# Efficient Mining and Recommendation of Sparse Data through Collaborative Filtering Technique in Medical Transcriptions

[#1] P. Hema, [#2] Mr. N. Sowri Raja Pillai
[#1]*PG Scholar, Department of Computer Science*
*Manakula Vinayagar Institute of Technology*
*Pondicherry, India*
[#2]*Assistant Professor, Department of Computer Science,*
*Manakula Vinayagar Institute of Technology*
*Pondicherry, India.*
[#1] phemacse@gmail.com
[#2] sowrirajacse@gmail.com

*Abstract*—**The Recommendation technique plays a major role in today's real time scenarios. Recent researchers focus on data mining based on the difficulties of recommendation techniques associated with cluster of data. A new methodology for recommendation technique is proposed in this paper. It is related with the information of user's current selection and previous information of the specific user or group of users. At last, the concluding recommendation is made based on weighing the features of the user's history. In the proposed system, Medical Record datasets is taken as an input and based on the user's selection and Disease type, the prediction is done. The operation is implemented using Google App Engine, a cloud platform.**

*Keywords —**dynamic recommendation, dynamic features, multiple phases of interest.***

## I. Introduction

Collaborative filtering is one most successful approach to recommendation reported in the literature. It automates the "Word of Mouth" recommendation by suggesting products liked by other consumers who showed similar preference patterns as the target consumer. A serious limitation of the collaborative filtering approach is the sparsity problem, referring to the situation where insufficient historical transactions are available for inferring reliable consumer similarities.

The medical transactional data is represented in order to recommend the concerned medicines and disease type to the doctors and patients. Under this consumer-product graph, the proposed method explores global graph structure to facilitate collaborative filtering under sparse data. A link analysis recommendation algorithm is developed based on the similar ideas implemented in Web graph analysis algorithms. The proposed algorithm was tested using a sample datasets that is currently available. The entire operation is implemented using Google App Engine, a cloud platform. The Login module is implemented using Google OAuth.

### 1) Recommendation Systems

Recommendation as a social process plays an important role in many applications for consumers, because it is overly expensive for every consumer to learn about all possible alternatives independently. Depending on the specific application setting, a consumer might be a buyer (e.g., in online shopping), an information seeker (e.g., in information retrieval), or an organization searching for certain expertise. Recommender systems have been developed to automate the recommendation process. Examples of research prototypes of recommender systems are: PHOAKS , Syskills and Webert, Fab ,and GroupLens.
These systems recommend various types of Web resources, online news, movies, among others, to potentially interested parties. Large-scale commercial applications of the recommender systems can be found at many ecommerce sites, such as Amazon, CD Now, Drugstore, and Movie Finder. These commercial systems recommend products to potential consumers based on previous transactions and feedback. They are becoming part of the standard e-business technology that can enhance e-commerce sales by converting browsers to buyers, increasing cross-selling, and building customer loyalty .One of the most commonly-used and successful recommendation approaches is the collaborative filtering approach. When predicting the potential interests of a given consumer, such an approach first identifies a set of similar consumers based on past transaction and product feedback information and then makes a prediction based on the observed behavior of these similar consumers. Despite its wide spread adoption, collaborative filtering suffers from several major limitations including sparsity, system scalability, and synonymy.

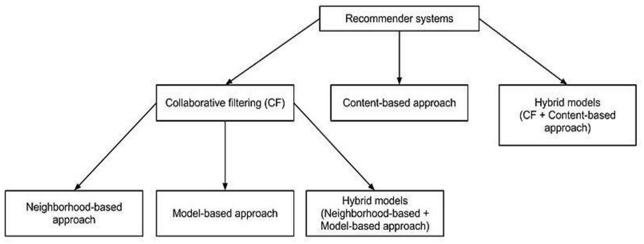
Recommender systems
Collaborative filtering (CF)
Content-based approach
Hybrid models (CF + Content-based approach)
Neighborhood-based approach
Model-based approach
Hybrid models (Neighborhood-based + Model-based approach)

Figure 1.1 System Structure

The sparsity problem is focused here, which refers to the lack of prior transactional and feedback data that makes it difficult and unreliable to predict which consumers are similar to a given consumer. For instance, the recommender systems used by online bookstores use past purchasing history to group consumers and then make recommendations to an individual consumer based on what the other consumers in the same group have purchased. When such systems have access only to a small number of past transaction records (relative to the total numbers of the books and consumers), however, determining which consumers are similar to each other and what their interests are becomes fundamentally difficult.

