ENSEMBLE METHODS FOR TOP-N RECOMMENDATION

A Thesis

Submitted to the Faculty

of

Purdue University

by

Ziwei Fan

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science

May 2018

Purdue University

Indianapolis, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Xia Ning, Chair

Department of Computer and Information Science

Dr. Mohammad Al Hasan

Department of Computer and Information Science

Dr. Murat Dundar

Department of Computer and Information Science

Approved by:

Dr. Shiaofen Fang

Head of the Graduate Program

To my parents

ACKNOWLEDGMENTS

I want to express my sincere gratitude to my advisor Dr. Xia Ning for her continuous support in my master studies. Her guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my master studies. I also want to thank my thesis committee, Dr. Mohammad Al Hasan and Dr. Murat Dundar for their help in finishing my thesis.

Next, I want to thank all labmates, Wen-hao Chiang, Junfeng Liu, Evan Burgen and Bo Peng for the interesting discussions, for the continuous support , and for the fun we have had in the last two years.

I also want to thank the support and resources provided by IUPUI. I want to thank Nicole Wittlief as well for her support and patience.

Last but not least, I want to thank my parents sincerely for their financial and emotional support. Without their support, I would have never been able to study in IUPUI and finish the thesis.

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Fan Ziwei M.S., Purdue University, May 2018. Ensemble Methods for Top-N Recommendation. Major Professor: Xia Ning.

As the amount of information grows, the desire to efficiently filter out unnecessary information and retain relevant or interested information for people is increasing. To extract the information that will be of interest to people efficiently, we can utilize recommender systems. Recommender systems are information filtering systems that predict the preference of a user to an item. Based on historical data of users, recommender systems are able to make relevant recommendations to users. Due to its usefulness, Recommender systems have been widely used in many applications, including e-commerce and healthcare information systems. However, existing recommender systems suffer from several issues, including data sparsity and user/item heterogeneity.

In this thesis, a hybrid dynamic and multi-collaborative filtering based recommendation technique has been developed to recommend search terms for physicians when physicians review a large number of patients' information. Besides, a local sparse linear method ensemble has been developed to tackle the issues of data sparsity and user/item heterogeneity.

In health information technology systems, most physicians suffer from information overload when they review patient information. A novel hybrid dynamic and multi-collaborative filtering method has been developed to improve information retrieval from electronic health records. We tackle the problem of recommending the next search term to a physician while the physician is searching for information about a patient. In this method, I have combined first-order Markov Chain and multi-collaborative filtering methods. For multi-collaborative filtering methods, I have de-

veloped the physician-patient collaborative filtering and transition-involved collaborative filtering methods. The developed method is tested using electronic health record data from the Indiana Network for Patient Care. The experimental results demonstrate that for 46.7% of test cases, this new method is able to correctly prioritize relevant information among top-5 recommendations that physicians are truly interested in.

The local sparse linear model ensemble has been developed to tackle both the data sparsity and the user/item heterogeneity issues for the top-n recommendation. Multiple local sparse linear models are learned for all the users and items in the system. I have developed similarity-based and popularity-based methods to determine the local training data for each local model. Each local model is trained on Sparse Linear Method (`SLIM`) which is a powerful recommendation technique for top-n recommendation. These learned models are then combined in various ways to produce top-N recommendations. I have developed model results combination and model combination methods to combine all learned local models. The developed methods are tested on a benchmark dataset and its sparsified datasets. The experiments demonstrate 18.4% improvement from such ensemble models, particularly on sparse datasets.

1. INTRODUCTION

As the amount of information grows, the desire to extract the information that will be of interests to users is increasing. One of the important techniques to discover what users like or dislike is recommender system. For example, in Amazon online shopping website, there are tens of millions items for users to select. It is impossible for users to find interesting items by clicking all items one by one. Recommender systems can help users efficiently discover what they need or like by utilizing their browsing or rating history. Users can easily discover the interesting items in the provided recommendations. However, there are still some challenges in recommender systems, which we discuss below.

One of the challenges of recommender systems is data sparsity. Users can only provide feedbacks for a small set of items compared to a large pool of items. For most of the users, ratings on the majority of items are missing. Without having enough data, it is difficult to accruately project the interests of a user. For example, when we utilize the idea of collaborative filtering to make recommendation for users, the recommendation is less reliable when the data sparsity problem is severe. The basic idea of collaborative filtering is "similar users like similar items". We calculate the similarity between users based on common rated items between users. The higher the data sparsity is, the lower reliability of these calculated similarities are. If the data is highly sparse, we are unable to extract reliable neighborhoods. Another challenge of recommender systems is user/item heterogeneity. When we train a global model for all users and items, such as the popular matrix factorization model, the recommender systems fail for certain users/items. Users may have diverse interests and items may belong to so many different branches. When we train all users and items in a single global model, it is hard for the global model to accurately make recommendations for specific users/items.

Recommender systems techniques have been widely applied to E-commerce platforms and healthcare systems. In healthcare systems, most physicians suffer from information overload when they review patient information. Recommender systems are able to efficiently prioritize relevant information for physicians. In E-commerce, only users' similarities and items' similarities will be considered. However, in healthcare systems, more similarities should be considered when we apply collaborative filtering technique to healthcare systems, for example, physicians' similarities, patients' similarities and search terms' similarities. To make more accurate recommendations for physicians, we should also consider the dynamics of physicians' searching history.

This dissertation focuses on improving information retrieval from electronic health records by using dynamic and multi-collaborative filtering. It also covers the work of local sparse linear method ensemble which tackles the issues of data sparsity and user/item heterogeneity. I have developed a new dynamic and multi-collaborative filtering approach by using electronic health record data from the Indiana Network for Patient Care. This new method is able to correctly rank relevant information among the top-5 recommendations that physicians are truly interested in. I also tested the local sparse linear method ensemble on a benchmard dataset. The experiments demonstated a great improvement from such ensemble methods particularly on sparse datasets.

1.1 Thesis Outline

The remaining of the thesis is organized as follows. In Chapter 2, I will introduce some necessary background of recommender systems. In Chapter 3, I will describe the work of improving information retrival from electronic health records using dynamic and multi-collaborative filtering in detail. In Chapter 4, I will describe the work of local sparse linear method ensemble for top-n recommendation in detail. Finally, I will summarize the dissertation in Chapter 5.

2. BACKGROUND OF RECOMMENDER SYSTEMS

Recommender systems are personalized information filtering techniques [1]. Recommender systems predict users' preference based on users' historical interactions with items. Recommender systems have been widely used in many applications. For example, in health information technology system, recommendation techniques have been applied to physicians recommendation problem [2, 3], drugs recommendation problem [4] and nursing care plan recommendation [5], etc.

2.1 Formulations of Recommendation Problems

Based on different inputs and outputs of recommender systems, there are two main problems in recommender systems [6]. The first problem is rating prediction problem. The input of this problem is typically rating data (explicit feedback). For example, in the Douban website, users can rate movies and books by giving ratings in the range of $[1, 5]$. The input of this problem is usually expressed as a user-item rating matrix. Each element in the rating matrix is the rating given by a user for an item. One important characteristics of this rating matrix is that the rating matrix is very sparse. In the rating matrix, only very few elements are known while most of them are unknown. The goal of rating prediction problem is to fill out the rating matrix by using existing known ratings. The second problem is top-n recommendation problem. Different from rating prediction problem, the prediction values of top-n recommendation are not important. In practice, the input of this problem is typically binary data (implicit feedback). For example, if a user has purchased an item or watched a movie, the implicit feedback of this user to this item is 1, otherwise, it will be unknown or 0. The goal of top-n recommendation problem is to generate an ordered list of N items which will be of interest to the user.

2.1.1 Rating Prediction

This problem is to predict the numerical rating value of a particular item given by a certain user. The training data for this problem is typically rating data which indicates the user preferences for items. For m users and n items, the training data is an incomplete user-item rating matrix of size $m \times n$. In the rating matrix, every known element indicates the known preference given by a user to an item. The missing (or unknown) values are predicted. The popular technique to solve this rating prediction problem is matrix factorization, which will be discussed in Section 2.2.2.

2.1.2 Top-N Recommendation

Different from rating prediction, top-n recommendation problem tries to generate a ranking list of items that will be of interest to a certain user instead of accurately predicting the user preferences for items. In most of scenarios, the recommendations are typically shown as a list of items to users instead of displaying the recommended ratings to users. It is also difficult to collect ratings (explicit feedback) compared with implicit feedback. For example, it is much easier for users to just click on the link instead of providing an exact value of preference. Moreover, the standards of rating items are different for different users. The input for this problem is typically binary feedback. The feedback can be any form of users' behaviors, such as browsing, purchasing and deleting (negative feedback). One classical work of solving this problem is Bayesian Personalize Ranking [7], which applied pair-wise idea to solve the top-n recommendation problem.

2.2 Collaborative Filtering Models

The basic idea of Collaborative Filtering models is utilizing collective power from similar users or similar items to make recommendations. We can first calculate the similarities between users by using their item rating profiles. For example, if two

users have similar ratings on two movies, their similarity will be high. By using the similarities, we are able to make inference about the unobserved feedbacks. One of the challenges in collaborative filtering models is data sparsity, which means the training rating matrices are highly sparse. Most of the ratings are unobserved. There are typically two types of collaborative filtering methods [8], including the neighborhood-based collaborative filtering and model-based methods [9].

2.2.1 Neighborhood-Based Collaborative Filtering

Neighborhood-based collaborative filtering methods are the earliest collaborative filtering methods. The unobserved ratings are predicted based on users' neighborhoods or items' neighborhoods or both of them. Based on different ways to define the neighborhoods, neighborhood-based methods can be classified into two categories, user-based collaborative filtering and item-based collaborative filtering.

User-Based Collaborative Filtering

In this method, we utilize the ratings of similar users to the target user A to make the recommendations for target user A [6]. The basic idea of this method is first to identify a set of users who share similar taste with the target user, then compute the weighted average of the ratings from that the set of similar users to make prediction as follows,

$$\hat{r}_{u,i} = \mu_u + \sum_{v \in S(u)} \text{sim}(u, v) \times (r_{v,i} - \mu_v), \tag{2.1}$$

where $\hat{r}_{u,i}$ is the estimated rating of item i by user u, μ_u is the average rating of user u, $\text{sim}(u, v)$ is the user-user similarity between user u and user v and $S(u)$ is the set of top-k similar users to user u. The similarity can be measured by using users' rating profile on items and be of different forms, such as cosine similarity. Given a user-item

rating matrix R of size $m \times n$, the cosine similarity between user u_1 and user u_2 is calculated as follows,

$$\text{sim}(u_1, u_2) = \frac{R(u_1, :)R(u_2, :)^{\mathrm{T}}}{\|R(u_1, :)\|\|R(u_2, :)\|}, \tag{2.2}$$

where m is the number of users and n is the number of items and $R(u_1, :)$ is the u_1-th row of the rating matrix R.

