Arab J Sci Eng
DOI 10.1007/s13369-016-2410-1

RESEARCH ARTICLE - COMPUTER ENGINEERING AND COMPUTER SCIENCE

Building Accurate and Practical Recommender System Algorithms Using Machine Learning Classifier and Collaborative Filtering

Asma Sattar[1] · **Mustansar Ali Ghazanfar**[1] · **Misbah Iqbal**[1]

Received: 1 July 2016 / Accepted: 14 December 2016

Abstract Recommender systems use machine learning and data mining techniques to filter unseen information and predict whether a user would like a particular item. A major research challenge in this field is to make useful recommendation from available set of millions of items with sparse ratings. A large number of approaches have been proposed aiming to increase accuracy, but they have ignored potential problems, such as sparsity and cold start problems. From this line of research, in this research work, we have proposed a novel hybrid recommendation framework that combines content-based filtering with collaborative filtering that overcome aforementioned problems. Our experimental results show that this performance of proposed algorithm is better or comparable with the individual content-based approaches and naive hybrid approaches, while it eliminates various problems faced by recommender systems.

Keywords Recommender systems · Collaborative filtering · Content-based filtering

1 Introduction

1.1 Recommender Systems

Recommender systems are significant part of e-commerce and information ecosystems. They are the powerful methods that enable users to filter through large amount of product space and information. There is plethora of information available on Internet, such as videos in Reel (Reel.com), YouTube (Youtube.com) and Netflix (netflix.xom), music in MyStrand (myStrand.com) and Last.fm (last.fm), and electronic resources in Delicious (delecious.com), CleverSet (cleverSet.com) and Amazon (amazon.com). Everyday, we are overwhelmed with options and choices. What to eat? What shoes, dress, etc., to buy? Which movie to watch? Which novel to read? Moreover, sizes of decision domains are growing massively; for instance, Netflix has more than 17,000 movies in its selection database [5] and there are 410,000 titles in Kindle store in Amazon [15]. Information discovery in such a massive size data is a potential challenge. Even a simple decision, such as *'which movie should a user watch this weekend?'* can be a difficult task without any previous direct knowledge about that particular candidate. Without using any filtering approach, it is not possible for a user to find something that is useful for them. Recommender system (RS) is an information filtering technique and a software tool that suggest items to users relevant to their use, and hence ease the process of decision making for a particular user.

Industries are showing significant interest in recommender system. Amazon, one of the famous e-commerce company, has been using collaborative filtering to recommend products to customers for a decade. Netflix, perhaps the most popular movie recommendation site, enhanced the recommender system technology underneath their service of movie rental

✉ Asma Sattar
asma_uett@yahoo.com

Mustansar Ali Ghazanfar
mustansar.ali@uettaxila.edu.pk

Misbah Iqbal
misbah_iqbal2k9@yahoo.com

[1] Department of Software Engineering, Faculty of Telecommunication and Information Engineering, University of Engineering and Technology, Taxila, Pakistan

Published online: 09 February 2017

at $1M through Netflix Prize [5]. Recommended items for a particular user are represented as a ranked list. A recommender system predicts which services or products are most suitable based on the constraints and preferences of users. User's preference can be collected from ratings of users for a particular item or inferred from user's action on the system.

1.2 Formalization of Recommender Systems

There are two basic elements of recommender system—users and items, where users rate items on a particular scale. We represent users by U and items rated by user by I. In this work, $\mathcal{U} = \{u_1, u_2, \ldots, u_N\}$ denotes user-set, where $|\mathcal{U}| = N$ is the total number of users who are using the system. $\mathcal{I} = \{i_1, i_2, \ldots, i_M\}$ denotes item-set, where $|\mathcal{I}| = \mathrm{M}$ is the total number of items in the system. User's profile consists of various characteristic of user, such as rating assigned to various items, gender, age, location. Item profile consists of characteristic that are defining the item. For example, in case of movie recommender system, each movie can be represented by actors, director, year of release, type of movie, sound tracks, description of movie, production company, etc. Ratings of a user 'u' for a particular item 'i' are denoted by $r_{i,u}$, $(r_{i,u}|(i, u) \in \mathcal{D})$, where $\mathcal{D} \subset \mathcal{I} \times \mathcal{U}$. $\mathcal{D}$ is a set of user–item rating pairs. We represent total number of ratings provided by all users with $|\mathcal{D}| = T$. Typically, every user rates small number of available items that are represented by $|\mathcal{D}| = T \lll |\mathcal{I} \times \mathcal{U}| == M \times N$. In reality, practical systems have $T\ /\ M \times N \cong 0.01$. Set of ratings of user for items can be represented through $M \times N$ rating matrix called user–item rating matrix.

1.3 Categories of Recommender Systems

There are five main classes of recommender systems: collaborating filtering [27], content-based filtering (CBR) [18], knowledge- or ontology-based systems [37], demographic-based filtering and hybrid recommender systems [18]. In this research work, our focus is on collaborative filtering and content-based filtering.

1. *Collaborative Filtering (CF)*

   The most popular algorithm for recommendation is collaborative filtering (CF), and it uses rating behavior of other users in the system for making prediction/recommendation. This algorithm assumes there exists a user–item rating matrix where row of matrix represents user profile and column represents item profile.

   CF-based systems can be categorized into two subcategories: *model-based CF* and *memory-based CF* [22]. In this research work, our focus is on memory-based collaborative filtering. In memory-based collaborative filtering, ratings of users provided for items are used to make recommendation. The first automated collaborative filtering known as K-nearest neighbor (KNN) collaborative filtering was introduced in GroupLens Usenet article recommender [41]. KNN classifies an instance by finding its nearest neighbors and picking the most common class among the neighbors. Two famous types of memory-based CF are item-based collaborative filtering and user-based collaborative filtering [15].

   Various similarity measures are used to find similarity between instances (user/item). Two widely used measures are cosine- and correlation-based similarity measures. In correlation-based approach, the Pearson correlation is used for finding similarity. The Pearson correlation [8] between two users u_a and u_b is computed as follows:

   $$sim(u_a, u_b) = \frac{\sum_{i \in I_{u_{a,u_b}}} \mu_{i,u_a} \mu_{i,u_b}}{\sqrt{\sum_{i \in I_{u_{a,u_b}}} \mu_{i,u_a}^2 \sum_{i \in I_{u_{a,u_b}}} \mu_{i,u_b}^2}},$$

   where $\mu_{i,u} = r_{i,u} - r_u$.

   Adjusted cosine similarity is another widely used technique to find similarity between two instances. Formally, the similarity between two users u_a and u_b can be found:

   $$sim(i, j) = \frac{\sum_{u \in U} (R_{u,i} - R_u)(R_{u,j} - R_u)}{\sqrt{\sum_{u \in U} (R_{u,i} - R_u)^2} \sqrt{\sum_{u \in U} (R_{u,j} - R_u)^2}},$$

   where $R_{u,i}$ denotes rating of user 'u' for item 'i,' and R_u denotes average rating of user 'u.' Output range of Pearson coefficient and adjusted cosine similarity is from 1 to -1, where 1 implies both users are perfectly similar to each other, -1 implies both users are very dissimilar to each other, and 0 implies there is no similarity relation. Although systems based on CF tried best in past and have been successful, it suffers from potential problems, such as sparsity and cold start problems. The algorithm proposed in this research work solves the sparsity and cold start problems.

2. *Content-Based Filtering*

   Content-based recommender systems analyze description of items and/or set of items that are previously rated by user and build a profile or model of user's interests which is based on object's feature rated by the user. A profile is structured representation of a user interests and used to recommend new items to the user. The recommendation process involves matching the attributes of user profile with the attributes of the item, and this provides the results that show level of interest of user in that particular object. Content-based information filtering systems require proper techniques to represent items, produce user profile and strategies for comparing user and item profile representation.

1.4 Problem Statement and Design Objectives

By doing careful study of the literature, we found that current state-of-the-art algorithms are not practical, ideal or flexible for the applications of real world. This is because process of generating recommendations is complex, and its quality depends on various factors like may be some algorithms perform well provided a dense, but it produces poor quality recommendations for the sparse dataset. Most of work proposed in the literature assume that dataset is static. In practical, however users keep on registering to the system as a result dataset is imbalance and new users have rated less items, which are denoted as cold start users. In the literature, researchers divide the dataset into training and testing and choose the training set with dense user/item profile, which can make evaluation strategy questionable. The performance of recommendation algorithm should be checked over imbalance dataset.

This research work aims to propose a novel framework of hybrid recommender system that combines collaborative filtering with various content-based approaches and improve the quality of recommendation for sparse dataset. We have proposed various algorithms to accomplish the following design objectives:

- *Accuracy*: Accuracy is one of the most important design objective of recommender systems. Although the current state-of-the-art algorithms, especially the ones which are proposed in Netflix competition, are quite accurate, they get good accuracy for specific dataset having particular characteristics. Moreover, these algorithms ignore design objectives such as cold start, imbalance and sparse dataset, and hence, they become impractical for applications of the real-world recommender system. We want a recommendation algorithm to be accurate for sparse dataset.
- *Scalability*: We need a recommendation algorithm that is scalable with the increase in data. Less the time algorithm takes to produce quality recommendation, more scalable it is.
- *Cold Start user*: In practical, users keep on registering to the system as a result dataset is imbalance and new users have rated less items, which are denoted as cold start users. In the literature, researchers divide the dataset into training and testing and choose the training set with dense user/item profile, which can make evaluation strategy questionable. The Performance of recommendation algorithm should be checked over imbalance dataset.
- *Cold Start item*: When some new items are added in the system, it is not possible to get rating of those items from significant number of users, such items are called cold start items. We need an algorithm that produce quality recommendations under presence of such cold start items in the system.
- *Sparsity*: Presence of cold start items and users make the dataset sparse. The hybrid algorithm proposed in this research work provides accurate recommendation under presence of sparse data.

Through experiments over two different datasets, we show that our proposed hybrid recommender system gives more accurate results than individual ones and also better than naive hybrid recommender systems. Moreover, it overcomes some of the traditional problems in the recommender system, such as cold start and sparsity problems.

This paper has been organized as follows. Section 2 gives an overview of existing algorithms used for recommendations and describes the limitations of the current state-of-the-art algorithms. Section 3 describes the experimental setup. In Sect. 4, item-based and user-based collaborative filtering, various content-based algorithms and novel hybrid approaches are discussed in detail. All results are presented in Sect. 5 followed by Sect. 6 which gives conclusion of research done in this thesis along with possible future advancement in this work.

2 Related Work

In this section, we give overview of content-based approaches, neighborhood-based approaches (CF) and various hybrid approaches.

