are given below.

Items are the objects which are seen or purchased by the users and are recommended by the recommender systems to the users. Items which are used in the recommender systems can be any object such as books, songs, movies and news.

The person or customer who gets recommendations from recommender systems is known as user of that system. Recommender systems are oftentimes personalized systems therefore user information is the most important information that is used by recommender systems. Examples of user information include user's preference for the related items, information about the relation between users, trust levels between users [4], etc.

Interactions between users and items are known as transactions, for instance rating score given to an item by a user [5]. Recommender systems mainly use four types of transactions, namely unary ratings (for example, if user has read/seen/purchased the item), binary ratings (for example, user 'like' or 'dislike' the item), numerical ratings (for example, user rates an item on 1-10 scale or 1-5 stars) and ordinal ratings (for example, user rates an item as 'agree', 'strongly agree', 'neutral', 'disagree', 'strongly disagree') [9].

Recommender systems are mainly grouped under four application domains, namely e-commerce (for example, books, computers and cameras), content (for example, documents, posts, news and articles), entertainment (for example, movies, music and TV shows) and service (for example, consultation and traveling) [6, 7].

2.1.2 Recommender Systems and User Modelling

Recommender Systems are used to provide custom and user-oriented recommendations to their users. These systems use knowledge about user preferences to make these recommendations. To get this knowledge, recommenders collect user preferences and using those, they generate user models. These can be collected by two different means - Explicit User Modelling or Implicit User Modelling.

- **Explicit User Modelling**

This technique uses user input to create the User Models. Users provide the feedback [3] on items themselves in various forms such as:

- Ratings
- Like or Dislike
- Survey

- Review Comments

This type of feedback is more accurate and more dependable as users explicitly state their opinions towards the items, hence it is preferred by researchers over the implicit type. But due to its need of high user effort, it is less convenient for the users and thus leads to relatively fewer rating ratio per user. Another aspect of this type of user data collection is that these user ratings may also differ in their qualities. For example, the quality of a rating of an item might be affected by how big a choice the user made while deciding on this item. A user might rate a song or a movie with less scrutiny than a laptop computer or a holiday resort. According to a study by Amatriain et al. [10], extreme explicit ratings have a higher consistency than mild feedback.

- **Implicit User Modelling**

In this technique, the user model is created implicitly by the system. Rather than using explicit user feedback about items, the system interprets user behaviors [4, 5, 6] to determine user interest or feedback on a certain item. For example, if a user spends more time than a predefined threshold value on browsing or viewing an item, it implies that the user is interested in said item and this information can then be used by the recommender to build the user model [4, 5]. Or if a user purchases an item, it can be interpreted as a positive feedback on the item. Below is a list of sample user behaviors that could be used to get implicit feedback [5].

- Time Tracking
- Reading Progress
- Keyboard/Mouse inputs
- Clicks
- GPS Location
- Eye Tracking

- Microphone input
- Facial Expression

Since Implicit feedback does not require users to provide feedback, it is more convenient for the users though it might be less accurate than the ratings provided explicitly by the users.

2.1.3 Recommendation Approaches

Depending on the technique used to collect and estimate the user ratings, there are mainly two types of approaches for content recommendations [6]: content-based approach and collaborative filtering-based approach. Some hybrid approaches, which use a combination of both techniques, are in use as well. The approaches used in this thesis are described below.

Content-Based Approach

This type of approach outputs recommendation scores for items that are yet seen by the user based on their similarity with already seen items by the user [1] using the feedback provided on them. The unseen items with highest scores are recommended in this type of system. For example, while recommending a movie or a TV series, the recommender will look at patterns such as the common genre, the artists, the director or the production company in the movies or shows rated positively by the user and then recommend movies or TV series that have a high similarity on these characteristics [1].

The methodology of content-based approach is, at its core, related to information retrieval [11] and information filtering areas [4]. These content-based approach usually works with items that have textual information because of its heavy reliance on text-based applications in the field of information retrieval and filtering. Furthermore, these systems improve upon the traditional means of information retrieval and filtering by keeping user data in the form of user profiles.

Content-based approach provides recommendations for unseen items to users by basically comparing and matching the values for different attributes in the user profiles with the values for the unseen items by those users. These different attributes are simply a representation of the user preferences stored in the user profiles in the system [1]. Content-based recommendations generally use three components to perform the recommendations – content analyzer, profile learner and rating predictor [1, 7].

The two main algorithms used in this thesis to perform analysis of text-based items are TF-IDF (Term-Frequency-Inverse Document Frequency) and LDA (Latent Dirichlet Allocation). These are described in detail in below sections.

One of the main algorithms used in this thesis for content profile modelling is TF-IDF, which stands for Term Frequency-Inverse Document Frequency. Term frequency is a measure of how often a term appears in a document and inverse document frequency is a measure of how much information the term provides and how rare the term is throughout the document collection. Using these two functions, the TF-IDF weight can be computed. The TF-IDF weight is often used in information retrieval and filtering, this weight is a measure of how important the term is in the whole collection of documents or corpus [12]. A term that is common through the document as well as the entire corpus is not valued as high as a term that is common through the document but is seldom mentioned in the corpus. Thus, the TF-IDF weight can differentiate between important words and unimportant words. The terms with the highest TF-IDF weight in a document can collectively represent the main characteristic or the core of the document [12]. These can then be used to find similarity between different documents. The documents with similar high weighted words are considered similar.

Another main algorithm used in this thesis for content profile modelling is LDA, which stands for Latent Dirichlet Allocation [13]. TF-IDF is unable to provide significant decrease in description length and does not disclose much about the statistical structure between different documents or within the same document. LDA overcomes these limitations.

LDA is a generative probabilistic model that has often been used by researchers for topic modeling as well as word-sense disambiguation [14]. It has also been used in DNA research [15, 16, 17], and query search [18]. Hence LDA, similar to TF-IDF, is a widespread and flexible approach for performing information retrieval tasks.

LDA discovers the latent topics in the corpus by using the observable variables of words using Bayesian statistics [13]. In the model, a Dirichlet distribution is used, hence giving it the name LDA. It is similar to TF-IDF in the sense that both are 'a bag of words' models and therefore the positions of words do not matter in a document. When applied to document, LDA categorises hidden topics which can then be used to determine the similarity between different documents.

Collaborative Filtering Approach

This type of approach emphasizes on the fact that most people use recommendations received from others while making decisions for various activities [19, 20]. For example, before buying a book or watching a movie, many people look for reviews online or from their peers. This type of system provides recommendations using the fact that people who shared similar interests or opinions previously will share the same interests or opinions again [4].

The model predicts the satisfaction values of a user for an unseen item by using the previous ratings of that item provided by other users. To be precise, the satisfaction function $f(u,i)$ for a user u with respect to an item i is predicted using the output of the satisfaction function

$f(u_j,i)$ for other users who are similar to the current user, with respect to the same item i. Two users from the entire user set U are considered similar by comparing their respective rating histories. In this system, if two users rated similar items similarly then they are considered similar, also known as neighbors. Once the system determines the neighbors for a particular user, it outputs the items that are rated highly by those neighbors but are yet seen by the user as recommendations for that user.

The pure collaborative filtering approach only uses the rating information of other similar users to recommend items. It does not use any other information pertaining to the items. Hence, these types of recommenders do not care about the application domain and are able to recommend any kind of items e.g. music, book, videos [25].

One main algorithm used in this thesis for collaborative filtering is matrix factorization. For creating recommendations for items, model-based algorithms use a user-item rating matrix [21] for learning a predictive model. Initially some offline preprocessing is required by this category of recommenders to learn the model. Once the model is learned, recommendation scores are generated using these models [6]. In this thesis, the main category of model-based algorithms used is Matrix Factorization.

In Matrix Factorization or Latent Factor Model approach, latent factors are obtained from the rating history of users. The items and users are both converted into a factor vector representation. The similarities between these vectors are then used to calculate the similarities between users and items [26]. The Singular Value Decomposition (SVD) method is one common way to extract the latent factors [27]. SVD method calculates the user-item rating matrix's best lower rank estimate values [28]. It generates smaller vector for items by joining various items

that have a high co-relation and co-occurrence into a single factor. Hence, more accurate and relevant recommendations are produced by considering the related terms as a single latent factor.

2.2 Related Work

In this section, we review the related work done by researchers in this field of interest. We start with discussing some of the work done in modelling user profiles for recommendations. Then we go over related work on recommendations using content-based approach and collaborative filtering-based approach.

2.2.1 User Profile Modelling for Recommendations

Ouaftouh and Zellou [29] proposed a clustering-based approach for user profile generation to personalize information systems by offering recommendations. The authors of this paper mainly based their approach on a proposed user profile similarity function measure. They introduced user profile as collection of information characterizing the user, their preferences and their interaction environment with the system. To construct a user model, they selected various attributes such as interest, profession, gender, country, languages, postal code, age, etc. Using these they were able to create communities of user profiles by gathering the similar profiles, which constitutes the Clustering Phase of their approach. They equated the act of finding out the similarity between two user profiles to comparing the attributes values of each user's profile and figuring out how much they match. To compare the attributes, suitable similarity functions had to be utilized since not all the attributes can be compared in the same way. For example, the comparison of attributes like interests or profession would require a different approach as compared to attributes like age or postal code. In this paper, authors proposed a similarity function which would measure the similarities between two profiles as the weighted combination of similarities between their attributes as calculated by similarity metric. They used two different

approaches to calculate the similarities between attributes of different user profiles namely, syntactic based similarity measure and semantic based similarity measure. Once the similarities between two profiles were calculated, they used the Agglomerative Hierarchical Clustering (AHC) method to find the groups of users with similar profiles.

Weib et al. [30] introduced an approach to personalize digital multimedia content based on user profile information. The authors designed this system for personalized electronic program guide (EPG) in the TV domain or for automated and personalized recommendation of online video websites etc. To achieve this, they developed a profile generator that generates user profiles representing the user characteristics automatically and a content-based recommender that approximates the user's interest in content that is yet seen by the user by matching their profile to metadata descriptions of the content. The Profile Generator collects all the explicit ratings and other data and extract the implicit ratings by interpreting user behavior to create user profiles. They defined explicit profiling as the ratings or explicit opinions users have given about an item after consuming its content and implicit profiling as the data that is interpreted indirectly by evaluating user's behavior and extracting their preferences from this evaluation. Their approach for explicit profiling allows them to collect ratings by the user, after consuming the content, on the whole item or a single attribute value. For the implicit profiling, the basis of their mechanism is the duration the user consumes the content of an item. For example, a longer consumption duration for the user on an item like a video implies their preference for the item. For this, they chose the relative duration instead of absolute duration because the amount of time required varies significantly over users. Then using a weighted combination of these ratings, they created the user model that would then be used by the recommender engine to predict unseen items for the user. They decided to use a content-based recommender system to keep the system

independent of any feedback channel and make it self-sufficient and also for some privacy reasons.

Zagheli et. al [47] proposed a new model for updating user profiles based on previously recommended items as well as semantic similarity of terms calculated using distributed representation of words. The authors of this paper proposed to do this by first using a log-logistic model to update user profiles based on recommendation history and then using semantic similarity of terms to enrich the user profiles. Their main idea behind this approach is that each user profile needs to be constructed based on the terms taken from a previous successful recommendation to some users that have a high semantic similarity with the user. They did this by creating a profile language model based on the estimated semantic similarity score between terms and a user profile. The final model is created by linearly interpolating the original model and the profile language model. They only considered English documents in this paper. They compared simple text recommendations with no profile updating, with profile updating based on recommendation history and with profile updating based on semantic similarity to their proposed semantic-aware profile updating hybrid model that combines the previous two approaches into one as described above.

In these reviewed papers, researchers have used different approaches for user profile modelling such as clustering-based approach for user profile generation; explicit ratings and implicit ratings to generate user profile; user profile generation with no profile updating feature, user profile updating based on recommendation history and based on semantic similarity. While in our thesis, we have used users' explicit interactions, implicit interactions and users' social share interactions (the data shared by users on other social networking websites). Then we use weighted values for each interaction and calculate a user feedback score. In other words, we have

used users' feedback for generating user profiles and then we used those user profiles with recommendation approaches for generating recommendations.

2.2.2 Recommendations based on Content

In this sub-section, we will review some of the content-based recommender systems proposed by the researchers in the past.

IJntema et al. [32] proposed an extension to the Hermes Framework (Frasincar et al. [33]), a framework for building a news personalization service, called Athena. Athena uses user profiles to collect and record terms and concepts present in different news items viewed by the user. The framework then utilizes this information by using a traditional method based on TF-IDF, as well as several ontology-based methods to recommend new articles to the user. To build user profiles they kept track of the news articles the user has already seen. To implement approaches such as concept equivalence, binary cosine, and Jaccard, they extracted a set of concepts from the articles that the user has read as the user profile. For other approaches like semantic relatedness or ranked comparison, they created a vector with distinct concepts from the articles that the user has read and then calculated a weight for them. While performing content-based recommendation, before applying TF-IDF on the documents, they first removed the stop-words and stemmed the remaining words to their root words. Then they created a vector that contains the TF-IDF weights for the terms. This vector is then compared with a similar vector constructed for a user as part of the user profile by using all the documents read by the user as the total set of documents. The cosine similarity of these two vectors determines the similarity between the news item and the user. While performing semantic-based recommendation, they used an ontology that covers the different relations or concepts while recommending news articles. They also performed ranked semantic recommendation in which they used a similarly constructed user profile to the content-

based recommendation. In this paper, authors compared ranked recommender or the ontology-based recommender with traditional TF-IDF based recommender system for news personalization service.

Zhao et al. [34] proposed a system that can extract feature words, topics and reveal the most probable research area from research supported by NIH according to their titles. The authors of this paper have achieved this by using TF-IDF and LDA algorithms on the titles of 2000 NIH supported research in the 2017 fiscal year retrieved from the Research Portfolio Online Reporting Tools. They first pre-processed that raw data and extracted distinct terms from it. Then they extracted the feature words from the dataset using TF-IDF. They did this by calculating the TF-IDF weight to decrease the impact of more common but less important terms and to increase the impact of less common but more important terms. After this they used LDA to perform Topic Modelling. They extracted ten topics and three corresponding feature words for each topic and results showed that it was an effective topic-based modelling for NIH titles.