Item-Based Collaborative Filtering

Similar to the user-based collaborative filtering, item-based collaborative filtering utilizes the ratings of similar items to the target item to predict the rating of the target item by the target user [1]. First, we identify the set of similar items to the target item. Then the ratings from the set of similar items, which has been rated by the target user, will be used to predict if the target user likes the target item as follows,

$$\hat{r}_{u,i} = \mu_i + \sum_{k \in S(i)} \text{sim}(i, k) \times (r_{u,k} - \mu_k), \tag{2.3}$$

where $\hat{r}_{u,i}$ is the estimated rating of item i by user u, μ_i is the average rating of item i, $\text{sim}(i, k)$ is the item-item similarity between item i and item k and $S(i)$ is the set of top-k similar users to user u. The similarity can be measured by using items' rating profile by users. Given a user-item rating matrix R of size $m \times n$, the cosine similarity between user i_1 and user i_2 is calculated as follows,

$$\text{sim}(i_1, i_2) = \frac{R(:, i_1)^{\mathrm{T}}R(:, i_2)}{\|R(:, i_1)\|\|R(:, i_2)\|}, \tag{2.4}$$

where m is the number of users and n is the number of items.

2.2.2 Model-Based Methods

Model-based methods have been widely used and studied in recommender system [10]. Among the model-based methods, latent factor models are considered to be the state-of-the-art models in recommender system.

Matrix Factorization

Matrix factorization is proposed by Koren et al [10]. Matrix factorization has become popular in solving rating prediction problem. The basic idea of matrix factorization is that the user's preference on the item is calculated as the inner product of the user's latent vector and item's latent vector shown as equation 2.5. The latent vectors of users and items are learned from training data.

$$\tilde{r}_{u,i} = q_i^{\mathrm{T}} p_u, \tag{2.5}$$

where q_i is the item i's latent vector whose dimensionality is d and p_u is the user u's latent vector whose dimensionality is d. To learn the q_i and p_u, the following problem is optimized by using stochastic gradient descent,

$$\min_{p,q} \sum_{(u,i)\in K} (r_{ui} - q_i^{\mathrm{T}} p_u) + \lambda(\|q_i\|^2 + \|p_u\|^2), \tag{2.6}$$

where K is the set of the (u, i) pairs whose ratings are observed.

2.3 Related Work of Top-N Recommendation

In this section, I will introduce some related work of top-n recommendation. The work of top-n recommendation can be categorized to three types, including point-wise top-n recommendation, pair-wise top-n recommendation and list-wise top-n recommendation.

2.3.1 Sparse Linear method (SLIM)

Ning and Karypis [11] proposed the sparse linear method (`SLIM`) for top-n recommendation. The basic idea of `SLIM` is that the user's preference over an item is modeled as linear aggregation over the items that the user purchased before. The `SLIM` model learns the item-item coefficient matrix by incorporating the L1 regularization, which introduces sparsity. The predicting score is formulated as follows,

$$\tilde{r}_{u,i} = R(u,:)W(:,i), \tag{2.7}$$

where we have m users and n items, $\tilde{r}_{u,i}$ is the estimated user preference of user u on item i, R is the user-item rating matrix of size $m \times n$, $R(u,:)$ is the u-th row of the binary user-item purchase matrix, W is the item-item coefficient matrix of size $n \times n$ and $W(:,i)$ is the i-th column of the coefficient matrix W. To learn the coefficient matrix W, the following problem is solved,

$$\begin{aligned} \min_{W} \quad & \frac{1}{2}\|R - RW\|_F^2 + \frac{\beta}{2}\|W\|_F^2 + \lambda\|W\|_{\ell_1} \\ \text{s.t.} \quad & W \geq 0, \mathrm{diag}(W) = 0. \end{aligned} \tag{2.8}$$

where the constaint $\mathrm{diag}(W) = 0$ is applied to avoid the useless solution. When there is no constraint $\mathrm{diag}(W) = 0$, the optimal solution of W will be identical matrix, which means for an item will only recommend itself so as to minimize the loss.

2.3.2 Bayesian Personalized Ranking

Rendle et al [7] proposed Bayesian Personalized Ranking (BPR) to solve top-n recommendation by using the idea of pair-wise ranking. Different from previous methods, BPR directly optimized the ranking. BPR optimized the ranking statistics AUC (area under the ROC curve). Rendle et al [7] proposed to use stochastic gradient descent for learning the parameters of BPR. The idea of pair-wise ranking was applied

to BPR in the following way. The probability of user u preferring item i to item j is expressed as:

$$p(i >_u j, |\Theta) = \frac{1}{1 + e^{\hat{x}_{u,i} - \hat{x}_{u,j}}}, \tag{2.9}$$

where $\hat{x}_{u,i} = U(u,:)V(:,i)$, U is the latent matrix of all users and V is the latent matrix of all items, Θ is denoted as the set of parameters. The probability is calculated by using the difference between prediction scores of the user to two items.

The definition of AUC of user u is written as:

$$AUC_u = \frac{1}{|I_u^+||I \setminus I_u^+|} \sum_{i \in I_u^+} \sum_{j \in I \setminus I_u^+} \mathrm{I}(x_{u,i} > x_{u,j}), \tag{2.10}$$

where I is the entire item set, I_u^+ is the set of items which user u has provided possitive feedbacks to, $I \setminus I_u^+$ is the set of item which users do not provide any feedback to. The AUC cannot be directly optimized, so Rendle et al [7] smooths the AUC by using the differentiable loss $\mathrm{I}(x_{u,i} > x_{u,j}) = \ln \frac{1}{1+e^{\hat{x}_{u,i} - \hat{x}_{u,j}}}$. The objective function of BPR becomes:

$$\frac{1}{|I_u^+||I \setminus I_u^+|} \sum_{i \in I_u^+} \sum_{j \in I \setminus I_u^+} \ln \frac{1}{1 + e^{\hat{x}_{u,i} - \hat{x}_{u,j}}}. \tag{2.11}$$

The parameters are learned by using stochastic gradient descend. At each iteration, a user-item-item triplet $< u, i, j >$ is randomly selected.

2.3.3 CLiMF: Collaborative Less-is-More Filtering

Shi et al [12] proposed CLiMF to solve top-n recommendation by applying the idea of list-wise ranking. Different from BPR, CLiMF directly optimized another ranking statistics Mean Reciprocal Rank (MRR) instead of AUC as BRP [7] suggested. The difference between MRR and AUC is that MRR cares more about the positions of recommendations in the ranking list. MRR measures how highly ranked is the first relevant recommendation item. CLiMF utilized the idea of list-wise ranking while

BPR used the idea of pair-wise ranking. To evaluate the ranking list, Reciprocal Rank (RR) of user u is measured as follows,

$$RR_u = \sum_{i=1}^{N} \frac{Y_{u,i}}{R_{u,i}} \prod_{j=1}^{N} (1 - Y_{u,j} \mathrm{I}(R_{u,i} < R_{u,j})), \tag{2.12}$$

where N is the number of items, $Y_{u,i} = 1$ if user u prefers item i, $R_{u,i}$ refers to the ranking position of item i in the recommendation list of user u and $\mathrm{I}() = 1$ when $R_{u,i} < R_{u,j}$. However, RR_u is not differentiable. Shi et al [12] approximated the $\mathrm{I}(R_{u,i} < R_{u,j})$ by using $\mathrm{I}(R_{u,i} < R_{u,j}) = \frac{1}{1+e^{x_{u,i}-x_{u,j}}}$, where $x_{u,i} = U(u,:)V(:,i)$, U is the latent matrix of all users and V is the latent matrix of all items. The lower bound of MRR is found by using Jensen's inequality and the concavity of logarithm funciton. The objective function of CLiMF will become as follows,

$$\sum_{i=1}^{N} [\ln \frac{1}{1+e^{x_{u,i}}} + \sum_{j=1}^{N} \ln(1 - Y_{u,j} \frac{1}{1+e^{x_{u,i}-x_{u,j}}})]. \tag{2.13}$$

The parameters are learned by using gradient descent.

3. IMPROVING INFORMATION RETRIEVAL FROM ELECTRONIC HEALTH RECORDS USING DYNAMIC AND MULTI-COLLABORATIVE FILTERING

3.1 Introduction

When we consider buying a book on Amazon's Website, we often benefit from items listed in a section called "Recommended for you." These recommendations, generated by a method called Collaborative Filtering (*CF*) [8], suggest items of possible interest based on what other customers have viewed and purchased. Often, these suggestions are very useful and lead to additional purchases. However, when physicians search the electronic health records (EHRs) with regard to a particular patient problem, the EHRs do not make suggestions for potentially useful information. Instead, it requires physicians to go through the same manual, cumbersome and laborious process of searching for and retrieving information for similar patients/problems every single time.

In this chapter, we present *DmCF*, a novel hybrid ***D***ynamic and ***m***ulti-***C***ollaborative ***F***iltering method, for information recommendation when physicians search for information from patient EHRs. *DmCF* integrates the following two key ideas:

- collaborative filtering, which prioritizes information items based on what similar physicians have searched for on similar patients; and
- dynamic modeling, which foresees future information items of interest based on how physicians search for information items over time.

Here, *dynamics* refers to the information retrieval patterns over time (e.g., in which order different information items are searched for; which information item will be typically searched for after a certain information item has been retrieved). *Multi-collaborative filtering* (*mCF*) refers to that multiple types of similarities (e.g., physi-

cian similarities, patient similarities and information similarities) are integrated to score information items of possible interest. *DmCF* models information retrieval dynamics by a first-order Markov Chain (*MC*), and combines *MC* transition probabilities (discussed in Section 3.5) with *mCF* scores to produce final recommendation scores for future interested information items. *DmCF* recommends the information items with the highest scores to physicians. We tested *DmCF* on a real dataset from the Indiana Network for Patient Care (INPC). Our experimental results demonstrate 22.3% improvement from *DmCF* over *MC* models on top-1 recommendation (i.e., only the top recommended information item is considered), and for 46.7% of all the test cases, *DmCF* is able to include information items that are truly interesting to the physicians among its top-5 recommendations.

3.2 Literature Review

The most relevant research to our work is from Recommender Systems, a research area that originated in computer science. In particular, top-N recommender systems, which recommend the top-N items that are most likely to be preferred or purchased by users, have been used in a variety of applications in e-commerce. There are typically two categories of collaborative filtering methods [8]. The first category is neighborhood-based collaborative filtering methods [9], which leverage information from similar users and/or similar items to generate recommendations. The second category is model-based methods, particularly latent factor models which learn user and item latent factors and determine user preference over items using the factors. Recent recommendation methods also include deep learning based approaches [13], in which user preferences, item characteristics and user-item interactions can be learned in deep architectures.

Dynamic recommender systems have been developed to recommend information of interest over time. Popular techniques include latent factor transition approaches [14], and Markov models [15] that model the transitions among latent factors capturing

information preference; state space approaches [16, 17] that model the transitions across different states over time; point processes [18] and other statistical models [19] that learn probabilities of future events.

Recommendation methods have been recently used to recommend and prioritize healthcare information, due to the rapid growth of information available about individual patients and the tremendous need for personalized healthcare [20]. Current applications of recommender systems in healthcare include recommending physicians to patients on specific diseases [2,3]; recommending drugs [4], medicine [21] and therapies [22]; and recommending nursing care plans [5], etc.