2.1 Neighborhood-Based Approaches

The entire training dataset is used as a model in K-nearest neighbor-based algorithms. For prediction of unseen data instance, KNN-based algorithm search through the training data and find K most similar instances to the problem instance. Prediction attribute of these most similar instances is combined and summarized to find prediction of unseen problem instance. Collaborative filtering is good in a way because it can recommend out of the box [9]. For example, a user who loves to watch action movies can also enjoy romantic movies.

CF-based systems has two sub-categories, namely model-based and memory-based CF. Our focus in this research is memory-based CF. Memory-based collaborative filtering is further divide into two sub-categories user-based collaborative filtering and item-based collaborative filtering.

- *User-Based CF*: User-based collaborative filtering uses rating of similar user of a target user to predict rating of a target item. It involves following steps:
  - Find similarity of active user with all users in training set using similarity metrics.
  - Select top 'n' users, these are very similar to active user.
  - Ratings of these users are normalized and predicted rating is calculated from weighted combination of all selected neighbor's ratings. The predicted rating of an item for 'i' for user 'u,' $P_{u,i}$ can be calculated mathematically as:

    $$P_{u,i} = \frac{\sum_{ksimilarusers} S_{u,N} \times R_{u,N}}{\sum_{ksimilarusers} S_{u,N}}$$

    where $S_{u,n}$ is the similarity between two users, and R is the rating of a user similar to active user.
  - Items having highest rating are then recommended to user.
- *Item-Based CF*: Item-based CF (Sarwar et al. [44]) is the most popular example of memory-based approaches. It only examines the profile of an active user and compute the similarity of target item with all items already rated by an active user. It involves following steps:
  - In first phase, item rated by the active users is retrieved.
  - In next phase, similarity of retrieved items is computed with target item. Cosine similarity can be used for this purpose [44].
  - In last phase, prediction $P_{u,i}$ of most similar items is made.

    $$P_{u,i} = \frac{\sum_{ksimilaritems} S_{i,N} \times R_{i,N}}{\sum_{ksimilaritems} S_{i,N}}$$

    where $S_{i,N}$ is the similarity between two items, and R is the rating of an item similar to a target item.
  - Items having highest ratings are then recommended to user.

  Practically many recommender systems used to evaluate large item sets (e.g., Cdnow.com, Amazon.com, Bellcore, Ringo). Active user may purchase well under 1percent of the items in these systems. 1 percent of 3 million books is 300,000 books. Accordingly KNN-based recommender system is unable to recommend items to active users. Typical recommender systems suffer serious problem of scalability with millions of users and items. Nearest neighbors-based algorithm needs computation that grows with the increase in number of items and/or users. Improved accuracy and better scalability result in making an item-based approach more favorable in various cases [2,11]. Item-based approach provides reasoning behind prediction because user do not know about those allegedly like-minded users, but they are familiar with those items which they preferred previously.

  Largest e-commerce sites that have implemented collaborative filtering directly operate with stress. User-based similarity computation process in neighborhood-based collaborative filtering method is the bottleneck of the system which turns the whole process of recommendation unsuitable for real-time systems. In order to ensure high scalability, model-based recommender systems can be used as they have potential to operate at high scale. Main idea here is to isolate the prediction generation and neighborhood generation process.

2.2 Content-Based Approaches

Content-based recommender systems analyze description of items and/or set of documents that are previously rated by the user and build a profile or model of user interests, which is based on objects feature rated by a particular user. Profile is structured representation of a user interests and used to recommend new items to the user. There are various systems that rely on content-based filtering to help users in finding information on World Wide Web.

Items' description can be automatic, where algorithms are used for feature extraction from item's description or manual where domain experts explain the items. Also, recent social tagging Web sites, e.g., flicker can be used to describe an item by allowing the user to tag various items.

In CBF, words are being utilized as features. Majority of CBF systems use plain words and some other used n-grams [17]. Jiang et al. were topics that are combination of words occurring as social tags in CiteULike [30]. Giles et al. used citations in similar way as words are used and called this approach CC-IDF which is based on standard TF-IDF method. Some other systems also adapted this CC-IDF idea or used this method as baseline. Kahani and Zarrinkalam take authors as features to determine similarities by examining number of authors which two items shared [50]. All features of an item are not equally important as all of them are not describing items feature or information needs of user equally well, so giving weights to features is a key for successful CBF. The most popular one is TF-IDF approach.

Features that are retrieved only from the content are not enough. For example, content-based recommender system for movie recommendation is not based on the movie content. Instead it consider terms in the metadata, i.e., description, title or cast names. Some approaches extract text from abstract [14], and some extract text from title, from header, foreword, introduction, bibliography and from author provided keywords [14]. In addition, some researchers have extracted terms from external sources like citation context and social tags [29].

2.3 Hybrid Algorithms

Both collaborative filtering (CF) and content-based filtering (CBF) recommender systems bear many problems such as cold start, sparsity, gray sheep and over specialization problem. Hybrid Recommender systems that combines CF and CBF overcomes these problems.

Bruke categorized hybrid recommender system in seven categories [10]:

- Weighted—First category is *'Weighted'* in which score of recommended item is calculated using some weighting scheme to combine results from all available techniques of recommendation present in the system, and examples include Pazzani [40].
- Switching—Second category is *'Switching'* which uses switching criteria to switch between the individual techniques, for example, hybrid algorithm presented by Billsus and Pazzani [7].
- Mixed—Third category is *'Mixed'*, and it presents recommendation from different recommender techniques in the system at different times, and examples include Cotter and smith [12].
- Feature Combination—Fourth category is *'feature combination'* in which features from various knowledge sources are derived and combined together and given as input to a single recommendation algorithm. Example includes Basu et al. [4].
- Cascade—Fifth category is *'cascade'* in which strict priority is given to recommenders. Examples include EntreeC [9].
- Feature Augmentation—Sixth category is *'feature augmentation'* in which classification or rating of an item is obtained using one recommendation technique and this information is incorporated by second technique to produce useful results. Examples include Melville [34].
- Meta-Level—Seventh category is *'meta-level'* category in which model is generated by applying one technique of recommendation and resultant model is given as input to second recommendation technique. Examples include Fab [3].

Various hybrid algorithms [28] have been proposed to solve recommendation problems. Some of these combine collaborative filtering with content-based filtering [34]. Melville applied content-based filtering to convert sparse user rating matrix into the full rating matrix. Cotter and smith [12] proposed an approach that allows CF and CBF to generate separate recommendations and then combine their prediction directly. Some combined collaborative filtering technique with knowledge-based techniques or demographic filtering [35]. Ahmed Mohammed and K. Alsalama proposed a framework is built upon the integration of content-based approach and association rules [1].

Melville in 2002 proposed hybrid algorithm which is combination of collaborative filtering and naive Bayes classifier. In this hybrid framework, naive Bayes classifier is first trained on the documents (documents describe items rated by every user), and then unrated items are replaced by the prediction from this classifier. Resulting matrix is used to find neighbors, and prediction is produced using Pearson correlation. This approach is better than individual classifiers (CF and naive Bayes) and direct combination of both. Problem in this approach is that it is not scalable [34]. Salehi and Nakhai Kamalabadi [42] compared performance of two classifiers support vector machines (SVMs) and naive Bayes in his research paper. He concluded that naive Bayes classifier is better than SVM under text enrichment via external knowledge base.

Sarwar et al. [43] presented simplest hybrid approach for movie recommendation which is linear combination of collaborative filtering and naive Bayes (CBF approach). This linear combination is not the best solution. Researchers did not test proposed system on real-world end users; they just followed Herlocker's [26] suggestions. There are many problems in this system due to which it does not produce accurate recommendations. Good et al. [24], Sarwar et al. [45] and Park et al. [39] combined CBF and CF by adding information filtering agents or Filter Bolts. For example, in 1998 Sarwar et al. used simple agent like spell checker to analyze new document in the domain of news and rate it in order to reduce sparsity. These approaches claimed that CF integrated with information filtering agents performs well as compared to simple CF in terms of accuracy. Problem with above approaches is that quality of recommendation is heavily dependent on training of individual agents, and it may not be desired in various cases, specifically given limited resources.

Badaro et al. [2] presented a hybrid approach which is weighted combination of user-based CF and item-based CF. This hybrid approach is intended to solve the sparsity problem and overcome the challenge of accuracy in recommender system by incorporating user's and item's correlation simultaneously. In 2011, [50] proposed a hybrid algorithm, which is integration of cloud-model CF and user-based CF subsystems to solve the sparsity problem. This overall system looks like black box.

Pazzani [40] provides a way to learn profile of interests of user to recommend news articles or web pages. He presented a hybrid approach where content profile of user is used to find similar users. Here problem is that if content of profile is erroneous, it will result in poor recommendation. Simon Dooms [13] introduced algorithm named as in-memory content-based recommendation and efficiently distributed it across multiple machines in distributed memory environment. Urszula KuÅelewska [31] analyzed the ratings of user on a product using clustering methods and then proposed techniques that help to identify profile of user. He has used similarity measures like cosine correlation and Euclidean distance to find similarity.

Most of researcher's work did not pay attention to preferences of users in different categories of product. Salehi et al. [42] proposed a hybrid framework to solve aforementioned problem where recommendation is generated using results of the two modules explicit (utilize explicit-based attributes)—and implicit (utilize implicit-based attributes)-based module. Most of the hybrid work is done by combining various recommendation strategies. In addition to this, customer data provided by different data mining techniques can be combined with other information to produce recommendations [33].

Sharif et al. proposed a hybrid algorithm that incorporate both co-occurrence and content similarity in order to solve the problem of limited preference information about item in case of item-based recommender system and evaluated it on Yahoo! Front Page 'Today Module User Click Log' dataset [47].

Le Hoang Son [48] in his research presented the mathematical definition of fuzzy recommender system and proposed hybrid approach user-based fuzzy CF that integrates user-based similarity calculated through rating history with the fuzzy similarities calculated on the basis of demographic data of a user in order to determine final prediction. Fang et al. [16] used position of user and its contextual information to capture preferences of user in context of indoor shopping. Carrer-Neto, Valencia-Garca, Hernndez-Alcaraz and Garca-Snchez [11] utilized social and knowledge networks in proposed hybrid recommender system for cinematographic domain. Ghazanfar and Prgel-Bennett are the first ones who proposed hybrid algorithm that makes reliable recommendations and improve performance of system under presence of gray sheep users. Furthermore, researches have proposed hybrid recommender system that integrates individual recommender systems to avoid limitations of the individual recommender systems [10,20,32]. Mustansar Ali Ghazanfar [23] proposed a framework by combining different kernals which are built from features and rating information vectors to improve recommendation performance under sparse dataset. Although kernel-based techniques are known to give excellent performance, recommender systems are challenging because of the size of the datasets, but its performance is not good under sparse dataset and cold user/item scenarios.