The previous studies have proposed to study the collaborative filtering algorithms in bipartite graphs. In such graphs, one set of nodes represents products, services, and information items for potential consumption. The other set represents consumers or users. The transactions and feedback are modeled as links connecting nodes between these two sets. The central research hypothesis is that the proposed link analysis recommendation algorithm is able to extract useful link structure information from the consumer-product graph and facilitates more effective recommendation with sparse transactional data.

## II BACKGROUND

Since the importance of internet increases day to day, the information binded to the web also increases. It is essential to deliver the content much faster and appropriate to the users. A Personalized recommendation is needed to deliver the content in a desirable way to improve customer satisfaction and retention.

In the Existing system, the recommendation is partially made based on the prediction, and hence that result proven to be less accurate and thus it could not be taken in to the real time consideration. In the previous approaches, Movie rating is taken as an example scenario, and based on the rating awarded by the users, the prediction is done.

## III PROBLEM DEFINITION

The involved ratings can reflect similar users' preferences and provide useful information for recommendation. Correspondingly, in order to enable the algorithm to catch up with the changing of signals quickly and to be updated conveniently, a set of dynamic features are proposed based on time series analysis (TSA) technique, and relevant ratings in each phase of interest are added up by applying TSA to describe users' preferences and items' reputations. Then a personalized recommendation algorithm by adaptively weighting the features according to the amount of utilized rating data. The experimental results show that the proposed algorithm is effective with dynamic data and significantly outperforms previous algorithms.

In most cases, the drifting of users' preferences or items' reputations is not too rapid, which makes it possible to describe temporal state of them by using some features. Firstly a way to make use of profiles to extend the co-rating relation was introduced, and then a set of dynamic features to reflect users' preferences or items' reputations in multiple phases of interest is proposed, and after that an adaptive algorithm for dynamic personalized recommendation.

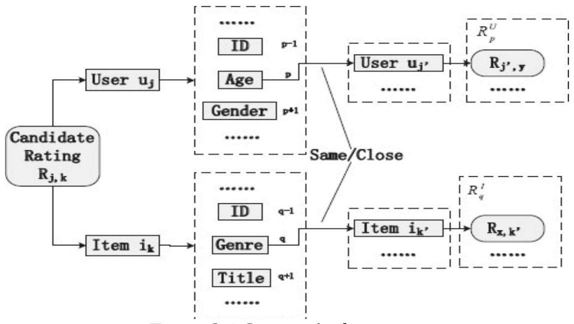

Figure 3.1 System Architecture

*a) Relation Mining of Rating Data*

For the sparsity of recommendation data, the main difficulty of capturing users' dynamic preferences is the lack of useful information, which may come from three sources – user profiles, item profiles and historical rating records. Traditional algorithms heavily rely on the co-rate relation, which is rare when the data is sparse. Useful ratings are discovered using the co-rate relation, which is simple, intuitional and physically significant when we go one or two steps along, but it strongly limits the amount of data used in each prediction.

*b) Dynamic Feature Extraction*

Users' preferences or items' reputations are drifting, thus deal must be done with the dynamic nature of data to enhance the precision of recommendation algorithms, and recent ratings and remote ratings should have different weights in the prediction. Three kinds of methods were proposed in concept drift to deal with the drifting problem as *instance selection*, *time-window* and *ensemble learning*. These methods help to make progress in precision of dynamic recommendation, but they also have their weaknesses: decay functions cannot precisely describe the evolution of user preferences and only isolating transient noise cannot catch up with the change in data. So we propose a set of dynamic features to describe users' multi-phase preferences in consideration of computation, flexibility and accuracy. It is impossible to learn weights of all ratings for each user, but it is possible to learn the general weights of ratings in the user's different phases of interest if the phases include ranges of time that are long enough.