3.3 Terminologies, Definitions and Notations

Table 3.1.: Notations

notation	description
$y/p/t/v$	a physician/patient/term/visit
$\vec{T}(y,p,v)$	a search term sequence of y on p in visit v
$\mathcal{S}_y(y)$	a set of physicians similar to y
$\mathcal{S}_p(p)$	a set of patients similar to p
$\mathcal{S}_t(t)$	a set of terms similar to t

In EHR systems, there is no measurement similar to numerical rating values in Amazon that can be used to quantitatively assess how much a physician is interested in a certain information item. In this case, we take a type of implicit feedback as a qualitative measurement. That is, if a physician searches for an information item from a patient's EHR data, the physician is considered as interested in that information item during the diagnostic process of the patient, and that information item is useful for/relevant to the diagnosis of the patient. Thus, to evaluate whether a physician is interested in an information item on a patient, we can check whether the physician searches for the information item from the patient's EHR data. Since search is typically done through submitting a search term, we use the two terms "search term"

and "information item" exchangeably, and the problem becomes to recommend the next search term that a physician is interested in on a certain patient.

In this chapter, a physician is denoted as y, a patient is denoted as p, and a search term is denoted as t. A sequence of search terms that a physician y searches for on a certain patient p during a certain patient visit v is represented as

$$\vec{T}(y, p, v) = \{t_{v_1} \rightarrow t_{v_2} \rightarrow \cdots \rightarrow t_{v_k} | y, p\}, \tag{3.1}$$

where t_{v_k} is the k-th search term during visit v. Note that a physician may have multiple search sequences on a single patient during different visits. The physician to who, we recommend the next search term on a patient is referred to as the *target physician*, and the corresponding patient is referred to as the *target patient*. A set of physicians/patients similar to the target physician y/target patient p is denoted as $\mathcal{S}_y(y)/\mathcal{S}_p(p)$, respectively. A set of search terms similar to a particular search term t is denoted as $\mathcal{S}_t(t)$. The size of a set S is denoted as $|S|$. Additional notations will be introduced when they are used (e.g., in Section 3.7). Table 3.1 presents the important notations that we use in this chapter.

3.4 Overview of the Dynamic and Multi-Collaborative Filtering Method – *DmCF*

In this research work, we tackle the problem of recommending the next search term to a physician while the physician is searching for information about a patient. The key idea is to analyze search patterns in order to make recommendations for potentially useful, other information to the physician. To do so, we score and prioritize possible recommendations based on the following two criteria combinatorially:

- which terms the physician has searched for on the patient already and
- which terms similar physicians have searched for on similar patients.

The first criterion considers the search dynamics under the assumption that the past behavior of physicians is a reasonable approximation for the standard of care [23, 24],

and their future behavior follows a same standard of care. Thus, future search terms can be inferred from previously searched terms and their orders. The second criterion considers patient similarities and physician similarities. The underlying intuition is that patients share commonalities and similar patients stimulate similar information retrieval patterns by physicians. Likewise, physicians share commonalities which result in similar search patterns on patients.

We propose a hybrid method which we call *DmCF* that considers search dynamics and multiple similarities for the next search term recommendation. *DmCF* consists of two scoring components. The first component is designed to address search dynamics through a first-order Markov Chain [25]. The score of a possible search term from this dynamics-based scoring component is denoted as $\text{Score}_{\text{DYN}}$. The second component is to score search terms based on similarities via multi-collaborative filtering. The score of a possible search term from this similarity-based scoring component is denoted as Score_{CF}. Thus, *DmCF* scores a next possible search term t for a physician y on a patient p after a sequence of searches $\vec{T}(y,p,v)$ (Equation 3.1) as a linear combination of $\text{Score}_{\text{DYN}}$ and Score_{CF}, that is,

$$\text{Score}(t|\vec{T}(y,p,v)) = (1-\alpha)\cdot \text{Score}_{\text{DYN}}(t|\vec{T}(y,p,v)) + \alpha \cdot \text{Score}_{\text{CF}}(t|\vec{T}(y,p,v)), \quad (3.2)$$

where $\alpha \in [0,1]$ is a weighting parameter.

Table 3.2.: Methods

notation	method description
DmCF	dynamic and multi-collaborative filtering method (Section 4.3)
foMC	first-order markov chain-based scoring method (Section 3.5.2)
ypCF	physician-patient-similarity-based *CF* scoring method (Section 3.6.2)
TptCF	transition-involved patient-term-similarity-based *CF* scoring method (Section 3.6.3)
simP2Y	patient-first similarity identification (Section 3.6.2)
simY2P	physician-first similarity identification (Section 3.6.2)

In this work, if a score is generated from a certain method X, a superscript ^{X} will be included on the score notation (e.g., Score^X, $\text{Score}^X_{\text{DYN}}$ or $\text{Score}^X_{\text{CF}}$). In general, a

superscript ^{X} indicates an associated method X. All possible terms are first scored using the scoring function in Equation 3.2. The top-scored terms are recommended as the next possible search terms. The first-order Markov Chain-based scoring and the multi-collaborative filtering-based scoring will be discussed in Section 3.5 and Section 3.6, respectively. Table 3.2 lists all the methods in this work.

3.5 Markov Chain-based Scoring

3.5.1 Background on Markov Chains

Markov Chain (MC) [25] represents a very fundamental dynamic modeling scheme based on the Markovian assumption. The Markovian assumption states that in a sequence of events $(e_0, e_1, e_2, \cdots, e_{t-1}, e_t)$, each event only depends on a small set of previous consecutive events but independent of any earlier events. An MC models a sequence of events so that each of the events follows the Markovian assumption. The Markovian assumption is statistically represented as $P(e_t|e_0, e_1, e_2, \cdots, e_{t-1}) = P(e_t|e_{t-k}, \cdots, e_{t-2}, e_{t-1})$, where $P(e_t|E)$ is the probability of observing event e_t given the previous event sequence E. The number of previous events that e_t depends on (i.e., k in $P(e_t|e_{t-k}, \cdots, e_{t-2}, e_{t-1})$) defines the order of the MC. A special MC is first-order MC, in which each event only depends on its immediate precursor. MC has been demonstrated to be very effective in modeling, approximating and analyzing real-life sequence data [25].

3.5.2 First-Order Markov Chain-based Scoring – *foMC*

We use a first-order MC as the dynamic model to simulate the sequence of terms that a physician y searches for on a patient p during a visit. This method is referred to as ***f***irst-***o***rder ***M***arkov ***C***hain, denoted as *foMC*. For a sequence $\vec{T}(y, p, v) =$

$\{t_{v_1}, t_{v_2}, \cdots, t_{v_k}|y,p\}$, *foMC* calculates a dynamics-based score $\text{Score}^{foMC}_{\text{DYN}}$ of a next possible search term t after t_{v_k} as the transition probability from t_{v_k} to t, that is,

$$\text{Score}^{foMC}_{\text{DYN}}(t|\vec{T}(y,p,v)) = P(t|t_{v_k}), \tag{3.3}$$

where $P(t|t_{v_k})$ is the transition probability from t_{v_k} to t in a first-order *MC*. The transition probability $P(t_j|t_i)$ from a term t_i to another term t_j in a first-order *MC* is calculated as the ratio of the total frequency of transitions from t_i to t_j over the total frequency of all transitions from t_i to any terms, that is,

$$P(t_j|t_i) = \Big[\sum_{\vec{T}(y,p,v)} h(t_i \rightarrow t_j|\vec{T}(y,p,v))\Big] \Big/ \Big[\sum_{\vec{T}(y,p,v)} \sum_{(t_i \rightarrow t_k) \in \vec{T}(y,p,v)} h(t_i \rightarrow t_k|\vec{T}(y,p,v))\Big], \tag{3.4}$$

where $(t_i \rightarrow t_k) \in \vec{T}(y,p,v)$ represents that $(t_i \rightarrow t_k)$ is in $\vec{T}(y,p,v)$, $h(t_i \rightarrow t_j|\vec{T}(y,p,v))$ is the frequency of the transitions from t_i to t_j in $\vec{T}(y,p,v)$. Thus, $\text{Score}^{foMC}_{\text{DYN}}$ as in Equation 3.3 is not specific to a particular physician or patient, but corresponds to clinical practices that are summarized from all available physicians and patients.

3.6 Multi-Collaborative Filtering-based Scoring

3.6.1 Background on Collaborative Filtering

Collaborative Filtering (*CF*) is a popular technique in Recommender Systems [8] for recommending items to a target user. The fundamental idea of *CF* is that "similar users like similar items". User-based *CF* methods first identify similar users to the target user, and then recommend to the target user the items that are preferred by similar users. Item-based *CF* methods first identify items similar to the target user's preferred items, and then recommend to the target user such similar items. Thus, *CF* methods heavily depend on the calculation of user similarity and item similarity. A typical way to calculate user similarity is to represent each user using her preference

profile over items, and calculate user similarity as the item preference profile similarity. Likewise, a typical way to calculate item similarity is to represent each item using its preference profiles across users, and calculate item similarity as the user preference profile similarity. The user similarity function and item similarity function in *CF* are often pre-defined, and thus the recommendations based on similarities can be easily interpreted. *CF* is particularly powerful when user and item data are sparse, which is often the case in real-life applications. *CF* is also well-known for its scalability on large-scale problems, particularly when the user similarity and item similarity can be calculated in parallel trivially.

3.6.2 Physician-Patient-Similarity-based CF Scoring – *ypCF*

We developed a *CF* method that generates search term recommendations from similar physicians and patients. This method first identifies similar physicians and similar patients (discussed in Section 3.6.2) and then scores terms searched by similar physicians on similar patients (discussed in Section 3.6.2). This method is referred to as ph***y***sician-***p***atient-similarity-based ***C***ollaborative ***F***iltering, and denoted as *ypCF*.

Identifying similar physicians and similar patients

We developed two approaches to identify the set of similar physicians and the set of similar patients, depending on which set is identified first.

Patient-First Similarity Identification – *simP2Y* In the first approach, a set of patients similar to the target patient p is first identified, and then based on the similar patients, a set of physicians similar to the target physician y is then selected. This approach is denoted as *simP2Y* (i.e., from ***P***atients to ph***Y***sicians). In *simP2Y*, the set of patients similar to the target patient p is represented as

$$\mathcal{S}_p^{\mathrm{P2Y}}(p) = \{p_1, \cdots, p_{k_p} | p\}, \tag{3.5}$$

and is composed of the top-k_p most similar patients to the target patient p (patient-patient similarity will be discussed later in Section 3.7). Given $\mathcal{S}_p^{\text{P2Y}}(p)$, a set of physicians similar to the target physician y is represented as

$$\mathcal{S}_y^{\text{P2Y}}(y|p) = \{y_1, \cdots, y_{k_y} | \mathcal{S}_p^{\text{P2Y}}(p)\}, \tag{3.6}$$

and selected as follows: first, physicians who have ever searched for same terms on p and on one or more patients in $\mathcal{S}_p^{\text{P2Y}}(p)$ are identified. From such physicians, the top-k_y most similar physicians to y are selected into $\mathcal{S}_y^{\text{P2Y}}(y|p)$ (physician-physician similarity will be discussed later in Section 3.7).