We compare the proposed algorithm with the individual classifiers, collaborative filtering, a naive hybrid algorithms (average of classifier and collaborative filtering) and switching hybrid approaches [19].

3 Experimental Setup

3.1 Datasets

Commonly movies dataset are used in various recommender system, so we used movie dataset from different film recommendation sites. Large datasets of movies are available, which allows to test scalability of a algorithm. As these datasets are commonly used in the literature, so it is better to use these datasets in order to evaluate proposed algorithm with state-of-the-art algorithms and set benchmark.

- *MovieLens 100K Ratings (SML)*: This dataset is represented by 'SML' in this work. It consists of 943 users, 1682 movies and 100,000 ratings. The integer scale is used for rating, i.e, from 1 (worse) to 5 (excellent). Many research projects , e.g., Sarwar et al. [43,44] have used this dataset.
- *FilmTrust (FT)*: We created this dataset by crawling the FilmTrust Web site. The dataset retrieved contains 1592 users, 1930 movies and 28,645 ratings on a floating point scale of 1 (worse) to 10 (excellent) (with a difference of 0.25). This dataset adequately captures new item and new user cold start problem adequately. This dataset contains imbalance data, i.e, some items are rated by single user, while others are rated by thousands of users. Similar is the case for the users.

  In order to evaluate the results for more sparse data, we have reduced the sparsity of these datasets by removing those users and items whose ratings in dataset are numerous and consider only those users and items for which there are small number of ratings in the dataset (Tables 1, 2).

3.2 Getting Additional Features About Movies

In addition to the rating information of movies (SML/FT), we have used content description of the movies. We have crawled additional information about movies from the Internet movie database (IMDB: http://www.imdb.com). We used jmdb, (http://www.jmdb) to match the URL's and titles provided in SML dataset with those given in IMDB. We have used tags, keywords, category, additional information, number of chapters, subtitle, sound encoding, labels, novels, book, production dates and company, sound track and language information shown in Table 3.

3.3 Feature Extraction

Information extraction is a technique that search for specific data pieces in documents containing natural language and then extract structured information from them. Structured and searchable database is constructed after information

Table 1 Benchmarking results under various scenarios for FilmTrust dataset

Cold start user scenario		MAE
Best approach	$KNNHybrid_{IB}^{NB}$	1.25
Literature approaches	*NBIBCF*	1.54
	Switching NBCF [6]	1.53
Cold Start Item Scenario		
Our best approach	$KNNHybrid_{IB}^{SVM}$	1.25
Literature approaches	*SVMIBCF*	1.83
	Switching SVMCF [6]	1.50
Sparsity scenario		
Our best approach	$KNNHybrid_{IB}^{NB}$	1.38
Literature approaches	*NBIBCF*	3.02
	Switching NBCF [6]	2.01

Table 2 Benchmarking results under various scenarios for SML dataset

Cold start user Scenario		MAE
Our best approach	$KNNHybrid_{IB}^{NB}$	1.01
Literature approaches	*NBIBCF*	1.06
	Switching NBCF [6]	1.05
Cold start item scenario		
Our best approach	$KNNHybrid_{IB}^{NB}$	0.86
Literature approaches	*NBIBCF*	1.005
		0.92
Sparsity scenario		
Our best approach	$KNNHybrid_{IB}^{SVM}$	0.74
Literature approaches	*SVMIBCF*	0.779
	Switching SVMCF [6]	0.76

Table 3 Information about movies crawled from internet movie database

Crawled information	Description of information
Keywords	Keywords for a movie (variable length)
Summary	Summary of the movie (variable length)
Tags	Tags given to a movie (variable length)
Actor	Actors/Actresses in movie
Director	Director of a movie
Producer	Producer of a movie
Editor	Editor of a movie
Production company	Company producing movie
Technical	Film negative format, color info, disk info, quality of video, etc.
Sound track	Lyrics information, singer information
Language	Movie's original language (e.g, English, Urdu, French)
Novel/book	Books have descriptions similar to those for movies, so we can obtain features such as author, year of publication and genre.
Subtitle	Subtitle of the movie
AKA-title	Movie is also known as
Release information	Country where it was released

extraction from massive textual data. In this research work, we have crawled description (textual information) of movies from Internet movie database (IMDB). After that, we created profile of item using its description.

Users' profile can be built using two main techniques. It can be built using the description of the items in which a user is interested. Another technique is to build users' profile by maintaining history of user's interaction with the system. This history can be gathered implicitly (e.g., analyzing the time spent by user on web page) and explicitly (e.g., rating provided by user). In this research work, we have used both techniques but in later technique we have used only explicit feedback. After crawling the description of movies from IMDB, we have performed preprocessing steps.

a) *Preprocessing:*
In preprocessing, strings of characters in documents are transformed into such representation which is suitable for machine learning algorithm. At first, document is converted into tokens (sequence of digits and letters) and then it is modified through following steps [49].

(a) *Removal of Html and other tags:* Tags like html are removed in this step.

(b) *Removal of stop words:* Stop word are those words which occurs frequently throughout document but have little meaning. They belong to syntactical classes like conjunctions, preposition, particles, articles, pronouns, adjectives. anomalous verbs and adverbs. To complete this task, we have customized the list stop words provided by Google (ranks.nl/resources/stopwords.html).

(c) *Perform stemming:* In this step, inflections and case information are removed from word and mapped it into the same stem, e.g., the words recommending, recommender , recommended and recommendation are all mapped to a single word root recommend. We have used Porter stemmer algorithm [25] for this task.

b) *Indexing:*
Usually vector of weighted index terms is used to represent document. Most commonly used technique for document representation is vector space model. In this technique, each document is represented by the vectors of words. Matrix known as word-by-document matrix represented by 'A' is a collection of all documents. Each entry of matrix denote the occurrence of particular word

in the document, i.e., $A = a_{w,d}$, where as a weight of a word 'w' in document 'd' is represented by $a_{w,d}$. This is typically sparse matrix as every word is not present in every document.

We assume that the total number of documents in collection is n_d, n_w being the number of words remaining in collection after preprocessing steps (stop word removal and stemming) and then document frequency '*DF*' of a particular word 'w' in the collection can be represented as DF_w and it determines the number of times particular word appears in whole collection. $TF_{w,d}$ determines the frequency of a word 'w' in a document 'd.' There are several approaches like word frequency weighting, Boolean weighting, TF-IDF weighting and entropy weighting which can be used to calculate $a_{w,d}$ [49].We have used TF-IDF approach because it is the simplest approach and widely used in the literature [38,49].

c) *Term Frequency–Inverse Document Frequency ($TF - IDF$):*

It is the popular approach used to calculate $a_{w,d}$. Weight of a word 'w' in document 'd'' is computed using frequency of that word in document ($TF(w, d)$) and inverse document frequency $IDF(w)$ (computed using equation a). When determining the relevant words of a query, $TF(w, d)$ gives equal importance to every word. This creates problem because some words have no or little discrimination power in determining relevance. For example, if we have computer industry related collection of documents, then term 'computer' exists in almost every document. Some mechanism is required to attenuate effect of such frequently occurring terms in collection of documents and inverse document frequency $IDF(w)$ is used for this purpose. The word has high IDF if it occurs in only one document.

$$\text{IDF(w)} = log \frac{n_d}{DF(w)}$$

Composite weight $a_{w,d}$ (entry of word-document matrix) is calculated using following formula:

$$a_{w,d} = TF(w, d) \times IDF(w, d)$$

The above equation shows that TF-IDF does not consider length of documents, which will create problem if documents vary in length. This problem can be eliminated by normalizing the weights:

$$a_{w,d} = \frac{TF(w, d) \times IDF(w, d)}{\sum_{j=1}^{n_w} [TF(w, d) \times IDF(w, d)]^2}$$

Indexing step produces an output of bag of words in which documents are represented as attribute value (token and corresponding weight value) pair.

d) *Dimension reduction:*

There are large number of features in attribute-value representation of a document (one dimension for every unique word in the collection of documents after stop word removal and stemming). If number of documents are huge, then machine learning algorithms cannot process data because of memory and processing power limitations. To resolve this problem, various feature selection techniques are used without considerable loss of information. For this purpose, various techniques are used which improves generalization accuracy and avoid overfitting. We have used feature selection for this purpose. In this technique, feature space is reduced by eliminating noisy words (words having small signal to noise ratio or words having no or little discrimination power). Different approaches are used in this technique. We have used document frequency (DF) thresholding and term frequency (TF) thresholding, which reduce the dimensions without losing accuracy [46].

- *DF Thresholding:* In this approach, document frequency of each and every word in training set is calculated and those words have DF less than fixed threshold are removed. The assumption behind this approach is that these words have very low discriminating power and have less or no contribution in determining category
- *TF Thresholding:* In this approach, term frequency of each and every word in document is calculated and words having TF less then fixed threshold are removed.

  We have trained text categorization approaches on the selected features to predict unknown ratings. Zhang and Iyengar [51] claimed that recommender systems and text categorization share number of characteristics. For example, each user can be considered as a document and item rated by user can be considered as word in a document. Mooney and Roy [38] proposed another approach of using content features of an item for book recommender system. Here, each item was represented as a vector of bags of words and ratings provided by users were considered as class labels. Melville et al. [34] and some other researchers have used set of movies rated by a user to build profile of a user. We have also used this approach in our work.

3.4 Evaluation Metrics

In order to evaluate the performance of recommender system, several metrics are used but due to lack of standardization it is difficult to compare results of published algorithms. Herlocker et al. [27,36] provides the overview of various metrics.

So far, most of the published work has focused on accuracy measure, which can be classified into three categories. These are rank accuracy metrics, classification accuracy metrics and predictive accuracy metrics [36]. We have evaluated our algorithms using predictive accuracy metrics. We have actual rating of an item for a particular user, and our main task is to predict rating of item for a user and then compute difference between both ratings. All records in recommendation dataset are presented as <uid,mid,r>, where 'r' is the rating of a movie 'mid' rated by user 'uid.' If 'r_{ix}' is the actual rating on object 'x' by user 'i,' and '$r\prime_{ix}$' is the rating predicted by recommender system on object 'x' for user 'i' and 'D^{test}' is test dataset, then MAE can be defined as:

$$\text{MAE} = \frac{1}{|\mathcal{D}^{test}|} \sum_{r_{i,x \in D^{test}}} |r_{ix} - r\prime_{ix}|.$$

We have selected this metric because we can benchmark our results with other state-of-the-art algorithms. We mostly use MAE measure throughout the paper, because it is defacto standard for benchmarking the recommender systems. This measure has been used by various researchers in their projects, e.g., [8,20,21,41]. The Aim of a recommender system is to minimize the MAE.

3.5 Evaluation Methodology

We have conducted fivefold cross-validation by dividing the dataset into training and testing set randomly and recorded the average results. We again divided our training set into train and test set in order to measure parameter's sensitivity. In order to learn parameter, we conducted fivefold cross-validation on train set.