Shao et al. [35] proposed a recommender system for Stack Overflow. They presented a model which would recommend questions to potential answerers according to their field of interest and expertise levels. To do this, they used two prediction components. To predict the topics of the questions and label it with the proper category, they used Latent Dirichlet Allocation (LDA) to perform this task [36]. Then they used the traditional feature-based approach to recommend appropriate developers to answer this question. They used the LDA algorithm to process the dataset and extract the topics and the structure of the questions which were then used to cluster similar questions together. They created the user profiles by extracting various user characteristics from these posts such as user reputation, membership length, number of accepted answers, upvotes etc. They then input the features from the question clusters and user profile into

a trained random forest classifier to predict the answerers. To evaluate their model, they collected information of each user who had answered a particular question and if that user was present in their predicted answerer users set, then they counted that as a true positive. They then divided this true positive value by the total number of answerers on the question to get their accuracy measure. The authors concluded that they were able to get good results when they recommended around 10 to 20 answerers.

Above papers described several approaches for content-based recommendations such as TF-IDF based recommendation approach; ontology-based recommendation approach; and LDA based topic modelling for recommendations. In our thesis, we followed similar approaches for content-based recommendations such as topic-based recommendations using LDA algorithm and term-based recommendations using TF-IDF algorithm. Finding similar contents using content-based approach is one of the important strategies we use for our proposed recommendations.

2.2.3 Recommendations based on Content and Collaborative Filtering

In this sub-section, we will review the papers which have used a hybrid (content and collaborative filtering) approach or have compared content-based and collaborative filtering approaches.

Wilson and Chaudhury et al. [38] proposed an approach to enhance standard collaborative filtering recommenders by using Latent Dirichlet Allocation (LDA) to learn hidden properties of items, depicted in terms of topic proportions and derived from their textual description. They inferred a user's topic preferences in a similar way as the items, based on the user's rating history. They used a combined similarity measure involving rating overlap and similarity in the hidden topic space to compute similarity between users. This allowed them to alleviate the sparsity

problem in recommended items and improve quality of generated recommendations. Each item's (movie) metadata was represented as a document (movie summary). They applied LDA on the document corpus to get the latent topic space for each item and performed the topic modelling task on each item. After this they created topic-distribution vectors for the users, which constitutes their user profile, in the same latent topic space. After modelling all the users and representing them in the same space, they calculated the similarity between users as a product of rating overlap-based similarity and latent topic similarity. Then they used this hybrid similarity measure while building the user neighborhood, or in other words the set of users similar to a user, instead of the standard rating overlap-based similarity. In their experiments, they compared the performance of their hybrid neighborhood based collaborative filtering recommender with the simple user based collaborative filtering recommender and item based collaborative filtering recommender. Their proposed hybrid neighborhood collaborative filtering recommender outperformed the other two approaches.

Salehi and Kmalabadi [48] proposed a learning material recommendation approach which is based on modelling of materials in a multidimensional space of material's attribute. They modeled every user by a matrix that considers multi-attribute of materials. This approach adapts to the user's preference and changing interests. It uses both content-based filtering and collaborative filtering as well as some hybrid approaches. The contextual information about an item is used to model user's learning preference. Their main objective was to develop a recommendation algorithm that considers the attributes of learning materials while performing recommendations for the users. They made user profiles for the learners based on the attribute values of items that the user consumed and rated in the form of a personal preference matrix for the user [48]. Apart from content-based and collaborative filtering, they also proposed a few

hybrid approaches namely, Most Frequently Visited materials, Most Similar Visited materials to target learner, Most Similar Visited to the Most Similar Learners and Most Frequently Visited to the Most Similar Learners. These hybrid approaches differ in the way of how they calculated the scores to form the Matrix representations for the weight vectors. They concluded that their approach of considering attributes of materials was better than traditional recommendation approaches.

These papers described different recommendation approaches such as hybrid approach for recommendations by combining collaborative filtering-based recommender with LDA based similarity measures; and material recommendation approach which uses both content-based filtering and collaborative filtering as well as some hybrid approaches. In our thesis, we compared collaborative filtering recommendations with content-based recommendations using LDA algorithm and TF-IDF algorithm.

2.3 Summary

In this chapter we have reviewed some of the concepts used in the rest of this thesis and also discussed the related work done in the area of user profile modelling and recommendations based on content and collaborative filtering. From these reviews, we can see that most of the proposed user profile modelling [29], [30] use different types of user information such as user attributes, user implicit/explicit ratings and recommender systems [32], [35], [38] uses different types of recommendation approaches for generating recommendations such as content-based approach, collaborative filtering approach and hybrid approach.

While a recommender system can generate a very good recommendation using content-based approach, the user may be interested in the results from collaborative filtering approach.

Finding the most relevant article, the most popular article or the most similar article based on mutual relationship between the article and the user is not always the solution for recommender system. Recommender systems need to be trained on the users' past preferences and their historic interactions so as to build best possible user profile for individual user and generate most relevant recommendations for them.

CHAPTER 3

UTILIZING USER FEEDBACK DATA FOR CONTENT RECOMMENDATIONS

This chapter presents the proposed methods of utilizing user feedback data for content recommendations. The proposed methodology includes two major recommendation approaches: content-based filtering approach and collaborative filtering approach. In content-based filtering approach, we use TF-IDF algorithm and LDA algorithm for content profile modelling and user profile modelling. In collaborative filtering approach, we use user feedback score and Matrix Factorization algorithm for generating recommendations. In the following sections, we explain the overall system architecture for recommendations, content-based filtering model, collaborative filtering model and recommendation engine.

3.1 System Architecture

This research mainly focuses on utilizing user feedback data for generating content recommendations using content-based approach and collaborative-filtering approach. In content-based approach, content profile and user profile are created according to the content information and user's interaction history. These profiles are generated using two techniques that are TF-IDF [1, 11] and LDA Algorithm [1, 42].

In the previous chapter, we reviewed content-based filtering approach and collaborative filtering approach. In content-based filtering, we reviewed TF-IDF algorithm and LDA algorithm and in collaborative filtering approach, we reviewed Matrix Factorization algorithm.

Now we want to compare recommendations generated by content-based filtering approach and collaborative-filtering approach for the CMS dataset which we are using in this research work. Furthermore, we want to compare LDA based content profile and user profile

representation with TF-IDF based representation to see which one could generate better recommendations when being used in content-based filtering approach.

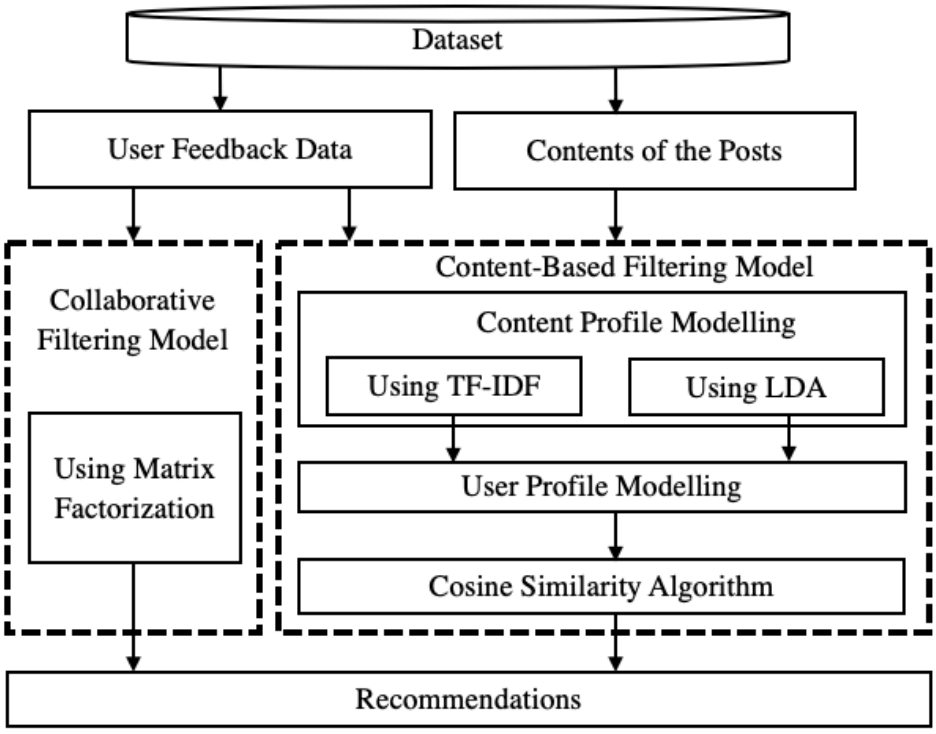

Figure 1: Overall System Architecture for Recommendations

To do that in a systematic way, we have designed the overall system architecture that has enabled this comparison. Figure 1 shows the overall system architecture for recommendation systems which is followed in this research project.

This system architecture has two major components for generating recommendations: content-based filtering model and collaborative filtering model. This chapter describes both components of this system in detail, the input and the output at each step and the algorithms which are involved in this framework.

The first step is to extract required user implicit/explicit feedback data and contents of the posts from the dataset. Both extracted items (user feedback data and contents of the posts)

are the input for the content-based filtering model and user feedback data is the only input for the collaborative filtering model. In content-based filtering model, first, we model content profiles using TF-IDF algorithm and LDA algorithm; second, we model user profile using user feedback data and last, we feed user profile and content profiles to cosine similarity algorithm for profile matching and generating recommendations. In collaborative filtering model, we use matrix factorization algorithm with user feedback data and generate recommendations.

3.2 Content-Based Filtering Model

The first component of the overall system architecture is content-based filtering model. This component is further divided to show the detailed content profile modelling architecture including input data corpus, corpus pre-processing, content profile modelling algorithms; detailed user profile modelling; and content-based recommendations using similarity algorithm. Figure 2 shows detailed content-based filtering model architecture from input as the dataset to the output as the content recommendations.

3.2.1 Content Profile Modelling

The research methodology is carried out using two different content profile modeling approaches (LDA and TF-IDF) to compare their recommendation results. This section covers each of the components of the architecture shown in Figure 2:

1. Corpus Pre-Processing
2. Content Profile Modelling using LDA
3. Content Profile Modelling using TF-IDF

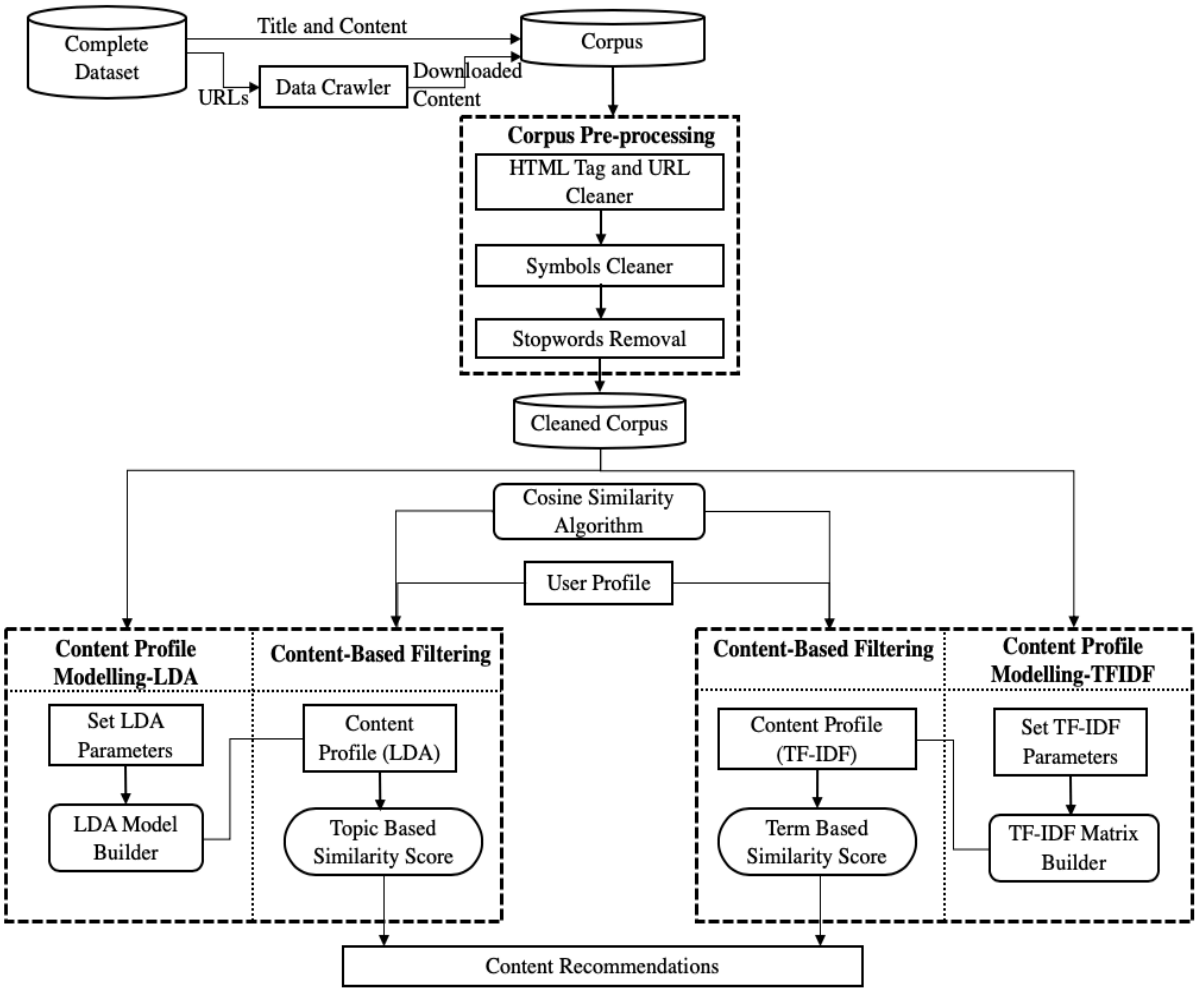

Figure 2: Detailed Content-Based Filtering Model

- **Corpus Pre-Processing**

The framework takes the corpus as the input. The corpus is taken from the dataset which contains title and contents of the documents/posts. This CMS dataset is used in the whole research project for providing effective content recommendations to individual users. There are 6900 documents in the dataset known as posts. Some posts are extemporaneously given while others are downloaded from the internet using data crawling technique based on the post's URL information. We have only kept posts in English in the final corpus. The downloaded contents of the posts are saved in a new table in the same dataset.