Physician-First Similarity Identification – *simY2P* The second approach is to first identify a set of physicians similar to the target physician y, and then based on the similar physicians, to identify a set of similar patients. This approach is denoted as *simY2P* (i.e., from ph***Y***sicians to ***P***atients). In *simY2P*, the set of similar physicians is represented as

$$\mathcal{S}_y^{\text{Y2P}}(y) = \{y_1, \cdots, y_{k_y} | y\}, \tag{3.7}$$

and has the top-k_y most similar physicians to y. Based on $\mathcal{S}_y^{\text{Y2P}}(y)$, a set of patients similar to the target patient p, denoted as

$$\mathcal{S}_p^{\text{Y2P}}(p|y) = \{p_1, \cdots, p_{k_p} | \mathcal{S}_y^{\text{Y2P}}(y)\}, \tag{3.8}$$

is identified as patient p's top-k_p most similar patients on whom physicians in $\mathcal{S}_y^{\text{Y2P}}(y)$ have ever searched for same terms as on p.

Collaborative Filtering in *ypCF*

From $\mathcal{S}_y(y)$ and $\mathcal{S}_p(p)$ (either $\mathcal{S}_p^{\text{P2Y}}(p)$ and $\mathcal{S}_y^{\text{P2Y}}(y|p)$, or $\mathcal{S}_y^{\text{Y2P}}(y)$ and $\mathcal{S}_p^{\text{Y2P}}(p|y)$), a set of physician-patient-term triplets, denoted as $\mathcal{S}_{ypt}^{ypCF}(\mathcal{S}_y(y), \mathcal{S}_p(p)) = \big\{\langle y_i, p_j, t_k\rangle | y_i \in$

$\mathcal{S}_y(y), p_j \in \mathcal{S}_p(p), t_k \in \vec{T}(y_i, p_j, v_l), \forall v_l\}$, is constructed. That is, $\mathcal{S}_{ypt}^{ypCF}(\mathcal{S}_y(y), \mathcal{S}_p(p))$ has all the $\langle y_i, p_j, t_k\rangle$ triplets such that physician $y_i \in \mathcal{S}_y(y)$ has searched for term t_k for patient $p_j \in \mathcal{S}_p(p)$. Thus, for a sequence $\vec{T}(y, p, v) = \{t_{v_1}, t_{v_2}, \cdots, t_{v_k}|y, p\}$, the score $\text{Score}_{\text{CF}}^{ypCF}$ of a next possible search term t is calculated as follows:

$$\text{Score}_{\text{CF}}^{ypCF}(t|\vec{T}(y,p,v)) = \bar{f}(\langle y, p, \cdot\rangle) + \frac{\sum_{\langle y', p', t\rangle \in \mathcal{S}_{ypt}^{ypCF}} \hat{f}(y', p', t) \cdot \text{sim}_y(y, y') \cdot \text{sim}_p(p, p')}{\sum_{y', p': \exists \langle y', p', t\rangle \in \mathcal{S}_{ypt}^{ypCF}} \text{sim}_y(y, y') \cdot \text{sim}_p(p, p')}, \tag{3.9}$$

where $\bar{f}(\langle y, p, \cdot\rangle) = \sum_{t:\langle y,p,t\rangle \in \mathcal{S}_{ypt}^{ypCF}} f(\langle y, p, t\rangle) / \sum_{t:\langle y,p,t\rangle \in \mathcal{S}_{ypt}^{ypCF}} 1$, and $\hat{f}(\langle y', p', t\rangle) = f(\langle y', p', t\rangle) - \bar{f}(\langle y', p', \cdot\rangle)$, $f(\langle y', p', t\rangle)$ is the frequency of the triplet $\langle y', p', t\rangle$ (i.e., how many times y' searches for t on p' in total); $\bar{f}(\langle y, p, \cdot\rangle)$ is the average frequency of all possible terms that y searches for on p; $\hat{f}(\langle y, p, \cdot\rangle)$ is the centered frequency for $\langle y, p, \cdot\rangle$ (i.e., shifted by $\bar{f}(\langle y, p, \cdot\rangle)$) in order to reduce the bias from searches with different frequencies; and $\text{sim}_y(y, y')$ and $\text{sim}_p(p, p')$ are the similarity between y and y', and the similarity between p and p', respectively (discussed in Section 3.7). The intuition behind the scoring scheme in Equation 3.9 is that the possibility that y searches for t on p after a sequence of searches is the aggregation of 1). the average possibility of y searching for arbitrary search terms (i.e., the first term in Equation 3.9), and 2). the possibility that similar physicians search for t on similar patients (i.e., the second term in Equation 3.9).

3.6.3 Transition-Involved Patient-Term-Similarity-based CF Scoring – *TptCF*

The order in which a physician searches for different terms could indicate a diagnosis process, and therefore the search order deserves additional consideration. We developed a new patient-term-similarity-based *CF* scoring method that involves the transitions among search terms. Patient similarities and term similarities are considered in this method, which is different from those in *ypCF* (i.e., physician similarities

and patient similarities in *ypCF*). This method is referred to as ***T***ransition-involved ***p***atient-***t***erm-similarity-based ***C***ollaborative ***F***iltering, denoted as *TptCF*.

TptCF aggregates from all similar patients the transitions from the last search term in a sequence $\vec{T}(y, p, v)$ (Equation 3.1) to another search term. Specifically, *TptCF* identifies a set of patients $\mathcal{S}_p(p)$ similar to the target patient p and a set of terms $\mathcal{S}_t(t_{v_k})$ similar to the last search term t_{v_k} in $\vec{T}(y, p, v)$. The set $\mathcal{S}_t(t_{v_k})$ contains the terms with term-term similarity (discussed in Section 3.7) to t_{v_k} above a threshold β. Then *TptCF* looks into what physicians search for on patients in $\mathcal{S}_p(p)$ after they searched for a similar term in $\mathcal{S}_t(t_{v_k})$. The underlying assumption is that similar patients stimulate similar patterns of search sequences. Thus, the score $\text{Score}_{\text{CF}}^{TptCF}$ of a next possible search term t is calculated as follows:

$$\begin{aligned}\text{Score}_{\text{CF}}^{TptCF}(t|\vec{T}(y,p,v)) = \sum_{p' \in \mathcal{S}_p(p)} \{ & \frac{\text{sim}_p(p,p')}{\sum_{p'' \in \mathcal{S}_p(p)} \text{sim}_p(p,p'')} \\ & \times \sum_{t' \in \mathcal{S}_t(t_{v_k})} \frac{g(t' \to t|p')\text{sim}_t(t_{v_k},t')}{\sum_{t'' \in \mathcal{S}_t(t_{v_k})} g(t'' \to t|p')} \},\end{aligned} \tag{3.10}$$

where $g(t' \to t|p')$ is the frequency of transitions from term t' to term t for patient p' from all possible searches on p', $\text{sim}_t(t_{v_k}, t')$ is the term-term similarity between t_{v_k} and t' (discussed in Section 3.7).

3.7 Similarity Calculation

Physician-Physician Similarities – sim_y We first represent each physician y using a vector of search term frequencies, denoted as $\mathbf{v}$. Each dimension of $\mathbf{v}$ corresponds to a term, and the value in each dimension of $\mathbf{v}$ is the total frequency that the corresponding term has been searched by y. Note that the frequency is aggregated from all the patients that y searches on. This representation scheme is very similar to the bag-of-word representation in text mining [26]. Given the

representation, the similarity between two physicians y and y' is calculated as the cosine similarity between $\mathbf{v}_y$ and $\mathbf{v}_{y'}$, that is,

$$\text{sim}_y(y, y') = \cos(\mathbf{v}_y, \mathbf{v}_{y'}). \tag{3.11}$$

The intuition is that the search term distribution indicates physician specialties and expertise, and physicians of similar specialties and expertise are considered similar.

Patient-Patient Similarities – $\mathbf{sim}_p$ Similarly as for physicians, each patient is also represented using a vector of term frequencies, denoted as $\mathbf{u}$. Each dimension of $\mathbf{u}$ corresponds to a term, and the value in each dimension of $\mathbf{u}$ is the total frequency of the corresponding term searched for by all physicians. The term distribution represents the health histories of the patient, and thus a reasonable patient representation. Given the representation, the similarity between two patients p and p' is calculated as the cosine similarity between $\mathbf{u}_p$ and $\mathbf{u}_{p'}$, that is,

$$\text{sim}_p(p, p') = \cos(\mathbf{u}_p, \mathbf{u}_{p'}). \tag{3.12}$$

Term-Term Similarities – $\mathbf{sim}_t$ Each term t is represented using a vector of patient frequencies, denoted as $\mathbf{w}$. Each dimension in $\mathbf{w}$ corresponds to a patient, and the value in each dimension of $\mathbf{w}$ is the total frequency that term t is searched for by all physicians. The term-term similarity between terms t and t' is calculated as the cosine similarity between $\mathbf{w}_t$ and $\mathbf{w}_{t'}$, that is,

$$\text{sim}_t(t, t') = \cos(\mathbf{w}_t, \mathbf{w}_{t'}). \tag{3.13}$$

The underlying assumption is that if two terms are frequently searched for on the same patient, they are considered as similar in their medical meanings and relatedness.

3.8 Materials

3.8.1 Data

Table 3.3.: Statistics of INPC Dataset

statistics	INPC	CUTOFF (06/26/2013)		CUTOFF (07/18/2013)		CUTOFF (08/15/2013)		CUTOFF (09/03/2013)	
		train	test	train	test	train	test	train	test
$\#p$	13,819	6,669	587	8,471	624	10,852	472	12,014	372
$\#y$	2,121	1,267	126	1,542	147	1,818	126	1,948	105
$\#t$	9,781	5,334	665	6,550	654	7,952	532	8,657	461
$\#\vec{T}$	24,183	10,385	648	13,677	692	18,166	535	20,492	414
$\text{len}(\vec{T})$	69,770	28,789	2,568	38,553	2,506	51,272	1,831	58,146	1,482
$\text{len}(\vec{T})/\#p$	5.049	4.317	4.375	4.551	4.016	4.725	3.879	4.840	3.984
$\text{len}(\vec{T})/\#\vec{T}$	2.885	2.772	3.963	2.819	3.621	2.822	3.422	2.837	3.580

In this table, $\#p$ is the number of patients; $\#y$ is the number of physicians; $\#t$ is the number of terms; $\#\vec{T}$ is the number of sequences; $\text{len}(\vec{T})$ is total length of sequences; $\text{len}(\vec{T})/\#p$ is average length of sequences per patient and $\text{len}(\vec{T})/\#\vec{T}$ is average length of sequences.

The data we use for experiments come from the Indiana Network for Patient Care (INPC) [1]. The INPC is Indiana's major health information exchange, and offers physicians access to the most complete, cross-facility virtual electronic patient records in the nation. Implemented in the 1990s, the INPC collects data from over 140 Indiana hospitals, laboratories, long-term care facilities and imaging centers. We extracted the INPC search logs that were generated between 01/24/2013 to 09/24/2013. Table 3.8.1 presents the statistics of the INPC dataset. Figure 3.1 presents the distribution of sequence length in the dataset. It is notable that search sequences are typically very short (on average 2.89 search terms per each sequence). Figure 3.2 presents the distribution of the number of unique terms for each patient. On average, each patient has 3.85 unique search terms. The short sequences and small number of unique search terms per patient make the recommendation problem difficult, because the available data are very sparse.