4 Proposed Meta-Level Hybrid Algorithms

Accuracy and scalability are two of the most important design objectives of recommender systems. Though collaborative filtering (CF) is one of the most popular recommendation techniques used by recommender systems; however, it does not provide both accuracy and scalability. Also content-based filtering alone does not provide accurate results and suffers from various problems. When sufficient and clear information is given, the conventional algorithms demonstrate great performance, but with the addition of items and users in the rating matrix, CF algorithm gradually uncovers few weaknesses. Content-based filtering approaches consider all items rated by user in order to predict rating of a test item, so we require a hybrid algorithm that adjusts with the dynamic change of a rating matrix and considers only similar items of a test item to make useful prediction of a test item.

In this section, we first discuss background concepts related to individual classifiers. Next, we present outline of our proposed hybrid algorithm. In last section, we discuss recorded problems solved by novel hybrid algorithm.

4.1 Background

In this section, we give overview of various content-based approaches.

- *Bayesian Classifier*: The basic idea behind the Bayesian classifier is to predict class for the values of features. Examples are grouped into same class if they have common values for features. Target class is discrete which may not be binary. In case of movie recommendation, rating given to movie is considered as target class. Bayes rule is used to predict class of movie given features of that movie. The learning agent develops probabilistic model of features of movie in which features are dependent on each other and then the model is used for the prediction of classification of new examples. For a movie 'd' and class 'C'

$$\begin{aligned} CMAP &= \text{argmax}_{c \in C} P(c|d) \\ &= \text{argmax}_{c \in C} \frac{P(d|c)P(c)}{P(d)} \\ &= \text{argmax}_{c \in C} \frac{P(x_1, x_2, \ldots x_n|c)P(c)}{P(x_1, x_2, \ldots x_n)} \end{aligned}$$

Determining category of x_k by determining for each y_i class

$$P(Y = y_i|X = x_k) = \frac{P(Y = y_i)P(X = x_k|Y = y_i)}{P(X = x_i)}$$

- *Naive Bayes*: Naive Bayes classifier assigns the class label to instances which are represented as feature vector. Class labels are actually the ratings given to the movie. In naive Bayes, feature is assumed to be independent of each other, i.e., each feature contributes to probability independently. For a movie 'd' and class 'C'

$$\begin{aligned} CMAP &= \text{argmax}_{c \in C} P(c|d) \\ &= \text{argmax}_{c \in C} \frac{P(d|c)P(c)}{P(d)} \\ &= \text{argmax}_{c \in C} P(d|c)P(c) \\ &= \text{argmax}_{c \in C} P(x_1, x_2, \ldots.., x_n|c)P(c) \end{aligned}$$

where x_1 to x_n are features of movie d.

Assumptions:

– *Bag of Words Assumption:* Position of words in feature does not matter.

- *Conditional Independence:* Probability of features is independent of the given class.

$$P(x_1, x_2, \ldots.., x_n|c) = P(x_1|c) \times P(x_2|c) \ldots P(x_n|c)$$
$$CNB = \text{argmax}_{c \in C} P(c_j) P(x|c)_{x \in X}$$

- *Support Vector Machines*: The aim of this approach is to minimize the upper limit of generalization error via maximizing the margin that separates hyperplane and data. Basically SVMs learn a linear decision rule $h(x) = sign\{wx + b\}$, where 'w' is a weight vector and 'b' is threshold. It takes n examples from training set as input $S_n = ((x_1, y_1), .., (x_n, y_n)), x_i \in \Re^N, y_i \in \{-1, +1\}$. In case of linearly separable input set S_n, SVM searches for optimal margin hyperplane that separates the closest training examples with maximum Euclidian distance and solve the optimization problem [49].

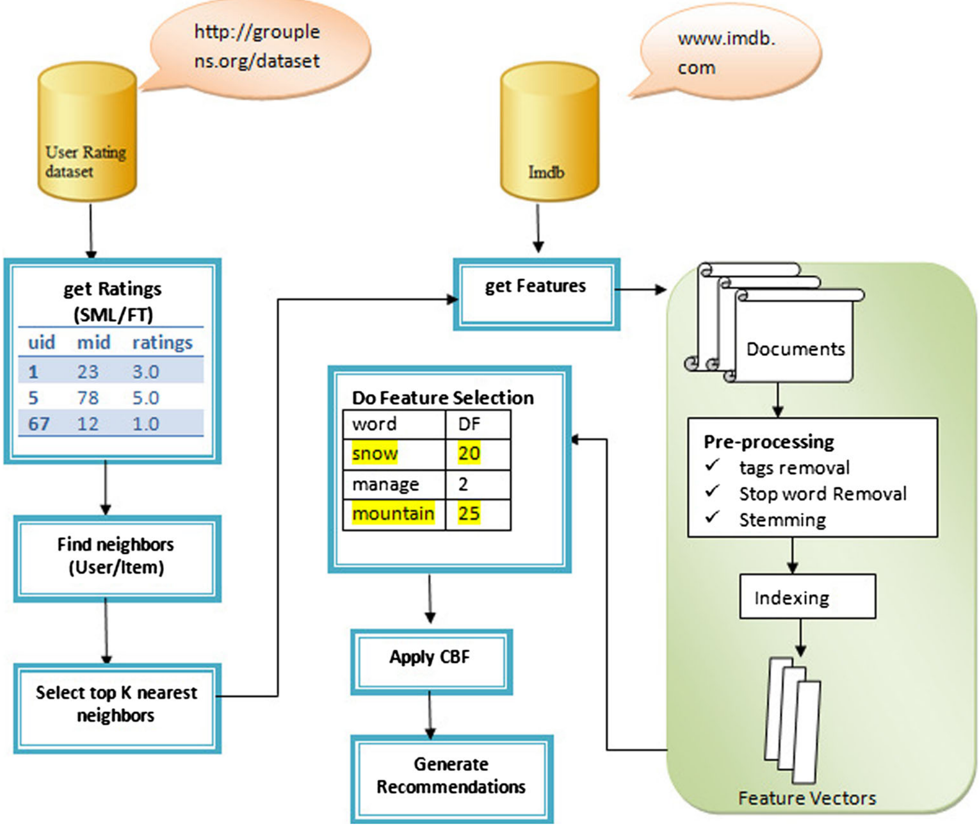
$$\begin{aligned} &\text{minimize} \quad 1/2||w||^2 \\ &\text{Constraints} \quad y_i(w.xi + b) \geq 1, i = 1, \ldots.., l \end{aligned}$$

By introducing the soft margin hyperplane, SVM constructs the hyperplane with maximum margin having some training error and a penalty term is added to minimization problem (sum of the deviations ξ)

$$\begin{aligned} &\text{minimize} \quad 1/2||w||^2 + C \sum\nolimits_{i=1}^{1} \xi_i \\ &\text{Constraints} \quad (1)\ y_i(w.xi + b) \geq 1 - \xi_i, i = 1, \ldots.., l \\ &(2)\ \xi_i > 0, \forall_i \end{aligned}$$

Here, C is weight of the penalty term (for Misclassifying training examples). For large value of C, SVM takes more time because of time taken by exhaustive search to minimize the number of examples that are misclassified. In case of multi-class classification problem, SVM treats it as a collection of various binary classification problems.

- *Decision tree*: It is a classification tree, which is used for learning a classification function that concludes dependent variable's value given the values of independent attribute. It includes discovering useful pattern in complex and large bulk of data. All specialists and theoreticians are searching continually to make process more accurate, cost-effective and efficient [6].



uid	mid	ratings
1	23	3.0
5	78	5.0
67	12	1.0

word	DF
snow	20
manage	2
mountain	25

Fig. 1 Proposed algorithm's architecture demonstrating the flow of work starting from the crawling of data followed by preprocessing, feature selection and then using meta-level hybrid algorithm (combination of collaborative and content-based approaches) to provide recommendations

4.2 Architecture of Proposed Algorithm

The architecture diagram (Fig. 1) shows general framework demonstrating whole procedure starting from use of rating datasets (SML and FT) for finding neighbors of a test/target item (item for which we want to predict rating), crawling information of these neighbors from IMDB dataset, and then performing preprocessing steps and feature selection technique leading to selected feature which at the end are used as input for content-based approaches for making prediction of a target item. These predictions are used to make useful recommendation for an active user.

4.3 Novel Hybrid Algorithm Combining Content-Based Approaches with CF

- *Use of K-nearest neighbor concept of Item-based Collaborative Filtering:*
  In this version of proposed algorithm, content-based approaches are built on neighbors of a target item. Figure 2 demonstrates the idea of finding neighbors of a target item. Here, D is set/group of all items in training set and D' is set of items rated by active user. In order to predict rating of a target item for active user, proposed algorithm first searches the K-nearest movies of a target item in the domain of D' set. Adjusted cosine similarity between target item and all items in D' set is computed using equation given below:

$$sim(i, j) = \frac{\sum_{u \in U} (R_{u,i} - R_u)(R_{u,j} - R_u)}{\sqrt{\sum_{u \in U} (R_{u,i} - R_u)^2} \sqrt{\sum_{u \in U} (R_{u,j} - R_u)^2}}$$

  where i is test item, j is one of the item from D' set, U are the users in training set who have seen both movies i nd j. $R_{u,i}$ is rating of a user $u \in U$ for item i, and R_u is average rating of user u and $R_{u,j}$ is rating of a user $u \in U$ for item j.
- *Use of K-nearest neighbor concept of User-based Collaborative Filtering:*
  In this version of proposed algorithm, content-based approaches are built on items rated by neighbors of a target user. In order to predict rating of a target item for active user, proposed algorithm first searches the K-nearest users of a target user in the domain of D' set. Adjusted cosine similarity between target user and all users in D' set is computed using equation given below:

$$sim(i, j) = \frac{\sum_{c \in I_{ij}} (R_{i,c} - R_i)(R_{j,c} - R_j)}{\sqrt{\sum_{c \in Item} (R_{i,c} - R_i)^2} \sqrt{\sum_{c \in Item} (R_{j,c} - R_j)^2}}$$

  where set of items rated by both users $'i'$ and $'j'$ is denoted by I_{ij}, R_i and R_j represent average rating for user i and j, respectively. After experiments, we found that it is not good approach as results we got are not better.
- *Building content-based approaches on the selected neighbors:*
  We have trained our content-based algorithm over the features of the neighbors of a target item (item-based version) or items rated by neighbors of target user (user-based version) and built a classifier over these selected neighbors. So the basic idea here is to train classifier over K-nearest neighbors. We have combined item-based and user-based CF with SVM, naive Bayes, Decision tree and Bayesian classifier, respectively.