The corpus has to be cleaned initially by removing HTML tags, URLs, hashtags, symbols, and punctuation marks. This is done so as to simplify the content to the bare essence of the corpus and then algorithms like TF-IDF and LDA are applied on the cleaned corpus. The conversion of source files into simple text is a bit complicated since the posts are in different formats such as HTML and XML. The use of regular expressions proved very helpful in extracting the text from the source files. After all the cleaning, the corpus contains cleaned, text-only format of post contents.

Other punctuations do not require any special handling. Thus, their removal is simple and easily done. These punctuations include the question mark, hash mark, dollar sign, exclamation mark, arithmetic symbols, tilde, percentage sign, comma, ampersand, quotes, braces, colon, period and semi colon. Basically, any symbol that is not alphabets or numerical digits is removed from the corpus.

Afterwards we remove formatting such as bold, italics, underlines, blockquotes and subscripts from the text as well. This is required so that these terms get represented as readable and do not get passed to the algorithm as separate ineligible words during the content profile modelling. The simplified form of each post is the cleaned version, to be used by the content profiling model.

a	an	and	are	as	at	be	by	from
has	he	in	i	is	it	of	on	same
should	so	some	such	that	the	this	to	them
there	then	very	was	were	who	which	with	will….

Figure 3: List of some common stopwords.

In the last step, we remove stopwords from the corpus. The stopwords removal is an important step for content-based filtering model. Figure 3 shows a list of common stopwords. In natural language processing, there are many libraries which have pre-defined stop-word list.

They detect stopwords accordingly and remove them from the whole corpus. The stopwords are the words which are present in almost all the documents, but they do not contain any important information about the documents.

- **Content Profile Modelling using LDA**

In this section, we use LDA algorithm for content profile modelling. The cleaned corpus, mentioned in the previous section, are given as an input to the topic modelling process. In addition to that, we also set parameters for the LDA algorithm, for example, number of topics and number of keywords per topic. LDA then generates a topic model using this information. That model then gets consumed by our custom code to create a topic-based scored documents. For the topic modeling on this corpus, we choose the LDA model introduced by Blei et al. [13].

Described below is the main formula that the joint distribution of a topic mixture θgiven by LDA and the parameters α and β, z, w:

$$p(\theta, z, w|\alpha, \beta) = p(\theta|\alpha) \prod_{n=1}^{N} p(z_n|\theta) p(w_n|z_n, \beta) \quad (3.1)$$

Where,

- N is the number of topics and M is the number of keywords in N,
- α is the parameter of the Dirichlet prior on the per-document topic distributions,
- β is the parameter of the Dirichlet prior on the per-topic word distribution,
- m is a specific document in the corpus,
- θ is the topic distribution for document m,
- z is the set of N topics in document m,
- and w is set of M words in document m.

LDA model uses this formula for all the documents of the corpus to generate scores for each topic for each document and then every document is assigned to the topic which has the highest score. The LDA algorithm is defined below:

Algorithm 1: Latent Dirichlet Allocation

Input: Corpus **D**
Output: Topic assignments **Z,** Document-Topic Matrix **M**
(1) Step 1: [Make a pass through all documents in corpus **D** for each topic update]
(2) **foreach** iteration **do**
(3) **repeat**
(4) update document/topic distribution
(5) update topic to words assignments
(6) compute topic score using (3.1)
(7) until convergence
(8) end **for**
(9) Step 2:
(10) update topic/word assignments **Z**
(11) update document-topic matrix **M** [Global score update for all the documents]
(11) end

In this way, LDA assigns topic scores to all the documents and at the end we get a topic-based model which has topics with respect to documents and their scores.

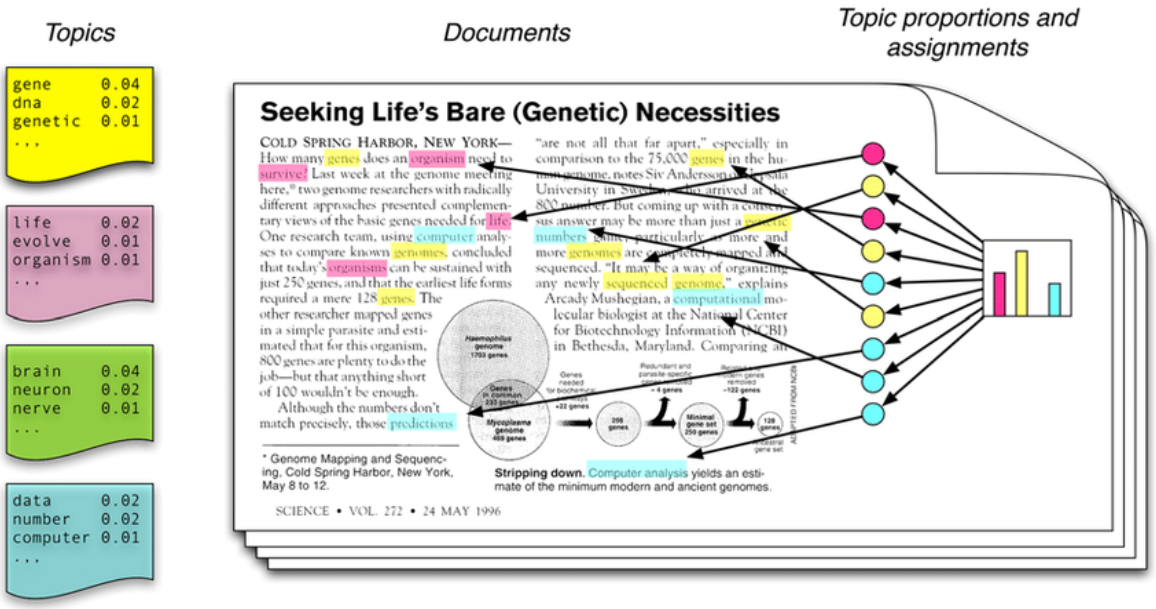

Figure 4: Example of LDA algorithm

In Figure 4, an example is shown for LDA topic modelling [13]. The example shows four topics with different keywords and documents with topic proportions and assignments. The topics are assigned to each of the documents by applying LDA algorithm. Column bars (on the right side) represent topic scores for the document. As we can see, for the top document, one topic is missing, and thus only three bars are shown in the bar graph, which means the document only covers three topics. Also, the bar for yellow topic is greater than bars for other topics. Therefore, the topic in yellow is assigned to this document. In a similar way, topics can be assigned to the rest of the documents in the corpus.

- **Content Profile Modelling using TF-IDF**

Another approach which we use to model content profile is TF-IDF. The TF-IDF formula is used by the matrix model builder to create a word-value matrix using the cleaned corpus.

Term Frequency – Inverse Document Frequency (TF-IDF) is one of the popular measurement techniques in the field of Information Retrieval to calculate term weight values in the vector space model. It makes the following assumptions while measuring the term weights [41]–

- TF assumption – if a term occurs many times in a document then it is closely related to that document
- IDF assumption – if a term occurs in most of the documents in the corpus then it is not very descriptive term for any document.
- Normalization assumption – if a document is longer than another then it should not affect their comparison result between them.

Only the current document is considered while calculating the TF value. For TF value calculation, we choose the TF-IDF model introduced by Salton et al. [49]. Where $f_{t,d}$ is the frequency of the term t in a document d.

$$TF(t,d) = \log(1 + f_{t,d}) \tag{3.2}$$

Inverse document frequency is used to depict a term's occurrence over all the documents. This value is used with the term frequency measurement to decrease the significance of terms such as this, is, etc. that appear in many documents [31]. This means that if a term has a high occurrence in a single document but does not appear too many times in the rest of the documents then that term has a high importance for that document and provides more information about that document than other terms.

$$IDF(t) = \log\frac{|D|}{d(t)} \tag{3.3}$$

Where *D* (corpus) is the set of all the documents and *d(t)* is the number of documents that contain the term t in them. Therefore, the TF-IDF weight of a term t in a document d is –

$$TF - IDF(t,d) = TF(t,d) \times IDF(t) \tag{3.4}$$

The TF-IDF algorithm is defined below:

Algorithm 2: Term Frequency – Inverse Document Frequency

Input: Corpus **D**
Output: TF-IDF weights, Document-Term Matrix **M**
(1) Step 1: [Make a pass through all the documents in D (corpus)]
(2) **foreach** document in **D do**
(3) **repeat**
(4) compute IDF (inverse document frequency) using (3.3)
(5) **foreach** term **do**
(6) **repeat**
(7) compute TF (term frequency) for each term using (3.2)
(8) compute TF-IDF weights for each term using (3.4)
(9) end **for**
(10) end **for**
(11) Step 2:
(12) update document-term matrix **M** [Global score update for all the documents]

(13) end

In this way, the algorithm assigns TF-IDF scores to all the terms and at the end we get a documents-term matrix which has terms with respect to the documents and their scores.

3.2.2 User Profile Modelling

User profile consists of a vector of weights which depict the user's preference. Each weight in the vector corresponds to a certain set of interactions or aspect related to the user/item interaction. These interactions can be of any type and their selection mostly depends on the type and the domain of dataset used by the recommender. For example, author name, title, content etc. could be appropriate attributes for the dataset of books. The weights can be calculated by a multitude of information retrieval and content-based filtering techniques [1].

In this thesis, we focus on developing a user preference-based recommendation so as to generate more user-oriented models and collect more reliable recommendation scores. The process involves representing the user preferences as a collection of interactions that are grouped under categories for calculating the user profiles. After examining the user's historical data and interactions, we calculate the user's preference score pertaining to those categories.

The interactions or user feedback data which show user/item interactions internally on the CMS such as like, comment, impression and clickthrough come under the category known as *Direct Interaction*. The interactions which show data is shared externally by the users on their social networking websites such as shares, re-shares, reply, retweets come under *Social Share* category. And the interactions like reading duration and the number of reads which show reading progress made by a user internally on the CMS itself come under the *Reading Statistics* Category. The flow diagram of the approach related to user's preference score calculation for the three categories is shown in Figure 5.

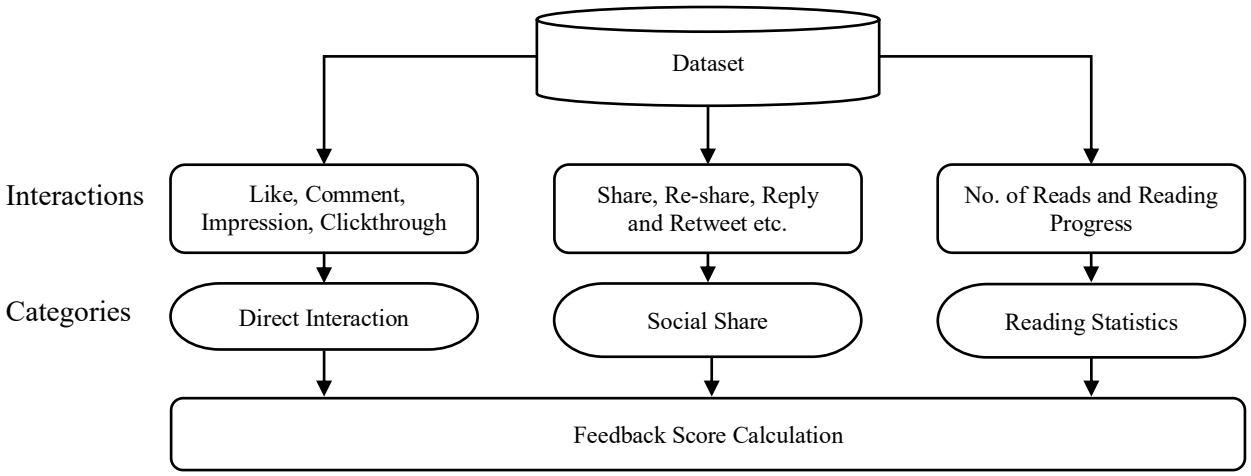

Figure 5: User Feedback Score Calculation

The methodology includes the following steps. First, user/post implicit interactions and explicit interactions are extracted from the dataset. Second, these interactions are categorized based on the type of interactions. Interactions are grouped under three main categories which are *Direct Interaction*, *Social Share* and *Reading Statistics*. Third, score is calculated for each category and overall feedback score is calculated for each user and post pair.

The *Direct Interaction* category score is calculated which consists of user/post interactions made by users on the CMS itself that are direct share, direct impression, direct re-share, direct like, direct comment and direct clickthrough. Equation 3.5 is used for *Direct Interaction* score calculation which is given below.

$$\boldsymbol{D_{up} = w_i * di_{up} + w_s * ds_{up} + w_l * dl_{up} + w_c * dc_{up} + w_r * dr_{up} + w_t * dt_{up}} \quad (3.5)$$

Where,

- $\boldsymbol{D_{up}}$ = Direct Interaction score for one user on one post
- $\boldsymbol{w_i, w_s, w_l, w_c, w_r, w_t}$ are weights for impression, share, like, comment, re-share and clickthrough respectively.

- $di_{up}, ds_{up}, dl_{up}, dc_{up}, dr_{up}, dt_{up}$ are total numbers of direct impressions, direct shares, direct likes, direct comments, direct re-shares and direct clickthrough made by a user on a post respectively. The direct impressions and direct clickthrough are numerical values while direct shares, direct likes and direct re-shares are binary values.

The *Social Share* category score is calculated which consists of user/post interactions made by users from CMS to other social networking websites including Facebook, Twitter and LinkedIn. Equation 3.6 is used for *Social Share* score calculation which is given below.

$$S_{up} = w_{sh}[fs_{up}+ts_{up}+ls_{up}] + w_{im}[fi_{up}+ti_{up}+li_{up}] + w_{cl}[fc_{up}+tc_{up}+lc_{up}] + w_{co}[fm_{up}+tm_{up}+lm_{up}] + w_{re}[fr_{up}+tr_{up}] + w_{li}[fl_{up}+ll_{up}] + w_{fa}[tf_{up}] \quad (3.6)$$

Where,

- S_{up} = Social Share score for one user on one post
- $w_{sh}, w_{im}, w_{cl}, w_{co}, w_{re}, w_{li}, w_{fa}$ are weights for share, impression, clickthrough, comment, reply, like and favourite respectively.
- $fs_{up}, ts_{up}, ls_{up}$ represent whether a post is shared by a user on Facebook, Twitter and LinkedIn respectively. They are binary values, if a user shared a post then value is 1 otherwise 0.
- $fi_{up}, ti_{up}, li_{up}$ are numbers of impressions made by a user on a post on Facebook, Twitter and LinkedIn respectively. They are numerical values.
- $fc_{up}, tc_{up}, lc_{up}$ are numbers of clickthrough made by a user on a post on Facebook, Twitter and LinkedIn respectively. They are numerical values.
- $fm_{up}, tm_{up}, lm_{up}$ are numbers of comments made by a user on a post on Facebook, Twitter and LinkedIn respectively. They are numerical values.