[1]IRB Protocol # 1612682149 "Supporting information retrieval in the ED through collaborative filtering".

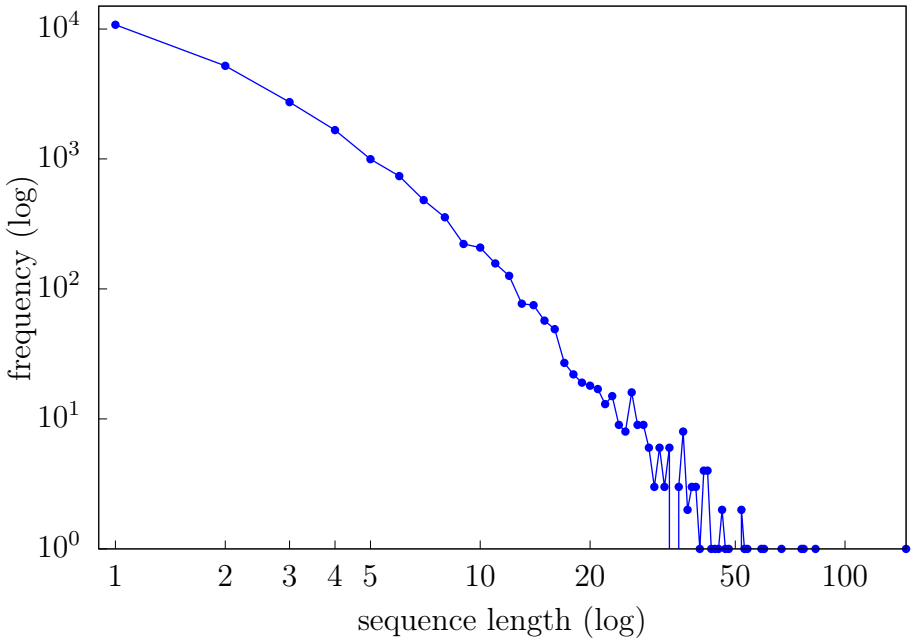

Fig. 3.1.: Distribution of INPC sequence length

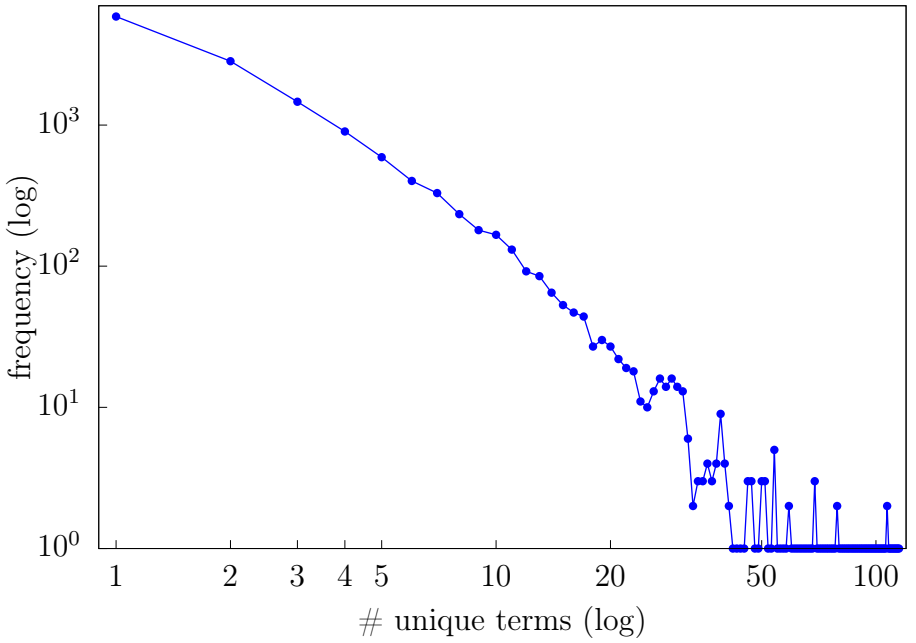

Fig. 3.2.: Distribution of INPC # unique terms per patient

3.8.2 Experimental Protocols and Evaluation Metric

We use the following experimental protocol to evaluate our methods on the INPC dataset: all the search sequences are split by the same cut-off time. Any searches

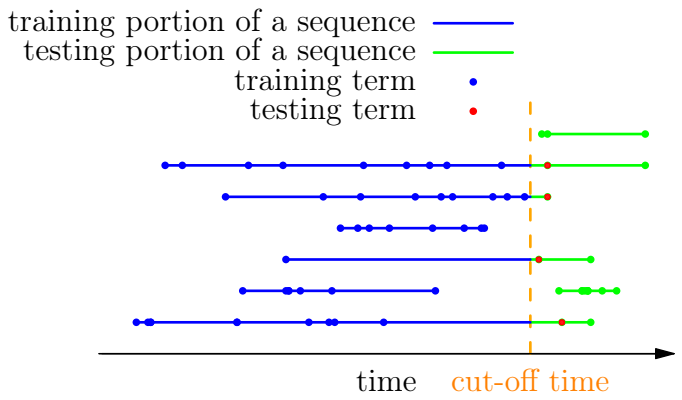

Fig. 3.3.: CUTOFF experimental protocol

before the cut-off time are in the training set, and any searches after the cut-off time are in the test set. The models are trained using only training set, for example, the transition probabilities (Equation 3.4) are constructed only using the search sequences and terms in training set, and the various similarities (Equation 3.11, 3.12 and 3.13) are calculated only from the training set. This protocol is referred to as cut-off cross validation, denoted as CUTOFF. Figure 3.3 demonstrates the CUTOFF experimental protocol. We use the cut-off time 08/15/2013. This cut-off time is selected because sufficient search terms from a majority of the search sequences are retained in training set before the cut-off time and meanwhile sufficient search sequences have testing terms after the cut-off time. We also try other different cut-off times, including 06/26/2013, 07/18/2013 and 09/03/2013. After the split, the statistics for the training and test data are presented in Table 3.8.1 (in "CUTOFF" rows). This CUTOFF setting is close to the realistic scenario, that is, all the data before a certain time should be used to predict information after that time. However, a shortcoming of CUTOFF is that many early search sequences may not have test terms, and many late search sequences will not have anything in the training set. Sequences that do not have test terms are still used to train models. Sequences that do not have

training terms are not used. For those sequences which have terms after the cut-off time, only the first one of the terms after the cut-off time will be used for evaluation.

The model performance is measured using Hit-Rate at N (HR@N). For a sequence, a hit is defined as a recommended term that is truly the next search term. HR@N is the percentage of testing sequences that have a hit and the hit appears among the top-N recommended terms. Higher HR@N values indicate better performance.

3.9 Experimental Results and Discussions

3.9.1 Overall Performance

We compare *foMC*, *ypCF*, *TptCF* and *DmCF*, as well as their variations, in our experiments. Table 3.4 presents the best performance of each method. Overall, *DmCF-ypCF* with *simP2Y* is the best method because 4 out of 5 results of *DmCF-ypCF* with *simP2Y* are the best among all the methods. With parameters α=0.2, $|\mathcal{S}_p|$=1 (i.e., 1 similar patient) and $|\mathcal{S}_y|$=1 (i.e., 1 similar physician), *DmCF-ypCF* with *simP2Y* outperforms the simple *foMC* at 22.3%, 20.2%, 26.0%, 16.7% and 18.1% on HR@1, HR@2, HR@3, HR@4 and HR@5, respectively. The second best method is *ypCF* with *simP2Y* because it has better results overall than the rest methods. With parameters $|\mathcal{S}_p|$=1 and $|\mathcal{S}_y|$=1, *ypCF* with *simP2Y* outperforms the simple *foMC* at 23.3%, 19.5%, 20.1%, 10.3% and 8.9% on HR@1, HR@2, HR@3, HR@4 and HR@5, respectively. It is notable that although *ypCF* is significantly better than *foMC*, the best *DmCF-ypCF* with *simP2Y* has a weight α=0.2 on the *ypCF* scoring component, but a weight 1-α=0.8 on the *foMC* scoring component. This indicates the importance of search dynamics in recommending the next search terms. It is also notable that the optimal *DmCF-ypCF* with *simP2Y* corresponds to a very small number of similar patients ($\mathcal{S}_p$=1) and physicians ($\mathcal{S}_y$=1). This demonstrates the effectiveness of *DmCF-ypCF* in identifying most relevant information and leveraging such information for term recommendation.

Table 3.4.: Overall Performance Comparison with CUTOFF (08/15/2013)

method	sim	α	$\lvert\mathcal{S}_p\rvert$	$\lvert\mathcal{S}_y\rvert$	β	HR@1	HR@2	HR@3	HR@4	HR@5
foMC	-	-	-	-	-	**0.202**	**0.297**	**0.338**	**0.378**	**0.393**
ypCF	*simP2Y*	-	1	1	-	**0.249**	**0.355**	**0.406**	0.417	0.428
		-	50	2	-	0.215	0.336	0.393	**0.424**	0.441
		-	100	2	-	0.222	0.342	0.393	0.422	**0.443**
	simY2P	-	1	1	-	**0.262**	0.292	0.305	0.310	0.320
		-	1	10	-	0.254	**0.329**	0.350	0.368	0.378
		-	2	5	-	0.237	0.312	**0.357**	0.372	0.381
		-	3	20	-	0.230	0.312	0.355	**0.381**	0.393
		-	10	1	-	0.211	0.273	0.336	0.374	**0.398**
TptCF	-	-	160	-	0.1	**0.213**	0.279	0.303	0.322	0.331
	-	-	480	-	0.9	0.189	**0.290**	0.320	0.340	0.355
	-	-	480	-	0.1	0.200	0.284	**0.329**	0.355	0.378
	-	-	500	-	0.1	0.200	0.282	0.327	**0.357**	**0.379**
DmCF-ypCF	*simP2Y*	0.2	1	1	-	**0.247**	0.357	**0.426**	**0.441**	0.464
		0.5	1	1	-	0.245	**0.363**	0.422	0.439	0.464
		0.2	100	2	-	0.226	0.351	0.404	0.430	**0.467**
	simY2P	0.5	3	5	-	**0.254**	0.329	0.353	0.379	0.426
		0.1	3	2	-	0.230	**0.346**	0.366	0.402	0.432
		0.1	1	20	-	0.230	0.331	**0.391**	0.424	0.447
		0.1	1	1	-	0.222	0.331	0.383	**0.430**	0.447
		0.2	1	1	-	0.222	0.323	0.378	0.426	**0.449**
DmCF-TptCF	-	0.8	60	-	0.4	**0.228**	0.307	0.335	0.359	0.379
	-	0.7	40	-	0.1	0.213	**0.312**	0.348	0.376	0.398
	-	0.8	200	-	0.1	0.213	0.303	**0.353**	0.376	0.400
	-	0.6	5	-	0.1	0.209	0.297	0.344	**0.383**	0.406
	-	0.1	1	-	0.1	0.200	0.310	0.346	0.381	**0.413**

In this table, the column "sim" corresponds to similarity identification methods; α is the weight on CF component in *DmCF*; $\lvert\mathcal{S}_p\rvert$ is the number of similar patients; $\lvert\mathcal{S}_y\rvert$ is the number of similar physicians; β is the similarity threshold to identify similar terms. The best performance of each method under each metric is **bold**. The best overall performance of all methods under each metric is **underlined**.