*c) Adaptive Weighting Algorithm*

As features like *feas, d* ($s$ = 1, 2, ..., $d$ = 1, 2, ...) gained by applying *Multiple Phase Division* are all normalized rating values, in other words, as content of user and item profiles have been quantified in the feature extraction, it is convenient for us to organize them for accurate rating estimation by adaptive weighting. Sizes of the relevant subsets are also recorded in MPD and could reflect data density. These features are incorporated for recommendation with a linear model since they are homogeneous and it is efficient to learn their weights.

In most dynamic scenarios, there are mainly two issues that prevent accurate prediction of ratings – the sparsity and the dynamic nature. Since a user could only rate a very small proportion of all items, the $U \times I$ rating matrix is quite sparse and the amount of information for estimating a candidate rating is far from enough. While latent factor models involve most ratings to capture the general taste of users, they still have difficulties in catching up with the drifting signal in dynamic recommendation because of sparsity, and it is hard to physically explain the reason of the involving.

The dynamic nature decides that users' preferences may drift over time in dynamic recommendation, resulting in different taste to the items in different phases of interest, but it is not well studied in previous studies. So hence an alternative recommendation technique is needed to improve the prediction and make it accurate. In the proposed system, Medical Record datasets is taken as an input and based on the user's selection and Disease type, the prediction is done. The newly proposed system also uses the collaborative filtering technique. But in an alternative way, in such a way that it increases the probability of correctness in the prediction

## IV. Scheme Description

A new recommendation technique is implemented in the proposed system. Unlike the previous approach, each and every user interaction is taken in to consideration in order to make the accurate prediction.

Before this method, the user could only rate a small proportion of the all items. So the crisis arises at this point in order to make recommendation based on the small data that is currently in hand. A complete medical transactions among the patients and doctors is taken in to consideration. Hence the recommendation is made to the doctor based on different aspects that is being currently mined from the cluster of data.

Eg : As the user searches for any disease and related medicines, the history is maintained in a separate table. The factors taken in to consideration are Age, Gender, Location, Search results, Search disease, Searched Medicines. So all these factors are made as a separate datasets and it is taken as an input. So if a new user comes, the data is fetched from these histories and it is recommended to the doctors. Additionally Google App Engine (OAuth) is used for Login Authentication.

*1) Authentication (Google Oauth)*

A user is permitted to log in only if the authorization is done. So, in this module, the authentication process is carried out. Instead of the normal sign up process that is carried out in all applications, Google OAuth (Open Authentication.) is used in this Module. This saves the sign up time, and unwanted fake entries to the applications. So all the user credentials are obtained from google. Google APIs use the OAuth 2.0 protocol for authentication and authorization.

Google supports common OAuth 2.0 scenarios such as those for web server, installed, and client-side applications. OAuth 2.0 is a relatively simple protocol. The application should be registered with Google. Then the application requests an access token from the Google Authorization Server, extracts a token from the response, and sends the token to the Google API that you want to access.

After an application obtains an access token, it sends the token to a Google API in an HTTP authorization header. It is possible to send tokens as URI query-string parameters, but we don't recommend it, because URI parameters can end up in log files that are not completely secure. Also, it is good REST practice to avoid creating unnecessary URI parameter names. Access tokens are valid only for the set of operations and resources described in the scope of the token request. For example, if an access token is issued for the Google+ API, it does not grant access to the Google Contacts API. You can, however, send that access token to the Google+ API multiple times for similar operations.

