A Survey of Recommendation Systems and Performance Enhancing Methods

Liao Ke

M.Sc. Thesis
UNIVERSITY OF HELSINKI
Department of Computer Science

Helsinki, January 17, 2019

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty Faculty of Science	Laitos — Institution — Department Department of Computer Science

Tekijä — Författare — Author
Liao Ke

Työn nimi — Arbetets titel — Title
A Survey of Recommendation Systems and Performance Enhancing Methods

Oppiaine — Läroämne — Subject
Computer Science

Työn laji — Arbetets art — Level M.Sc. Thesis	Aika — Datum — Month and year January 17, 2019	Sivumäärä — Sidoantal — Number of pages 58

Tiivistelmä — Referat — Abstract

With the development of web services like E-commerce, job hunting websites, movie websites, recommendation system plays a more and more importance role in helping users finding their potential interests among the overloading information. There are a great number of researches available in this field, which leads to various recommendation approaches to choose from when researchers try to implement their recommendation systems.

This paper gives a systematic literature review of recommendation systems where the sources are extracted from Scopus. The research problem to address, similarity metrics used, proposed method and evaluation metrics used are the focus of summary of these papers. In spite of the methodology used in traditional recommendation systems, how additional performance enhancement methods like machine learning methods, matrix factorization techniques and big data tools are applied in several papers are also introduced.

Through reading this paper, researchers are able to understand what are the existing types of recommendation systems, what is the general process of recommendation systems, how the performance enhancement methods can be used to improve the system's performance. Therefore, they can choose a recommendation system which interests them for either implementation or research purpose.

ACM Computing Classification System (CCS):

H.3.3 [Information Search and Retrieval]: Information filtering

Avainsanat — Nyckelord — Keywords
literature review, recommendation systems, content based filter, collaborative based filter, hybrid filter

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Contents

1 Introduction

Personalized services have been quite popular nowadays. Many of personalized services are based on recommendation systems. There are wide applications of recommendation systems. In E-commerce, recommendation system can provide customers with their potential interest products [23]. Video sites generate a list of TV programs to help users select what they would like to watch [25]. Job recommendation systems can help users find their interesting roles [2]. Audio services provide users their potential liking songs [12].

There are three major types of recommendation systems: collaborative based, content based and hybrid filters. Content based filter makes recommendation by finding similar items to items which are liked by the users, collaborative filter makes recommendation by finding similar users' liked items. They both have limitations, especially when there are cold start problems. Content based filter performs poorly when there are limited items in the dataset, while collaborative filter makes poor recommendation results when there are limited number of users in the dataset. That is why some recommendation system propose hybrid filter which combines the two filters.

Despite that there are many different researches using different filters, there are also various performance enhancement methods used in recommendation system to improve its performance. Machine learning methods like NLP can help extract the feature descriptor [22, 16], k-means can help cluster the users or items to simplify the computation of similarities [20]. K nearest neighbors can help select most similar users or items to the target user or item [27, 9]. Matrix factorization methods can also help recommendation system since it can reduces the dimension of overly complicated feature descriptors which is a problem existing for big dataset [12, 18, 5]. It decomposes the original user item rating matrix to two matrices, whose product is the resulting lower dimension matrix which can approximate the original rating matrix [12]. During the process, big data framework tools like Hadoop and Spark can be used to perform the matrix decomposition in parallel while it runs the process of computation of two matrices in separate [4, 14, 24]. They can also be used to speed up the computation process of similarity metrics and making predictions.

Performance of recommendation systems can be evaluated in various approaches based on the difference between the predicted ratings and real ratings. It includes MAE, which is the mean of absolute errors of ratings [9], NMAE, which is the normalized version of MAE[9], RMSE, which gives more penality to the prediction errors[9], Precision and Recall which is based on finding the overlapping items between the recommendation list and users liked item list [25], and F-measure which combines the Precision and Recall.

Developing a suitable recommendation system is difficult if researchers have little idea about what is the general process of making recommendations, what are the approaches for each step in the process, what problems they might face while developing a recommendation system and how they can be addressed. This leads to the research questions of this paper: For each category of recommendation systems, (1) What are the trends of recommendation systems when implementing individual step for making recommendations and how they are applied? (2) What research problems they are trying to address? And the objective of this paper is to (1) Identify approaches to implement each step in the process, including for example similarity metrics, evaluation metrics, and some performance enhancement approaches; (2) Identify trends of research problems in recommendation systems. By identifying the existing problems, researchers can think about what are the topics to be explored to improve recommendation systems. And by identifying the approaches, researchers can get an idea of what are the approaches that are used by other papers hence they can choose a approach to implement or develop a new approach based on previous approaches.

In order to answer the above questions and achieve the above objectives, a systematic literature review is carried out. The first step is to get a list of papers which includes all three types of recommendation systems from the paper search engine, then Inclusion and Exclusion Criteria are applied to filter the results, which leads to the final paper list in Table 1. By going through the list of papers in Table 1 about these recommendation systems, the general process of making recommendations is figured out and present in Figure 1. In order to answer research question 1 and 2, the summary of each paper is made in Section 4 in the aspect of what research question it has, how the item or user features are represented, what similarity metric is used to measure the similarity between user or item features, how the ratings of items are predicted, how the final prediction is made, what dataset is used for experimenting the proposed approaches, and whether the proposed approach improves the state of art approaches. For those papers which do not follow this process, the summary is made differently. For example, if the paper only gives introduction to a specific approach, the summary also focuses on introducing the new approach. Besides, most of these papers have background section which introduces the basic approaches before introducing their proposed approaches, therefore, the Background section 3 which introduces these basic approaches is also included in this paper to help readers get familiar with the basic approaches before they get to know more advanced approaches introduced in the Results section 4.

This paper is organized as follows: Section 2 gives introduction to the background knowledge to help understand the summary of papers given in Section 4. It introduces types of recommendation systems, what are the basic similarity metrics to compute the similarity between users and items, how the performance enhancement approaches work and what are the basic performance evaluation methods. Section 3 introduces the method used for finding relevant literature and how these papers are chosen to form the final paper list 1. Section 4 provides a summary of final paper list, in the aspect of what are the research problems they are trying to solve, what are the similarity metrics used in their systems, how the ratings are predicted and how the final recommendation list is generated, and what are the experimental results. For those papers which do not follow the process that they only introduce a new method, how the method works is the focus of the summary. Finally, Section 5 draws a conclusion of the paper.

2 Background

This section gives introductions to basic concepts to help understand the review of list papers I found, including similarity metrics, performance evaluation methods that are used by recommendation systems.

2.1 Types of Recommendation systems

There are three types of recommendation systems: content based, collaborative based and hybrid methods which concatenate the two methods [3]. In content based recommendation system, the recommendations are made based on correlations between contents [3]. Items are represented with item vectors, then similarity based methods are used to compute the similarity between the item vectors and the item which are considered most similar to the target item, which can be an item which is like by the user, are recommended [3]. Apart from similarity based methods, there is also TF-IDF method which can determine the relative importance of recommended items [15].

Collaborative based recommendation systems make recommendations based on the relationship between the items and users' preferences [3]. There are two types of collaborative based methods, model and neighborhood based [3]. In model based approach, it constructs the user item rating matrix and generates a model from the matrix, then the recommendation is made based on the model's prediction result [3]. Neighborhood based approach also constructs a user item rating matrix and aims to predict the users' ratings on objects which are not rated by the user but considered similar to the objects which are liked by the user or are given ratings from similar users [3]. During this process, there are two approaches to find the similar objects or users, User Based Collaborative Filtering(UBCF) and Item based Collaborative Filter(IBCF) [3], where UBCF is focused on finding similar users to the target user and make recommendations from the items liked or given high ratings from the similar users to the target user [3], IBCF recommends similar objects to the object that is liked or given high ratings by the target user [3].

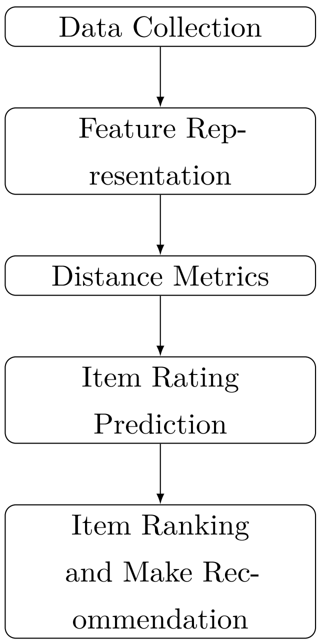

Figure 1: Process of Recommendation System

Since both content and collaborative based recommendation systems have its pros and cons, there are also quite many hybrid recommendation systems which utilize both these methods.

The process of recommendation system is shown in Figure 1. It first collects data by either crawling tool or using existing datasets like MovieLens. Then it selects features from users or items and represent them by either vectors or matrices, then it uses distance metrics to measure the similarity of users or items, after that, it predicts users' ratings on the items and generates recommendation by ranking the items based on the predicted ratings.

Different recommendation systems carry out these steps in different ways. Firstly items' or users' profiles are constructed by selecting features, while the type and number of features to be selected vary in each recommendation system based on the type of objects or topics that the researcher wants to focus on, the result of this process is feature vectors. Then the prediction of users' preferences can be made by either using similarity based methods or using some standalone methods like TF-IDF, which is commonly used in content based recommendation systems. The similarity between these feature vectors are computed using methods like Euclidean distance, cosine similarity, adjusted cosine similarity and so on. After that, prediction methods are used to rank the items based on the computed similarity or the standalone methods.

2.2 Similarity Metrics

Many recommendation systems use vectors to represent features extracted from users' or items' profiles, and they make the recommendation based on similarity metrics to find the most similar objects. There are many types of similarity metrics, here covers the most basic ones to help illustrate the idea of how the similarity are computed. Other similarity methods which are derived from the basics ones will be specifically explained in the results section.

Euclidean Distance

Euclidean distance method measures how close points lie to each other by the formula [9]

$$EuclideanDistance = \sqrt{(x_1 - y_1)^2 + \ldots + (x_N - y_N)^2} \tag{1}$$

Cosine Similarity

Cosine similarity uses the difference in the direction to calculate and the difference in angle is normalized to the interval $[-1, 1]$, where 1 implies the same direction and -1 means the opposite direction [4].

$$sim(A, B) = cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} \tag{2}$$

Correlation-based Similarity

Another similarity method which is widely used in user based collaborative filtering method to calculate the similarity between users is called Pearson correlation and it is given by [3]:

$$sim(a, b) = \frac{\sum_{p \in P}(r_{a,p} - \overline{r}_a)(r_{b,p} - \overline{r}_b)}{\sqrt{\sum_{p \in P}(r_{a,p} - \overline{r}_a)^2}\sqrt{\sum_{p \in P}(r_{b,p} - \overline{r}_b)^2}}, \tag{3}$$

where a, b represent user a and b, $r_{a,p}$ is the rating of user a for item p, P are the set of items who are rated by both a and b, and the computed similarity value is between -1 and 1 [3].

Adjusted Cosine Similarity

Derived from cosine-based method, adjusted cosine similarity metric takes the different rating scale between different customers into consideration, it is given by [23]:

$$sim_{i,j} = \frac{\sum_{u \in U}(R_{u,i} - \overline{R}_u)(R_{u,j} - \overline{R}_u)}{\sqrt{\sum_{u \in U}(R_{u,i} - \overline{R}_u)^2}\sqrt{\sum_{u \in U}(R_{u,j} - \overline{R}_u)^2}}, \tag{4}$$

Here the $\overline{R}_{u,i}$ is the rating of user i on item u, $\overline{R}_u$ is the average of the u-th user's ratings [23].

2.3 Performance Enhancement Approaches

2.3.1 Big Data Frameworks

Hadoop and Spark are the two common big data framework tools used to deal with large amount of data. They can also be used when running the collaborative or content based filter algorithms to improve the efficiency.

Hadoop is a distributed platform based on MapReduce operations so that large datasets can be processed in a cluster of computers [14]. It consists of Hadoop Distributed File System(HDFS) layer and MapReduce layer [14]. HDFS is a scalable and reliable file system that can store data and span large clusters of commodity servers [14]. In MapReduce, there is a master node and slave node, where the master node is responsible for: breaking down the dataset into blocks so that different block can be processed by different nodes at the same time, storing replication of input dataset, choosing block to run, assigning tasks to nodes and keeping track of tasks running on master and slave nodes [14]. MapReduce communicates data with HDFS and process it in parallel [14].