The *DmCF-TptCF* method is also slightly better than *foMC*. With parameters α=0.1, $\lvert\mathcal{S}_p\rvert$=1 and β=0.1, *DmCF-TptCF* outperforms *foMC* at -1.0%, 4.4%, 2.4%, 0.8% and 5.1% on HR@1, HR@2, HR@3, HR@4 and HR@5, respectively. However, *DmCF-TptCF* is significantly worse than *DmCF-ypCF* with *simP2Y*. The difference between *DmCF-TptCF* and *DmCF-ypCF* is that in *DmCF-ypCF*, the similarity-based scoring component (i.e., *ypCF*) does not consider search dynamics and only

Table 3.5.: Overall Performance Comparison with CUTOFF (06/26/2013)

method	sim	α	$\lvert\mathcal{S}_p\rvert$	$\lvert\mathcal{S}_y\rvert$	β	HR@1	HR@2	HR@3	HR@4	HR@5
foMC	-	-	-	-	-	**0.205**	**0.313**	**0.341**	**0.369**	**0.381**
ypCF	*simP2Y*	-	4	1	-	**0.261**	0.366	0.380	0.383	0.383
		-	50	1	-	0.259	**0.377**	0.398	0.414	0.418
		-	100	1	-	0.250	0.373	**0.403**	**0.418**	**0.431**
	simY2P	-	2	3	-	**0.302**	0.350	0.364	0.369	0.372
		-	3	1	-	0.287	**0.370**	0.397	0.414	0.421
		-	5	1	-	0.279	0.360	**0.401**	**0.423**	0.437
		-	10	1	-	0.262	0.349	0.397	0.421	**0.444**
TptCF	-	-	200	-	0.1	**0.207**	0.312	0.335	0.347	0.349
	-	-	220	-	0.1	0.204	**0.313**	0.343	0.350	0.353
	-	-	320	-	0.1	0.199	0.313	**0.347**	**0.361**	0.370
	-	-	380	-	0.1	0.194	0.312	0.346	0.356	**0.372**
DmCF-ypCF	*simP2Y*	0.3	4	1	-	**0.262**	**0.387**	0.415	0.437	0.449
		0.1	20	1	-	0.253	0.377	**0.420**	**0.449**	0.458
		0.2	20	1	-	0.258	0.381	0.420	0.449	**0.460**
	simY2P	0.6	3	10	-	**0.262**	0.370	0.407	0.438	0.455
		0.4	3	1	-	0.219	**0.380**	0.409	0.440	0.469
		0.2	3	4	-	0.227	0.375	**0.417**	0.441	0.463
		0.2	2	3	-	0.216	0.363	0.412	**0.451**	0.463
		0.1	5	1	-	0.228	0.373	0.417	0.443	**0.475**
DmCF-TptCF	-	0.7	5	-	0.1	**0.215**	0.310	0.352	0.381	0.392
	-	0.9	220	-	0.1	0.207	**0.324**	0.356	0.373	0.383
	-	0.8	10	-	0.1	0.208	0.312	**0.360**	0.384	0.394
	-	0.6	10	-	0.1	0.211	0.321	0.355	**0.386**	0.395
	-	0.5	10	-	0.1	0.208	0.318	0.353	0.381	**0.397**

In this table, the column "sim" corresponds to similarity identification methods; α is the weight on CF component in *DmCF*; $\lvert\mathcal{S}_p\rvert$ is the number of similar patients; $\lvert\mathcal{S}_y\rvert$ is the number of similar physicians; β is the similarity threshold to identify similar terms. The best performance of each method under each metric is **bold**. The best overall performance of all methods under each metric is **underlined**.

looks at the search terms that have ever been searched by similar physicians on similar patients, regardless of how such search terms transit to the search term of interest, while *TptCF* considers such transitions. The performance difference between *DmCF-TptCF* and *DmCF-ypCF* may indicate that the transition information captured in *TptCF* might overlap with that captured in *foMC* and thus combining them together will not lead to substantial gains. On the other hand, the information captured by

Table 3.6.: Overall Performance Comparison with CUTOFF (07/18/2013)

method	sim	α	$\lvert\mathcal{S}_p\rvert$	$\lvert\mathcal{S}_y\rvert$	β	HR@1	HR@2	HR@3	HR@4	HR@5
foMC	-	-	-	-	-	**0.210**	**0.292**	**0.325**	**0.341**	**0.348**
ypCF	*simP2Y*	-	5	1	-	**0.267**	0.347	0.358	0.364	0.366
		-	50	1	-	0.262	**0.358**	0.379	0.395	0.400
		-	100	1	-	0.257	0.358	**0.384**	**0.402**	0.412
		-	100	2	-	0.237	0.342	0.380	0.396	**0.413**
	simY2P	-	2	3	-	**0.289**	0.337	0.353	0.357	0.358
		-	1	100	-	0.283	**0.345**	0.353	0.357	0.358
		-	10	1	-	0.240	0.325	**0.379**	**0.410**	**0.426**
TptCF	-	-	260	-	0.1	**0.210**	0.286	0.301	0.312	0.329
	-	-	300	-	0.1	0.207	**0.289**	0.305	0.318	0.329
	-	-	380	-	0.1	0.208	0.288	**0.309**	0.324	**0.341**
	-	-	420	-	0.1	0.208	0.288	0.308	**0.325**	0.340
DmCF-ypCF	*simP2Y*	0.2	5	1	-	**0.267**	**0.364**	0.393	0.403	0.426
		0.1	50	1	-	0.256	0.355	**0.396**	**0.415**	0.428
		0.2	100	1	-	0.253	0.360	0.396	0.413	**0.431**
	simY2P	0.5	2	3	-	**0.251**	0.347	0.387	0.408	0.426
		0.4	2	4	-	0.250	**0.351**	0.392	0.413	0.431
		0.5	5	4	-	0.228	0.341	**0.397**	0.419	0.441
		0.2	5	1	-	0.228	0.335	0.389	**0.423**	0.436
		0.5	10	4	-	0.212	0.315	0.384	0.412	**0.447**
DmCF-TptCF	-	0.8	5	-	0.1	**0.218**	0.292	0.332	0.351	0.367
	-	0.8	300	-	0.1	0.215	**0.305**	0.328	0.345	0.351
	-	0.6	5	-	0.1	0.217	0.302	**0.340**	0.355	0.364
	-	0.5	5	-	0.1	0.215	0.302	0.338	**0.357**	0.364
	-	0.3	1	-	0.1	0.208	0.292	0.331	0.354	**0.367**

In this table, the column "sim" corresponds to similarity identification methods; α is the weight on CF component in *DmCF*; $\lvert\mathcal{S}_p\rvert$ is the number of similar patients; $\lvert\mathcal{S}_y\rvert$ is the number of similar physicians; β is the similarity threshold to identify similar terms. The best performance of each method under each metric is **bold**. The best overall performance of all methods under each metric is **underlined**.

ypCF methods could be complementary to that in *foMC* and thus integration of *ypCF* and *foMC* results in significant performance improvement.

In *DmCF-ypCF*, *simP2Y* is slightly better than *simY2P*. The *simP2Y* method first identifies patients similar to the target patient, and based on the identified similar patients identifies physicians similar to the target physician. The *simY2P* method identifies similar patients and similar physicians in the reversed order as in *simP2Y*.

Table 3.7.: Overall Performance Comparison with CUTOFF (09/03/2013)

method	sim	α	$\lvert\mathcal{S}_p\rvert$	$\lvert\mathcal{S}_y\rvert$	β	HR@1	HR@2	HR@3	HR@4	HR@5
foMC	-	-	-	-	-	**0.193**	**0.271**	**0.304**	**0.331**	**0.365**
ypCF	*simP2Y*	-	10	1	-	**0.261**	0.326	0.345	0.355	0.355
		-	20	1	-	0.261	**0.329**	0.353	0.365	0.367
		-	100	1	-	0.246	0.324	**0.374**	**0.399**	**0.406**
	simY2P	-	1	1	-	**0.278**	0.329	0.350	0.365	0.365
		-	2	3	-	0.271	**0.336**	0.360	0.379	0.384
		-	10	1	-	0.234	0.304	**0.372**	0.391	0.406
		-	5	1	-	0.242	0.331	0.362	**0.396**	0.408
		-	10	20	-	0.222	0.300	0.360	0.389	**0.413**
TptCF	-	-	180	-	0.1	**0.184**	0.246	0.271	0.290	0.304
	-	-	320	-	0.1	0.179	**0.266**	0.295	0.309	0.326
	-	-	500	-	0.1	0.174	0.261	**0.312**	**0.338**	**0.353**
DmCF-ypCF	*simP2Y*	0.2	10	1	-	**0.263**	0.336	0.377	0.389	0.411
		0.1	10	1	-	0.261	**0.338**	0.377	0.389	0.411
		0.1	100	1	-	0.234	0.331	**0.382**	**0.411**	0.425
		0.2	100	1	-	0.246	0.331	0.382	0.408	**0.428**
	simY2P	0.4	3	2	-	**0.242**	0.319	0.355	0.386	0.423
		0.4	2	1	-	0.234	**0.343**	0.384	0.391	0.418
		0.3	3	2	-	0.234	0.336	**0.389**	0.396	0.423
		0.2	4	5	-	0.220	0.333	0.374	**0.403**	0.425
		0.1	2	2	-	0.208	0.312	0.362	0.391	**0.435**
DmCF-TptCF	-	0.8	40	-	0.1	**0.208**	0.292	0.326	0.348	0.374
	-	0.8	20	-	0.1	0.198	**0.292**	0.321	0.345	0.379
	-	0.9	460	-	0.1	0.181	0.271	**0.338**	0.365	0.382
	-	0.9	480	-	0.1	0.184	0.271	0.333	**0.367**	0.382
	-	0.1	5	-	0.1	0.198	0.278	0.319	0.350	**0.389**

In this table, the column "sim" corresponds to similarity identification methods; α is the weight on CF component in *DmCF*; $\lvert\mathcal{S}_p\rvert$ is the number of similar patients; $\lvert\mathcal{S}_y\rvert$ is the number of similar physicians; β is the similarity threshold to identify similar terms. The best performance of each method under each metric is **bold**. The best overall performance of all methods under each metric is **underlined**.

The better performance of *simP2Y* over *simY2P* in *DmCF-ypCF* demonstrates that when physician search dynamics has been considered via *MC*, similar patients should be identified first and then based on identified similar patients, similar physicians should be identified. This may be because that when *MC* already considers all patients and all physicians (Equation 3.4), a more focused and more homogeneous group of patients similar to the target patient is more critical in order to complement to the

MC information. Since physicians may see many patients with different diseases, high physician similarity may be due to patients who are different from the target patient. If such physicians are first selected (e.g., in *simY2P*), similar patients identified from these physicians might be very different from the target patient. However, when no information about all the patients and all the physicians is considered like in *ypCF*, a diverse set of physicians and patients might be beneficial, and that could explain why in *ypCF*, *simY2P* actually outperforms *simP2Y* slightly.