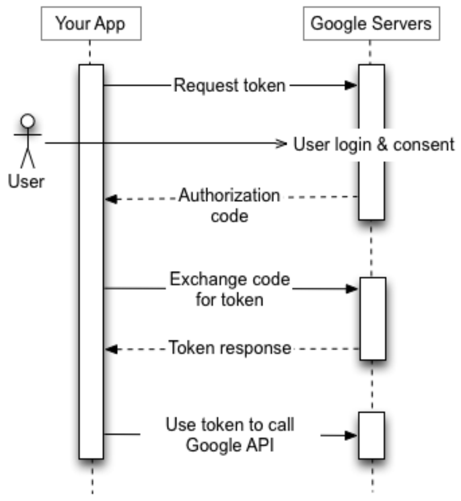

Figure 4.1 Oauth Design

*2) User History Acquisition*

Once the user is authenticated and logged in, the user is requested to enter some more necessary information. That is considered to be vital part in the project, since the data sets is formed based on the information acquired in this phase.

All the medical related information is acquired and maintained on a separate table. This table is referenced for the recommendation process. The Data store concept is used in order to store the information to the table. A data store is nothing but the cloud database, that is provided by the software giant Google.

App Engine Data store is a schema less object data store providing robust, scalable storage for your web application, with the following features:

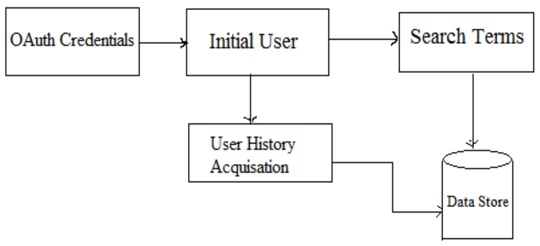

Figure 4.2 Acquisition Design

- No planned downtime
- Atomic transactions
- High availability of reads and writes
- Strong consistency for reads and ancestor queries
- Eventual consistency for all other queries

The Data store holds data objects known as *entities.* An entity has one or more *properties*, named values of one of several supported data types: for instance, a property can be a string, an integer, or a reference to another entity.

Each entity is identified by its *kind*, which categorizes the entity for the purpose of queries, and a *key* that uniquely identifies it within its kind. The Data store can execute multiple operations in a single *transaction.*

By definition, a transaction cannot succeed unless every one of its operations succeeds; if any of the operations fails, the transaction is automatically rolled back. This is especially useful for distributed web applications, where multiple users may be accessing or manipulating the same data at the same time. So once the data is gathered from the users, it is stored in data store as mentioned above.

*3) Collaborative Filtering*

Collaborative filtering is the concept of recommending technique. It is the process of filtering for information or patterns using techniques involving collaboration among multiple agents, viewpoints, data sources, etc. In narrower sense, collaborative filtering is a method of making automatic predictions (filtering) about the interests of a user by collecting preferences or taste information from many users (collaborating).

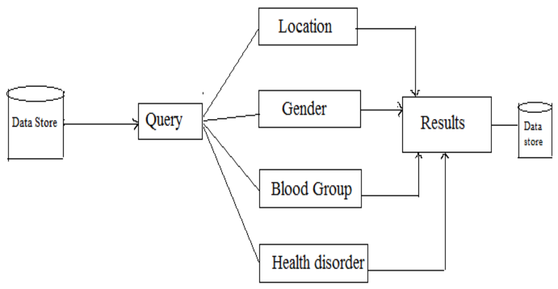

Figure 4.3 Filtering Mechanism

The collaborative filtering is implemented for being used to recommend the samples. Unlike the previous approaches, here each every user is taken in to consideration rather than taking some part of the data sets.

*4) Recommendation Technique*

The main objective of collaborative filtering is to recommend the samples . So in this scenario, the recommendation is done to the doctors on medical fields as follows.

Once the user logged through OAuth , the details related to medical terms are gathered from the user and stored in data store, then if some other user logs in and searches for any diseases, then the search results are fetched based on the collaborative filtering. So all the searches and results are being continuously monitored and stored by our algorithm.