Spark is also a distributed platform but it is based on Resilient Distributed Datasets (RDD) which is memory and computational efficient [24]. Spark consists of Spark context, cluster manager, work node and executor [24].

Spark context is responsible for interaction between user logic and spark cluster, cluster manager manages resource and schedules clusters, work node is responsible for computational tasks, where executor executes tasks [24].

2.3.2 Matrix Factorization

Matrix factorization is used to find latent models of users and items, therefore, the rating of target user on target item can be predicted using inner product of corresponding latent models [18]. Here latent models are trained aiming at minimizing the sum-of-square-error loss function, regularization is used to prevent overfitting [18]. The traditional matrix factorization algorithm is singular value decomposition(SVD), which decomposes the user item rating matrix P of m rows and n columns into three matrices [18]:

$$P = U \times C \times V^T \tag{5}$$

where U is an orthogonal matrix of m rows and columns, C is a diagonal matrix of m rows and n columns, V is an orthogonal matrix of n rows and columns [18]. The k largest singular values in diagonal matrix C forms a new diagonal matrix C_k, then we can get a new rating matrix P_k which is considered most similar to original rating matrix [18]:

$$P_k = U_k \times C_k \times V_k^T \tag{6}$$

2.3.3 Machine Learning Methods

This section briefly introduces machine learning methods used in the extracted papers.

K Nearest Neighbors(KNN):

KNN is a popular machine learning method used for classification. To classify a new sample point, the algorithm finds the k points which has minimum distances to the new point, and the class of new point is the class which has greater number of neighbors. Distance metrics are used to compute the distance between the points, the most popular distance metric is Euclidean distance.

K-means:

In machine learning methods, for the dataset without any labels, clustering methods need to be used, k-means is one of the most popular clustering approaches.

At the beginning, it randomly selects k initial cluster centroids, then it cluster all the points to their nearest cluster centroids based on Euclidean distance. Then it computes the mean of each cluster to be new cluster centroids, then it re-cluster all the points. This step iterates until the cluster centroids do not change.

Support Vector Machine(SVM):

SVM continually adjusts a hyperplane for classification or regression, the points which has minimum Euclidean distance to the hyperplane are the support vectors, and the minimum distance is the margin which SVM tries to minimize. SVM is sensitive to the kernel selection, there are linear kernel, Gaussian kernel, polynomial kernel and so on.

Natural Language Processing(NLP):

NLP is used for processing text and help machine understand human language, it includes bag of words model, TF-IDF model and RNN based models.

TF-IDF [2] is widely used in content based recommendation systems. In TF-IDF, TF is the term frequency which is the frequency of a term in a document [2], IDF is the invert document frequency, which is the log of the division of total number of documents and the number of documents that include the term [2]. TF-IDF is the product of TF and IDF [2].

Convolutional Neural Network(CNN):

CNN is widely used for extracting features from images or texts. It normally consists of convolution kernel, pooling layer, fully connected layers. Its structure and parameters can be changed to adapt the need and there are a great number of models for selection.

2.4 Performance Evaluation Methods

Performance Evaluation Methods are used to evaluate recommendation system's performance. Here are some commonly used evaluation metrics.

Mean Absolute Error (MAE):

MAE is defined by the following equation [9]:

$$MAE = \frac{\sum_{i=1}^{n} |x_i - y_i|}{n} \tag{7}$$

where x_i, y_i are the predicted rating and real rating from user i, n is the number of users [9].

Normalized Mean Absolute Error (NMAE) is the normalized version of MAE where error is expressed in percentage [9]:

$$NMAE = \frac{\sum_{i=1}^{n} |x_i - y_i|}{n(r_{max} - r_{min})} \tag{8}$$

where r_{max}, r_{min} are the maximum and minimum rating respectively.

Root Mean Squared Error (RMSE) :

Compared to MAE, RMSE gives more penalty to the prediction errors [9]:

$$RMSE = \frac{\sum_{i=1}^{n} (x_i - y_i)^2}{n} \tag{9}$$

Accuracy:

Accuracy is the number of correctly classified items divided by the total number of items.

Coverage:

Coverage is defined by [25]:

$$coverage(y, \hat{f}) = \frac{1}{n} \sum_{i=1}^{n} max_{j:y_{ij}=1} rank_{ij} \tag{10}$$

Where $rank_{ij} = |\{k : \hat{f}_{ik} \geqslant \hat{f}_{ij}\}|$.

Precision and Recall:

Precision denotes the ratio of correct recommended items to the number of recommended items [25], while Recall denotes the ratio of correct recommended items to the number of items that are liked like the user [25].

$$Precision = \frac{\sum_{u \in U} |R(u) \cap T(u)|}{\sum_{u \in U} |R(u)|} \tag{11}$$

$$Recall = \frac{\sum_{u \in U} |R(u) \cap T(u)|}{\sum_{u \in U} |T(u)|} \tag{12}$$

where U is the user set, $R(u)$ is the recommended item list generated for user u, $T(u)$ is the list of items which are liked by the user u [25].

F-measure:

F-measure combines the precision and recall metrics by the following equation [3]:

$$F - measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{13}$$

Mean Average Precision(MAP):

MAP is based on Average Precision, which is the average precision values for different validation datasets, MAP is the mean value of Average Precision values for different classes [16].

2.5 Abbreviations

This section gives the full list of abbreviations used in this paper and their full names.

KNN: K Nearest Neighbors

SVM: Support Vector Machine

FNN: Feedforward Neural Network

CNN: Convolutional Neural Network

VGG: Visual Geometry Group

MLP: Multi-layer Perceptron

LSTM: Long Short-Term Memory

NLP: Natural Language Processing

TF-IDF: term frequency–inverse document frequency

LDA: Linear Discriminant Analysis

LSI:	Latent Semantic Indexing
LSA:	Latent Sementic Analysis
PCA:	Principal Component Analysis
MAE:	Mean Absolute Error
NMAE:	Normalized Mean Absolute Error
RMSE:	Root Mean Squared Error
MAP:	Mean Average Precision
EER:	Equal Error Rate
NDCG:	normalized discounted cumulative gain
UBCF:	User Based Collaborative Filtering
IBCF:	Item based Collaborative Filter
HDFS:	Hadoop Distributed File System
RDD:	Resilient Distributed Datasets
OD:	origin-destination
ASR:	automatic speech recognition
API:	application programming interface
RFR:	relative feature rating
MRFF:	relative feature frequency
RFS:	relative feature score
UB-CF:	user based collaborative filtering
HUM-CF:	user based collaborative filtering with hybrid user model
ALS-WR:	alternating least-squares with weighted-λ-regularization
NALS-WR:	normalized ALS-WR
HTML:	Hypertext Markup Language
ID:	Identity Document
CRF:	conditional random field
SGNS:	Skipgram Negative Sampling

ROC: Receiver Operating Characteristic

MSD: Million Song Dataset

3 Methodology

This section describes the process of searching the paper, and paper selection based on inclusion and exclusion criteria, the extracted paper list and what kind of research problems these papers try to address and what similarity metrics, evaluation metrics, and additional performance enhancement approaches are used in these papers.

3.1 Literature Searching

Scopus is used here as the searching tool for relevant literature. It is a nice tool since it has a large database and provides a wide range of options for filtering out search results. The result of search process is retrieved on 18th of November in 2018.

In order to answer the research questions proposed in the introduction section, all three major types of recommendation systems (collaborative based, content based, and hybrid methods which utilize both collaborative and content based methods) need to be included in the result. Therefore, I applied first “(recommendation AND system) AND (collaborative OR content OR hybrid)” as my search string and I get 17921 document results, which is quite a huge number and it will be exhausted to go through such a large database. Therefore, many of them are filtered out by limiting the subject area to Computer Science, selecting Conference Paper and Article as the desired document type, choosing "Collaborative Filtering Recommendations", "Content-based Recommendation" and "Hybrid Recommender Systems" as keywords and limiting language to English. The number of documents I get is 490. The query I used is as follows:

TITLE-ABS-KEY ((recommendation AND system) AND (collaborative OR content OR hybrid)) AND (LIMIT-TO (SRCTYPE,"p") OR LIMIT-TO (SRCTYPE,"j")) AND (LIMIT-TO (DOCTYPE,"cp") OR

LIMIT-TO (DOCTYPE,"ar")) AND (LIMIT-TO (SUBJAREA,"COMP")) AND (LIMIT-TO (LANGUAGE,"English")) AND (LIMIT-TO (EXACTKEYWORD,"Collaborative Filtering Recommendations") OR LIMIT-TO (EXACTKEYWORD,"Content-based Recommendation") OR LIMIT-TO (EXACTKEYWORD,"Hybrid Recommender Systems"))

3.2 Inclusion and Exclusion Criterias

The inclusion and exclusion criterias are used during the process of paper selection.

Inclusion Criteria:

A paper is included if it:

- Gives an introduction to a specific recommendation system in either of these categories: collaborative based, content based, hybrid method which use both collaborative based and content based, a collaborative based or content based method with one or more additional performance enhancing methods including but not limited to association rule based, item response theory, machine learning methods.
- Is published in a conference or journal.
- Is written in English.

Exclusion Criteria:

A paper is excluded if it:

- Does not introduce a recommendation system.
- Gives an introduction to a recommendation system which does not belong to the categories as described in the inclusion criteria.
- Gives a systematic review of recommendation systems.
- Has a misleading or vague title.
- Does not have a unique study.

- Does not have a good enough abstract.
- Does not introduce performance evaluation metrics in experiment.
- Is not published in a conference or journal.
- Is not written in English.

3.3 Extraction of Papers

The final data extraction is chosen among the document results I get and is filtered based on their title, abstract and content according to the inclusion and exclusion criteria.

In order to get a collection of papers with good quality, there are three stages for filtering the papers. The first stage is title filtering stage, a paper which does not have a good title is filtered, the title needs to be clear, not misleading or vague. The second stage is abstract filtering stage, the abstract needs to provide a brief summary of the paper which describes the process of recommendation system in general instead of focusing on explaining one aspect or concept. The third stage is filtering based on the content of the paper, the content needs to be sufficiently well, the papers with experiments are preferred and they need to provide performance evaluation metrics if they have experiments. Besides, the study used in the paper needs to be unique, which means a paper is discarded if there is already one similar paper selected.

From the 490 papers that I get in the search process, 81 papers are selected by going through the title. After going through the abstracts and contents, 27 papers are chosen according to Table 1, which are categorized according to whether they are content based, collaborative based or hybrid approaches.

Based on the research question of this paper, the research problems and approaches used to implement recommendation systems are extracted from the listed papers, including research problems as listed in Table 3, similarity metrics as listed in Table 4, and evaluation methods as listed in Table 5.

By going through the papers, I found that most of research papers are trying to propose a recommendation system to address the information overload problem and data sparsity problem, which often occurs when there is big amount of data, some papers try to address the cold start problem, which occurs when making recommendations to new users in the system, or when there are new items with few ratings in the system. Some papers are proposed to improve the state of art approaches like similarity metrics, some of them try to apply a recommendation system in a specific domain, for example, [27] applies a recommendation system to recommend timing plans for traffic signal control. Some papers aims to improve recommendation accuracy by proposing new approaches or integrating existing approaches.

For similarity metrics, there are Euclidean distance approaches, cosine based, adjusted cosine based and correlation based approaches. For evaluation metrics, MAE, precision and recall are used by most of the papers, RMSE and coverage are also popular evaluation metrics, while F-measure, MAP and NDCG are not used frequently.

Among the paper list, some of them also utilize some performance enhancement approaches, which includes matrix factorization, machine learning and big data frameworks. They are listed in Table 2.

4 Results

This section gives summary of the list of papers I found in the literature search process, there are three categories of filtering methods: content based, collaborative based and hybrid. Therefore, it would be better to introduce papers which use these filters first to help readers understand how recommendation filters work beforehand.

For these categories, I will focus on introducing what problem the paper is trying to address, what similarity method is used, which dataset the method is applied to if there is, what evaluation method is used to measure the system's performance, and the result it gives.