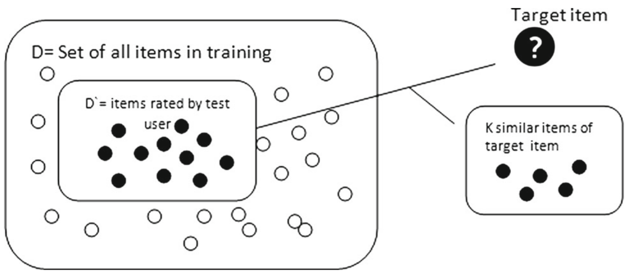

Target item
?
D= Set of all items in training
D`= items rated by test user
K similar items of target item

Fig. 2 K-nearest neighbor selection from domain of items

4.4 Naive Bayes Combined with Item-Based CF

In this novel approach, the similarity between the target item and items rated by target user is computed. The features of these items are extracted and employed for model creation. First, we get movies seen by the active user in test set.

These movies are target movies for which we have to make predictions. Then, for a target movie ‘m’-nearest neighbors in the train set are found using item-based collaborative filtering. Adjusted cosine similarity formula is used to find similarity between movies. Top K most nearest neighbors from the list are selected. Stored features of these movies from train set are obtained and dataset name ‘Train NB’ is created. Build naive Bayes model over this dataset. Now model is ready to make prediction for the target item as shown in Algorithm 1. It is shown empirically in result section that the proposed hybrid algorithm performs better than individual content-based algorithms.

Algorithm 1 is the novel algorithm that combines K-nearest neighbor concept with the naive Bayes classifier. In step 1, we loop through all the users of test set S_{test} and get the user id in step 2. In steps 3–11, we loop through movies seen by test user in test set. These are the target movies for which we need to predict rating for a test/target user. After finding the movie id in step 4, we find the similarity of a target movie with all movies seen by the target user in training set S_{train}. Similarity is computed using adjusted cosine similarity as shown in step 5. We have arranged all these items in list in descending order of similarity value so that top elements are more similar to a target item. Ids and features of top k-selected neighbors of movies are obtained. If a movie has some features (step 8), then its features are added in dataset '$NBtrain$'. This is leaning dataset on which our model is built and used it for prediction. After training the model , next step is to evaluate prediction (Step 12). In this step, we find actual rating for a movie and predicted rating. After getting the difference between two ratings, we add it to mean absolute error.

Algorithm 1 : $KNNHybrid_{IB}^{NB}$
S_{train} Training Set ;
S_{test} Test Set
M_t Target Item/Movie
k Number of nearest neighbors
Output: Mean Absolute Error Of Prediction

```
1: for ($U_t \in S_{test}$) do
2:     $U_{t\_id} \leftarrow$ test user id
3:     for ($M_t \in S_{test}$) do
4:         $M_{t\_id} \leftarrow$ test movie id
5:         $N_k \leftarrow$ get $K$-nearest neighbors of $M_{t\_id}$ in Strain using Adjusted Cosine sim.
           $sim(i, j) = \frac{\sum_{u \in U}(R_{u,i} - R_u)(R_{u,j} - R_u)}{\sqrt{\sum_{u \in U}(R_{u,i} - R_u)^2}\sqrt{\sum_{u \in U}(R_{u,j} - R_u)^2}}$
6:         Get Features and Rating of selected neighbor movies,
7:         if ($Features! = NULL$) then
8:             Create Learning dataset '$NBTrain'$ of features of selected neighbors of $Mt\_id$ once
9:         end if
10:        Build and Train Naive Bayes over all movies in '$NBTrain'$
11:    end for
12:    Evaluation:
           • Get Features and Rating of $M_{t\_id}$ from $S_{test}$
           • Create Test dataset '$NBTrain'$
           • Find Actual rating and Predicted Rating
           • Add MAE
13: end for
14: Calculate Mean Absolute error For all $U_t \in S_{test}$
```

4.5 Naive Bayes Combined with User-Based CF

In this novel hybrid Algorithm 2, the similarity between target/test user and all users in system is computed using adjusted cosine similarity formula. Then features of movies rated by K-nearest neighbors of target user are extracted and employed for model creation.

4.6 Solution to Recorded Problems

Prediction based on sparse rating matrix results in poor quality of predictions. Sparsity problem in collaborative filtering is solved by novel approach as predictions in it are based on the features not ratings. We have created scenario of cold start users and cold start item in the dataset to check performance of proposed algorithm over skewed/imbalance dataset. The experimental results claim that proposed hybrid approach performs good as compare to other approaches. Features of movies (extracted from Internet movie data base) include category, certification, aspect ratio, additional information, number of chapters, subtitle, sound encoding, labels, novels, book, production dates and company, sound track, quotes, language, etc. They are presented in machine-readable form to remove problem of limited content analysis in CBF.

Algorithm 2 : $KNNHybrid_{UB}^{NB}$
S_{train} Training Set ;
S_{test} Test Set
M_t Target Item/Movie
k Number of nearest neighbors
Output: Mean Absolute Error Of Prediction

```
for ($U_t \in S_{test}$) do.
    $U_{t\_id} \leftarrow$ test user id
    $N_k \leftarrow$ get $K$ nearest neighbors of $U_{t\_id}$ in Strain using Adjusted Cosine sim.
    $sim(i, j) = \frac{\sum_{c \in I_{ij}}(R_{i,c} - R_i)(R_{j,c} - R_j)}{\sqrt{\sum_{c \in Item}(R_{i,c} - R_i)^2}\sqrt{\sum_{c \in Item}(R_{j,c} - R_j)^2}}$
    for ($M_t \in S_{test}$) do // test movies
        $M_{t\_id} \leftarrow$ test movie id
        for ($M_train \in M_k$) do // movies rated by k neighbors of target user
            $M_{train\_id} \leftarrow$ train movie id
            Get Features and Rating of train movies
            if ($Features! = NULL$) then
                Create Learning dataset '$NBTrain'$ of features of selected neighbors once
            end if
            Build and Train Naive Bayes over all movies in '$NBTrain'$
        end for
        Evaluation:
            • Get Features and Rating of $M_{t\_id}$ from $S_{test}$
            • Create Test dataset '$NBTrain'$
            • Find Actual rating and Predicted Rating
            • Add MAE
    end for
Calculate Mean Absolute error For all $U_t \in S_{test}$
```

Table 4 Results of novel hybrid approaches in terms of MAE for SML and FilmTrust Datase.The best results are shown in bold form

Sr. no	Approaches	SML-MAE	Film Trust-MAE
Novel hybrid approaches			
1	$KNNHybrid_{IB}^{NB}$	**0.76**	**1.65**
2	$KNNHybrid_{IB}^{SVM}$	0.77	1.66
3	$KNNHybrid_{IB}^{Bayes}$	0.76	1.65
4	$KNNHybrid_{IB}^{DecisionTree}$	0.76	1.651
5	$KNNHybrid_{IB}^{NBMultinomial}$	0.77	1.65
6	$KNNHybrid_{IB}^{Perceptron}$	0.76	1.65
7	$KNNHybrid_{IB}^{BayesNetGenerator}$	0.761	1.651
8	$KNNHybrid_{IB}^{Bagging}$	0.765	1.65
9	$KNNHybrid_{IB}^{MultiBoosting}$	0.761	1.651
10	$KNNHybrid_{UB}^{SVM}$	0.88	1.92
11	$KNNHybrid_{UB}^{NB}$	0.86	1.89
12	$KNNHybrid_{UB}^{DecisionTree}$	0.85	1.83
13	$KNNHybrid_{UB}^{Bayes}$	0.86	1.89

5 Results and Discussion

Table 4 shows that results of novel item-based version of hybrid approaches are good. However, the user-based version of hybrid algorithm has poor results because in this version we are considering features of all items rated by neighbor users. In the rest of the paper, results of item-based version are discussed.

We have compared results of proposed algorithms with content-based approaches and naive hybrid approaches NBIBCF (taking average of prediction produced by item-based CF and naive Bayes) and SVMIBCF (taking average of prediction produced by item-based CF and SVM). Performance of proposed algorithm is evaluated, and best approach was choosen for each scenario (cold start and sparsity scenario). At the end, results of proposed algorithms were benchmarked by comparing it with naive and switching [19] hybrid approaches (Fig. 3).

5.1 Performance Comparison with Content-Based Approaches in Terms of MAE

Novel meta-level hybrid approach proposed in this research work considers K-nearest neighbors of target items from the domain of items rated by target user, but content-based approaches consider all items rate by the target user in order to make prediction. Results of novel hybrid approaches are compared with content-based approaches to see the difference in performance due to squeezing of user profile. Figures 4 and 5 show improvement in performance of individual classifier, when it is combined with item-based CF in terms of mean absolute error on SML and FT. It shows that novel hybrid approaches (combining CBF and item-based similarity concept) perform better as compared to individual content-based classifiers.

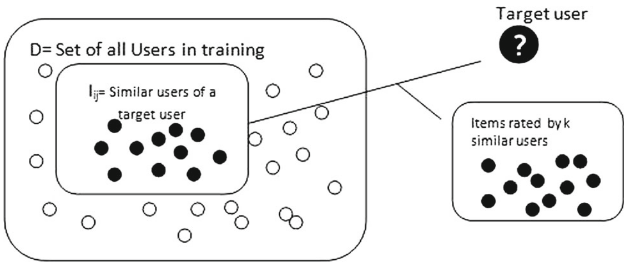

Target user
?
D= Set of all Users in training
I_{ij}= Similar users of a target user
Items rated by k similar users

Fig. 3 K-nearest neighbor selection from domain of users

The percentage decrease in MAE of $KNNHybrid_{IB}^{SCM}$ Algorithm is found to be 6.1 and 1.78% over SVM for SML and FT, respectively. The rate diminish in MAE of $KNNHybrid_{IB}^{NB}$ algorithm is discovered to be 12.04 and 3.51% over naive Bayes for SML and FT, respectively. The percentage decrease in MAE of $KNNHybrid_{IB}^{DecisionTree}$ algorithm is found to be 9.63 and 0.54% over Decision tree for SML and FT, respectively. The rate diminish in MAE of $KNNHybrid_{IB}^{Bayes}$ algorithm is discovered to be 15.18 and 10.81% over Bayesian classifier for SML and FT, respectively.