- fr_{up}, tr_{up} are replies made by a user on a post on Facebook and Twitter respectively. These interactions are represented as binary values.
- fl_{up}, ll_{up} represent whether a post is liked by a user on Facebook and LinkedIn respectively. These interactions are shown as binary values, if user liked a post then value is 1 otherwise 0.
- tf_{up} represents whether a post is favored by a user on Twitter. These interactions are represented as binary values.

The *Reading Statistics* category score is calculated which consists of user/post implicit interactions made by users on CMS itself that are number of reads and reading progress. Equation 3.7 is used for *Reading Statistics* score calculation which is given below.

$$R_{up} = w_n * rn_{up} + w_p * rp_{up} \tag{3.7}$$

Where,

- R_{up} = Reading Statistics score for one user on one post
- w_n, w_p, are weights for number of reads and reading progress respectively.
- rn_{up}, rp_{up}, are number of reads on a post made by user and reading progress on a post made by user respectively. Reading Progress is a percentage value measuring how many percent of content has been read by the user.

Lastly, the *Feedback Score* is calculated by adding scores from all the categories together with the weights defined by us for each category. The equation used to calculate the Feedback Score is shown below.

$$FS_{up} = (W_x * D_{up}) + (W_y * S_{up}) + (W_z * R_{up}) \tag{3.8}$$

Where,

- FS_{up} = Feedback score for one user on one post
- D_{up} = Direct Interaction score for one user on one post, calculated using Equation 3.5
- S_{up} = Social Share score for one user on one post, calculated using Equation 3.6
- R_{up} = Reading Statistics score for one user on one post, calculated using Equation 3.7
- W_x = Weight for Direct Interaction category
- W_y = Weight for Social Share category
- W_z = Weight for Reading Statistics category

In this way, we generate Feedback Score while considering all the categories and by adjusting weight, we can put different weights on different categories. We are going to define the weight values chosen for this work for each interaction in the next chapter.

The next step is to model user profile for the individual users. So, after calculating score for each category and the overall feedback score, we can model user profile for each user. We use these calculated scores together with the content profiles of the posts that the user has interacted with to get the user profile for each user. In this way, we create multiple user profiles based on the different combinations of user/post interaction categories (Direct Interaction, Social Share, Reading Statistics and Feedback Score) and content profiles (LDA and TF-IDF).

3.2.3 Content-Based Recommendations

One typical problem caused by the data sparsity is the cold start problem. Cold start problem in recommender systems is relevant for new users, who have no preferences or historical interactions recorded, and for new items, which have no usage or interaction information existing. The content-based approach can naturally deal with the new-item cold-start problem because it mainly relies on the matching between content features of the item and the user profile, not on usage or rating of items. To deal with new-user cold-start problem, we use popularity-

based strategy. The most popular posts are identified based on popularity of the posts with maximum views and recommended to the new users.

For content-based recommendations, we use Cosine similarity algorithm in this thesis. Cosine similarity is considered as a heuristic based method in content-based filtering approaches [1]. For content-based recommendations, the Cosine similarity algorithm is used to find similarities between the items and the users. This is done by applying the cosine similarity to the TF-IDF weights vectors or the LDA weight vectors in the profiles.

To perform the recommendations of items for the user, the user profile is compared for similarity with the unseen items' profiles [41]. This way, the items whose profiles are the closest or the most similar to the user's profile are ranked the highest and thus are outputted by the content-based recommenders as recommendations for the user [56].

Since each item has its own weight vectors created using either TF-IDF or LDA and every user profile is also represented by weight vectors of the implicit scores as well. Their similarity is measured by simply calculating the cosine of the angle between the two vectors. If the cosine of an angle between two vectors are close to 1, which means that the vectors are parallel or in other words, they are similar to each other. If the cosine is 0 then that implies perpendicularity between two vectors or in other words, they are totally different with each other. The equation for calculating the cosine similarity between two vectors A and B is,

$$similarity(A,B) = \cos(\vec{A},\vec{B}) = {\vec{A}.\vec{B}}\Big/{|\vec{A}| \times |\vec{B}|} \tag{3.9}$$

Where, A is the weight vector of an item calculated in content profile and B is the weight vector of the user profile.

3.3 Collaborative Filtering Model using Matrix Factorization

We choose to write the code for Collaborative Filtering recommender system ourselves even though there are various open source tools available for this. Figure 6 shows complete architecture of collaborative filtering using matrix factorization. To implement matrix factorization, we can use different types of decomposition techniques. In this thesis, the algorithm we use for matrix decomposition is SVD (Singular Value Decomposition). The model builder uses SVD to produce a matrix for all given users and all given posts. Suppose M is a $m \times n$ (m is number of rows and n is number of columns) matrix whose entries come from the variable K, which is the variable of non-negative real numbers. The factorization is called as a singular value decomposition of **M** of the form where:

$$\mathbf{M} = \mathbf{U}\,\mathbf{\Sigma}\,\mathbf{V}^{*} \tag{3.10}$$

- **U** is an $m \times m$ unitary matrix over K (if $K = R$, unitary matrices are orthogonal matrices),
- **V** is an $n \times n$ unitary matrix over K, and $\mathbf{V}^{*}$ is the conjugate transpose of **V**.
- **Σ** is a diagonal $m \times n$ matrix with non-negative real numbers on the diagonal,

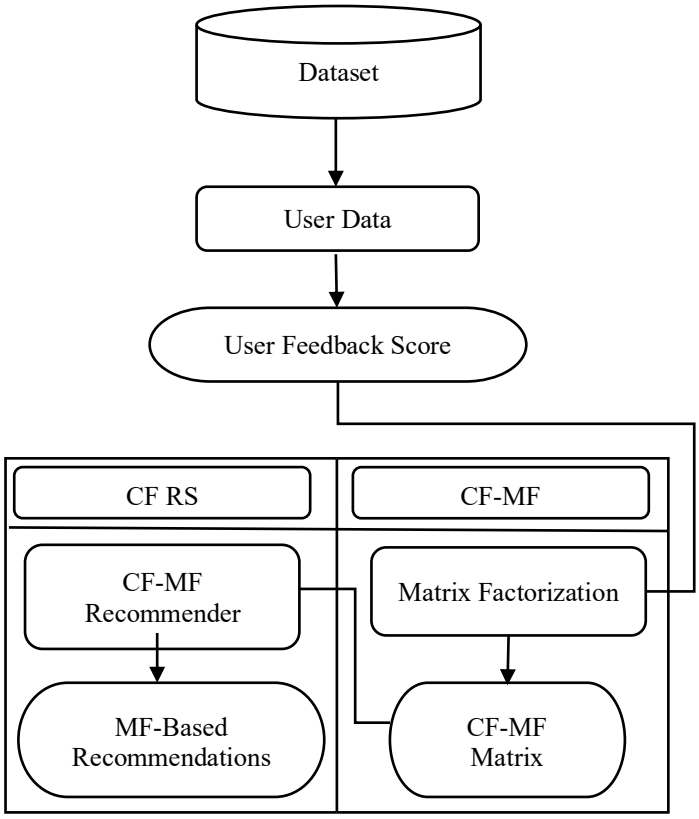

Figure 6: Collaborative Filtering using Matrix Factorization architecture

The diagonal entries σ_i of $\mathbf{\Sigma}$ are known as the singular values of $\mathbf{M}$. A common convention is to list the singular values in descending order. In this case, the diagonal matrix is uniquely determined by $\mathbf{M}$. In this way, it calculates the whole matrix using SVD approach.

The biggest advantage of CF-MF (Collaborative Filtering-Matrix Factorization) is that it does not care about the items and does not need any content information about them for making the recommendations. It bases recommendations only on the similarities between users and their ratings (or other types of feedbacks). It works on the principle that if a group of users shared similar interests in the past then they will share the similar interests in the future.

3.4 Summary

In this chapter, we explain the methodology of our research in details. Since each user is unique and motivated differently, recommendations provided for one user may not work for another user. Some users are interested in finding contents they already read while others may be interested in finding new content. Therefore, we use content-based approach to generate recommendations as well as collaborative filtering-based recommendations.

We have given a brief introduction of the overall system architecture for recommendations. Then we have described detailed content-based filtering model which includes content profile modelling, user profile modelling and content-based recommendations. Also, we have described collaborative filtering model using matrix factorization. The detailed content profile modelling provides the idea about the techniques which are used to generate content profiles. We have explained the methodology involved in generating user feedback score. Furthermore, we have described the different approaches used in generating recommendations for the users.

Our recommendation model uses the user's historical activity data to generate the best possible recommendations for the individual user. Our assumption is that the scores generated using yesterday's user data can be effectively used to predict today's articles for the user. In our next chapter, we will discuss the experiment we performed to evaluate our approach, including an implementation of the overall system.

CHAPTER 4
EXPERIMENTS

In this chapter, we discuss the experiment we performed in order to determine the accuracy of our proposed recommendation system. First, we start with how the CMS data is collected. The content and interaction data which are used to implement the proposed system are also explained in this subsection. Second, we present the programming languages and libraries which are used to design the overall system. Third, we describe the evaluation phase of the proposed recommendation system and show the results which are obtained from the testing phase of the system. After that, the evaluation metrics that are used to appraise the proposed system are given in the next subsection. Finally, we show the comparison between the proposed approaches with analysis and discussion on those results.

4.1 The CMS Dataset

We perform the experiment to validate the accuracy of our proposed recommendation system. We start with implementing the proposed algorithms. As we have defined the system in modules, we implement different modules separately.

The dataset we use in this experiment is a company's CMS dataset. The duration of data collection is more than four years starting from February 27, 2013 to September 28, 2017. We use this CMS dataset to generate recommendations in our experiment for effective content suggestions to the individual user of the dataset. The dataset has two major entities: posts and users. The current dataset consists of 250 users, 6900 posts as well as the user/post interactions. In order to build the entire model, we collect the posts' information and users' information from the server using SQL queries. The main post information we extracted are described below.

- **Post ID:** It is a unique value provided to each individual post. It is very important for the content profile generation as it is used as an index value.
- **Post Title:** This feature describes the title of the post. Every post has a title and is available in the dataset. So, we use this feature for content profile generation.
- **Post URLs:** This feature provides the URL address of the post. It stores URL information for every post.
- **Post Content:** This feature is available in the dataset for some posts and for the remaining posts, we downloaded the contents of the posts using data crawler based on their URLs. As our recommender system is not directly connected to the online service, we extract the data from the internet and made it available to our system. We compose the list of posts' URLs to be used for downloading of data. Then we store the downloaded content along with other features.

Similarly, we extract user information from the dataset. The current dataset has around 250 users in total and some users have hundreds of interactions with the posts. We only extract major interactions to generate user profile for our recommendation system. For each of the users for whom we want to generate recommendations, we extract the following information related to user interactions.

- **User ID:** It represents a unique ID for each individual user. It is very important for the user profile generation as it is used as an index value for it.
- **Direct Share:** It provides information about whether a user has *shared* a post directly on the CMS. For example, if a user has shared a post on CMS directly, then interaction value for that user-post pair is 1 otherwise 0.

- **Direct Impression:** It provides information about the number of *impressions* directly provided by the user on the CMS for a post. It is a numerical value.
- **Direct Re-share:** It provides information about whether a user has *re-shared* a post directly on the CMS. The interaction value for *direct re-share* is represented by a binary value for each user-post pair.
- **Direct Like:** It provides information about whether a user has *liked* the post directly on the CMS. These user-post interaction values are binary.
- **Direct Comment:** It provides information about whether a user has *commented* on the posts directly on the CMS. These user-post interaction values are binary.
- **Direct Clickthrough:** It provides information about the number of times a user has *clicked* the post directly on the CMS. These interactions are numerical values.
- **Twitter Share:** It provides information about whether a user has *shared* post from CMS to other social networking websites, in this case, the social networking website is Twitter. These interactions are binary values.
- **Twitter Impression:** It provides information about the number of *impressions* provided by users on the post on Twitter website. These interactions are numerical values.
- **Twitter Retweet:** It provides information about whether a user has *retweeted* on the post on Twitter. These interactions are binary values.
- **Twitter Favorite:** It provides information about whether a user has marked post as *favourite* on Twitter. These interactions are binary values.
- **Twitter Reply:** It provides information about whether a user has *replied* on the post on Twitter. These interactions are binary values.

- **Twitter Clickthrough:** It provides information about the number of times the post has been *clicked* on Twitter. These interactions are numerical values.
- **Facebook Share:** It provides information about whether a user has *shared* post from CMS to other social networking websites, in this case, the social networking website is Facebook. These interactions are binary values.
- **Facebook Impression:** It provides information about the number of *impressions* provided by users on the post on Facebook website. These interactions are numerical values.
- **Facebook Re-share:** It provides information about whether a user has *re-shared* a post on Facebook. These interactions are binary values.
- **Facebook Like:** It provides information about whether a user has *liked* the post on Facebook. These interactions are binary values.
- **Facebook Comment:** It provides information about whether a user has *commented* on Facebook. These interactions are binary values.
- **Facebook Clickthrough:** It provides information about the number of times the post has been *clicked* on Facebook. These interactions are numerical values.
- **LinkedIn Share:** It provides information about whether a user has *shared* post from CMS to other social networking websites, in this case, the social networking website is LinkedIn. These interactions are binary values.
- **LinkedIn Impression:** It provides information about the number of *impressions* provided by users on the post on LinkedIn. These interactions are numerical values.
- **LinkedIn Like:** It provides information about whether a user has *liked* the post on LinkedIn. These interactions are binary values.