Apart from these categories, a specific section which discusses performance enhancement approaches is given to help researchers get an idea about what kind of approaches are used to improve recommendation accuracy or reduce

Category	**Paper**	**Publish Year**
Content-based Algorithms	[23]	2005
	[4]	2011
	[13]	2013
	[2]	2015
	[11]	2015
	[27]	2015
	[21]	2015
	[14]	2015
	[22]	2016
	[26]	2018
Collaborative Filtering Algorithms	[20]	2013
	[10]	2013
	[12]	2014
	[5]	2014
	[25]	2016
	[6]	2016
	[7]	2017
	[28]	2017
	[18]	2017
	[24]	2018
Hybrid Filtering Algorithms	[8]	2010
	[3]	2011
	[15]	2012
	[17]	2014
	[19]	2017
	[9]	2017
	[16]	2017

Table 1: Final paper list

Category	**Paper**	**Publish Year**	**Method**
Big data frameworks	[4]	2011	Hadoop
	[14]	2015	Hadoop
	[24]	2018	Spark
Matrix factorization	[12]	2014	SVD
	[5]	2014	TagGSVD++
	[18]	2017	NALS-WR
Machine learning	[23]	2005	KNN
	[3]	2011	KNN
	[20]	2013	k means
	[11]	2015	Bayesian network
	[27]	2015	KNN
	[21]	2015	SVM
	[22]	2016	NLP
	[16]	2017	deep learning
	[9]	2017	KNN
	[26]	2018	CNN

Table 2: Performance Enhancement Approaches

Research Problems	**Paper**
Information overload	**[23], [13], [6], [28], [15], [19], [22], [26], [4], [14], [24]**
Improve similarity metric	**[2], [7]**
Data sparsity problem	**[20], [10], [17], [18], [5], [12]**
Cold start problem	**[11], [3], [9]**
Reduce time consumption	**[4], [14], [24]**
Improve recommendation accuracy	**[20], [25], [15],[16]**
Domain specific problem	**[13], [27], [21],[8]**

Table 3: Research Problems

Similarity Metrics	**Paper**
Euclidean distance	**[23], [2], [27], [12]**
Cosine based similarity	**[13], [25], [3], [15],[19],[4], [24]**
Adjusted cosine	**[28]**
Correlation based similarity	**[10], [6], [3],[15],[19], [16]**

Table 4: Similarity Metrics

Evaluation Method	**Paper**
MAE	**[23], [20], [10], [28], [17], [19], [19], [5], [24]**
RMSE	**[3], [19], [26]**
Precision, recall	**[13], [11], [25], [8],[22]**
Coverage	**[25], [3], [15], [22]**
F-measure	**[3]**
MAP	**[22]**
NDCG	**[27]**

Table 5: Evaluation Method

data processing time. These approaches include big data framework tools that are used to preprocess the data and facilitate the recommendation process, machine learning and matrix factorization approaches that can help reduce the recommendation time and improve the recommendation accuracy.

4.1 Content-based Recommendation Systems

This section provides the summary of papers which use content based recommendation systems, they all make recommendations based on finding the items that are considered to be similar to items that users liked in the history. The difference lies in the distance metrics for computing the similarities, and prediction methods for predicting users' ratings.

As is discussed in the background section, recommendation systems typically make recommendation to users based on similarity metrics, however, in commercial applications like e-commerce, there are huge amount of items, and the items that are rated or purchased by the users take a small amount over all the items, this leads to sparsity problem. In [23], which is published in 2005, it addresses this problem using content-based and clustering recommendation algorithm [23].

It first splits the unvalued items into several clusters using apriori-knowledge and predict their ratings, then the nearest neighbors of these items are found to make the recommendation [23]. The initial sets are constructed with the examples with known classes, then the similarities between every two sets are computed using the following equation [23]:

$$IDF(i) = \frac{1}{|c_l| \bullet |c_k|} \sum_{x_l \in c_l}^{x_k \in c_k} s(x_l, x_k) \tag{14}$$

Every two sets which are considered most similar to each other based on the above equation are united if they belong to the same class, the number of clusters k is the number of sets which are left [23]. Then a random cluster center is picked for each cluster, and self-organizing map is used to cluster based on the cluster centers, the examples of unknown classes can be classified according to the examples of known classes within the same cluster [23]. Hence the similarity of customers can be computed using cosine-based based similarity [23]. After the clustering process, the rating of customer u for product i can be computed based on the following equation [23]:

$$P(u,i) = \overline{R}_u + \frac{\sum_{n \in neighbor} sim(u,n) \times (R_{n,i} - \overline{R}_n)}{\sum_{n \in neighbor} (|sim(u,n)|)} \tag{15}$$

Here $sim(u,n)$ denotes the similarity between user u and its neighbor, $R_{n,i}$ is customer n's rating on item i, $\overline{R}_u$ and $\overline{R}_n$ are average rating of customer u and n respectively [23].

The algorithm is tested under their own dataset, which has 100000 ratings with scale 1-5 from 943 customers on 1682 movies, items with rating 1, 2 or 3, 4 or 5 are classified into three classes [23]. The result is evaluated using MAE method. The proposed method is compared with another two systems, one is collaborative based method which uses correlation-based similarity method to find the customers which have similar taste to the target user and predict their ratings as the target user's rating, another method uses cosine base method to find similar items and apply their ratings to the unrated items [23]. The result shows that the cosine based method improves only slightly compared to collaborative based method, but the proposed system improves the accuracy significantly [23].

Web curation services like Pinterest, Scoop.it, Storify suggest a collection of contents to users based on topics or keywords, where the recommendation of contents are challenging and it is the problem that [13] tries to solve [13]. It uses cosine based similarity method for computing the similarity, TF-IDF method for prediction [13]. Precision is used to evaluate its performances [13].

This paper is published in 2013, it represents each topic as a document vector with 7 different feature extraction approaches(TopicTitleDesc, PageTitle-Summ, PageContent, LSIPageTitleSumm, LSIPageContent, LDAPageTitle-Summ and LDAPageContent) for indexing and user profiling [13], where the feature descriptors can be generated based on title, summary descriptions, page content or by applying LDA and LSI to both page level and content level [13]. The topic index generated by TopicTitleDesc, for example, is represented as $I(t_i, TopicTitleDesc)$.

Each user is profiled based on his or her curated topics ct^{u_i}, followed topics ft^{u_i}, or both $ctft^{u_i}$ [13]. Since the users can be profiles with seven different approaches, the users' profiles generated from different approaches are represented differently($P = P_{src}(u_j, type_u)$) [13].

The topic recommendation is achieved by using TF-IDF method where the scores are used to rank the topics [13]:

$$Score(u_j, t_i, src, type_u, type_I) = \sum_{e \in t_i \cap u_j} tf(e, P) \times idf(e, I) \tag{16}$$

Here t_i denotes topic i, u_j is the user j, $P = P_{src}(u_j, type_u)$, and $I = \cup_{t_i} I(t_i, type_I)$ is the topic index [13].

The dataset used in the experiment is extracted from Scoop.it during October and November 2012, and it includes 22000 unique topics covering 2 million pages, the author performs an offline training testing recommendation study using the dataset, where the training dataset covers topics created by 845 active users who follow at least 20 topics and random selections of 10% of their followed topics, the remaining followed topics forms the test dataset [13]. It compares the performance of content based recommendation systems in Scoop.it under different 7 types of feature(index, user profile) representation approaches and 3 sources of data profiling(TopicTitleDesc, PageTitleSumm, PageContent) configurations [13]. Therefore, the training-test data is applied in 147 different configuration combinations, and the top-n recommendations are made based on ranking the score computed using TF-IDF method where n varies between 5, 10, 20 and 30 [13]. Precision is selected as the evaluation metric.

The result shows that the precision vary from <10% to almost 30%, which means the profiling and indexing methods do make a difference [13]. The application of LDA in the feature extraction stage does not show significant improvement, but LSI helps [13]. It also shows that the source of data matters, the page content contributes the result compares to the system with data only extracted from title and description [13]. And the system with the three sources performs the best over other systems [13].

In job recommendation system, jobs are described with required qualifications, and each candidate is assigned a value for each qualification, however, in job descriptions, the requirements can be an exact value(E), a range with lower limit(L), a range with upper limit(U), or a range with both lower and upper limit(LU) [2]. Traditional job recommendation systems cannot cover all the cases. In [2], which is published in 2015, it proposes a job recommendation system addressing the problem of matching people and jobs in job recommendation system by extending Minkowski distance method while structuring the job descriptions and candidates' profiles [2].

In [2], a job is represented with a vector $J = (j_1, j_2, \dots, j_i, \dots, j_n)$, where each j_i belong to one of E,L,U,LU [2]. A weight vector $W = (w_1, w_2, \dots, w_i, \dots, w_n)$ and a candidate employee vector $C = (c_1, c_2, \dots, c_i, \dots, c_n))$ are also defined that w_i is the corresponding weight of j_i and c_i is the computed value of attribute j_i [2]. The vector J is classified into four vectors J_1, J_2, J_3, J_4 depending on which category(E,L,U,LU) the attributes belong to [2].

After constructing the vectors, the suitability for each class is computed differently according to the following equations [2]:

$$S_E = -(\sum_{i=1}^{n} w_i|j_i - c_i|^p)^{\frac{1}{p}}, p \in R^* \tag{17}$$

$$S_L = (\sum_{k=1}^{n} w_k|j_k - c_k|^p)^{\frac{1}{p}} - (\sum_{l=1}^{n} w_l|j_l - c_l|^p)^{\frac{1}{p}}, c_k \geqslant j_k, c_l < j_l, k \neq l \tag{18}$$

$$S_U = (\sum_{v=1}^{n} w_v|j_v - c_v|^p)^{\frac{1}{p}} - (\sum_{u=1}^{n} w_u|j_u - c_u|^p)^{\frac{1}{p}}, c_u \geqslant j_u, c_v < j_v, u \neq v \tag{19}$$

$$S_L U = -(\sum_{m=1}^{n} w_m|j_m - c_m|^p)^{\frac{1}{p}}, p \in R^* \tag{20}$$

Here p represents the distance variable, while $p = 1$ means Manhattan distance method is used, $p = 2$ means Euclidean distance is used [2].

The resulting suitability S_{CJ} of candidate C for job J is the sum of the above suitability and they are considered as a match if the suitability is above a requirement value which is set beforehand [2].

In its experiment, dataset is taken from Kaggle [1], 100 different IT jobs are retrieved and 8 skills are selected as attributes, the weight vector is defined by experts on job seeking and recruiting domain [2]. Then the corresponding suitability value is computed for jobs for each candidate, and the jobs with top 5 suitability values are recommended to the candidate [2]. The proposed method's recommendation result is promising and the result shows that the choice of distance method does not make any difference to the system [2].

In [11] which is published in 2015, it applies a content based recommendation system by applying Bayesian network model with TrueSkill algorithm to model users' preferences, which can help address the "cold start" and sparsity problems in recommendation systems [11].

Users are represented by a vector $A = 1, \ldots, k$, the features including items and ratings from user u_1 and u_2 who rates the same product as u_1,are represented by a vector $X = x_1, \ldots, x_k$, the rating vector $r = r_1, \ldots, r_k$ is taken from k users for a same item, $s_{i,j}$ denotes the preference of user i for feature j in item p, $p_{i,j}$ denotes the item attribute performance for user i, d_j is the rating difference between users for item j [11]. After constructing these vectors, the algorithms runs for each item, it gets two random users u_1 and

u_2 who have rated this item j, rating probability $p(r|s,A)$ is computed based on joint probability $p(s,p,t|r,A)$ defined by TrueSkill, then the posterior distribution of user preferences, which is denoted by the skill vector s here, is computed based on the following equations [11]:

$$p(s|r,A) = \frac{p(r|s,A)p(s)}{p(r|A)} \tag{21}$$

$$p(s|r,A) = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} p(s,p,t|r,A)dpdt \tag{22}$$

After knowing the preferences of users, the rating prediction can be computed by the following equation [11]:

$$p(d > 0) = 1 - \phi(\frac{\mu_i - \mu_j}{\sigma_i^2 + \sigma_j^2 + 2\beta^2}) \tag{23}$$

Here μ_i and σ_i are the mean and standard deviation for user i's preferences, β is the performance variance for each user's preference [11]. The item is recommended to the target user u_1 if the probability is more than 0.5 [11].

In its experiment, MovieLens and TripAdvisor datasets are used, where MovieLens dataset includes 100000 ratings from 943 users and 1682 movies, TripAdvisor dataset includes 10000 ratings from 9999 users and 67 hotels [11]. The proposed method is evaluated using precision. The result shows that the rating prediction works well even if there are only a few item ratings from users, therefore, the "cold start" problem can be addressed [11].

Another paper [27] published in 2015 proposes a recommendation system for traffic signal control to find best matching time plans for various intersections conditions [27]. Euclidean method is used as the similarity metric [27]. K-nearest neighbor method is used to find similar traffic conditions to the target traffic condition, then it predicts their matching degree, and timing plans can be predicted by analyzing the history timing plans data of the similar traffic conditions [27].