Table 3.5, Table 3.6 and Table 3.7 present the best performance of all the methods for cut-off times 06/26/2013, 07/18/2013 and 09/03/2013, respectively. Overall, *DmCF-ypCF* achieves the best performance over the other methods on the different cut-off times. The trends among different methods as identified from cut-off time 08/15/2013 remain very similar for the other cut-off times. Note that as using later cut-off times, training data become more as shown in Table 3.8.1, and the performance of each method over different cut-off times tends to become worse. For example, the performance of *foMC* model decreases in general over different cut-off times. This may be due to the increasing heterogeneity among patients as more patients in the system. Table 3.5 presents the best performance of all the methods for cut-off time 06/26/2013. Overall, *DmCF-ypCF* with *simP2Y* and *simY2P* are the best methods because 4 out of 5 results of *DmCF-ypCF* with *simP2Y* and *simY2P* are the best among all the methods. The best HR@1 is achieved with the parameters $|\mathcal{S}_p|$=2 (i.e., 2 similar patients) and $|\mathcal{S}_y|$=3 (i.e., 3 similar physicians) of *ypCF* with *simY2P* method. The best HR@2, HR@3, HR@4 and HR@5 are achieved by the *DmCF-ypCF* with *simP2Y* and *simY2P* methods. The best HR@2, HR@3, HR@4 and HR@5 are better than the performance of *foMC* with the improvements of 23.6%, 23.2%, 22.2% and 24.7%. Table 3.6 presents the best performance of all the methods for cut-off time 07/18/2013. Overall, *DmCF-ypCF* with *simP2Y* and *simY2P* are the best methods because 4 out of 5 results of *DmCF-ypCF* with *simP2Y* and *simY2P* are the best among all the methods. The best HR@1 is achieved with the parameters $|\mathcal{S}_p|$=2 (i.e., 2 similar patients) and $|\mathcal{S}_y|$=3 (i.e., 3 similar physicians) of *ypCF* with

simY2P method. The best HR@2, HR@3, HR@4 and HR@5 are achieved by the *DmCF-ypCF* with *simP2Y* and *simY2P* methods. The best HR@2, HR@3, HR@4 and HR@5 are better than the performance of *foMC* with the improvements of 24.7%, 22.2%, 24.0% and 28.4%. Table 3.7 presents the best performance of all the methods for cut-off time 09/03/2013. Overall, *DmCF-ypCF* with *simP2Y* and *simY2P* are the best methods because 4 out of 5 results of *DmCF-ypCF* with *simP2Y* and *simY2P* are the best among all the methods. The best HR@1 is achieved with the parameters $|\mathcal{S}_p|$=1 (i.e., 1 similar patient) and $|\mathcal{S}_y|$=1 (i.e., 1 similar physician) of *ypCF* with *simY2P* method. The best HR@2, HR@3, HR@4 and HR@5 are achieved by the *DmCF-ypCF* with *simP2Y* and *simY2P* methods. The best HR@2, HR@3, HR@4 and HR@5 are better than the performance of *foMC* with the improvements of 26.6%, 28.0%, 24.2% and 19.2%. Overall, the best performance is achieved by the method *DmCF-ypCF*. The trends are also similar for different cut-off times.

Comparing *ypCF* and *TptCF*, it is notable that *ypCF* is significantly better than *TptCF*, even though in *TptCF* more patients similar to the target patient are used to achieve its optimal performance. In *TptCF*, only terms from similar physicians and patients that are similar to the term of interest are considered in calculating the scores (Equation 3.10). However, in *ypCF*, all the terms from similar physicians and patients are used. The improved performance of *ypCF* compared to that of *TptCF* may indicate that using more possible terms could benefit recommendation. On the other hand, both *foMC* and *TptCF* consider term transitions, while *TptCF* considers term transitions only among similar terms on similar patients. The experimental results show that *TptCF* performs worse than *foMC*. This may indicate that if term transition is a major factor in determining next search term, transitions from more diverse patients should be integrated.

Figure 3.4, 3.5, 3.6, 3.7 and 3.8 present HR@1, HR@2, HR@3, HR@4 and HR@5 of *DmCF-ypCF* with *simP2Y* over different α values (Equation 3.2) when $|\mathcal{S}_y| = 1$ and $|\mathcal{S}_p| = 1$. As the weight α increases from 0, that is, as the *CF* takes place in the term scoring (Equation 3.2), the performance of *DmCF* in terms of HR@1,