- **LinkedIn Comment:** It provides information about whether a user has *commented* on LinkedIn. These interactions are binary values.
- **LinkedIn Clickthrough:** It provides information about the number of times the post has been *clicked* on LinkedIn. These interactions are numerical values.
- **Number of reads Internally:** It provides information about how many times a user has *read* the post directly on the CMS itself.
- **Reading Duration Internally:** It provides information about the user's *reading duration* for a specific post directly on the CMS. It shows the percentage of the post the user has finished reading.

All of the above given user interactions are used in this experiment. The user profiles are generated based on these interactions. First, we categorize the interactions into three categories as described below.

- **Direct Interaction:** This category consists of interactions which show user/post interactions directly on the CMS, for example, direct share, direct impression and direct like. Table 1 shows statistical information about this category.

Table 1: Statistical information about Direct Interaction category

Description	Number of Interactions
Total number of direct interactions	20868
Total number of users with direct interactions	150

- **Social Share:** This category consists of information which shows user shared/liked posts from CMS to the other social networking websites, for example, Twitter share, Facebook share and LinkedIn share. Table 2 shows statistical information about this category.

Table 2: Statistical information about Social Share category

Description	Number of Interactions
Total number of social shares	28363
Total number of users with social shares	165

- **Reading Statistics:** This category consists of user's post reading details, for example, number of reads and reading duration on the CMS. Table 3 shows statistical information about this category.

Table 3: Statistical information about Reading Statistics category

Description	Number of Interactions
Total number of posts with reading stats	10985
Total number of users with reading stats	134

Table 4: User interactions and default weight values

User Interactions	**Interaction Category**	**Default Weight Values**
Direct Share	Direct Interaction	3
Direct Impression	Direct Interaction	1
Direct Re-share	Direct Interaction	5
Direct Like	Direct Interaction	5
Direct Comment	Direct Interaction	5
Direct Clickthrough	Direct Interaction	5
Twitter Share	Social Share	3
Twitter Impression	Social Share	1
Twitter Retweet	Social Share	5
Twitter Favorite	Social Share	5
Twitter Reply	Social Share	5
Twitter Clickthrough	Social Share	5
Facebook Share	Social Share	3
Facebook Impression	Social Share	1
Facebook Reply	Social Share	5
Facebook Like	Social Share	5
Facebook Comment	Social Share	5
Facebook Clickthrough	Social Share	5
LinkedIn Share	Social Share	3
LinkedIn Impression	Social Share	1
LinkedIn Like	Social Share	5

LinkedIn Comment	Social Share	5
LinkedIn Clickthrough	Social Share	1
Number of reads	Reading Statistics	5
Reading duration	Reading Statistics	5

The predefined weights for all of these interactions are shown in Table 4. We generate overall feedback score considering all these three categories. We also calculate feedback score by considering each category separately to identify which category has greater impact on the recommendations. For the feedback score calculation for each category, we used predefined weight provided in the CMS dataset.

4.2 Model Implementation

In this research, we have used MAC Operating System Sierra and we installed all required libraries and platforms. The main programming language used for the model implementation in this thesis project is Python. Python is a high-level language that provides us the versatility required for this project. For the model implementation, we used Python 2.7 version using IDLE which is the default python IDE and for the compilation and interpretation of the results we used command prompt window. One reason we chose Python is that most of the frameworks and libraries used in the field of Machine Learning and Data Mining are available in Python and are regularly maintained.

We used many libraries to implement our system. We used Pandas library to build the data frames and regular expressions to clean the data. To model the user interest on a given post, we aggregate all the interactions the user has performed on an item by a weighted sum approach and apply a log transformation to smooth the distribution. Other libraries we used include Scikit-Learn [43], Mallet [44] and MySQL Connector.

Scikit-Learn [43] is an open source machine learning library for Python Language. It provides a diverse collection of algorithms for supervised and unsupervised machine learning via

an interface in Python. In this thesis, we use Scikit-Learn library for implementing TF-IDF algorithm and Cosine Similarity algorithm.

Mallet [44] is a statistical natural language processing library. It is widely used for document classification, clustering, topic modelling, information extraction etc. In this thesis, we use Mallet for the purpose of topic modelling which is essentially a probabilistic model of the words appearing in a corpus.

We also use SQL (Structured Query Language), a popular language for accessing and manipulating databases, and MySQL Connector in Python code for extracting the data required for this project before pre-processing and cleaning it so as to prepare it as an input for the text processing algorithms.

4.3 Model Evaluation

The model evaluation is, as the name suggests, used to evaluate the performance of our approaches. It takes the ranked lists of recommended items from the recommender systems and then performs calculations to quantify the results. In this project, we compare the output of the three recommendation approaches (TF-IDF, LDA and CF) mentioned in Chapter 3 by using the metrics depicted by the confusion matrix. We have chosen the precision, recall and f1-score as the evaluation metrics.

The complete process for this thesis work is outlined as stages with the components in Figure 7. First stage is to get the dataset, second stage is to compute recommendations and third stage is to evaluate and compare recommendations. Parallelization is possible for some sub-stages but is not performed in this thesis work.

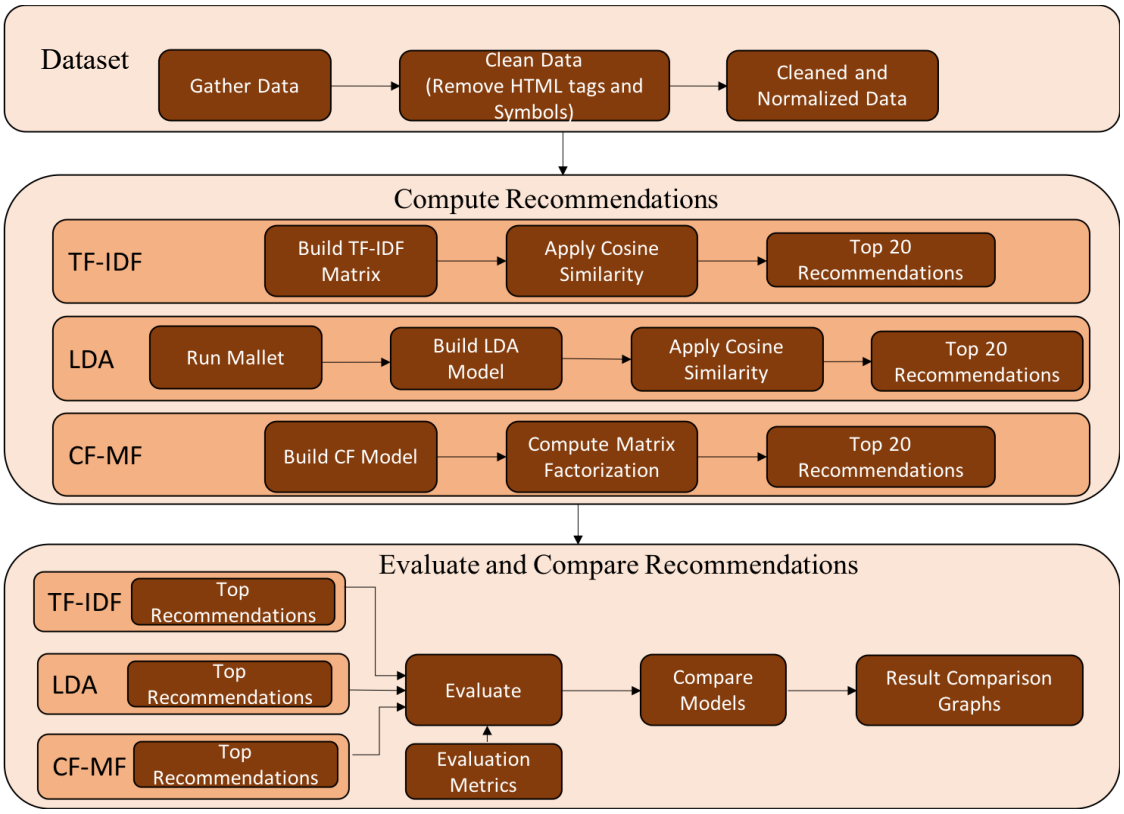

Figure 7: Complete process of evaluation and comparison of recommendations

4.4 Evaluation Metrics

In the proposed model, the dataset includes the posts, the users and the interactions that users have with the posts in the system. To evaluate the proposed system, the feedback score table needs to be divided into 2 parts. One part is needed for training phase of the proposed approach, the second part is needed for the testing phase of the proposed approach. Simple holdout approach is used, in which a random data sample (20% in this case) are kept aside in the training process, and exclusively used for testing. All evaluation metrics reported here are computed using the test set. With the train part of the table, the proposed system is trained and with the test part of the table the proposed system is evaluated.

To evaluate the proposed recommendations, we use the decision support metrics: precision, recall and f1-score [31]. We choose these metrics for the evaluation of our system because most of the recommender systems in the related literature work [6, 1, 77, 68] have used same evaluation metrics in their evaluation process. ROC (Receiver Operating Characteristic) curve is another popular option, which represents the combination result of precision and recall. Since f1-score summarizes the precision and recall of ROC curve and can be used in place of ROC curve [26], we have chosen to use f1-score instead of ROC curve in this work.

Root Mean Square Error (RMSE) metric is another metric which can be used to test the prediction accuracy and this metric calculates the standard deviation between predicted implicit feedback score and the actual implicit feedback score [31]. According to [45] and [14], RMSE cannot reflect the real user experience. Users in the real recommender system applications are only interested in whether an item is recommended or not. But RMSE metric measures the accuracy by only relying on the predicted ratings (or feedback scores). The evaluated feedback score related to a post is not as important as the predicted classification of the post: like or not like. Because of these reasons, we have chosen not to use RMSE to evaluate the proposed recommendation approaches.

To use these metrics, the predicted scores and the actual scores should be converted to the binary scale: positive and negative. A relevant post for a specific user-post pair means that this post is a good recommendation for the user. We categorize the recommendation results as shown in Table 5 to calculate the evaluation metrics.

Table 5: Categorization of recommendation results using confusion matrix

Confusion Matrix	**Actual Positive**	**Actual Negative**
Predicted Positive	True Positive	False Positive
Predicted Negative	False Negative	True Negative

The posts whose scores are marked as positive (Actual Positive) and recommended to the users by the proposed system (Predicted as Positive) are considered as True Positive. The posts whose scores are marked as positive (Actual Positive) but not recommended to the users by the proposed system (Predicted as Negative) are considered as False Negative (FN). The posts whose ratings are marked as negative (Actual Negative) but recommended to the users by the proposed system (Predicted as Positive) are considered as False Positives (FP). The posts whose score are marked as negative (Actual Negative) and not recommended to the users by the proposed system (Predicted as Negative) are considered as True Negatives (TN). After predicting the recommendation results, we calculate the precision, recall and f1-score metrics for each recommendation approach by using Equation 4.1, Equation 4.2 and Equation 4.3, respectively.

$$Precision = \frac{TP}{TP+FP} \tag{4.1}$$

Precision can be described as the ratio of number of posts predicted as positive and defined as positive to the number of posts that are predicted as positive. This ratio shows the probability of a recommended post is liked by the user.

$$Recall = \frac{TP}{TP+FN} \tag{4.2}$$

Recall can be defined as the ratio of the number of posts predicted as positive and defined as positive to the number of posts which are defined as positive in the system. This ratio shows the probability of a liked post is recommended by the system to the user.

$$F1 - Score = 2 * \frac{Precision*Recall}{Precision+Recall} \tag{4.3}$$

F1-score is the harmonic mean of the precision and recall. It conveys balance between the precision and recall metrics.

4.5 Results and Analyses

We generate recommendations using two different approaches: content-based approach and collaborative filtering approach. The time taken to generate collaborative filtering recommendations is less as compared to content-based recommendations. The collaborative filtering approach takes approximately 12 minutes in generating recommendations with evaluation results while TF-IDF approach takes approximately 16 minutes and LDA approach takes approximately 17 minutes in generating recommendations. The possible reason for the difference in their running time (the collaborative filtering approach and content-based approaches) is that in content-based filtering approach, the content profile and user profile both are generated in order to provide recommendations while in collaborative filtering only user feedback score-based matrix is generated for providing recommendations.

In the content-based approach we use two different algorithms to generate content profiles: TF-IDF and LDA. We have generated four content profiles by using these two algorithms which are described below.

- **TF-IDF (title only):** In this profile, we have considered *titles* of the posts only and then we have implemented TF-IDF algorithm on the *titles* of the posts to generate this content profile.
- **TF-IDF (content):** In this profile, we have considered whole *contents* of the posts and then we have implemented TF-IDF algorithm on the *contents* of the posts to generate this content profile.
- **LDA (title only):** For this profile, we have considered *titles* of the posts only and then we have implemented LDA algorithm on the *titles* of the posts to generate this content profile.

- **LDA (content):** In this profile, we have considered *contents* of the posts and then we have implemented LDA algorithm on the *contents* of the posts to generate this content profile.

In this way, we have generated 4 different content profiles by using two algorithms. We have also generated 4 different user interaction categories which are defined below:

- **Direct Interaction:** This category considers the feedback score only from *direct interaction* category.
- **Social Share:** This category considers the feedback score only from *social share* category.
- **Reading Statistics:** This category considers the feedback score only from *reading statistics* category.
- **Feedback Score:** This category considers the overall feedback score which is aggregated score of all three categories.

Then user profiles are calculated by using above described user interaction categories and by using 4 content profiles. In this way, in total 16 user profiles are created by all possible combinations of content-profiles and user-profiles.

Afterwards, we have matched all of these four content profiles with all user profiles by using cosine similarity algorithm and we compute recommendations. These recommendations are then evaluated and compared for all these models. We have also implemented collaborative filtering-based model by using matrix factorization to compare our content-based recommendations with the collaborative filtering-based recommendations.

4.5.1 Evaluation Results for *Top-K* Recommendations

In the context of recommendations, we are most likely to be interested in recommending *top-k* posts to the user as it makes more sense to compute precision, recall and f1-score metrics for the first *k* posts instead of all the posts. The *top-k* precision, *top-k* recall and *top-k* f1-score is defined by *k* where *k* a is integer set by us to match the *top-k* recommendations.