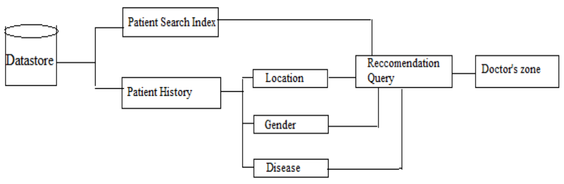

Figure 4.4 Recommendation Mechanism

At this case, when the user ( Patient ) visits the doctor, the user profile is loaded to the doctors page, that contains what are all the searches that the user made, So based on the details and the patient's case, the doctors prescribe the medicines for the user. So now, this details also included to the data store that is being separately maintained.

The efficient filtering is done by taking the following factors in to considerations. Age , Gender, Location, Previously prescribed medicines, Patient search histories. So based on all the above factors, the Medicines are recommended, and also including this, the new medicines are also suggested and hence it is stored to the data store.

## V Conclusion

In this project, an efficient technique for mining sparse data using collaborative technique is used. The outcome of the mined data is recommended for medical industry and its associated fields. The recommendation is done using the novel dynamic personalized recommendation algorithm for sparse data, in which lot of factors are taken in to consideration. Unlike the previous approaches done so far, the recommendation is done based on the user search results and the their associated medical histories. Efficient mining technique is applied in order to query the user medical data in order to recommend it to the associated doctor. A complete user interface is built in order to search and filter the cluster of data. A set of dynamic features are designed to describe the preference information based on collaborative filtering technique, and finally a recommendation is made by adaptively weighting the features using information in multiple phases of interest.

## Reference

[1]. Xiangyu Tang and Jie Zhou, “Dynamic Personalized Recommendation on Sparse Data”, IEEE transactions on knowledge and data engineering, 2013.
[2]. Probabilistic Memory-Based Collaborative Filtering, Kai Yu, Anton Schwaighofer, Volker Tresp, Xiaowei Xu, and Hans-Peter Kriegel
[3]. Matrix Factorization Techniques For Recommender Systems, Yehuda Koren, Yahoo Research Robert Bell and Chris Volinsky, AT&T Labs—Research
[4]. Y. Koren, Collaborative filtering with temporal dynamics, Communications of the ACM 53 (4) (2010) 89–97.
[5]. G. Adomavicius, A. Tuzhilin, Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions, IEEE Trans. Knowl. Data Eng. 17 (6) (2005) 734–749.
[6]. B. M. Sarwar, G. Karypis, J. A. Konstan, J. Riedl, Item-based collaborative filtering recommendation algorithms, in: WWW, 2001, pp. 285–295.
[7]. P. Brusilovsky, A. Kobsa, W. Nejdl (Eds.), The Adaptive Web, Methods and Strategies of Web Personalization, Lecture Notes in Computer Science, Springer, 2007.
[8]. S. Boutemedjet, D. Ziou, Long-term relevance feedback and feature selection for adaptive content based image suggestion, Pattern Recognition 43 (12) (2010) 3925–3937.
[9]. B. M. Kim, Q. Li, C. S. Park, S. G. Kim, J. Y. Kim, A new approach for combining content-based and collaborative filters, J. Intell. Inf. Syst. 27 (1) (2006) 79–91.
[10]. G. Prassas, K. C. Pramataris, O. Papaemmanouil, Dynamic recommendations in internet retailing, in: ECIS, 2001.
[11]. S. Rendle, C. Freudenthaler, L. Schmidt-Thieme, Factorizing personalized markov chains for next-basket recommendation, in: Proceedings of the 19th WWW, ACM, 2010, pp. 811–820.
[12]. B. Efron, T. Hastie, I. Johnstone, R. Tibshirani, Least angle regression, The Annals of statistics 32 (2) (2004) 407–499.
[13]. B. Boser, I. Guyon, V. Vapnik, A training algorithm for optimal margin classifiers, in: Proceedings of the fifth annual workshop on Computational learning theory, ACM, 1992, pp. 144–152.