The intersection is represented as a vector $I = (r_1, r_2, \ldots, r_m)$, where $r_i = (l_1, l_2, l_3, l_4, l_5)$ and l_1, l_2, l_3, l_4, l_5 represent U-turn lane, left turn lane, straight lane, straight and right turn lane, right turn lane respectively [27]. It considers traffic conditions as users and they are represented by origin-destination(OD) matrices, timing plans C are considered as items and traffic indicators like delay time are considered as ratings given by the users to the items, and a user-item ratings matrix is produced [27].

The similarity between each two traffic conditions a and b is computed using Euclidean Distance [27]:

$$sim(a,b) = \frac{1}{1+\sqrt{\sum(a_i - b_i)^2}} \tag{24}$$

After computing the similarities, k nearest neighbor(KNN) is used to choose k most similar traffic conditions to the target traffic condition, and the average delay rating of them are used to predict rating of the target traffic condition a according to time plan c [27]:

$$pred(a,c) = \frac{\sum_{b \in k} sim(a,b) * r_{b,c}}{\sum_{b \in k} sim(a,b)} \tag{25}$$

Where $r_{b,c}$ is the rating that traffic condition b gives timing plan c [27]. After this step, an ordered recommendation list $(c_1, c_2, \ldots, c_i, c_{i+1}, \ldots)$, which is compared with another timing plan list $(c'_1, c'_2, \ldots, c'_i, c'_{i+1}, \ldots)$ for evaluation [27]. NDCG acts as the performance indicator [27]:

$$nDCG_p = \frac{DCG_p}{IDCG_p} \tag{26}$$

Where DCG_p is the accumulated DCG at rank position p and $IDCG_p$ is the maximum DCG before position p [27].

In experiment, it test the system's performance in a simulation network of single intersection with 4 roads [27]. Three timing plans are predicted, where timing plan 1 emphasizes more on phase 1 and 2, timing plan 2 emphasizes more on phase 3 and 4, timing plan 3 does not give any preferences [27]. The result shows that timing plan 3 performs best [27]. In real applications, the matrix of delay time can be sparse, therefore, another simulation experiment with sparse degree 75% is carried out, the dataset includes 40 ODs and 40 timing plans, the result shows that the NDCG values of most ODs are nearly 1, which is quite promising [27]. The proposed method is also compared with Webster method, which is the benchmark method for traffic signal control, the result shows that with the proposed method, the delay time is greatly reduced compared to the Webster method [27].

In [21] which is published in 2015, it presents a content based music recommendation system for spoken documents using a multisource approach that non-linguistic characteristics of audio suck as the speaker's identity, language, gender and environment properties are used [21]. Here, automatic speech recognition(ASR) with low and high resources are used to extract the acoustic vector with speaker, language, gender attributes [21]. For evaluation, a corpus of audio clips are taken from CreativeCommons.org videos, the result shows that the system reduces the equal error rate(EER) to half of the bag-of-words' model [21].

Bags-of-words consist of audios and transcripts from LDC's Switchboard Phase 1 are used to train a model for feature extraction, the feature vectors are extracted from the large vocabulary ASR, TF-IDF is used to compute the word frequencies for each document [21]. Frequencies of attributes including speaker, language, gender are generated for each document using Kaldi deep neural network and this is the generated bags-of-senones [21]. Bags of pseudoterms are also extracted as characteristic of environment from each audio clip and corresponding TF-IDF values are computed [21]. For each audio clip, an acoustic i-vector is extracted and it is normalized using Garcia-Romero [21]. Latent Sementic Analysis(LSA) and Principal Component Analysis(PCA) are used to reduce the dimension of the data [21]. A classifier is trained for each feature, back-end fusion classifiers are trained for multiple features, three fusion classifiers score fusion, feature fusion and hybrid fusion can be trained with fusion scores, features or their combination respectively [21]. Logistic regression and SVM models are trained for each feature and each user, then fusion classifiers are applied to test scores for each audio clip [21].

Audios stripped from Creative Commons internet videos are used as dataset, which are divided into train, dev, and test lists, the test set is scored with each classifier which can be logistic regression, SVM model, or one of the two models combined with fusion classifier, the EER is computed for each feature in the test set [21]. The result shows that the acoustic i-vectors performs better than the baseline bags-of-words, and the use of ASR also reduces the error rate, dimensional reduction with LSA slightly increases the error rate for bags-of-pseudoterms and decreases the error rate for bags-of-words [21]. The bags-of-senones performs best out of bags-of-words and bags-of-pseudoterms [21]. SVM performs better than the logistic regression model. In the aspect of fusion classifiers, score fusion performs the best over feature fusion and hybrid fusion [21].

4.2 Collaborative Based Recommendation Systems

This section provides the summary of papers which use collaborative based recommendation systems, they all make recommendations based on the items that are liked by the similar users to the target user. The difference lies in the distance metrics for computing the similarities of users, and prediction methods for predicting users' ratings.

In [20] which is published in 2013, a collaborative based recommendation system for movie recommendation is proposed to address the data sparsity and poor prediction problem in recommendation systems when the amount of recommendable items increases [20]. K-means clustering method is used to find similar users to the target user based on their ratings on items, then slope one algorithm is applied to predict the user's rating on the target item [20]. MovieLens dataset is used as dataset and the result shows that the proposed method performs better than other recommendation methods [20].

In order to reduce the amount of users needs to be used to compute the prediction ratings, k-means clustering method is used [20]. Firstly random k cluster centers are picked, then the algorithm tries to minimize the euclidean distance between each data point and its nearest cluster center by continually updating cluster centers until the objective function is minimized [20].

After clustering the users, the user rating vector set SU which consists of ratings of similar users to the target user is used to calculate the average deviation $dev_{j,i}$ between target item j and other item i [20]:

$$dev_{j,i} = \sum_{uinS_{j,i}(SU)} \frac{r_{u,j} - r_{u,i}}{c_{j,i}} \tag{27}$$

Where $S_{j,i}(SU)$ denotes the user rating vector set SU that evaluate item j and i, $c_{j,i} = card(S_{j,i}(SU))$ is the quantity of the set $S_{j,i}(SU)$ [20]. Then the rating of user u on item j is predicted using the following equation [20]:

$$P_{u,j} = \frac{\sum_{i \in S_{j,i}(SU)-\{j\}} (dev_{j,i} + r_{u,i}) c_{j,i}}{\sum_{i \in S_{j,i}(SU)-\{j\}} c_{j,i}} \tag{28}$$

Then items are sorted according to the predicted ratings decreasingly that the item with highest rating is recommended to the target user [20].

In experiment, MovieLens and Fingerhut Inc E-Commerce data sets are used for training and testing the model, MAE is selected as the evaluation method. It compares the system's performance with various values of k (5,10,15,20 and 25) and it finds out that the system with 15 clusters performs best [20]. The paper compares the proposed method with user based, item based collaborative filtering methods and standard Slope One algorithm, and the result shows that the proposed method performs best while all the algorithms improves the accuracy when there are more users [20].

Another paper [10] published in 2013 also tries to address the data sparsity problem, it proposes a new similarity metric since there are deviations in traditional similarity methods that will reduce the recommendation accuracy when the data is sparse [10]. The new similarity metric uses an impact factor ϵ to adjust the deviations oft traditional similarity metrics, it also applies λ to adjust the weight of user based collaborative filtering and item based collaborative filtering when predicting the items' ratings [10].

The proposed similarity metric uses a similarity factor ϵ, and it is computed by [10]:

$$\epsilon = \frac{|I_{U_aU_b} \times I_{U_aU_b}|}{|I_{U_a} \times I_{U_b}} \tag{29}$$

Where $I_{U_a}, I_{U_b}, I_{U_aU_b}$ denotes the items rated by user U_a, U_b, both U_a and U_b respectively. Then the similarity is given by [10]:

$$sim'(U_a, U_b) = \epsilon \times sim(U_a, U_b) \tag{30}$$

Where $sim(U_a, U_b)$ denotes the similarity measured using traditional similarity metrics, and its deviation is adjusted by ϵ to improve the recommendation system [10].

The rating of user a on item i is then given by [10]:

$$\begin{aligned} R_{a,i} &= \lambda \times \left(\overline{R_a + \frac{\sum_{U_x \in U} sim'(U_a, U_x) \times (R_{x,j} - \overline{R_x})}{\sum_{U_x \in U} sim'(U_a, U_x)}}\right) + \\ &= (1-\lambda) \times \left(\overline{R_i + \frac{\sum_{I_y \in I} sim'(I_i, I_y) \times (R_{a,y} - \overline{R_y})}{\sum_{I_y \in I} sim'(I_i, I_y)}}\right) \end{aligned} \tag{31}$$

Where $\overline{R_a}, \overline{R_x}$ are average ratings of user U_a and U_a on items, $\overline{R_i}, \overline{R_y}$ are average ratings of items I_i and I_y, λ is used to adjust the weight of users' similarity and items' similarity for predicting the rating [10].

The system is evaluated in MovieLens dataset, and MAE is the evaluation metric [10]. The results shows that the usage of ϵ can improve the recommendation result, and the similarity with impact factor ϵ that is computed based on adjusted cosine similarity performs the best [10].

In [25] which is published 2016, it proposes a TV recommendation method based on collaborative filtering [25].

It builds user-tag model and program-tag model where user-tag model describes the users' viewing preferences and program-tag model describes the type of programs [25]. Programs and tags are represented using numerical values, the 19 tags of tv programs are their types including "romantic", "action", "historical" and so on [25]. Users are tagged by their preferences over director, actor, region and type [25]. TF-IDF is used to compute the importance degree of a tag for users and programs [25]. The user-tag model in the form of matrix is given by [25]:

$$\frac{T_i}{T} \times \left(log(\frac{n}{n_i})\right) \tag{32}$$

Where T_i and T are the time that viewer spent on tag i and all tags [25]. n_i and n are the number of users who viewed tag i and number of all users respectively [25].

The program-tag model in the form of matrix is given by [25]:

$$\frac{x}{X} \times \left(log(\frac{N}{N_i})\right) \tag{33}$$

Where x and X are the number of tag i and total tags in the tag list [25].

After generating the two models, every user has a user-tag vector that can be used to compute the similarity between the target user and other users, the similarity metric used is cosine similarity [25]. Then users are sorted based on their similarities to the target user and a list of users with greatest similarities are considered as the reference users [25]. When making the program recommendation list, the dot product result of user-tag matrix and program-tag matrix is sorted that the top 20 programs are recommended to the user [25].

The dataset is taken from the viewing record in a province for three months, the algorithm is evaluated using precision, recall, coverage and average popularity level [25]. The average popularity represents how novel the program is and it is defined as [25]:

$$AveragePopularity = \frac{\sum_u \sum_{i \in R_u} log(1 + popularity(i))}{\sum_u \sum_{i \in R_u} 1} \tag{34}$$

The result shows that the precision and recall rate are 9.5% and 12.9% respectively, where coverage and popularity level are 11.8% and 1.92 respectively [25].

In [6] which is published in 2016, it proposes a collaborative based recommendation system that takes the context information of user into consideration to help find similar users in commercial recommendation system [6]. It first calculates the relations between user locations using location attenuation function, then it uses Pearson similarity metric to compute the users' similarities, after that, it predicts the users' preferences and makes recommendations [6].

For the target user i and every other user j, it finds a union of items that are rated by both user i and j, then it defines a location based attenuation function as follows [6]:

$$f(|l_{ui} - l_{uj}) = \frac{1}{1 + \alpha |l_{ui} - l_{uj}|} \tag{35}$$

Where α is the distance attenuation ranges from 0 to 1 depends on the how fast the user's location changes [6]. l_{ui} and l_{uj} are the location information of user i and j on item u respectively [6].

Then this function is multiplied with Pearson similarity metric to compute the similarity of users [6]. The users are sorted according to the computed similarities and the users ranked in the front forms a user set M_p, after that, the rating of user i on item p is defined by [6]:

$$P_{i,p} = \frac{\sum_{n \in M_p} sim_{p,n} \times R_{i,n}}{\sum_{n \in M_p} |sim_{p,n}|} \tag{36}$$

In the end, top n items with highest ratings are recommended to the target user [6].

The paper [7] published in 2017 proposes a new similarity metric which considers user interest information so that the similarity values are more accurate [6]. When the number of user ratings are not large, recommendation system computes similarity of user attributes, when there is large number of user ratings, the similarity is computed based on user ratings [7]. The paper integrates the two similarities and apply weights to both so that recommendation system can adapt to the need.