First, we have considered *Precision at k* which means the proportion of recommended items in the *top-k* set that are relevant. For example, assume that our *precision at 20* in a *top-20* recommendation problem is 80%. This means that 80% of the recommendation we made are relevant to the user. Mathematically, *Precision@k* is defined in in Equation 4.4:

$$Precision@k = \frac{\text{Number of recommended items @k that are relevant}}{\text{Number of recommended items @k}} \tag{4.4}$$

Second, we have considered *Recall at k* which means the proportion of relevant items found in the *top-k* recommendations. For example, assume that we computed *recall at 20* and found it is 70% in our *top-20* recommendation system. This means that 70% of the total number of the relevant items appear in the top-20 results. Mathematically, *Recall@k* is defined in Equation 4.5:

$$Recall@k = \frac{\text{Number of recommended items @k that are relevant}}{\text{Total number of relevant items}} \tag{4.5}$$

Third, we have considered *F1-score at k* which provides harmonic mean of the *precision@k* and *recall@k* metrics. Mathematically, *F1-score@k* is defined in Equation 4.6:

$$F1 - Score@k = 2 * \frac{Precision@k * Recall@k}{Precision@k + Recall@k} \tag{4.6}$$

So, instead of measuring average precision, average recall and average f1-score for our recommendations, we have measured *top-k* precision, *top-k* recall and *top-k* f1-score for all of our recommendation approaches where we set *k* as *10* and *20*. First, we compare recommendations from all the models based on *top-k* precision where *k* is set as *10 and 20.*

- ***Precision@10:*** This evaluation of the recommendations is based on precision metric where we evaluate precision score for *top-10* recommendations. Figure 8 shows precision@10 scores for all the algorithms with all possible comparisons.

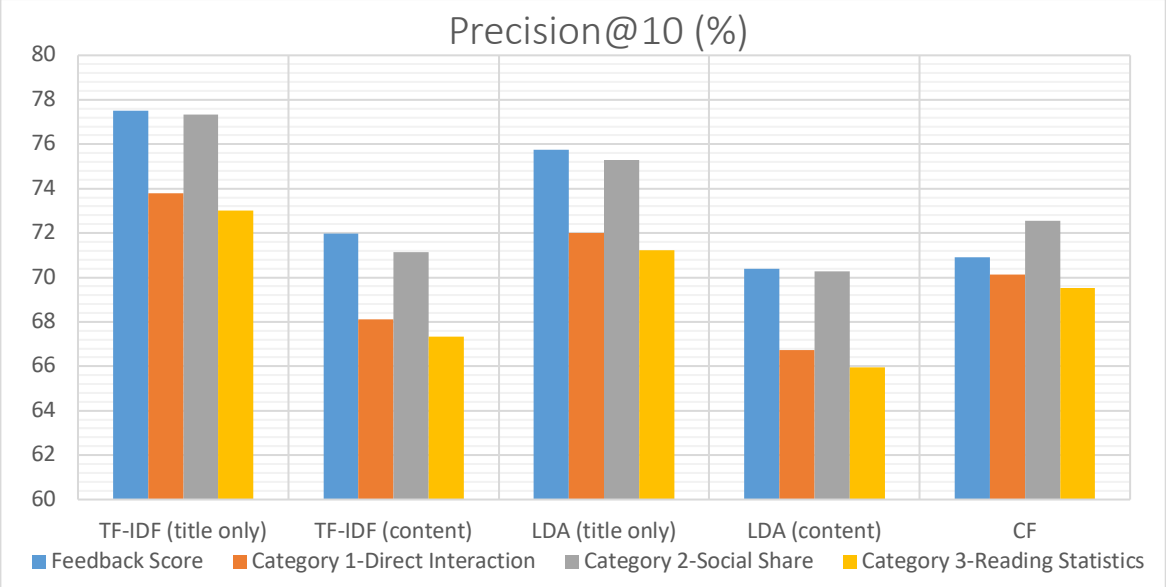

Figure 8: Evaluation of recommendations using Precision@10

First, we compare TF-IDF (title only) content profile with all four user profiles (Feedback Score, Direct Interaction, Social Share and Reading Statistics) and we get maximum value of *precision@10* from overall feedback score user profile which is 77.52% and lowest value of 73% from category 3 reading statistics user profile.

Second, we compare TF-IDF (content) based content profile with all four user profiles and we get maximum value of *precision@10* from feedback score user profile which is 71.98% and lowest value of 67.34% from category 3 reading statistics user profile.

Third, we compare LDA (title only) content profile with all four user profiles and we achieve maximum value of *precision@10* from feedback score user profile which is 75.76% and lowest value of 71.22% from category 3 reading statistics user profile.

Fourth, we implement LDA (content) based content profile with all four user profiles and we get maximum value of *precision@10* from feedback score user profile which is 70.38% and lowest value of 65.96% from category 3 reading statistics user profile.

Lastly, we implement CF using matrix factorization for all four user profiles and we achieve highest score of *precision@10* from social share user profile which is 72.54% and lowest score of 69.52% from category 3 reading statistics user profile for CF based recommendations.

By comparing all the results for Precision@10, we get maximum value from TF-IDF (title only) content profile for overall feedback score user profile which is 77.52% and we get lowest value of 65.96% for LDA (content) content profile for reading statistics user profile. The reading statistics is usually the lowest among all user profiles because we have a smaller number of interactions in this category as compared to other categories.

- ***Precision@20:*** In this part, the recommendations are evaluated using same precision metric formula but here we evaluate precision score for *top-20* recommendations instead of *top-10*. Figure 9 shows precision@20 scores for all the algorithms with all possible comparisons.

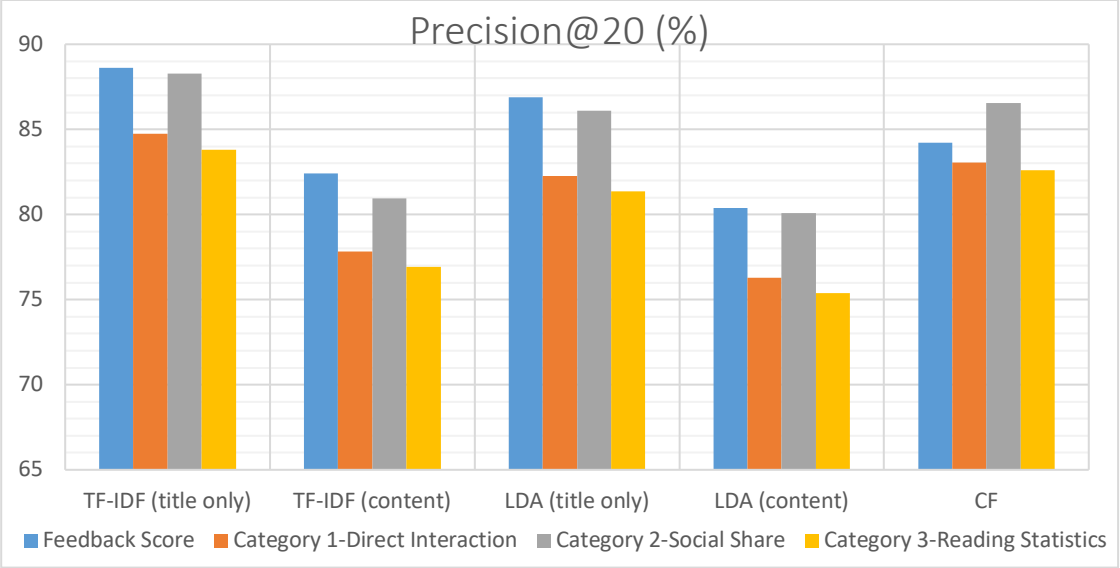

Figure 9: Evaluation of recommendations using Precision@20

Figure 9 shows the results for precision@20 for all the algorithms, we get maximum value from TF-IDF (title only) content profile for overall feedback score user profile which is 88.6% and we get lowest value of 75.38% for LDA (content) content profile for reading statistics user profile.

We can check the results shown in Figure 8 and Figure 9 for a comparative analysis of precision@10 and precision@20 for all the approaches. The figures depict that precision@10 values and precision@20 values show relatively same pattern for content-based approaches (TF-IDF and LDA) and for CF approach.

- ***Recall@10:*** This evaluation of the recommendations is based on recall metric where we evaluate recall score for *top-10* recommendations. Figure 10 shows recall@10 scores for all the algorithms with all possible comparisons. We can observe similar pattern here. The best result for recall@10 is achieved by using TF-IDF (title only) approach with overall feedback score user profile, and the worse performance is from LDA (content) approach with reading statistics user profile.

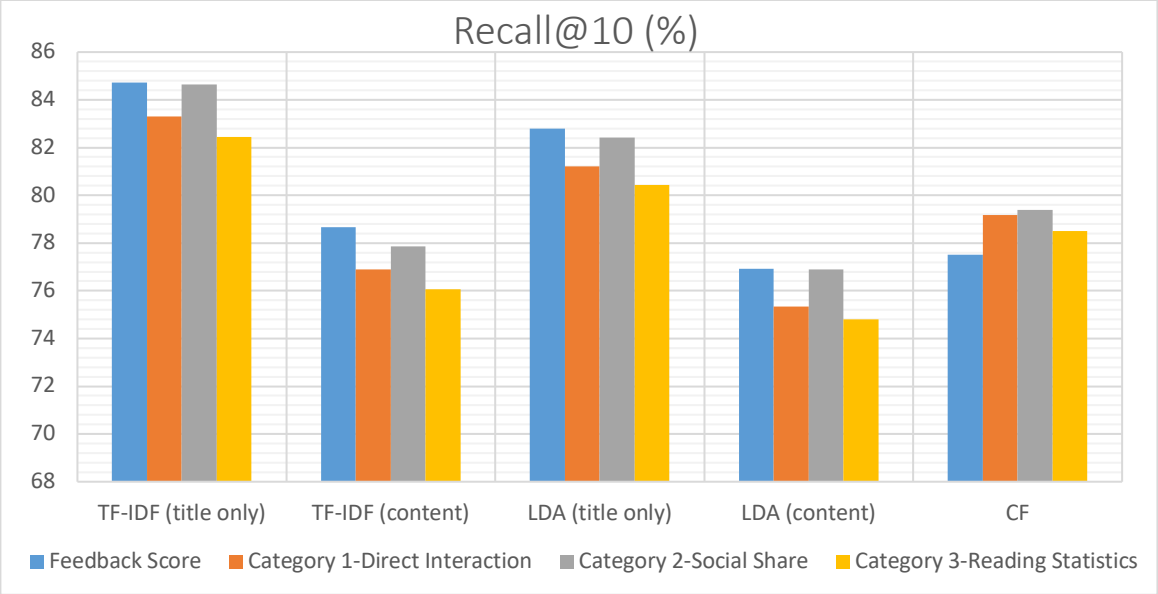

Figure 10: Evaluation of recommendations using Recall@10

- ***Recall@20:*** Recall@20 is computed by using same recall metric formula but we have measured recall score for top-20 recommendations this time. Figure 11 shows evaluation of recommendations using recall@20. Again, we observe similar patterns in terms of comparison between different approaches. Also, we can see that all the recall values are greater than 90%. It shows that when recommending top-20 results majority of the relevant posts can be identified by our algorithms.

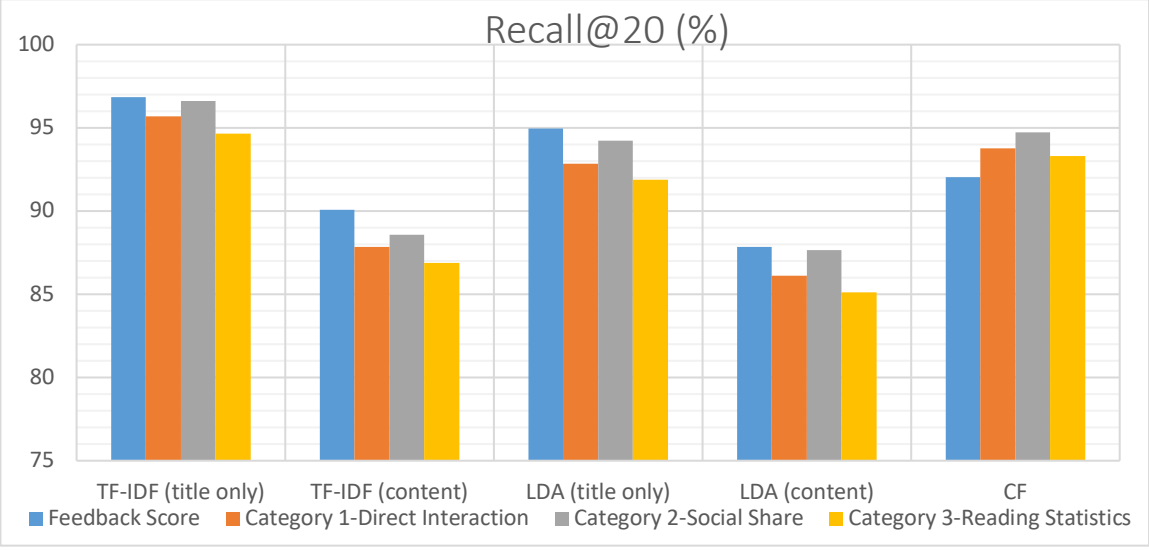

Figure 11: Evaluation of recommendations using Recall@20

From Figure 10 and Figure 11 we can compare between recall@10 and recall@20 for all the approaches. The figures depict that recall@10 values and recall@20 values show relatively same pattern for content-based approaches (TF-IDF and LDA) and for CF approach.

- ***F1-score@10:*** Figure 12 shows evaluation of recommendations using f1-score@10. As f1-score is harmonic mean of precision and recall, and therefore we are getting balanced results for precision@10 and recall@10.