5.2 Performance Comparison with Naive Hybrid in Terms of MAE

The results of novel hybrid approaches with naive hybrid approaches were compared. Figure 6 shows performance of naive hybrid and novel hybrid approaches on FilmTrust and SML dataset. Proposed hybrid approach performs better as compared to naive hybrid approaches. The percentage decrease in MAE of novel $KNNHybrid_{IB}^{SVM}$ algorithm is found to be 0.24 and 0.77% over naive SVMCF for FT and SML, respectively. The rate diminish

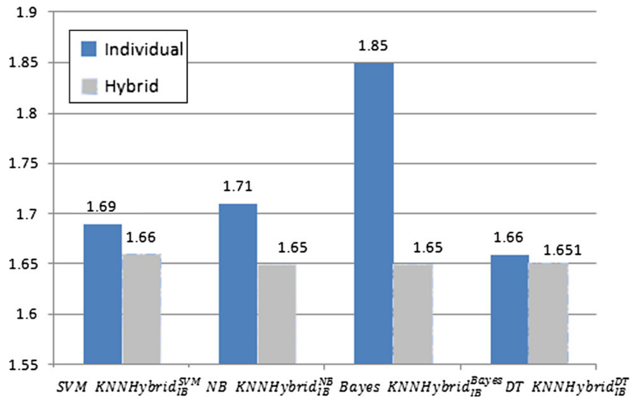


Fig. 4 Comparison of the novel hybrid algorithm with individual content-based approaches—x-axis: $Approach$, y-axis: MAE, $Dataset: FT$

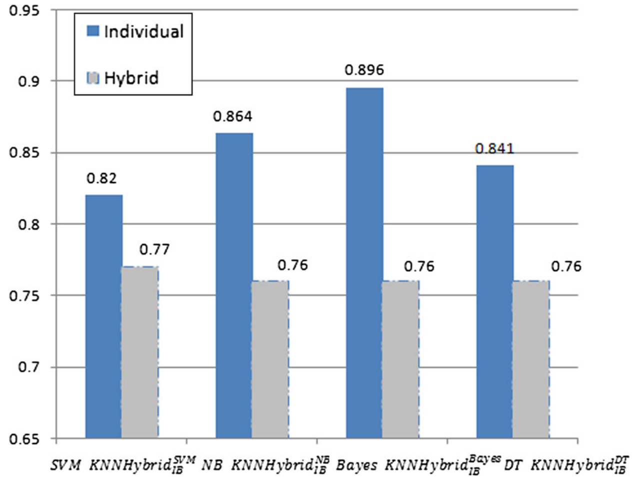


Fig. 5 Comparison of the novel hybrid algorithm with individual content-based approaches—x-$axis$: $Approach$, y-axis: MAE, $Dataset: SML$

in MAE of novel $KNNHybrid_{IB}^{NB}$ algorithm is discovered to be 1.32 and 2.56% over naive NBCF for SML and FT, respectively. The percentage decrease in MAE of novel $KNNHybrid_{IB}^{DecisionTree}$ algorithm is found to be 0.54 and 1.55% over naive DecisionTreeCF for SML and FT, respectively. The rate diminish in MAE of novel $KNNHybrid_{IB}^{Bayes}$ is discovered to be 1.32 and 2.56% over naive BayesCF for SML and FT, respectively (Fig. 7).

5.3 Performance Under Imbalance Dataset

We have evaluated the performance of novel hybrid algorithms for imbalance dataset.

- *Cold start user*: Cold start user scenario is generated by considering and evaluating those users who have rated very few movies in the dataset. Table 5 and Fig. 8 show that the novel hybrid approaches perform better

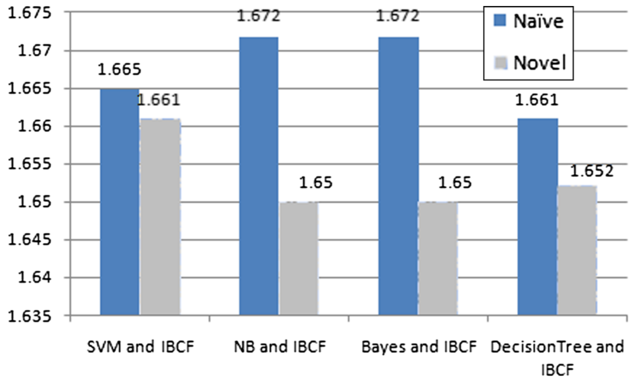


Fig. 6 Comparison of the novel hybrid algorithm with Nave Hybrid algorithms—x-$axis$: $Approach$, y-$axis$: MAE, $Dataset: FT$

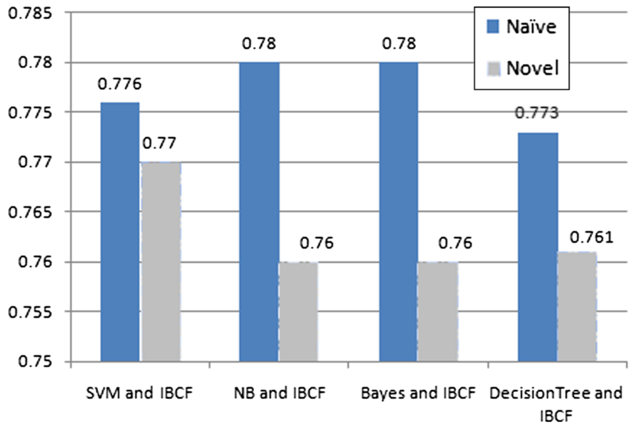


Fig. 7 Comparison of the novel hybrid algorithm with Nave Hybrid algorithms—x-$axis$: $Approach$, y-$axis$: MAE, $Dataset: SML$

Table 5 Results under cold start user scenario-FilmTrust

Sr. no	Approaches	FilmTrust-MAE
1	*NBIBCF*	1.54
2	*SVMIBCF*	1.36
3	$KNNHybrid_{IB}^{NB}$	**1.25**
4	$KNNHybrid_{IB}^{SVM}$	1.29

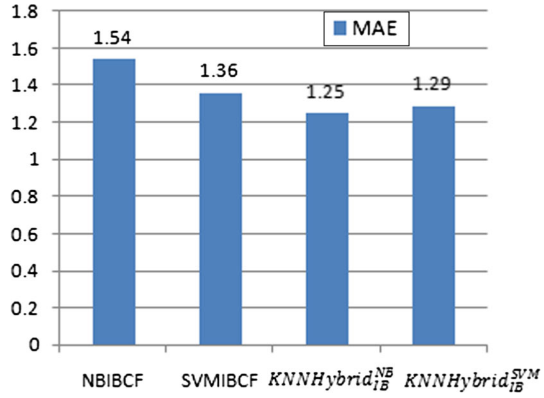


Fig. 8 Results under cold start user scenario FT—x-axis: $Approach$, y-axis: MAE, $Dataset: FT$

than naive hybrid approaches under cold start user scenario. The percentage decrease in MAE of NBIBCF over $KNNHybrid_{IB}^{NB}$ is found to be 18.83%, and SVMIBCF over $KNNHybrid_{IB}^{SVM}$ is found to be 5.15% in case of cold start user scenario in FT. The novel algorithm $KNNHybrid_{IB}^{SVM}$ performed the best under this scenario.

- *Cold start Item*: Cold start item scenario is created to evaluate novel hybrid approach. We have tested our proposed hybrid approach for cold start movies that has number of ratings less then 4, 6, 8 and 10. Table 6 shows results of novel hybrid approaches and naive hybrid approaches under cold start item scenario. Figure 9 shows results of naive hybrid approaches and novel hybrid approaches under cold start item scenario for FilmTrust dataset.

  Similarly, we have created cold start scenarios for SML dataset and found that performance of proposed algorithm is better than naive hybrid approaches under these scenarios.
- *Sparsity*: Sparse dataset is created by combining cold start users and cold start items to evaluate performance of proposed novel hybrid approaches under skewed dataset. Results, listed in Tables 7 and 8, show that the proposed novel hybrid approach performed best under skewed dataset. The percentage decrease in MAE, in the case of novel algorithm $KNNHybrid_{IB}^{NB}$ over the naive NBIBCF, is found to be 54.3 and 8.24% for the sparse FT and SML dataset, respectively. The rate diminish in MAE, in case of novel $KNNHybrid_{IB}^{SVM}$ over the naive SVMIBCF, is discovered to be 31.13 and 5.01% for sparse FT and SML dataset, respectively. The performance of $KNNHybrid_{IB}^{NB}$ is best as compared to other novel approaches presented in graph in case of sparse FT. The performance of $KNNHybrid_{IB}^{SVM}$ is best as compared to other novel approaches presented in graph for sparse SML (Figs. 10, 11).

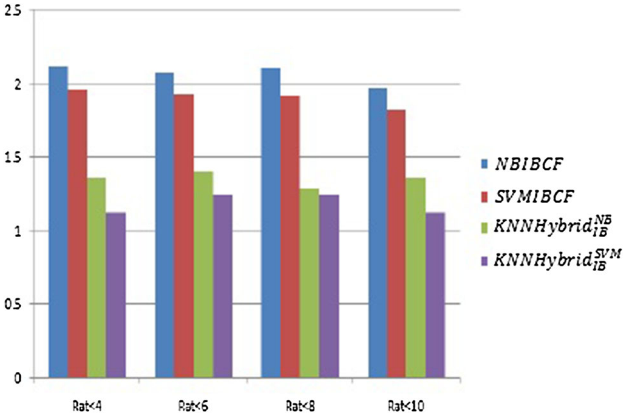


Fig. 9 Results under cold start item scenario for FT dataset—x-axis: $Rating$, y-$axis$: MAE, $Dataset$: FT

Table 7 Results of novel and naive hybrid approaches under sparsity scenario for FT dataset

Sr. no	Approaches	MAE-FT
1	*NBIBCF*	3.02
2	*SVMIBCF*	3.02
3	$KNNHybrid_{IB}^{NB}$	**1.38**
4	$KNNHybrid_{IB}^{SVM}$	2.08
5	*DecisionTreeIBCF*	1.56

Table 8 Results of novel and naive hybrid approaches under sparsity scenario for SML dataset

Sr. no	Approaches	MAE-FT
1	*NBIBCF*	0.85
2	*SVMIBCF*	0.779
3	$KNNHybrid_{IB}^{NB}$	0.78
4	$KNNHybrid_{IB}^{SVM}$	**0.74**

5.4 Benchmark Result

The purpose of this research is to provide good recommendations under cold start and sparsity scenario. We have combined various content-based filtering approaches with collaborative filtering approaches and chosen our best approach under these scenarios. In order to benchmark results, we have compared our best approaches, i.e., $KNNHybrid_{IB}^{NB}$ and $KNNHybrid_{IB}^{SVM}$ meta-level hybrid algorithms with the results of previous algorithms. The experimental results for

Table 6 Results under cold start item scenario for FT dataset

S. no	Approaches	FilmTrust-MAE			
		Ratings<4	**Ratings<6**	**Ratings<8**	**Ratings<10**
1	*NBIBCF*	2.11	2.08	2.10	1.97
2	*SVMIBCF*	1.96	1.919	1.914	1.83
3	$KNNHybrid_{IB}^{NB}$	1.363	1.4	1.29	1.16
4	$KNNHybrid_{IB}^{SVM}$	**1.25**	**1.25**	**1.25**	**1.125**
5	$KNNHybrid_{IB}^{DecisionTree}$	1.47	1.44	1.43	1.37