For computing similarity based on user ratings, it considers both users' rating similarity and interest tendency similarity and the two similarities are combined together to produce the similarity [6]. The similarity of user i and j is defined as follows [7]:

$$sim_{score}(i,j) = \left(\frac{1}{|I_{ij}|} \sum_{x \in I_{ij}} sim_1(i,j,x) \times sim_2(i,j,x) \right) \times sim_3(i,j) \tag{37}$$

Where $sim_1(i,j,x), sim_2(i,j,x)$ are the similarity of ratings and interest tendency of user i,j on item x respectively, $sim_3(i,j)$ is used to measure confidence coefficient of similarity and it is defined by [7]:

$$sim_3(i,j) = \frac{|I_i \cap I_j|}{|I_i \cup I_j|} \tag{38}$$

For computing similarity based on user attributes, the similarity is given by [7]:

$$sim_{Attr}(i,j) = \sum_{m \in Attr} \omega_m \cdot sim_{Attr}(i,j,m) \tag{39}$$

Where $sim_{Attr}(i,j,m)$ denotes the similarity of m attribute of user i and j, it is 1 if the attributes are the same and 0 if they are not [7]. ω_m is the weight of the attribute m [7].

Then the similarity based on user ratings and user attributes are combined as follows [7]:

$$sim(i,j) = \alpha \cdot sim_{Attr}(i,j) + \beta \cdot sim_{score}(i,j) \tag{40}$$

Where α, β are the weights of the two similarities [7]. After this step, users are ranked and K users N_K with greatest similarities are obtained [7]. Then the rating of target user i on item x is predicted as [7]:

$$r(i,x) = \frac{\sum_{k \in N_k} sim(i,x) \cdot r_{kx}}{\sum_{k \in N_k} sim(i,x)} \tag{41}$$

The movielens is used as the dataset, MAE is used to evaluate the system. The results shows that the proposed algorithm performs better than the contrast algorithm either under normal condition, cold start condition or data sparsity condition [7].

Since some users are strict and tend to give low ratings, some users are easy to give high ratings, simply predicts ratings without considering this will lead to inaccuracy. In [28] which is published in 2017, it uses emotional polarity of comments together with rating computed based on adjusted cosine to make recommendations [28]. It uses amazon products as dataset that includes reviews, product metadata and links [28].

It uses a python library TextBlob for emotion polarity computation, and the user's rating y is given by [28]:

$$y = \omega_i \times x_1 + \omega_j \times x_2 \tag{42}$$

Where x_1 denotes user's rating, x_2 denotes the computed emotion polarity [28]. ω_i, ω_j are their corresponding weights that the system tried to learn so that the difference between predicted ratings and real ratings are minimized [28].

For the rating part, it uses item based collaborative filtering method, since in user based recommendation method, all the ratings of items need to be saved for making recommendations, while in item based recommendation, the degree of similarity between items can be represented directly [28]. It uses adjusted cosine similarity to compute the similarity between items, then it predicts the rating of user u on item i as follows [28]:

$$P(u,i) = \frac{\sum_{N \in SimilarTo(i)} S_{i,N} \times NR_{u,N}}{\sum_{N \in SimilarTo(i)} (|S_{i,N}|)} \tag{43}$$

Where $S_{i,N}$ is the adjusted cosine similarity, $NR_{u,N}$ is the normalized ratings of users from [1,5] to [-1,1] [28].

MAE is used to evaluate the system's performance. The result shows that the ideal setting of weights are 0.7 for the predicted rating, 0.3 for the emotion polarity [28].

4.3 Hybrid Recommendation Systems

This section provides the summary of papers which use hybrid recommendation systems. Hybrid recommendation systems combines the collaborative recommendation system and content based recommendation system to make use of both of their advantages. Different papers combine the two filters differently, they can be assigned adjustable weights and used together or used separately under different circumstances. Besides, their choices of similarity metrics and evaluation metrics are different.

In [8] which is published 2010, it recommends the user's followees who are followed by the user on Twitter using various user profiling methods [8]. It compares the performance of various recommendation methods, including content based, collaborative based and hybrid recommendation systems [8]. The recommendation list of followees can either be generated according to a search query or the user's profile [8].

In the user profiling process, each user has his/her 100 most recent tweets, a set of followees who are followed by the user, and a set of followers who follow this user [8]. Thus, the user can be profiled by their tweets, their followees' tweets and their followers' tweets [8]. It uses Lucene platform to index and give weights to the features, the platform is similar to TF-IDF, but is more robust and scalable [8]. Either each user's tweets, his/her followees' tweets or followers' tweets, is represented as a vector, in this vector, each element is different word's weight, which are computed by Lucene's TF-IDF weighting metric [8].

It carries out an off-line evaluation of the recommendation systems, the dataset which includes 20000 users is imported from Twitter API and is split into training, test set [8]. It compares 9 different recommendation systems, which includes 4 content based filters S_1, S_2, S_3, S_4 where the users are profiled by their tweets, their followees' tweets, followers' tweets, a combination of the three tweets respectively [8], 3 collaborative filters S_5, S_6, S_7 where the users are profiled by their followees' tweets, followers' tweets, a combination of three tweets respectively [8], two hybrid ensemble filters S_8, S_9 where the scoring function is different, S_8 is scored based on the combination of a content based filter where the users are profiled by their tweets and a collaborative filter where the users are profiled by their followers' tweets, S_9 is scored based on users' rankings in the recommendation lists [8]. The recommendation list's size varies from 5 to 20. The systems' performances are measured using precision. The performance result by measuring how many predicted followees overlap with the actual followees of the target user shows that the profiling method of followees' tweets performs the best among others [8]. It also shows that the performance drops when the recommenation list gets larger. In addition, the collaborative based method S_6 and two ensemble methods S_8, S_9 performs better than other methods [8]. Another ranking method based on the position of relevant recommendations in the recommendation list shows that content based method performs better than the collaborative method [8].

Since the previous performance evaluation method neglects the fact that the result of recommendation list which does not overlap with existing followees might be the potential followees of the target user, the paper also carries out a live experiment which involves 34 users [8]. The result shows that the system can make a good recommendation that an average of 6.9 users in the recommendation list are the followees that participants are willing to follow [8].

In [3] which is published in 2011, it tries to address the cold start problem, which means the recommendation system does not perform good for making recommendations to new users or for new items added to the system [3]. It combines collaborative, content based filtering methods as well as demographic characteristics of users [3]. The result shows that the proposed method outperforms the conventional collaborative, content based and hybrid filtering methods, and it also addresses the cold start problem [3].

In the proposed method, cosine based similarity metric is used to measure the similarity between users in its collaborative filtering method, in its content based filtering, KNN is used to cluster the items so that similar items to the target item are selected, Pearson correlation based similarity method is used to find similar users for the target user. In its demographic filtering, KNN and cosine based similarity method are used to cluster the users into categories based on the demographic characteristics of users(e.g. gender, race, age, etc.) [3]. When a new user is added, his/her predicted rating for a target item uses only the user's cluster [3]. For new items, content based filtering and demographic characteristics of users are used for recommendation [3].

The three filters make prediction of users' ratings independently, since each of them has its advantage in a certain circumstance, they are assigned weights and combined together to make the final prediction according to the following equation [3]:

$$\hat{r}_{u,i} = \frac{\alpha, +\mathbf{DF}_{u,i} + \beta\mathbf{CB}_{u,i} + \gamma\mathbf{CF}_{u,i}}{\alpha + \beta + \gamma} \tag{44}$$

Where $\mathbf{DF}_{u,i}, \mathbf{CB}_{u,i}, \mathbf{CF}_{u,i}$ are the predicted ratings for user u on item i from demographic filtering, content based filtering, and collaborative filtering respectively, α, β, γ are their corresponding weights, and they change dynamically according to the number of available ratings of the target user [3].

In experiment, MovieLens is used as the dataset, RMSE and F-measure are used to evaluate the system's performance [3]. It compares the system's performance with different neighborhood sizes(K values of KNN method) and it finds out that 100 is the optimal setting for K [3]. Then it compares the proposed method with seven other approaches, including collaborative filtering using Pearson correlation similarity metric, content based filtering using cosine based similarity, demographic filtering using cosine based similarity, hybrid filtering which uses both collaborative and content based filtering, hybrid method which uses both collaborative and demographic filtering, hybrid method which uses both content based and demographic filtering, hybrid method which uses content based, collaborative and demographic filtering where the rating is predicted as the mean of the ratings from the three filtering methods [3]. The result shows that the proposed method outperforms all other seven methods in terms of RMSE and F-measure evaluation methods, it also gives better prediction results compared to other methods when there is cold start problem [3].

In [15] which is published 2012, it integrates content based recommendation algorithm with item based collaborative filtering to recommend webpages to users [15]. They are firstly used separately to generate recommendation lists, then the lists are combined to obtain the final recommendation list [15].

In its content based filtering, Vector Space Model is used to represent the features by vectors, the features are the words in the pages [15]. TF-IDF is used to assign weights to the words. Then the similarity between the vectors are calculated using cosine based similarity metric. In its item based collaborative filtering method, Tanimoto coefficient, which is the ratio of number of items which are liked by both two users to the number of items which are liked by either of them [15]. To combine the recommendation lists generated from content based and collaborative filtering methods, each of them are given a proportion [15]:

$$\alpha N_1 + \beta N_2 = N \tag{45}$$

Where N_1, N_2 are the number of items generated by content based and collaborative based method respectively [15], α, β are their corresponding coefficients. N is the number of final results [15].

In experiment, the authors developed a platform called scientific-research-online, it includes about 10000 web pages. The pages which are clicked by users are considered as their preferences [15]. They construct a table which records the page ids which are clicked by the users for computing the similarity of users. 3000 pages and 10 users from the platform are used as the dataset [15]. These users are asked to click pages in the beginning so each of them has a preliminary preference dataset, then for each user, the content based and collaborative filtering are used separately to generate 50 pages which are considered most similar to the preliminary dataset [15]. After that, the user is asked to mark 50 pages which really interest him/her [15]. And the ratio of the pages from either of filtering method to the actual liked pages are computed [15]. The result shows that the content based method performs slightly better than the collaborative method, the proposed method increase the accuracy rate by 9.87% than the content based filtering [15].

In [17] which is published in 2014, it tries to address the scalability and sparse problems exist in collaborative filtering algorithm by proposing a hybrid filtering method with review helpfulness features [17]. It compares this method with collaborative filtering based on hybrid user model, user based collaborative filtering, the result of experiment shows that the proposed method outperforms other methods in terms of accuracy and time consumption [17].

The method has off-line and on-line stages [17]. Items are classified according to their categories, users' preference information is also extracted from ratings and item features, review information is extracted from review ratings and item features [17]. It computes relative feature rating(RFR) and modified relative feature frequency(MRFF) values for both information, where RFR is given by the following equation [17]:

$$RFR(i,k) = \frac{FR(i,k)}{TR(i)} \tag{46}$$

Here $FR(i,k)$ is the sum of ratings that user i gives to feature k, and $TR(i)$ is the sum of ratings that the user gives to all items [17].

MRFF is computed by following equation [17]:

$$MRFF(i,k) = \frac{\sum_{j \in F_K \subset T_i} w_p \times \delta_p(R_{i,j})}{\frac{v}{2} \times TF(i)} \tag{47}$$

Where $p \in \{x | s \geq \frac{v}{2}\}$, $R_{i,j}$ is the user's rating on product j, $\delta_p(R_{i,j})$ is the effective rating that the user i gives to feature k, it is 1 if $p = R_{i,j}$ and is 0 otherwise [17]. F_k is the item sets with feature k, v is the maximum user rating in the system, $TF(i) = |T_i|$, $w_p = p - \frac{v}{2} + 1$ [17].

It also computes relative feature score(RFS) that user i gives to feature k in the aspect of both preference information and review helpfulness information, RFS is defined by [17]:

$$RFS(i,k) = \frac{2 \times Max \times RFR(i,k) \times MRFF(i,k)}{RFR(i,k) + MRFF(i,k)} \tag{48}$$

Here Max is the maximum possible rating values in the system [17]. After obtaining the RFS values, relevant similarity is used for computing similarity between users based on the following equation [17]:

$$S_{u,v} = \frac{\sum_{k=1}^{p+p} |(H_{u,k} - \overline{H}_{u,k}) \times (H_{v,k} - \overline{H}_{v,k})|}{\sqrt{\sum_{k=1}^{p+p}(H_{u,k} - \overline{H}_{u,k})^2} \times \sqrt{\sum_{k=1}^{p+p}(H_{v,k} - \overline{H}_{v,k})^2}} \tag{49}$$

Here $H_{u,k}$ are the combination of RFS values for both preference information from user u to feature k, $H_{v,k}$ are the combination of RFS values for both preference information from user v to feature k, $\overline{H}_{u,k}$ is the average RFS values from user u to feature k, $\overline{H}_{v,k}$ is the average RFS values from user v to feature k [17].