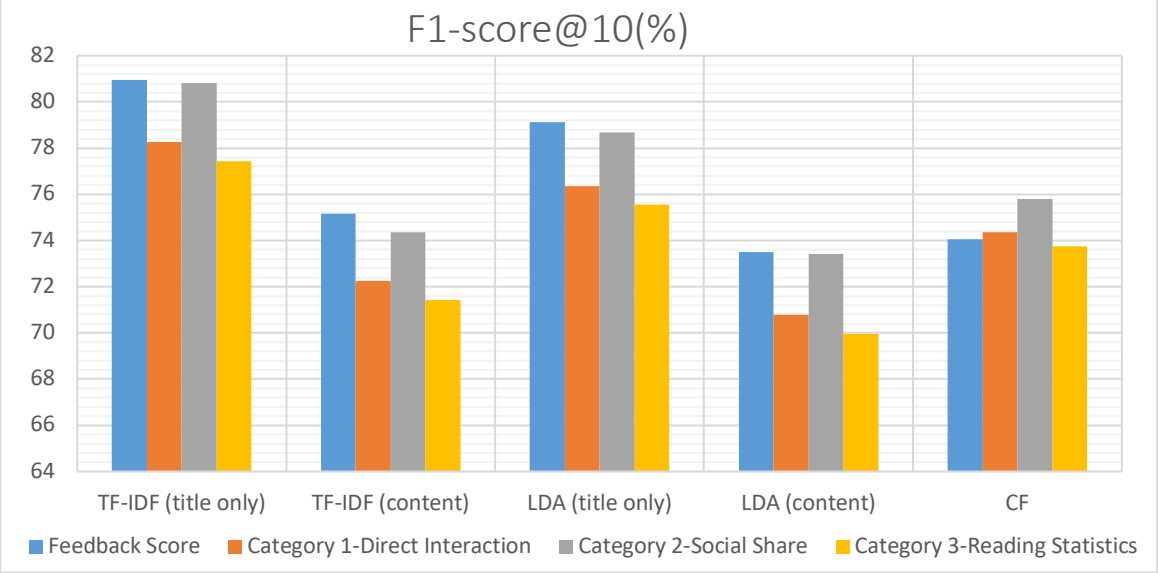

Figure 12: Evaluation of recommendations using F1-score@10

- ***F1-score@20:*** Similarly, we have measured f1-score for top-20 recommendations for all the approaches. Figure 13 shows evaluation using f1-score@20.

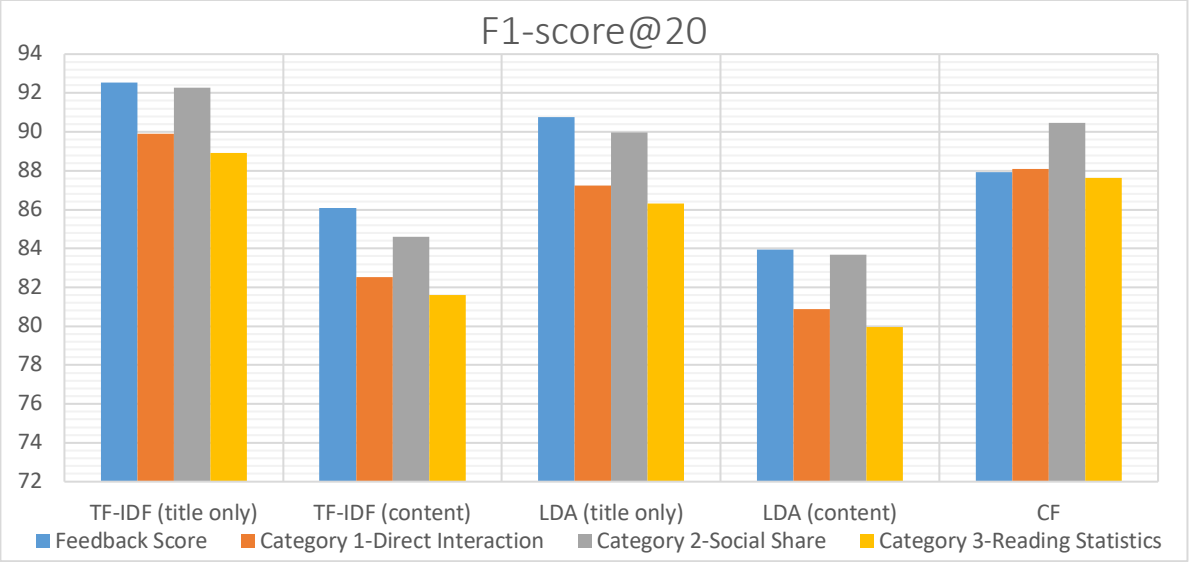

Figure 13: Evaluation of recommendations using Recall@20

From Figure 12 and Figure 13 we can provide a comparison between f1-score@10 and f1-score@20 for all the approaches. The figures depict that the f1-score@10 values and f1-score@20 values show comparatively similar pattern for content-based approaches (TF-IDF and LDA) and for CF approach.

4.5.2 Results Comparison and Discussion

The evaluation results for this overall model can be compared in many ways. Here we are considering three major ways to compare them: content profile based, user profile based, and approach based (content-based approach and collaborative filtering approach).

First, we are going to compare all evaluation results on the basis of content-profiles. As we have already mentioned that there are total four different content profiles, namely *TF-IDF (title only), TF-IDF (content), LDA (title only)* and *LDA (content)*. We are achieving better results from *TF-IDF (title only)* profile as compared to *LDA (title only)* profile. Similarly, we achieve better results from *TF-IDF (content)* profile as compared to *LDA (content)* profile. The result shows that TF-IDF approach is performing better than LDA approach for the content profile

generation for this CMS dataset. The possible reason behind this result is that LDA provides topic-based clustering with predetermined number of topics while in TF-IDF there is no such limitation. For LDA model, we have also tried different numbers of topics, for example: 10, 15 and 20. And we get best results by using 15 as the number topics, but still *TF-IDF* outperforms the *LDA* model. The TF-IDF divides post corpus at term level while LDA divides post corpus at topic level. For this CMS dataset, term level division works better than topic level division because the dataset consists of less varieties of the posts. The discussed topics in these posts are relatively concentrated. All posts are company/organization related, which are quite focused on a few topics. LDA probably works better if the dataset is bigger and there is a variety of topics in the dataset. Second observation we can make on the results is that *TF-IDF (title only)* and *LDA (title only)* profiles work better than *TF-IDF (content)* and *LDA (content)* profiles. The main reason is that title contains specific and key information about the post and content contains the whole article which sometimes may have distracted the key messages. Again, due to the relatively concentrated topics, recommending posts to the users based on *titles only* are more useful for this CMS dataset as compared to the recommendations based on whole *content*.

Second, we compare all the evaluation metrics on the basis of user profiles. We have in total four different categories of user profiles, namely *Feedback Score, Direct Interaction, Social Share* and *Reading Statistics*. The profiles are generated based on users' historical data. The result shows that for content-based approaches (TF-IDF and LDA), *Feedback Score* profile is performing the best among all four profiles and for collaborative filtering-based approach, *Social Share* profile performs better than other profiles. If we consider three profile categories: *Social Share, Direct Interaction* and *Reading Statistics,* the *Social Share* category performs better than other two, second best performance is from *Direct Interaction* profile and last one is *Reading*

Statistics profile. The major reason behind this result is that there are more types of social shares in this category. In other words, we can say in this category, we have more types of user/post interactions as compared to the other categories and data is largely shared by users on their social networking websites. As a comparison, *Reading Statistics* profile has only two types of interactions. Therefore, *Social Share profile* works better than *Direct Interaction* profile and *Reading Statistics* Profile. While comparing *Social Share* profile with *Feedback Score* profile, the overall *Feedback Score* profile works slightly better for content-based recommendation approaches. For collaborative filtering recommendation approach, *Social Share* category works better. The possible reasons behind this result are described as follows. First, for collaborative filtering approach, types of interactions play an important role for providing recommendations which are maximum in *Social Share* profile and when we aggregate all the categories together, the poor performance from other two categories have negative impact on overall result. Second, it is easier and more accurate to predict a *Social Share*, but less accurate to predict *Direct Interactions* or *Reading Statistics*.

Third, we compare all the evaluation metrics based on approaches. We used *content-based approach* and *collaborative filtering approach* for generating recommendations. In *content-based approach*, we used TF-IDF and LDA algorithms and in *collaborative filtering approach*, we used Matrix Factorization algorithm. The result shows that we are getting better results from *content-based approach* as compared to *collaborative filtering approach* for all the evaluation metrics. There can be many reasons for this result, one of the likely reasons is that this is a small CMS dataset with not many users, posts and their interactions. The density of matrix is 3.5% for this CMS dataset and the *collaborative filtering approach* works better if dataset is bigger and when matrix is dense. This analysis shows that content-profiles play a big role on

generating recommendations as compared to using only the user feedback data. In *content-based approach,* we consider content-profile as well as user profile for generating recommendations while in *collaborative filtering approach*, we consider feedback scores only for the recommendations. So, these results show that for this particular dataset combination of content-profile and user profile generates better recommendations as compared to providing recommendations only based on feedback scores using collaborative filtering.

4.6 Summary

In this chapter, we have explained the CMS dataset we used in our experiment, the experiment design and we have provided details of the implementation. By analyzing and evaluating our results, we showed that our proposed approaches provide good recommendations for the users of this CMS dataset. By comparing the results obtained by using content-based approach with that of the collaborative filtering approach, we can see that content-based approach work better for this CMS dataset.

CHAPTER 5
CONCLUSIONS AND FUTURE WORK

This chapter begins by summarizing the results of this research. After that major findings and contributions of research are presented. At the end, we present some potential directions for future work for generating content recommendations.

5.1 Summary of Results

The research evaluates the effectiveness of generating content profiles and utilizing user feedback data for providing content recommendations. It generates recommendation for the effective content suggestion to individual users. In total we generated four types of content profiles: *TF-IDF (title only), TF-IDF (content), LDA (title only), LDA (content)* and four types of user profiles: *Feedback Score, Direct Interaction, Social Share* and *Reading Statistics*. We matched all content profiles with all user profiles by using Cosine Similarity algorithm for content-based recommendations. We also generated collaborative filtering recommendations for all users by using matrix factorization technique with user feedback scores. Furthermore, we have compared the content-based recommendations against collaborative filtering recommendations to see which recommendations are more effective for content suggestions.

The TF-IDF based content profiles are compared with LDA based content profiles and analysis shows that TF-IDF based content profiles outperformed LDA based content profiles. The *TF-IDF (title only)* content profile with *Feedback Score* user profile was shown to be very effective on the tested CMS dataset and produced high *precision@k, recall@k* and *f1-score@k* values. The poor results appeared to have been generated from the worst profile combination – *LDA (content)* content profile with *Reading Statistics* user profile. This behavior is due to the

fact that for this CMS dataset, topic modelling is not that much effective as compared to TF-IDF modelling.

The content-based recommendations are then compared with collaborative filtering-based recommendations and results show that content-based recommendations work better for this CMS dataset as compared to collaborative filtering-based recommendations. Overall, the evaluation result shows that the combination of *TF-IDF (title only)* content profile with *Feedback Score* user profile generates the best recommendations for the users using this dataset.

5.2 Major Research Findings

The accuracy of recommendation algorithms and effectiveness of user or content profiling technique are highly depending on the CMS dataset. Our results show that:

- Including more information doesn't mean the better result. As we can see, *Title Only* category works better for this CMS dataset and the simpler model *TF-IDF* gives better result as compared to *LDA* model and *CF-MF* model;
- Different types of user interaction data may have different impacts on recommendation accuracy, therefore it is necessary to explore and investigate the optimal way of using them in a recommender system;
- For a relatively small and focused dataset, it is important to include all types of information to achieve a good result. As we can see, *CF-MF* model doesn't include content of posts and therefore its performance is worse than the content-based filtering model.

5.3 Major Research Contributions

- **Content profile generation:** This research has explored different ways of generating content profiles: using the full text vs. the title only as the base for content profile, using

term-level information vs. topic-level information to represent content. Four different content profiles are tested on their effectiveness for facilitating recommendations.

- **Utilizing user feedback data:** This research work also presents a unique way of utilizing user feedback data. First, we have extracted the feedback information that shows user/post interactions and then we have categorized those interactions based on the nature of interaction. We have calculated feedback score based on the interaction categories and tested their effectiveness of improving recommendation accuracy.
- **User profile generation:** By considering different categories of user feedbacks, we have generated four different user profiles for the same CMS dataset. Then, we compare all these user profiles to evaluate which user profile has greater impact on generating content-based recommendations. It is important that user preference should be well understood by the system and user profiles should reflect user's feedback score correctly.
- **Recommendations using CMS Dataset:** The proposed recommendation algorithms have been evaluated on a real CMS dataset. We have shown that it is necessary to compare different recommendation approaches for a given domain and dataset. A popular approach that works effective for many datasets may not work equally well in all datasets. It is very important to test and compare their performances to choose the best one.

5.4 Future Work

While some of the results presented in this thesis are appealing, a number of areas for future work exist that may greatly improve the effectiveness of the system.

To validate the proposed approaches in a different way, we could also compare the automatically generated user interaction weights with those that are pre-defined in the dataset. Firstly, this could be done by using a regression model or by using a classification algorithm to

find optimal weights based on the user interactions. Secondly, a set of users could be asked to provide weights explicitly to the interaction categories and generate an explicit feedback score-based user profile. After getting the weights from these approaches, a comparison could be made between our current results and the new ones.

In our proposed model, we extracted user interactions from the CMS dataset for user profile modeling, but in future we can add attributes which show user's demographic information as well such as age, gender, nationality and location. And based on this information, we can categorize users to generate new user profiles for providing recommendations.

In the implementation part of our proposed approach, we used a dataset which consists of users, posts and user/post interactions. The proposed algorithm can be used on other datasets in a different domain. Any item which can be represented via attributes can be used such as news, articles or movies. For example, if the proposed algorithm is implemented for the news domain, the content attributes could be news id, news category, headline, content, news source, etc., and user attributes could be news/user interactions such as likes, shares, comments, reading progress.

In our current approach, we compared content-based recommendations with collaborative filtering recommendations. In future, we can combine both of these recommendations to generate hybrid recommendations. And based on that we can compare hybrid recommendations with our current recommendations.

Personalized Recommendation is another good way of providing recommendations for individual users. In future, we can propose a model for personalized recommendations for each user instead of just generating content-based and collaborative filtering recommendations for all. Also, we can design an interface by which users can interact with the proposed system effectively.

APPENDICES

A.1 Sample Dataset

This appendix contains information about the CMS dataset structure and a visualization of the dataset showing its field names with their field values.