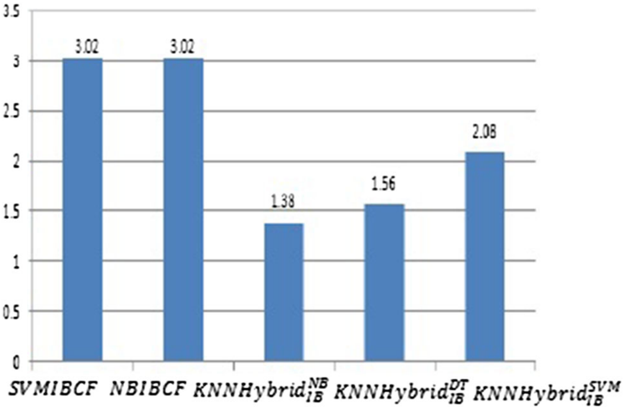


Fig. 10 Comparison of the novel and naive hybrid approaches under sparsity scenario—*x-axis*: $Approach$, *y-axis*: MAE, $Dataset: FT$

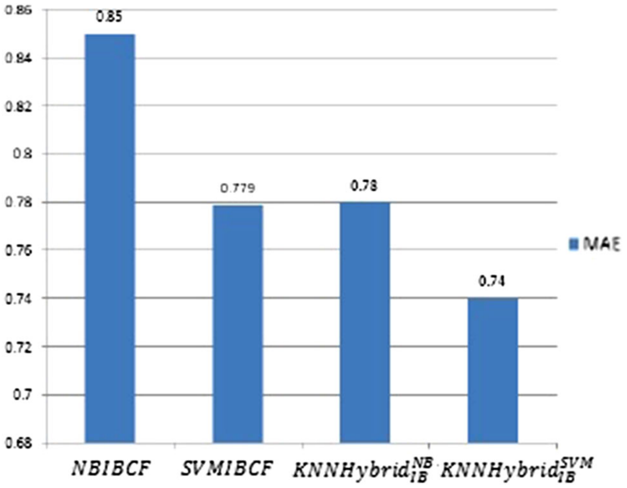


Fig. 11 Comparison of the novel and naive hybrid approaches under sparsity scenario—*x-axis*: $Approach$, *y-axis*: MAE, $Dataset: SML$

FilmTrust and SML, respectively, are listed in the Tables 1 and 2. The results claim that proposed hybrid approach performs good as compare to other approaches.

6 Conclusion and Future Work

The aim of this research work is to propose novel hybrid approach that is practical and can be used effectively to make recommendations under various scenarios. Although the current state-of-the-art algorithms, especially the ones which are proposed in Netflix competition are quite accurate but they have various drawbacks, for instance, they get good accuracy for specific dataset having particular characteristics. Moreover, these algorithms ignore design objectives such as cold start, imbalance and sparse dataset and hence they become impractical for applications of the real-world recommender system.

We propose meta-level hybrid recommendation algorithms by combining item-based collaborative filtering with content-based filtering and build content-based filtering model on the content of K-nearest neighbors of items. We have shown empirically that our novel hybrid approach perform better than others and also produce robust results for imbalance dataset and under cold start scenarios.

As future work, we would like to use various other feature selection approaches like principal component analysis that can reduce more features and might increase performance of classifiers. Moreover techniques such as singular value decomposition can be used to decompose user—item rating matrix to increase the performance.

Appendix 1: Learning the Optimal System Parameters

We have conducted various experiments to find out the effect of variation of different parameters on the prediction quality of proposed algorithm and determine optimal values of these parameters. We have described tuning of some important parameters over SML and FilmTrust dataset (Figs. 10, 11).

Appendix 1.1: Optimal Number of Neighbors of Item K

The neighborhood size was changed from 5 to 100 to find optimal number of neighbors (see the results on 5 to 50 neighbors with difference of 5, then see on 80 and 100) for FT and SML.

In Fig. 12, it is shown that decrease in mean absolute error with increase in neighborhood size for item-based CF, SVM, naive Bayes, Bayesian classifier and Decision tree. This is opposite to the item-based approach proposed by Sarwar et al. [44] in which MAE start increasing with the increase in size of neighbors and it is minimum for the small number of neighbors like less than 10 neighbors for SML dataset [44]. This is because he has not used any worthy weighting scheme and used weighted sum for the prediction generation formula. We are using adjusted weighted sum for prediction generation formula. MAE reached to minimum when number of neighbors (k) are 25 in case of SML. Then, it stays constant with the increase in neighbors.

Hybrid item-based approach proposed in this work is training content-based classifier like naive Bayes, SVM, Bayesian classifier, Decision tree over K-selected neighbors. So this parameter is very important. We have changed the neighborhood size from 5 to 100 and see its effect on the MAE. Figure 12 shows results of novel approaches $KNNHybrid_{IB}^{NB}$, $KNNHybrid_{IB}^{Bayes}$ and $KNNHybrid_{IB}^{DecisionTree}$ with the variation of number of

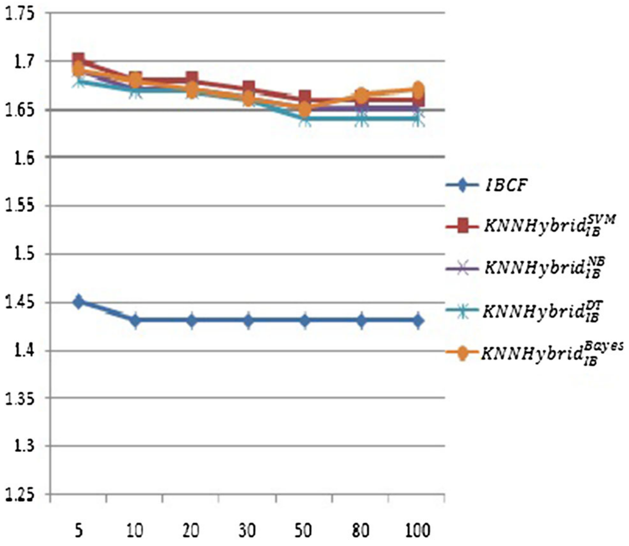


Fig. 12 Finding the optimal number of neighbors—*x*-*axis*: $Neighbors$, *y*-*axis*: MAE, $Dataset: FT$

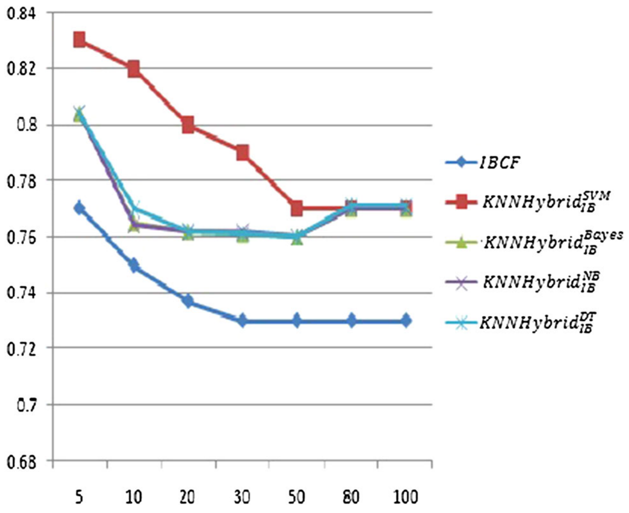


Fig. 13 Finding the optimal number of neighbors— *x*-*axis*: $Neighbors$, *y*-*axis*: MAE, $Dataset: SML$

neighbors (k) in MovieLens dataset. It shows that with increase in number of neighbors (from $k = 5$ to $k = 50$) MAE decreases, and then it increases little (at $k = 80$) and after that remains constant. These approaches show good results for $k = 50$. It shows $KNNHybrid_{IB}^{SVM}$ results with the variation of number of neighbors (k) in MovieLens dataset. Here, MAE decreases with increase in number of neighbors (from k=5 to k=30), and it is lowest at $k = 50$ and then it becomes constant (from $k = 50$ to $k = 100$). So for $K = 50$ it has shown best results ($MAE = 0.77$).

We have done same experiments using FT dataset. Here, optimal number of neighbors for item-based CF is 10 and for novel hybrid approaches is 50. Results of item-based CF, naive Bayes, SVM, Bayesian classifier and Decision tree are shown in Fig. 13.

Appendix 1.2: Optimal Value of TF and DF for Classifiers

We have changed the term frequency (it defines frequency of a word occurring in single documents). Words having TF more than some fixed threshold value are considered as features only. We changed the value of TF from 1 to 5 and analyzed the respective results of classifiers. Figure 14 shows results in the case of naive Bayes classifier. It shows that for TF=1 the error has lowest value. Similarly experiment was done on FilmTrust dataset, and the optimal value of DF was selected (3 in for Naive Bayes).

Document frequency (DF) defines frequency of a word occurring in a collection of documents. Words having DF less than fixed value are ignored. We changed the document

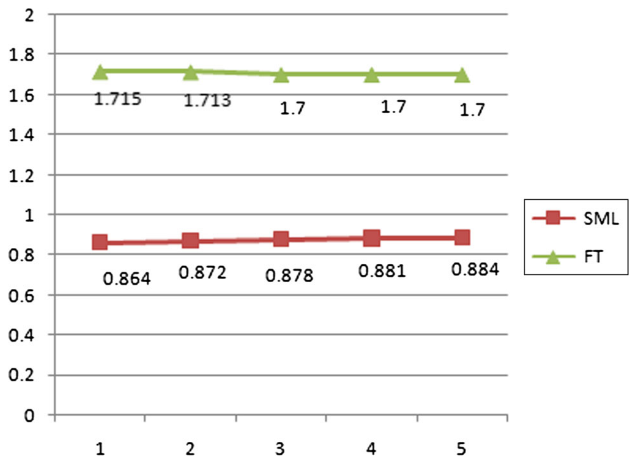


Fig. 14 Finding the optimal TF for naive Bayes— *x*-axis: TF, $y-axis$: MAE, $Dataset: FT$

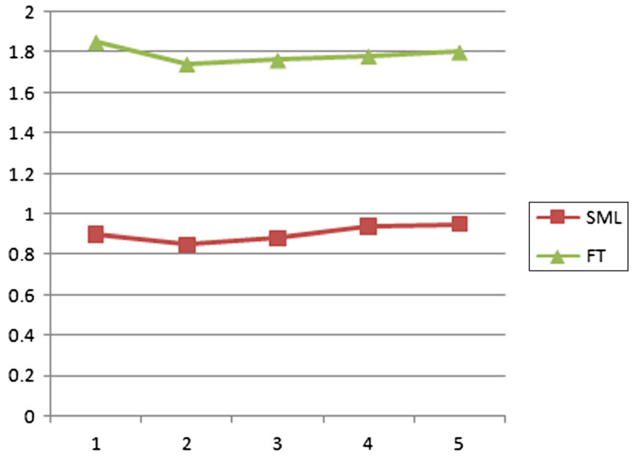


Fig. 15 Finding the optimal DF for naive Bayes - *x*-axis: DF, *y*-axis: MAE, $Dataset: FT$

frequency from 5 to 40 to analyze the results of different classifiers. Figure 15 shows the results in the case of naive Bayes classifier on SML and FilmTrust dataset. For $DF = 10$, the mean absolute error was found to be lowest in both cases.