After computing the similarity between users, the predicted rating for user u to item i is given as follows [17]:

$$P_{a,i} = \overline{r}_a + \frac{\sum_{u \in N} S_{a,u}(r_{u,i} - \overline{r}_u)}{\sum_{u \in N} S_{a,u}} \tag{50}$$

Where N is the user set which are considered similar to the target user a based on previous user similarity computation, $\overline{r}_a, \overline{r}_u$ are the average rating of user a and u respectively, $r_{u,i}$ is the rating of user u to item i [17].

After the ratings are predicted, top 10 items with highest ratings are recommended to the user.

A subset of Epionions dataset which contains 1317 users, 54985 items, 245042 ratings is used in the experiment, MAE is used as the evaluation method [17]. It compares the proposed method with user based collaborative filtering UB-CF, user based collaborative filtering with hybrid user model HUM-CF [17]. The result shows that the proposes method outperforms other methods.

In [19] which is published in 2017, it proposes a hybrid filtering method which utilizes good learners' ratings [19]. It compares this method with content based filtering with good learners' rating method, it finds out that the proposed method performs better in terms of MAE evaluation method [19].

In this model, learning materials, users, user ratings are represented as vectors d, u, s, and TF-IDF is used to compute learning materials' weight for each term. In its content based filter, cosine based similarity metric is used to compute the similarity between learner profiles and learning materials [19]. In its collaborative filter, Pearson correlation is used to compute the similarity between good learners and the active learner [19]. After this step, it predicts the rating from user a to learning material b by the following equation [19]:

$$pred_{a,p} = \overline{r}_a + \frac{\sum_{b \in N} sim(a,b)(r_{b,p} - \overline{r}_b)}{\sqrt{\sum_{b \in N} sim(a,b)}} \tag{51}$$

Where N is the number of neighbors who are considered similar to the target learner a, $r_{b,p}$ is the rating from learner b to learning material p, $\overline{r}_a, \overline{r}_b$ are the average ratings from learner a and b respectively, and $sim(a,b)$ is the similarity between the two learners [17].

108 Power Point slides are considered as learning materials. It carries out two experiments, experiment 1 is used to obtain the most suitable threshold for content and collaborative filtering methods, experiment 2 is used to measure the MAE value for the obtained thresholds, and finds the best threshold combination [17]. The result shows that the recommendation system with higher similarity threshold performs better [19]. It compares the proposed method with content based filtering with good learners' rating method, and it finds out that the proposed method improves the recommendation accuracy [19].

In [9] which is published in 2017, it introduces a tourism recommendation system to tourist. It combines content based, collaborative and demographic filtering methods and utilize also machine learning algorithms including KNN and decision tree [19]. Switching and weighted hybridization techniques are used when combine the three filtering methods, and the result shows that the hybrid method performs better than the single filtering method [9].

In this system, recommendation is made based on user's profile, previous appreciations and the type of activities to search [19]. The dataset is obtained from the e-tourism website TripAdvisor, where users are featured by id, login, age, etc [9]. Activities are featured by id, name, category, etc [9]. Then it constructs a rating table which includes users' ratings on activities [19].

In its user based collaborative filter, Tanimoto coefficient is used to measure the similarity between users, and it is given by [9]:

$$sim(i,j) = \frac{|f_i \cap f_j|}{|f_i| + |f_j| - |f_i \cap f_j|} \tag{52}$$

Where f_i, f_j are the item sets which interest user i and user j respectively, $f_i \cap f_j$ is the item set which interest both of them [9]. After computing the similarity between users, it finds 50 neighbors of the target user by using KNN algorithm [19].

In its content based filter, it first finds the activities that are not rated by the user, then it uses Euclidean distance to compute the similarity between each unrated activity between rated activities, after that, it finds the nearest activity to the not rated activity and predict its rating as the nearest activity's rating [9].

In its demographic filter, the same as before, it first finds the activities that are not rated by the user, then it creates a ID3 decision tree for each not rated activity, the nodes of decision tree are demographic information of users, the ratings of decision tree are their ratings of the activity [9]. After that, it uses the decision tree to predict the user's rating for the not rated activity.

To combine the three filters, it uses a weighted hybridization technique defined as follows [9]:

$$\hat{r}_w = \alpha \cdot \hat{r}_{DF} + \beta \cdot \hat{r}_{CB} + \gamma \cdot \hat{r}_{CF} \tag{53}$$

Where $\hat{r}_{DF}, \hat{r}_{CB}, \hat{r}_{CF}$ are the predicted ratings using demographic, content based, collaborative filtering method respectively, and α, β, γ are their corresponding weights. Cross validation is used to find the optimal setting of these coefficients that it tries to minimize the objective function as follows [9]:

$$\frac{\sum_{k=1}^{n} |\alpha \cdot \hat{r}_{DF} + \beta \cdot \hat{r}_{CB} + \gamma \cdot \hat{r}_{CF} - y_i|}{n} \tag{54}$$

Where $\alpha + \beta + \gamma = 1$, y_i is the actual rating value of the activity, n is the number of tested activities [9].

The hybridization technique called switching is also used here, it uses different filtering method under different circumstances, it can switch between content based, collaborative, demographic filtering methods as well as weighted hybridization recommendation system depending on whether there is no cold start problem, user cold start, or item cold start problem [9].

In experiment, it extracts dataset from TripAdvisor which includes 11737 reviews, 6576 users and 160 activities, MAE, NMAE, RMSE are used as evaluation methods [9]. It test the performance of collborative, content based, demographic filtering methods, weighted hybrid method, switching method, and combination of weighted and hybrid methods, the result shows that the hybrid methods performs better than the single filtering methods, and the combination of weighted and hybrid methods performs the best among all the methods [19].

4.4 Performance Enhancement Approaches

To further increase the performance of recommendation systems, many methods are used. For example, machine learning methods including KNN, k-means, decision tree are frequently used in recommendation systems, KNN is used to help find nearest neighbors to users or items, k-means can help cluster the users or items to groups hence can help simplify the process of similarity computation, decision tree can be used to help predict users' ratings on items. Some big data tools including Hadoop, Spark can also be used to help deal with large dataset. Apart from that, matrix factorization can be used to improve the performance of recommendation systems.

4.4.1 Big Data Frameworks

Since there are commonly large scale datasets in e-commerce system, it would be more efficient to utilize big data frameworks including Hadoop, Spark to facilitate the process of making recommendations. Hadoop and Spark are reliable, scalable and distributed processing platforms [4].

In [4] which is published in 2011, it implements a content based filtering algorithm on Hadoop, which has MapReduce operations for keyword extraction and generating content base recommendations [4].

In its content based filter, TF-IDF is used to represent each term's weight in the dataset [4]. Four MapReduce operations are used here, it first extracts key-value pair for each word, where the key is its identifies, values is its content, then the mapper is used to move the word from the key to value, the reducer is used to count the number of appearances for each word, the third operations consists of a reducer which computes the number of item descriptions which contains the word, the mapper moves the word's unique identifier to value, the last operation is a mapper which counts the TF-IDF value for each word [4].

It makes the recommendation by computing the similarity between the item attributes and user profile attributes [4]. Cosine similarity is used here for this purpose due to its simplicity and efficiency [4]. Three MapReduce operations are used here, the first operation computes the Euclidean norm for each vector, where the mapper moves the content variable from identifier to value, the reducer computes the Euclidean norm and put it to the identifier [4]. The second operation joints the item user vectors which share at least one word in its value, here the mapper moves the norm from identifier to value for each user item vector, reducer joints the two vectors [4]. The third operation computes the cosine similarity for each joint vector obtained before, where the mapper moved the unique identifier id for user and item from value to identifier position, and reducer counts the cosine similarity based on the vectors, the results form of this reducer is $((id_i, id_u), sim)$, where id_i, id_u are the unique identifiers for item i and user u, sim represents the cosine similarity between them [4]. Hence, the top-N recommendations can be made, it ranks the items for every user, and recommends the top N items with highest similarity [4].

In experiment, it downloads a subset of Wikipedia articles, to process these articles, it needs to filter out the stop words and HTML tags from the articles, then it computes the cosine similarity for each word user pair [4]. The result shows that the processing time for these operations increases dramatically that it follows a quadratic function when there are more than 900 files, while it follows a linear function when there are less documents [4].

In [14] which is published in 2015, it also proposes a content based filter using Hadoop to facilitate the preprocessing of data. It first loads the dataset to Hadoop Distributed File System(HDFS) which is scalable and reliable, the dataset used here is Amazon co-purchasing network metadata taken from Stanford University website [14]. Each product in the dataset has eight attributes, including Id, ASIN, Title, Group, Salesrank, Similar, Categories and Reviews, where title, sales rank, group, category and reviews attributes are removed during the map operation [14].

After loading the dataset, MapReduce operations are carried out to preprocess the dataset [14]. For each product, the map operation extracts customer id from review attributes, then it removes attributes which are not relevant and products whose similar field is empty, hence it associates products with customers and the resulting vector is key-value pairs for both products and customers [14]. Then it rearranges the vectors based on customer ID, this is followed by a remove operation, which process the vectors and recommend products to users based on products that are similar to what they have bought [14].

There are two types of content based filters used in the system. In the first content based filter, for every customer, the map operation removes the products which are purchased by the customer, then it adds two products which are similar to the purchased products, it removes the product if it is purchased or repeated in the recommendation list [14]. The reduce operation proceeds so that the top 5 recommendation results are printed in the output file [14]. In the other content based filter, the process is basically the same as the first one, but it adds all products instead of just two products when it generates recommendation list, and the products which repeat many times are also added to the recommendation list, the reduce operation prints the best 5 recommendation for each customer in the output file [14].

In experiment, Amazon co-purchasing network metadata is used as the dataset and it consists of 548522 products with the size 977.5 MB [14]. The result shows that for 977.5 MB data preprocessing takes about 90 minutes, while the first type of content based recommendation takes about 15 minutes for 266 MB data, the other one takes about 23 minutes for the same size of data [14].

Another paper [24] published in 2018 proposes an item based collaborative filtering algorithm with ALS algorithm and it runs the two algorithms in parallel on Spark [24]. It runs experiment with MovieLens dataset and compares this method with traditional single machine implementation [24]. The result shows that the proposed method increases the recommendation accuracy and reduces the processing time [14].

To deploy its collaborative filter on Spark, it first loads the MovieLens dataset to Resilient Distributed Datasets(RDD), then it changes the data vector from (userID, movieID, rating, timestamp) to (userID, movieID, rating) so that it simulates a user-item rating matrix [14]. To compute the similarity between movies, the movieID in the key field of key-value pair is moved to value field, and each two key-value pairs are joint together by the same user, then the similarity between the two movies are computed, which results in a list of ratings for each pair of movies [14]. The cosine based similarity is used to measure the similarity [14]. Then it computes the similarity between users with the same method and predict the rating from target user i on item c based on the user's neighbors' ratings [24]:

$$P_{i,c} = \hat{R}_i + \frac{\sum_{j \in N(i)} sim(i,j) \times (R_{j,c} - \overline{R}_j)}{\sum_{j \in N(i)} sim(i,j)} \tag{55}$$

where $sim(i,j)$ is the similarity between user i and j, $N(i)$ are the neighbors of user i, $R_{j,c}$ is the rating from user j on item c, $\overline{R}_i, \overline{R}_j$ are the average rating of user i, j respectively [24]. Based on the predicted ratings for all the items for the target user, top N items with highest ratings are recommended.

It runs ALS collaborative filtering algorithm in parallel, it aims to find a lower rank matrix to approximate the user item rating matrix, the lower rank matrix is the product of two matrices: user matrix and item matrix, while it tries to minimize the regularized loss function, which is the minimum least square between the low rank matrix and the original rating matrix [14]. In Spark, the processes of computing the user matrix and item matrix are executed in parallel, it first loads the data and extract user item features to RDD, then it computes the user item rating matrices in parallel, which are then used to generate a low rank matrix to approximate the original rating matrix, therefore, the recommendations based on predicted ratings can be made to users [24].