The first table shows information about the posts such as post ids, title of the post, content of the post, date when post was created, date when post was modified, post expiration date, status of the post, media (if post include any media information), post is editable or not and whether the post is shared on Twitter, Facebook or LinkedIn.

post_id	title	content	created	modified	start	expiration	status	media	editable	twitter	facebook	linkedin	category
275	social netwo	to assist cdn	2013-03-07	2013-03-13	NULL	NULL	2	media-175.p	0	1	1	1	NULL
276	startup tips	Testing for S	2013-03-07	2013-03-07	NULL	NULL	2	media-176.p	0	1	0	0	NULL
277	Microsoft	MaRS comm	2013-03-12	2013-03-13	NULL	NULL	2		1	1	1	1	NULL

The second table shows information about the users, including user ids (unique id provided to each user), post ids showing posts which were read by that user id, date showing when post was read by the user and information such as showing share/ impression/ retweet/ favorite/ reply/ clickthrough depicts interactions made by the user on that post id on Twitter, Facebook or LinkedIn.

id	post_id	user_id	date	twitter_share	twitter_impression	twitter_retweet	twitter_favorite	twitter_reply	twitter_clickthrough	facebook_share	facebook_impression
1	282	157	2013-02-27	0	0	0	0	0	0	1	4
2	256	146	2013-02-27	1	1526	0	0	0	23	1	114
3	259	146	2013-03-03	0	0	1	0	0	43	1	114
4	263	157	2013-03-05	1	13	0	0	1	54	1	70
5	261	163	2013-03-06	1	8403	1	0	0	12	1	269
6	251	164	2013-03-13	0	0	0	0	0	13	1	112
7	257	159	2013-02-28	0	0	0	0	1	56	1	454
8	272	146	2013-03-07	1	1574	1	1	0	6	1	114
9	255	157	2013-02-27	1	6	0	0	0	71	1	454
10	255	146	2013-02-27	1	1532	1	0	1	45	1	114
11	259	159	2013-03-01	1	1505	0	0	0	67	1	454
12	262	163	2013-03-06	1	8402	1	0	0	43	1	269
13	260	164	2013-03-06	0	0	0	1	1	1	1	113
14	274	146	2013-03-07	1	1575	1	0	0	5	1	114
15	256	164	2013-03-13	0	0	0	0	0	3	2	224
16	294	164	2013-03-14	0	0	1	0	0	0	1	112
17	300	164	2013-03-18	0	0	0	0	0	78	1	112
18	325	159	2013-03-21	1	1267	0	0	0	4	1	453
19	337	159	2013-03-26	1	1277	1	0	0	6	1	454

A. 2 Detailed Experimental Results

This appendix contains experiment result tables for Precision@10, Recall@10, F1-score@10, Precision@20, Recall@20 and F1-score@20, corresponding to the result graphs shown in Chapter 4.

Precision@10

Algorithms	Feedback Score	Category 1 Direct Interaction	Category 2 Social Share	Category 3 Reading Statistics
TF-IDF (title only)	77.52	73.80	77.32	73.00
TF-IDF (content)	71.98	68.12	71.14	67.34
LDA (title only)	75.76	72.00	75.30	71.22
LDA (content)	70.38	66.74	70.26	65.96
CF	70.92	70.12	72.54	69.52

Recall@10

Algorithms	Feedback Score	Category 1 Direct Interaction	Category 2 Social Share	Category 3 Reading Statistics
TF-IDF (title only)	84.72	83.31	84.63	82.44
TF-IDF (content)	78.66	76.90	77.86	76.05
LDA (title only)	82.79	81.20	82.42	80.43
LDA (content)	76.91	75.34	76.90	74.79
CF	77.50	79.16	79.40	78.51

F1-score@10

Algorithms	Feedback Score	Category 1 Direct Interaction	Category 2 Social Share	Category 3 Reading Statistics
TF-IDF (title only)	80.96	78.26	80.81	77.43
TF-IDF (content)	75.17	72.24	74.35	71.43
LDA (title only)	79.12	76.36	78.69	75.55
LDA (content)	73.50	70.78	73.42	69.96
CF	74.06	74.36	75.81	73.74

Precision@20

Algorithms	Feedback Score	Category 1 Direct Interaction	Category 2 Social Share	Category 3 Reading Statistics
TF-IDF (title only)	88.60	84.76	88.28	83.82
TF-IDF (content)	82.42	77.82	80.94	76.94
LDA (title only)	86.90	82.26	86.08	81.36
LDA (content)	80.38	76.28	80.08	75.38
CF	84.2	83.06	86.56	82.6

Recall@20

Algorithms	Feedback Score	Category 1 Direct Interaction	Category 2 Social Share	Category 3 Reading Statistics
TF-IDF (title only)	96.83	95.68	96.62	94.66
TF-IDF (content)	90.07	87.85	88.59	86.89
LDA (title only)	94.97	92.86	94.22	91.89
LDA (content)	87.84	86.11	87.65	85.13
CF	92.02	93.76	94.74	93.29

F1-score@20

Algorithms	Feedback Score	Category 1 Direct Interaction	Category 2 Social Share	Category 3 Reading Statistics
TF-IDF (title only)	92.53	89.89	92.26	88.91
TF-IDF (content)	86.07	82.53	84.59	81.61
LDA (title only)	90.75	87.24	89.96	86.30
LDA (content)	83.94	80.89	83.69	79.96
CF	87.93	88.08	90.46	87.62

REFERENCES

[1] Lops, P., de Gemmis, M., & Semeraro, G. (2011). Recommender systems handbook. Springer, pp. 1-186.

[2] Goldberg, D., Nichols, D., Oki, B. M., & Terry, D. (1992). Using collaborative filtering to weave an information tapestry. Communications of the ACM, ACM, 35 (12), pp. 61–70.

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

[4] Ricci, F., Rokach, L. & Shapira, B. (2011). Introduction to Recommender Systems Handbook, Springer, pp. 1-35.

[5] Rich, E. (1979). User Modeling via Stereotypes. In Cognitive Science: A Multidisciplinary Journal, ScienceDirect, Vol. 3, No. 4, pp. 329-354.

[6] Jannach, D., Zanker, M., Felfernig, A. & Friedrich, G. (2011). Recommender Systems: An Introduction. Cambridge University Press.

[7] Brahim, H. F. F. B. O. (2012). A comparison study of some algorithms in Recommender Systems, 2012 Colloquium in Information Science and Technology, IEEE Explore, pp. 130-135.

[8] Montaner, M., López, B. & De La Rosa, J. L. (2003). A Taxonomy of Recommender Agents on the Internet. Artif. Intell. Rev. Springer, Springer, 19 (4), pp. 285--330.

[9] Schafer, J. B., Frankowski, D., Herlocker, J. & Sen, S. (2007). The adaptive web. In P. Brusilovsky, A. Kobsa & W. Nejdl (ed.), Springer-Verlag, Springer, pp. 291--324.

[10] Amatriain, X., Pujol, J. M., & Oliver, N. (2009). I like it... i like it not: Evaluating user ratings noise in recommender systems. In User modeling, adaptation, and personalization, Springer, pp. 247–258.

[11] Salton, G. (1989). Automatic text processing: the transformation, analysis, and retrieval of information by computer. Addison-Wesley Longman Publishing Co., Inc. ISBN: 0-201-12227-8.

[12] Turney, P. D. & Pantel P. (2010). From Frequency to Meaning: Vector Space Models of Semantics. Journal of Artificial Intelligence Research, ACM, 37(1):141.

[13] Blei, D. M., Ng, A. Y. & Jordan, M. I. (2003). Latent Dirichlet Allocation. The Journal of machine Learning research, vol. 3, pp. 993–1022.

[14] Graber, J. H. B., Blei, D. M. & Zhu, X. (2007). A Topic Model for Word Sense Disambiguation. In EMNLP-CoNLL, Association for Computational Linguistics, pp. 1024-1033.

[15] Pritchard, J. K., Stephens, M. & Donnelly, P. (2000). Inference of Population Structure Using Multilocus Genotype Data. Genetics, NCBI, 155(2), pp. 945-959.

[16] Huelsenbeck, J. P., Jain, S., Frost, S. W. D. & Pond, S. L. K. (2006). A Dirichlet process model for detecting positive selection in protein-coding DNA sequences. Proceedings of the National Academy of Sciences, 103(16), PNAS, pp. 6263-6268.

[17] Shivashankar, S., Srivathsan, S., Ravindran, B. & Tendulkar, A. V. (2011). Multi-view methods for protein structure comparison using Latent Dirichlet Allocation. Bioinformatics, 27(13), NCBI, pp. 61-68.

[18] Wei, X. & Croft, W. B. (2006). LDA-based Document Models for Ad-hoc Retrieval. In Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '06, New York, USA, ACM. pp. 178-185.

[19] Mahmood, T. & Ricci, F. (2009). Improving recommender systems with adaptive conversational strategies. In Proceedings of the 20th ACM conference on Hypertext and hypermedia, ACM, pp. 73-82.

[20] McSherry, F. & Mironov, I. (2009). Differentially private recommender systems: building privacy into the net. In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, pp. 627-636.

[21] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P. & Riedl, J. (1994). GroupLens: an open architecture for collaborative filtering of netnews. Proceedings of the 1994 ACM conference on Computer supported cooperative work. ACM, pp. 175-186.

[22] Shardanand, U. & Maes, P. (1995). Social Information Filtering: Algorithms for Automating "Word of Mouth". In ACM Press, pp. 210-217.

[23] Hill, W., Stead, L., Rosenstein, M. & Furnas, G. (1995). Recommending and evaluating choices in a virtual community of use. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM Press/Addison-Wesley Publishing Co., pp. 194-201.

[24] Linden, G., Smith, B. & York, J. (2003). Amazon.com Recommendations: Item-to-Item Collaborative Filtering. IEEE Internet Computing 7 (1), pp. 76-80.

[25] Ping, H. Q. & Ming, X. (2012). Research on Several Recommendation Algorithms. Procedia Engineering 29 (0), ScienceDirect, pp. 2427 - 2431.

[26] Koren, Y., Bell, R. & Volinsky, C. (2009). Matrix Factorization Techniques for Recommender Systems. IEEE Computer, IEEE, 42 (8), pp. 30-37.

[27] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K. & Harshman, R. (1990). Indexing by latent semantic analysis. Journal of The American Society for Information Science, Willey Online Library, 41 (6), pp. 391-407.

[28] Sarwar, B. M., Karypis, G., Konstan, J. A. & Riedl, J. T. (2000). Application of Dimensionality Reduction in Recommender Systems: A case study. In Web KDD Workshop at the ACM SIGKKD.

[29] Ouaftouh, S., Zellou, A. & Idri, A. (2017) UPCAR: User Profile Clustering based Approach for Recommendation. ACM, ICETC, December 20–22, Barcelona, Spain.

[30] Weiß, D., Scheuerer, J., & Wenleder, M. (2008). A User Profile-based Personalization System for Digital Multimedia Content. DIMEA'08, September 10-12, 2008, Athens, Greece, ACM, 978-1-60558-248-1/08/09.

[31] Pazzani, M. J. (1999). A Framework for Collaborative, Content-Based and Demographic Filtering. Artif. Intell. Rev. Springer, 13 (5-6), pp. 393-408.

[32] IJntema, W., Goossen, F., Frasincar, F. & Hogenboom, F. (2010). Ontology-Based News Recommendation. EDBT, March 22–26, 2010, Lausanne, Switzerland. ACM, 978-1-60558-945-9/10/0003.

[33] Frasincar, F., Borsje, J., Levering, L. (2009). A Semantic Web-Based Approach for Building Personalized News Services. International Journal of E-Business Research 5(3), pp. 35–53.

[34] Zhao, G., Liu, Y., Zhang, W. & Wang, Y. (2018). TFIDF based Feature Words Extraction and Topic Modeling for Short Text. ICMSS, January 13–15, 2018, Wuhan, China, ACM.

[35] Shao, B. & Yan, J. (2017). Recommending Answerers for Stack Overflow with LDA Model. In Proceedings of Chinese CSCW '17, Chongqing, China, September 22-23, 2017, ResearchGate.

[36] Zhao W X, Jiang J & Weng J, et al. (2011). Comparing Twitter and Traditional Media Using Topic Models[M], Advances in Information Retrieval. Springer Berlin Heidelberg, Springer, pp. 338-349.

[37] Vasilescu, B., Filkov, V., Serebrenik, A. (2013). StackOverflow and GitHub: Associations between software development and crowdsourced knowledge, Social Computing (SocialCom), 2013 International Conference on. IEEE, pp. 188-195.

[38] Wilson, J., Chaudhury, S. & Lall, B. (2014). Improving Collaborative Filtering based Recommenders using Topic Modelling. IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT).

[39] Riedl, J. & Konstan, J. (1998). Movielens dataset.

[40] Cosley, D., Lam, S. K., Albert, I., Konstan, J. A. & Riedl, J. (2003). Is seeing believing? how recommender system interfaces affect users' opinions. In Proceedings of the SIGCHI conference on Human factors in computing systems, ACM, pp. 585–592.

[41] Littlestone, N. & Warmuth, M. K. (1994). The weighted majority algorithm. Inf. Compute. 108 (2), ScienceDirect, pp. 212-261.

[42] Balabanović, M. & Shoham, Y. (1997). Fab: content-based, collaborative recommendation. Commun. ACM, 40 (3), pp. 66-72.

[43] Scikit-Learn library. http://scikit-learn.org/stable/

[44] Mallet. MAchine Learning for LanguagE Toolkit. http://mallet.cs.umass.edu/

[45] NLTK. Natural Language Toolkit. https://www.nltk.org/

[46] Carenini, G. & Sharma, R. (2004). Exploring more realistic evaluation measures for collaborative filtering. In Proceedings of the 19th national conference on Artificial Intelligence, AAAI Press, pp. 749-754.

[47] Zagheli1, H. R., Zamani2, H. & Shakery, A. (2017) A Semantic-Aware Profile Updating Model for Text Recommendation. ACM, Rec Sys '17, August 27–31, 2017, Como, Italy.

[48] Salehi, M. & Kmalabadi, I. N. (2012). A Hybrid Attribute-based Recommender System for E-Learning Material Recommendation. International Conference on Future Computer Supported Education. Published by Elsevier B.V.

[49] Salton, G. & McGill, M. J. (1983). Introduction to modern information retrieval. ACM Digital Library.

[50] Gunawardana, A. & Shani, G. (2009). A Survey of Accuracy Evaluation Metrics of Recommendation Tasks. J. Mach. Learn. Res., ACM, 10, pp. 2935-2962.