Appendix 1.3: Optimal Value of Perimeter C for the SVM Classifier

Perimeter C is the cost parameter which is used to control trade-off between permitted training errors and rigid margins. It permits some misclassifications by constructing soft margins. A more accurate model can be created by increasing the value of parameter C but it may be over fitted. Similarly model may be under fit if value of C is very small. We chose $C = 10$ to avoid any under fitting and over fitting [34].

References

1. Alsalama, A.: A hybrid recommendation system based on association rules. Masters Thesis and Specialist Project Faculty of the Department of Computer Science, Western Kentucky University (2015)
2. Badaro, G.; Hajj, H.; El-Hajj, W.; Nachman, L.: A hybrid approach with collaborative filtering for recommender systems. In: Wireless Communications and Mobile Computing Conference (IWCMC), 2013 9th International (pp. 349–354). IEEE (2013)
3. Balabanović, M.; Shoham, Y.: Fab: content-based, collaborative recommendation. Commun. ACM **40**, 66–72 (1997)
4. Basu, C.; Hirsh, H.; Cohen, W. et al.: Recommendation as classification: using social and content-based information in recommendation. In: AAAI/IAAI (pp. 714–720) (1998)
5. Bennett, J.; Lanning, S.: The netflix prize. In: Proceedings of KDD Cup and Workshop (p. 35). vol. 2007 (2007)
6. Bhargava, N.; Sharma, G.; Bhargava, R.; Mathuria, M.: Decision tree analysis on j48 algorithm for data mining. In: Proceedings of International Journal of Advanced Research in Computer Science and Software Engineering, **3** (2013)
7. Billsus, D.; Pazzani, M.J.: User modeling for adaptive news access. User Model. User-Adapt. Interact. **10**, 147–180 (2000)
8. Breese, J.S.; Heckerman, D.; Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence (pp. 43–52). Morgan Kaufmann Publishers Inc. (1998)
9. Burke, R.: Hybrid recommender systems: survey and experiments. User Model. User-Adapt. Interact. **12**, 331–370 (2002)
10. Burke, R.: Hybrid web recommender systems. In: The Adaptive Web (pp. 377–408). Springer, Berlin (2007)
11. Carrer-Neto, W.; Hernández-Alcaraz, M.L.; Valencia-García, R.; García-Sánchez, F.: Social knowledge-based recommender system: application to the movies domain. Expert Syst. Appl. **39**, 10990–11000 (2012)
12. Cotter, P., Smith, B.: Ptv: Intelligent personalised tv guides. In: AAAI/IAAI (pp. 957–964) (2000)
13. Dooms, S.; Audenaert, P.; Fostier, J.; De Pessemier, T.; Martens, L.: In-memory, distributed content-based recommender system. J. Intell. Inf. Syst. **42**, 645–669 (2014)
14. Ekstrand, M.D.; Kannan, P.; Stemper, J.A.; Butler, J.T.; Konstan, J.A.; Riedl, J.T.: Automatically building research reading lists. In: Proceedings of the Fourth ACM Conference on Recommender Systems (pp. 159–166). ACM (2010)
15. Ekstrand, M.D.; Riedl, J.T.; Konstan, J.A.: Collaborative filtering recommender systems. Found. Trends Hum.-Comput. Interact. **4**, 81–173 (2011)
16. Fang, B.; Liao, S.; Xu, K.; Cheng, H.; Zhu, C.; Chen, H.: A novel mobile recommender system for indoor shopping. Expert Syst. Appl. **39**, 11992–12000 (2012)
17. Ferrara, F.; Pudota, N.; Tasso, C.: A keyphrase-based paper recommender system. In: Digital Libraries and Archives (pp. 14–25). Springer, Berlin (2011)
18. Ghazanfar, M.; Prugel-Bennett, A.: Building switching hybrid recommender system using machine learning classifiers and collaborative filtering. IAENG Int. J. Comput. Sci. **37** (2010)
19. Ghazanfar, M.A.: Robust, scalable, and practical algorithms for recommender systems. Ph.D. thesis University of Southampton (2012)
20. Ghazanfar, M.A.; Prugel-Bennett, A.: A scalable, accurate hybrid recommender system. In: Knowledge Discovery and Data Mining, 2010. WKDD'10. Third International Conference on (pp. 94–98). IEEE (2010)
21. Ghazanfar, M.A.; Prügel-Bennett, A.: The advantage of careful imputation sources in sparse data-environment of recommender systems: generating improved SVD-based recommendations. Informatica **13**, 61–92 (2013)
22. Ghazanfar, M.A.; Prügel-Bennett, A.: Leveraging clustering approaches to solve the gray-sheep users problem in recommender systems. Expert Syst. Appl. **41**, 3261–3275 (2014)
23. Ghazanfar, M.A.; Prügel-Bennett, A.; Szedmak, S.: Kernel-mapping recommender system algorithms. Inf. Sci. **208**, 81–104 (2012)
24. Good, N.; Schafer, J.B.; Konstan, J.A.; Borchers, A.; Sarwar, B.; Herlocker, J.; Riedl, J.: Combining collaborative filtering with personal agents for better recommendations. In: AAAI/IAAI (pp. 439–446) (1999)
25. Grasso, A.; Convertino, G.: Collective intelligence in organizations: tools and studies. In: Computer Supported Cooperative Work (CSCW), (pp. 1–13) (2012)
26. Herlocker, J.L.; Konstan, J.A.; Riedl, J.: Explaining collaborative filtering recommendations. In: Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work (pp. 241–250). ACM (2000)
27. Herlocker, J.L.; Konstan, J.A.; Terveen, L.G.; Riedl, J.T.: Evaluating collaborative filtering recommender systems. ACM Trans. Inf. Syst. (TOIS) **22**, 5–53 (2004)
28. Isinkaye, F.; Folajimi, Y.; Ojokoh, B.: Recommendation systems: principles, methods and evaluation. Egypt. Inform. J. **16**, 261–273 (2015)
29. Jack, K.: Mendeley: recommendation systems for academic literature. Presentation at Technical University of Graz (TUG) (2012)
30. Jiang, Y.; Jia, A.; Feng, Y.; Zhao, D.: Recommending academic papers via users' reading purposes. In: Proceedings of the Sixth ACM Conference on Recommender Systems (pp. 241–244). ACM (2012)
31. Kużelewska, U.: Advantages of information granulation in clustering algorithms. In: Agents and Artificial Intelligence (pp. 131–145). Springer, Berlin (2011)
32. Lucas, J.P.; Luz, N.; Moreno, M.N.; Anacleto, R.; Figueiredo, A.A.; Martins, C.: A hybrid recommendation approach for a tourism system. Expert Syst. Appl. **40**, 3532–3550 (2013)
33. Mathew, S.K.: Adoption of business intelligence systems in indian fashion retail. Int. J. Bus. Inf. Syst. **9**, 261–277 (2012)
34. Melville, P.; Mooney, R.J.; Nagarajan, R.: Content-boosted collaborative filtering (2009)

35. Middleton, S.E.; Shadbolt, N.R.; De Roure, D.C.: Ontological user profiling in recommender systems. ACM Trans. Inf. Syst. (TOIS) **22**, 54–88 (2004)
36. Mobasher, B.: Recommender systems. KI **21**, 41–43 (2007)
37. Mohanraj, V.; Chandrasekaran, M.; Senthilkumar, J.; Arumugam, S.; Suresh, Y.: Ontology driven bee's foraging approach based self adaptive online recommendation system. J. Syst. Softw. **85**, 2439–2450 (2012)
38. Mooney, R.J.; Roy, L.: Content-based book recommending using learning for text categorization. In: Proceedings of the Fifth ACM Conference on Digital Libraries (pp. 195–204). ACM (2000)
39. Park, S.-T., Pennock, D., Madani, O., Good, N., DeCoste, D.: Naïve filterbots for robust cold-start recommendations. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 699–705). ACM (2006)
40. Pazzani, M.: A framework for collaborative, content-based and demographic filtering. Department of Information and Computer Science. University of California, Irvine. Irvine, CA, 92697 (1999)
41. Resnick, P.; Iacovou, N.; Suchak, M.; Bergstrom, P.; Riedl, J.: Grouplens: an open architecture for collaborative filtering of netnews. In: Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work (pp. 175–186). ACM (1994)
42. Salehi, M.; Nakhai Kamalabadi, I.: A hybrid recommendation approach based on attributes of products using genetic algorithm and naive bayes classifier. Int. J. Bus. Inf. Syst. **13**, 381–399 (2013)
43. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Analysis of recommendation algorithms for e-commerce. In: Proceedings of the 2nd ACM Conference on Electronic Commerce (pp. 158–167). ACM (2000)
44. Sarwar, B.; Karypis, G.; Konstan, J.; Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th International Conference on World Wide Web (pp. 285–295). ACM (2001)
45. Sarwar, B.M.; Konstan, J.A.; Borchers, A.; Herlocker, J.; Miller, B.; Riedl, J.: Using filtering agents to improve prediction quality in the grouplens research collaborative filtering system. In: Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work (pp. 345–354). ACM (1998)
46. Sebastiani, F.: Machine learning in automated text categorization. ACM Comput. Surv. (CSUR) **34**, 1–47 (2002)
47. Sharif, M.A., Raghavan, V.V.: A large-scale, hybrid approach for recommending pages based on previous user click pattern and content. In: Foundations of Intelligent Systems (pp. 103–112). Springer, Berlin (2014)
48. Son, L.H.: HU-FCF: a hybrid user-based fuzzy collaborative filtering method in recommender systems. Expert Syst. Appl.: Int. J. **41**, 6861–6870 (2014)
49. Tsai, C.-F.; Hsu, Y.-F.; Lin, C.-Y.; Lin, W.-Y.: Intrusion detection by machine learning: a review. Expert Syst. Appl. **36**, 11994–12000 (2009)
50. Zarrinkalam, F.; Kahani, M.: Semcir: a citation recommendation system based on a novel semantic distance measure. Program **47**, 92–112 (2013)
51. Zhang, T.; Iyengar, V.S.: Recommender systems using linear classifiers. J. Mach. Learn. Res. **2**, 313–334 (2002)