It combines the predicted ratings from collaborative filter and ALS algorithm by a weight fusion model [24]:

$$\hat{r} = \alpha \cdot r^{(}1) + (1 - \alpha) \cdot r^{(}2) \tag{56}$$

Where $\hat{r}, r^{(}1), r^{(}2)$ are the final predicted rating, predicted rating from the collaborative filer and ALS algorithm respectively [24]. α is the weighted parameter.

In experiment, MovieLens is used as the dataset, in test the two algorithms in two different environment [14]. In the first environment, it runs the two algorithms separately. In the other environment, it runs them in parallel on Spark clusters with four nodes added one by one [14]. The result shows that the processing time is reduced when there more nodes added to the system, and the algorithms run in parallel on a single node executes faster than the first environment when they are run alone [24]. The evaluation metric used here is MAE, it compares the fusion model with two separate algorithms, the result shows that the fusion model has the lowest error compared to other methods, while ALS performs better than the item based collaborative filter [14].

4.4.2 Machine Learning Methods

On the podcast system, the volume is huge and it is difficult to tag the audio items, thus they do not have enough features for the similarity computation, which is the base of making recommendations [22]. In [22] which is published in 2016, it tries to address the problem by building a podcast recommendation system which uses text based features associated with the audios to generate latent embeddings based on Natural Language Processing(NLP) method, the generated embeddings can then be used for similarity computation among podcast items [22]. After that, a content based recommender with decision tree is used to predict whether the target user likes the item [22].

NLP represents words by vectors and it shows similarity between the vectors

[22]. To process Chinese sentence, the probability for a particular sequence of words needs to be computed, this is done by a word segmenter which is based on conditional random field(CRF) sequence model [22]. Chinese Wikipedia is used as the source of Chinese words segmentation, after getting the segmented words using the word segmenter, SGNS model is used to maximize the probability that the word and context pair comes from training corpus, which is to maximize the following objective equation [22]:

$$\sum_{w,c \in D^+} log \frac{1}{1+e^{-v_c \cdot v_w}} + \sum_{w,c \in D^-} log \frac{1}{1+e^{v_c \cdot v_w}} \tag{57}$$

Where w, c are the word sets and their contexts, v_w, v_c are their vector representations, D^+ are the correct word context pair from the training dataset, D^- are the incorrect word context pairs [22].

After learning the embeddings of all the words, it represents each podcast item as vectors including features such as tags, title, descriptive information and so on, then it computes the similarity between the items [22].

In experiment, it uses dataset from Ximalaya which is a podcast provider in China [22]. After obtaining the embeddings from the text based features from the source dataset, it predicts which category each audio item should belong to using NLP and compares the result with its actual category, precision is used as the evaluation method [22].

Then it builds a regression or decision tree based classifier to predict whether the user will like the podcast item [22]. And compares the predicted result with the true label, Receiver Operating Characteristic (ROC) is used here as the evaluation method [22]. The result is quite promising since the system can achieve around 80% precision value when it predicts the item's category, and it gives almost all true positive prediction when it predicts whether the user likes the podcast item [22].

In [16] which is published in 2017, it tries to address the 0/1 recommendation problem by proposing a hybrid filtering algorithm which combines user ratings and natural language text of the item, where natural language text is used to construct a content based filter, user ratings are used to construct a user based collaborative filter [16]. The result from the experiment on MovieLens dataset is promising.

It consists of a content based filter, a collaborative filter and a recommendation supervisor [16]. The recommendation supervisor is a two layers feedforward neural network(FNN) [22]. The content based and collabotive filers are used to predict whether the target user likes the target item, then a recommendation filter is used to compute the probability that the user likes the item to get a ranked list of items, then top 5 items are recommended to the user [16].

In the content based filter, doc2vec embedding is applied to the the natural language text of the target item, which can project the sparse data into a dense vector space, then a supervised classification model Random Forest is used to predict whether the user likes the item based on the dense vector space since it performs better than naive Bayes and SVM [16]. Accuracy, precision and recall are used to measure the performance of the supervised model [16].

In the collaborative filter, Pearson correlation based similarity metric is used to compute the similarity between the users, after finding the neighbors of the target user based on similarity, it predicts the rating of target user i on the target item p as the weighted average of ratings from the user's neighbors N_i using the following equation [16]:

$$\hat{r}_{i,p} = (\overline{r}_i + \frac{\sum_{j \in N_i} sim(i,j) \cdot (r_{j,p} - \overline{r}_j)}{\sum_{j \in N_i} sim(i,j)}) \cdot \frac{1}{6} \tag{58}$$

Here the predicted rating is scaled by $\frac{1}{6}$ to get probability output between 0 and 1 [22].

In the recommendation supervisor, it combines the recommendation result given by the content based and collaborative filter to get a ranked list for the user [16]. It learns a feedforward neural network, where the dataset includes the user-item rating pair, probability that the user likes the item from the collaborative filter, accuracy, precision and recall result from content based filter [16]. Then it predicts the probability that the user i likes the item p based on the following equation [16]:

$$P_{i,p} = f\Big(\alpha \cdot \hat{r}^{cb}_{i,p} + \beta \cdot \hat{r}^{cf}_{i,p} + \sum_j (\gamma_j \cdot CP_j)\Big) \tag{59}$$

where f is the softmax function, $\hat{r}^{cb}_{i,p}, \hat{r}^{cf}_{i,p}$ are the predicted rating from content based and collaborative filter respectively [16]. CP_j is the j_{th} component of the vector containing accuracy, precision and recall from previous filters [16]. The number of hidden layers, neurons for each hidden layer, activation function for each neuron from the network are chosen to optimize α, β, γ parameters [16]. After computing the probability, the items are ranked by the probabilities that top 5 items are recommended to the user. The performance of recommendation system is measure using MAP@5.

In experiment, MovieLens 1M dataset which includes 6000 users and 4000 movies is used [22]. The result shows that hybrid filter outperforms the content based and collaborative filters in terms of MAP@5 evaluation metric [22].

In [26] which is published in 2018, it proposes a hybrid recommendation system which can help users find interesting news articles using both text and image information [26]. It uses Convolutional Neural Network(CNN) to produce text eigenvector from the text, while VGG-16 model is used to extract image eigenvector from the image [26]. Then Multi-layer Perceptron(MLP) is used to classify the news based on the image and text eigenvectors [26]. Therefore, recommendation of news articles can be made. It compares this method with other methods including CNN, Long Short-Term Memory(LSTM) based on the text eigenvector, and it shows that this method performs the best [26].

Its CNN model consists of input layer, convolution layer, max pooling and fully connected layers [26]. The text from the articles are represented by matrix of words and is the input of input layer, the convolution layer obtains abstract information of the input, then max pooling layer further downsamples the data and extract important features from the output of convolutional layer, then it directs to the fully connected layer to classify the tag of the articles [26]. The image information is processed in a similar way, and it is dealt with by a specific CNN model VGG-16, it consists of 14 convolution layers and three fully connected layers where the last connected layer uses softmax function to produce a probability output [26].

Autoencoder is used to reduce the dimension of image eigenvectors, compared with linear dimension reduction method PCA which performs poorly when the data does not have a linear distribution or Gauss distribution, nonlinear method Autoencoder needs to be used in such cases [26]. Autoencoder consists of an encoder and a decoder, where encoder reduce the dimension of original data, and decoder reconstructs the data, its training process is to get a optimal weight of the network so that the reconstructed data is close to the original data [26].

After reducing the dimension of the image eigenvector, both the image and text eigenvectors are spliced to one vector, then this vector is sent to MLP, which consists of a fully connected layer, a dropout layer and another fully connected layer to classify the tag of the target article [26]. Therefore, the news can be recommended to the user based on its tag [26].

In experiment, the text dataset is crawled from www.news.ifeng.com and it includes 2278 news articles, the image dataset is obtained from Mirflicker dataset and it includes 10000 images [26]. It compares the proposed method to other methods including CNN, KNN, LSTM, MLP, Naive Bayes and SVM [26]. The result shows that CNN performs the best among CNN, KNN, LSTM methods, while in tradition machine learning methods, SVM performs better than Naive Bayes [26]. In general, the proposes method performs the best in terms of accuracy rate.

4.4.3 Matrix Factorization

In [12] which is published in 2014, it tries to address the data sparseness problem with matrix factorization techniques, it proposes a collaborative filter with SVD [12].

In the system, users are clustered based on the similarity metric Euclidean distance, SVD is used here for dimension reduction, which reduces the dimension of original user item rating matrix by finding the most approximate lower dimension matrix [12]. This process can help remove unimportant information which does not contribute much to find the correlation between variables and items [12]. Items are represented as vector, and they are also clustered based on Euclidean distance. After getting the clustered items, for each user, it first finds the similar users to the target user, then recommend items which are rated highly by the users [12].

In experiment, precision is used as the evaluation metric. Million Song Dataset(MSD) is the source of dataset, it consists of audio features and metadata for a million contemporary music tracks [12]. Last.fm from MSD is used as the dataset, it is a music web portal which has more than 30 millions users and millons of songs [12]. Out of the huge dataset, 10 users and their recording data are selected for experiment, it includes 22000 records and 8240 items. Users whose similarity with the target target user exceeds a threshold value are considered neighbors of the target user [12]. Different thresholds are tested during the experiment, the result shows that the precision and number of clusters decreases when the threshold value increases [12].

Another paper [5] tries to improve the traditional matrix factorization techniques by using user preferences information, it proposes an algorithm Tag-GSVD++ based on gSVD++ and tag sharing models [5]. The experiment shows that this method is more accurate than other approaches and can also help address the cold start problem.

In the proposed SVD++ algorithm, the rating from target user u on target item i is estimated using user, item and implicit feedback factors p_u, q_i, y_j based on the following equation [5]:

$$\hat{r}_{ui} = q_i^T \Big(p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j\Big) \tag{60}$$

Here $p_u \in \mathbb{R}^k, q_i \in \mathbb{R}^k, y_j \in \mathbb{R}^k$, k is the number of latent features, $N(u)$ is the item sets which are given implicit preference by the user [5].

Then it tries to minimize the loss function, which is the regularized square error loss between the predicted rating and real ratings. Stochastic gradient descent is used to faster this process [5].

The gSVD++ extends the SVD++ algorithm by considering also items' attributes, it computes the rating based on the following equation [5]:

$$\hat{r}_{ui} = \Big(q_i + |G(i)|^{-\beta} \sum_{g \in G(i)} x_g\Big)^T \Big(p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j\Big) \tag{61}$$

where $G(i)$ is the attributes related to item i, the value of $beta$ is set to 0 if there is no item attribute, and is set to be between 0 and 1 if there are attributes, $x_g \in \mathbb{R}$ is the latent variables from the metadata in the dataset [5].

Then it tries to minimize the loss function with regularization. Stochastic gradient descent is also used here [5]. It adapts the gSVD++ to be TagGSVD++ so that the tagging information shared across domains can be utilized, it is developed based on UserItemTags model and UserItemRelTags model [5].

In experiment, the movie dataset MovieLens and book dataset LibraryThing are used [5]. It compares the performance between six methods: Matrix factorization method based on SVD, SVD++, gSVD++, tags sharing based methods UserItemTags, UserItemRelTags ItemRelTags and TagGSVD++ [5]. MAE is used as the evaluation metric. Grid search is used to find the best setting of hyperparameters k, α, β, which are the number of latend features, learning rate, and regularization value respectively [5]. After finding the best parameters settings for all the methods, it test the performance of these methods, and the result shows that the proposed TagGSVD++ outperforms all other methods for all data sparsity levels [5]. It has also the best performance in cold start cases compared to other methods [5].

In [18] which is published in 2017, it proposes a collaborative filtering algorithm utilizing matrix factorization algorithm NALS-WR(normalized ALS-WR) [18] with Spark and compares it with other two matrix factorization algorithms single value decomposition(SVD) and ALS-WR [18], it aims at addressing the poor extendibility, data sparseness and low efficiency problems that exist in large dataset [18]. Spark is used here to improve the data processing speed. The experiment with MovieLens dataset shows that this method is more efficient compared to ALS-WR(alternating least-squares with weighted-λ-regularization) in Hadoop and SVD, it has better extendibility and sparseness resistance as well [18].

In this system, Apache Spark is used to facilitate the process of data since it is 10 times faster than Hadoop and it is suitable for iterative calculation in matrix factorization algorithms [18]. It has a Resilient Distributed Dataset(RDD) which is distributed and has multiple threads so that data can be processed by parallel, it is also resilient since it can prevent data loss [18].

The matrix factorization algorithm ALS-WR aims to find a low ranking matrix X which is most similar to original rating matrix by the following equation [18]:

$$X = UV^T \tag{62}$$

where $U \in C^{m \times d}, V \in C^{n \times d}$, here d is the number of features [18] [18]. Then it tries to minimize the sum-of-square-error loss function, and regularization is used to prevent overfitting [18].

Based on ALS-WR, NALS-WR is used in the system, it extends ALS-WR so that computation can be split on clusters, it tries to find two low dimensional matrices to approximate the original rating matrix P, where the loss function is sum-of-square-error loss with regularization [18]. The two low dimensional matrices user matrix $U_{i.}$ and item matrix $V_{j.}$ are computed based the following equations [18]:

$$\begin{aligned}
A_1 &= \frac{P_{i.}V_{ui}}{V_{ui}^T V_{ui} + \lambda n_{ui} I} - (P_{i.}V_{ui}(V_{ui}^T V_{ui} + \lambda n_{ui} I)^{-1})_{min} \\
B_1 &= (\frac{P_{i.}V_{ui}}{V_{ui}^T V_{ui} + \lambda n_{ui} I})_{max} - (P_{j.}^T U_{mj}^T (U_{mj}^T U_{mj} + \lambda n_{mj} I)^{-1})_{min} \\
U_{i.} &= \frac{A_1}{B_1} * s + t, i \in [1, m] \\
A_2 &= \frac{P_{.j}^T U_{mj}}{U_{mj}^T U_{mj} + \lambda n_{mj} I} - (P_{.j}^T U_{mj}(U_{mj}^T U_{mj} + \lambda n_{mj} I)^{-1})_{min} \\
B_2 &= (\frac{P_{.j}^T U_{mj}}{U_{mj}^T U_{mj} + \lambda n_{mj} I})_{max} - (P_{.j}^T U_{mj}(U_{mj}^T U_{mj} + \lambda n_{mj} I)^{-1})_{min} \\
V_{j.} &= \frac{A_2}{B_2} * s + t, j \in [1, n]
\end{aligned} \tag{63}$$

where $P_{i.}, P_{.j}$ are the rating matrix of user i and item j respectively [18]. V_{ui} is the feature matrix by user i, U_{mj} is the feature matrix for item j, n_{ui} is the number of items user i have commented, n_{mj} is the number of users who have commented on item j, λ is the regularization parameter for the loss function, s, t are the adjustable parameters depending on the rating scale of items [18]. The two matrices are updated iteratively during the process [18].

Then the rating for the target user i on target item j can be predicted by the following equation [18]:

$$\hat{p}_{ij} = \frac{(U_i^T V_j - (U_i^T V_j)_{min})}{(U_i^T V_j)_{max} - (U_i^T V_j)_{min}} * s + t \tag{64}$$

In experiment, MovieLens is used as the dataset, RMSE is used as the evaluation metric. The result shows that NALS-WR performs better than ALS-WR and SVD [18]. It also compares the runtime between the three algorithms, SVD is used stand-alone, while NALS-WR is used in spark, and ALS-WR is used with Hadoop [18]. It shows that for large dataset, NALS-WR is far more efficient than other two algorithms.

5 Conclusion

This study provides a systematic literature review of three major types of recommendation systems including content based filter, collaborative filter and hybrid filter which combines the content based and collaborative filters. It can help researchers choose which recommendation system is suitable for their research. It covers 27 papers extracted from Scopus, which is Elsevier's abstract and citation database of peer-reviewed literature launched in 2004.

Based on the brief summary of these papers, it shows that the recommendation system basically follows these steps, it first collects data by either creating their own dataset or using existing dataset, and the most commonly used dataset among these papers is MovieLens provided by GroupLens, which consists of a great amount of users, movies and their corresponding ratings. After collecting the dataset, some machine learning based approaches like NLP can be used to extract the interesting features from the dataset which results in a feature descriptor in the form of vector or matrix composed of vectors for users, items and their ratings. Then similarity metrics can be used to compute the similarity between these vectors, in content based filters, TF-IDF is commonly used to help find the similar users or items by measuring their weights and ranking the result. After computing the similarities, the rating for the target user on the target item is predicted for all the items, then the items are ranked by their ratings so that the recommendation list consists of the items with highest ratings can be generated.

Performance enhancement approaches can be used during the process. These include machine learning methods like NLP, k-means, k nearest neighbors, matrix factorization methods like SVD and its variations, and big data framework tools like Hadoop and Spark.

Performance of recommendation systems are evaluated in various approaches including MAE, RMSE, Precision and Recall, and F-measure which combines the Precision and Recall.

As for the research problems, most of papers try to address the information overload problem and data sparsity problem. Some papers try to address cold start problem or improve recommendation accuracy. Some papers propose new similarity metric methods based on previous approaches. Some papers try to reduce the time for training the model and making recommendations. Others try to adapt recommendation system to a specific domain.

References

[1] *MS Windows NT almalis2015fodradataset.* `www.kaggle.com/c/job-recommendation/data`. Accessed: 2010-09-30.

[2] Almalis, Nikolaos D, Tsihrintzis, George A, Karagiannis, Nikolaos, and Strati, Aggeliki D: *Fodra—a new content-based job recommendation algorithm for job seeking and recruiting.* In *Information, Intelligence, Systems and Applications (IISA), 2015 6th International Conference on*, pages 1–7. IEEE, 2015.

[3] Chikhaoui, Belkacem, Chiazzaro, Mauricio, and Wang, Shengrui: *An improved hybrid recommender system by combining predictions.* In *Advanced Information Networking and Applications (WAINA), 2011 IEEE Workshops of International Conference on*, pages 644–649. IEEE, 2011.

[4] De Pessemier, Toon, Vanhecke, Kris, Dooms, Simon, and Martens, Luc: *Content-based recommendation algorithms on the hadoop mapreduce framework.* In *7th International Conference on Web Information Systems*

and Technologies (WEBIST-2011), pages 237–240. Ghent University, Department of Information technology, 2011.

[5] Fernández-Tobías, Ignacio and Cantador, Iván: *Exploiting social tags in matrix factorization models for cross-domain collaborative filtering.* In *CBRecSys@ RecSys*, pages 34–41, 2014.

[6] Gao, Zhipeng, Lu, Zehui, Deng, Nanjie, and Niu, Kun: *A novel collaborative filtering recommendation algorithm based on user location.* In *Consumer Electronics-Taiwan (ICCE-TW), 2016 IEEE International Conference on*, pages 1–2. IEEE, 2016.

[7] Guo, Bo, Xu, Shiliang, Liu, Dongdong, Niu, Lei, Tan, Fuxiao, and Zhang, Yan: *Collaborative filtering recommendation model with user similarity filling.* In *Information Technology and Mechatronics Engineering Conference (ITOEC), 2017 IEEE 3rd*, pages 1151–1154. IEEE, 2017.

[8] Hannon, John, Bennett, Mike, and Smyth, Barry: *Recommending twitter users to follow using content and collaborative filtering approaches.* In *Proceedings of the fourth ACM conference on Recommender systems*, pages 199–206. ACM, 2010.

[9] Kbaier, Mohamed Elyes Ben Haj, Masri, Hela, and Krichen, Saoussen: *A personalized hybrid tourism recommender system.* In *2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)*, pages 244–250. IEEE, 2017.

[10] Mao, Jiumei, Cui, Zhiming, Zhao, Pengpeng, and Li, Xuehuan: *An improved similarity measure method in collaborative filtering recommendation algorithm.* In *Cloud Computing and Big Data (CloudCom-Asia), 2013 International Conference on*, pages 297–303. IEEE, 2013.

[11] Quispe, Laura Cruz and Luna, José Eduardo Ochoa: *A content-based recommendation system using trueskill.* In *Artificial Intelligence (MICAI), 2015 Fourteenth Mexican International Conference on*, pages 203–207. IEEE, 2015.

[12] Reddy, M Sunitha and Adilakshmi, T: *Music recommendation system based on matrix factorization technique-svd.* In *Computer Communication and Informatics (ICCCI), 2014 International Conference on*, pages 1–6. IEEE, 2014.

[13] Saaya, Zurina, Rafter, Rachael, Schaal, Markus, and Smyth, Barry: *The curated web: a recommendation challenge.* In *Proceedings of the 7th ACM conference on Recommender systems*, pages 101–104. ACM, 2013.

[14] Saravanan, S: *Design of large-scale content-based recommender system using hadoop mapreduce framework.* In *Contemporary Computing (IC3), 2015 Eighth International Conference on*, pages 302–307. IEEE, 2015.

[15] Shen, Yi, Yu, Jianjun, and Nan, Kai: *A hybrid recommender model for scientific research resources.* In *Wireless Communications, Networking and Mobile Computing (WiCOM), 2012 8th International Conference on*, pages 1–4. IEEE, 2012.

[16] Sottocornola, Gabriele, Stella, Fabio, Zanker, Markus, and Canonaco, Francesco: *Towards a deep learning model for hybrid recommendation.* In *Proceedings of the International Conference on Web Intelligence*, pages 1260–1264. ACM, 2017.

[17] Thuan, To Thi and Puntheeranurak, Sutheera: *Hybrid recommender system with review helpfulness features.* In *TENCON 2014-2014 IEEE Region 10 Conference*, pages 1–5. IEEE, 2014.

[18] Tu, Xiaohan, Liu, Siping, and Li, Renfa: *Improving matrix factorization recommendations for problems in big data.* In *Big Data Analysis (ICBDA), 2017 IEEE 2nd International Conference on*, pages 193–197. IEEE, 2017.

[19] Turnip, Rudolf, Nurjanah, Dade, and Kusumo, Dana Sulistyo: *Hybrid recommender system for learning material using content-based filtering and collaborative filtering with good learners' rating.* In *2017 IEEE Conference on e-Learning, e-Management and e-Services (IC3e)*, pages 61–66. IEEE, 2017.

[20] Wang, Jingjin, Lin, Kunhui, and Li, Jia: *A collaborative filtering recommendation algorithm based on user clustering and slope one scheme.* In *Computer Science & Education (ICCSE), 2013 8th International Conference on*, pages 1473–1476. IEEE, 2013.

[21] Wintrode, Jonathan, Sell, Gregory, Jansen, Aren, Fox, Michelle, Garcia-Romero, Daniel, and McCree, Alan: *Content-based recommender systems for spoken documents.* In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 5201–5205. IEEE, 2015.

[22] Xing, Zhou, Parandehgheibi, Marzieh, Xiao, Fei, Kulkarni, Nilesh, and Pouliot, Chris: *Content-based recommendation for podcast audio-items using natural language processing techniques.* In *Big Data (Big Data), 2016 IEEE International Conference on*, pages 2378–2383. IEEE, 2016.

[23] Xu, Bing, Zhang, Mingmin, Pan, Zhigeng, and Yang, Hongwei: *Content-based recommendation in e-commerce.* In *International Conference on Computational Science and Its Applications*, pages 946–955. Springer, 2005.

[24] Yang, Yongli, Ning, Zhenhu, Cai, Yongquan, Liang, Peng, and Liu, Haifeng: *Research on parallelisation of collaborative filtering recommendation algorithm based on spark.* International Journal of Wireless and Mobile Computing, 14(4):312–319, 2018.

[25] Yin, Fulian, Liu, Xiaowei, Ding, Wanying, and Zhang, Ruizhe: *Tag-based collaborative filtering recommendation algorithm for tv program.* In *Wavelet Active Media Technology and Information Processing (ICCWAMTIP), 2016 13th International Computer Conference on*, pages 47–52. IEEE, 2016.

[26] Yu, Boyang, Shao, Jiejing, Cheng, Quan, Yu, Hang, Li, Guangli, and Lü, Shuai: *Multi-source news recommender system based on convolutional neural networks.* In *Proceedings of the 3rd International Conference on Intelligent Information Processing*, pages 17–23. ACM, 2018.

[27] Zhao, YF, Wang, Fei Yue, Gao, Hang, Zhu, FH, Lv, YS, and Ye, PJ: *Content-based recommendation for traffic signal control.* In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pages 1183–1188. IEEE, 2015.

[28] Zhu, Yuanqing, Song, Wei, Liu, Lizhen, Zhao, Xinlei, and Du, Chao: *Collaborative filtering recommender algorithm based on comments and score.* In *Computational Intelligence and Design (ISCID), 2017 10th International Symposium on*, volume 1, pages 304–307. IEEE, 2017.