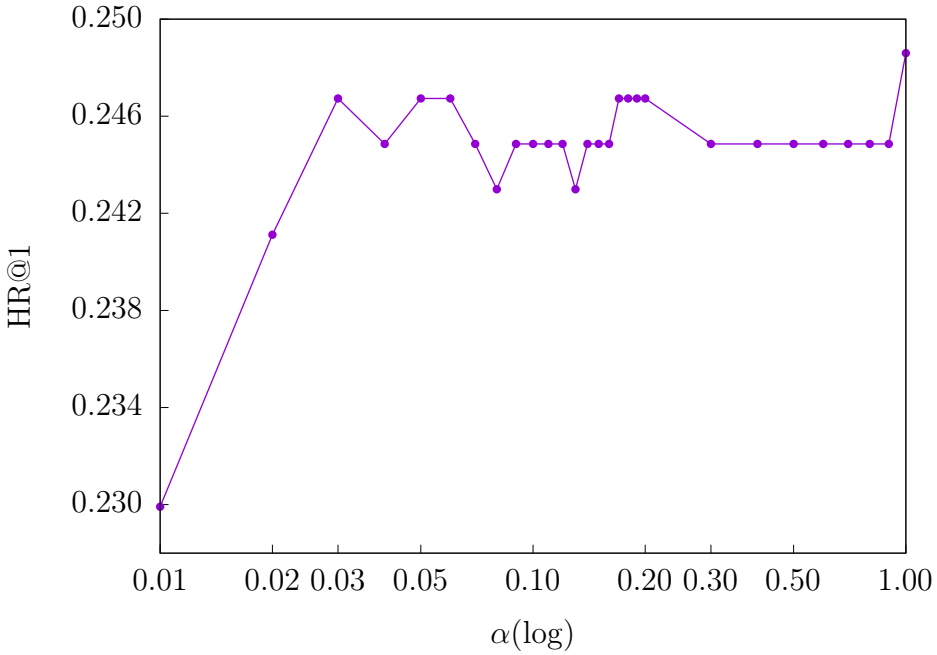

Fig. 3.4.: HR@1 over α values

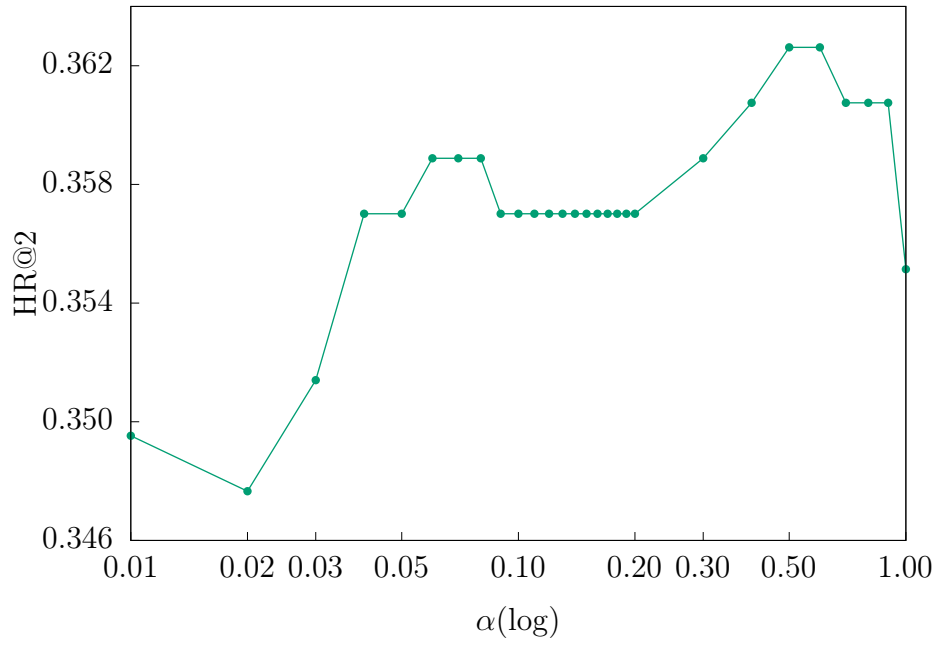

Fig. 3.5.: HR@2 over α values

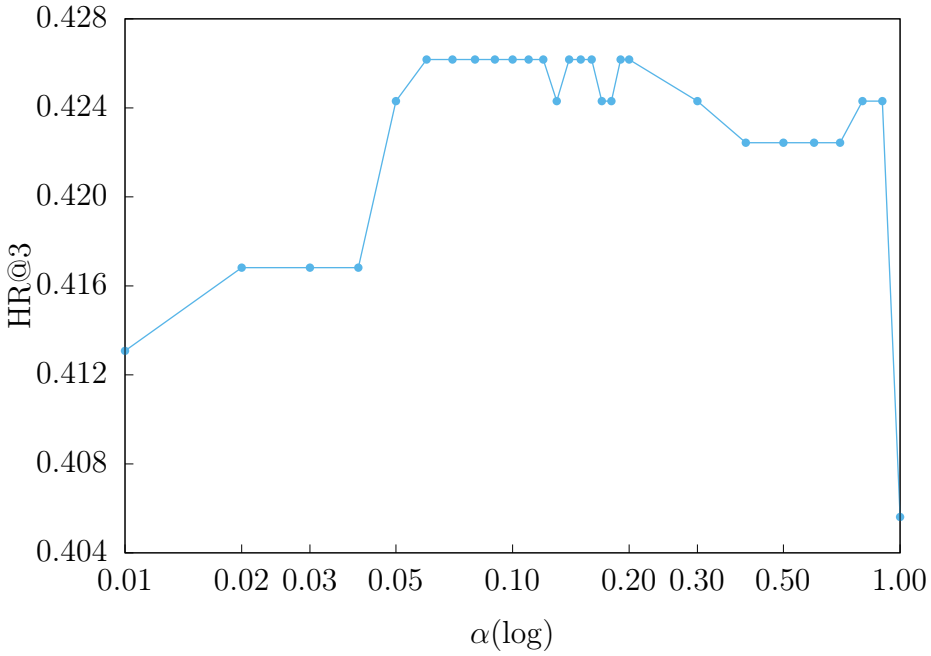

Fig. 3.6.: HR@3 over α values

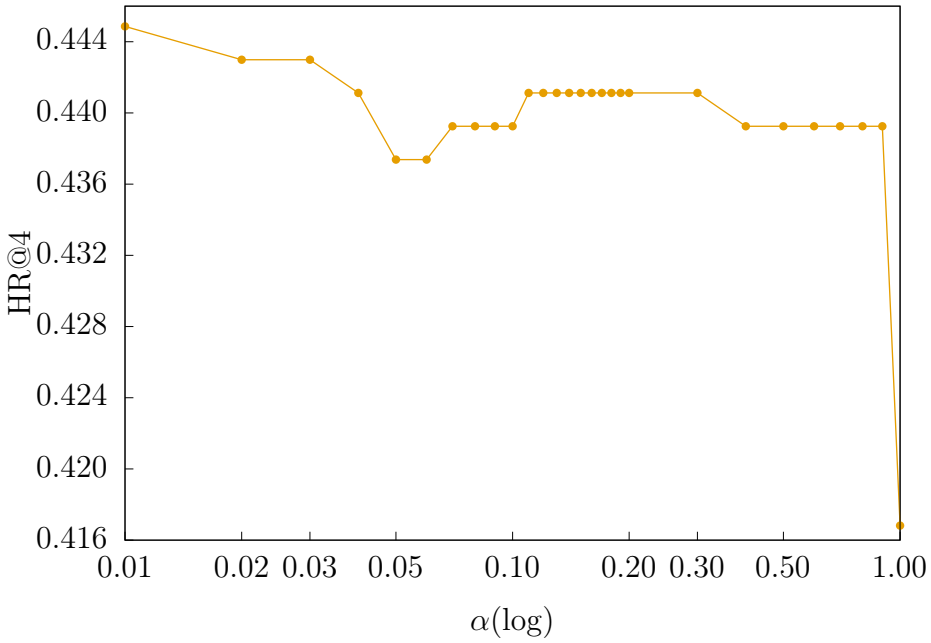

Fig. 3.7.: HR@4 over α values

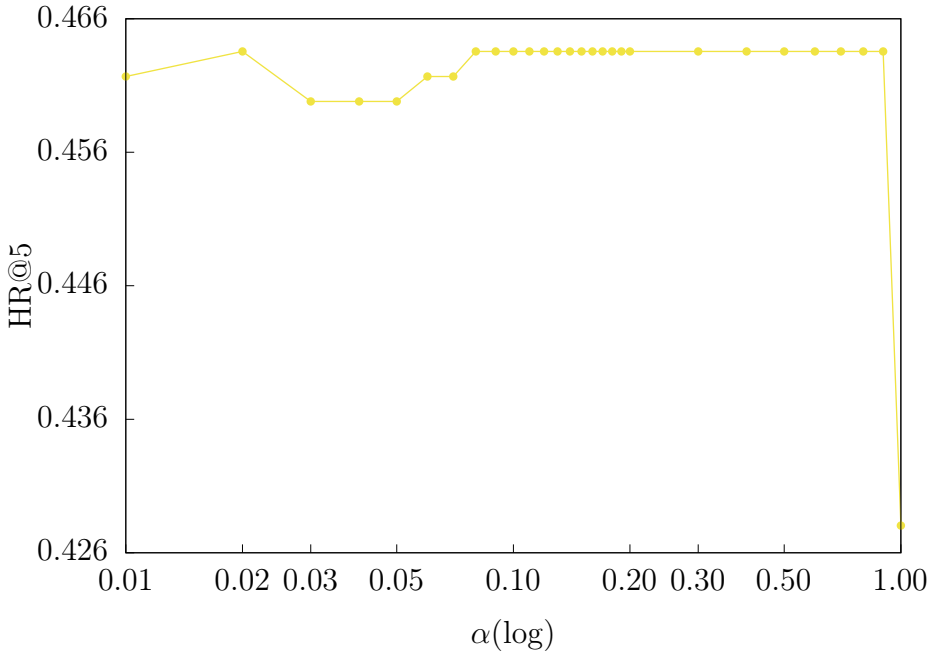

Fig. 3.8.: HR@5 over α values

HR@2 and HR@3 generally increases. For the performance of HR@4 and HR@5, as the weight α increases from 0, the performance slightly decreases, then increases and becomes stable (except when α=1 which means only the *CF* scoring component is considered.). This demonstrates the effect from *CF* scoring component in *DmCF*. As α further increases, the performance in general first gets better and then worse (except that the HR@1 performance reaches its best at α=1). This indicates that the dynamic scoring component and *CF* scoring component in *DmCF* play complementary roles for recommending terms, and thus considering their combination enables better recommendation performance than each of the two methods alone.

3.9.2 Similarity Analysis

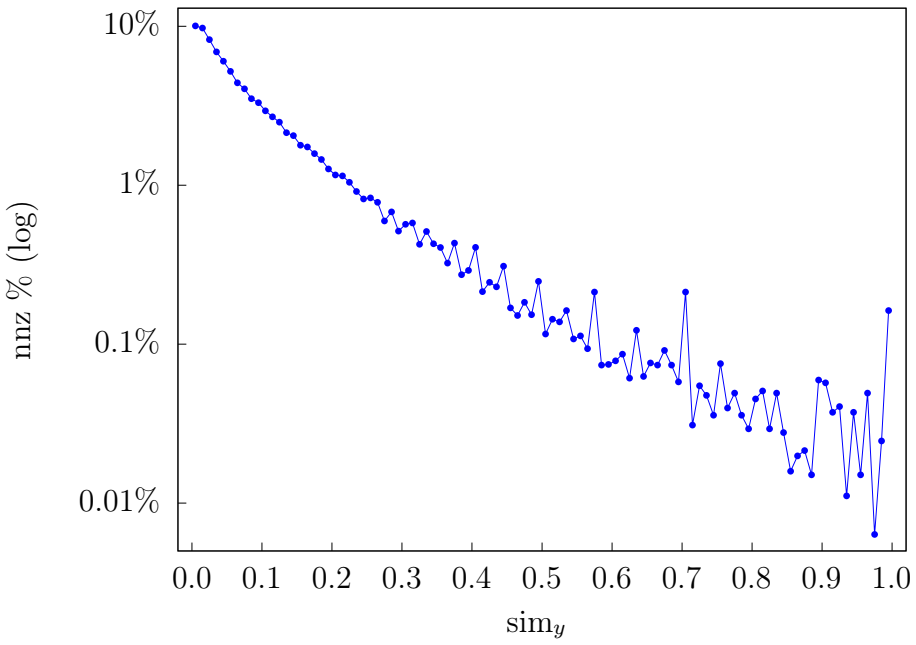

Fig. 3.9.: Physician-physician similarity distribution

Figure 3.9 and 3.10 present the distribution of non-zero physician-physician similarities (sim$_y$) and patient-patient similarities (sim$_p$), respectively. Figure 3.11 presents the distribution of non-zero term-term similarities (sim$_t$). For sim$_y$, 5.65% of physician-

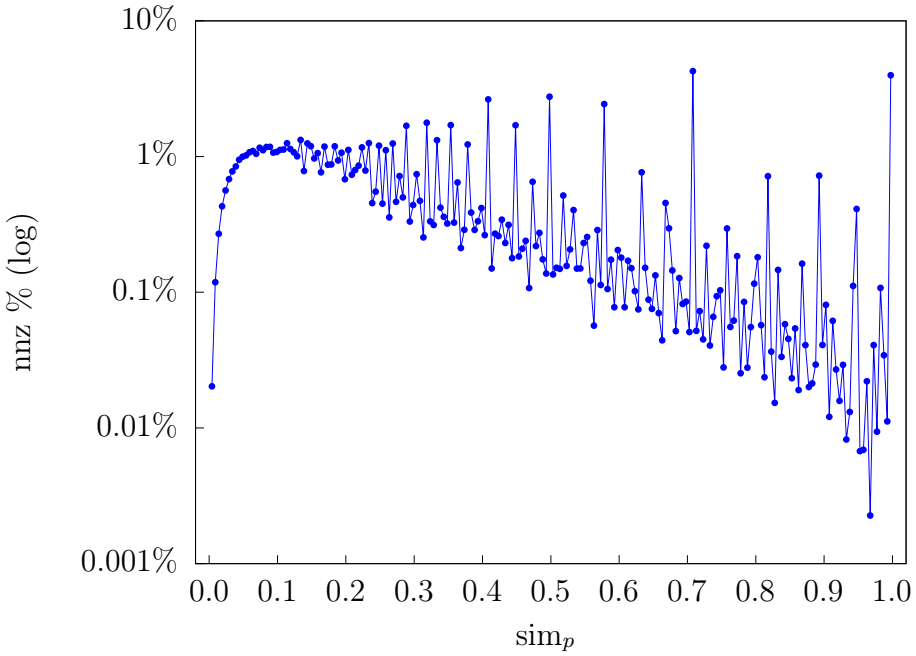

Fig. 3.10.: Patient-patient similarity distribution

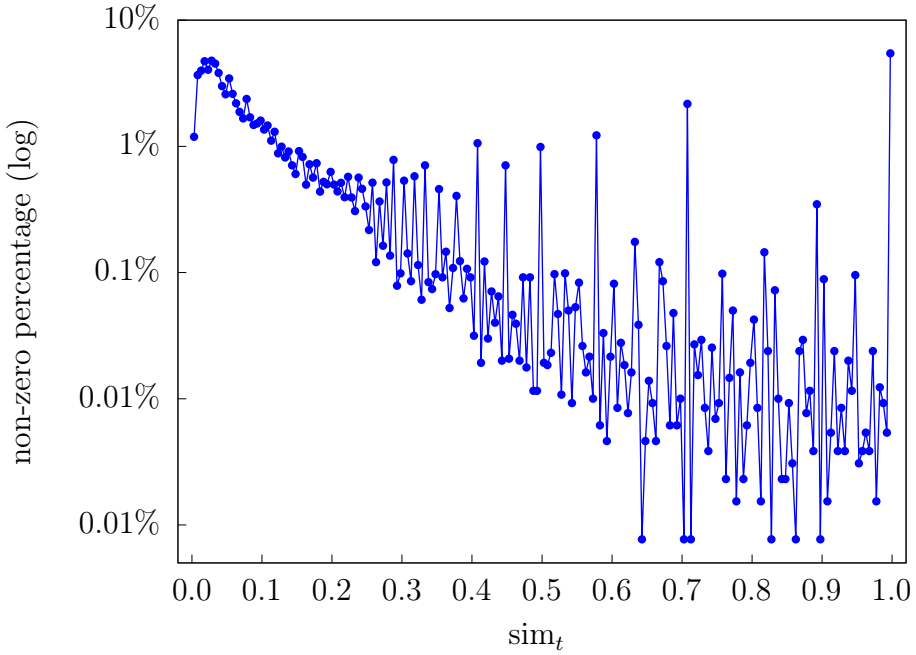

Fig. 3.11.: Term-term similarity distribution

physician similarities are non-zero, and 80.98% of the non-zero similarities are less than or equal to 0.2. For sim_p, 2.65% of the patient-patient similarities are non-zero, and 77.05% of the non-zero similarities are less than or equal to 0.5. For sim_t, only 0.28% of term-term similarities are non-zero, and 78.36% of the non-zero similarities are less than or equal to 0.3. Specially, there are some patients whose similarities with one another are relatively high (i.e., the peaks in Figure 3.10 on larger sim_p values). This also explains the advantages of *simP2Y* over *simY2P* and their performance in Table 3.4, because more patients with higher sim_p to the target patient provide better opportunities for *DmCF* to identify relevant information from such similar patients.

3.10 Conclusions

In this chapter, we presented our new dynamic and multi-collaborative filtering method *DmCF* to recommend search terms relevant to patients for physicians. *DmCF* combines a dynamic first-order Markov chain model and a multi-collaborative filtering model in order to score and prioritize search terms. The collaborative filtering model leverages the key idea originating from Recommender Systems research, and uses patient similarities, physician similarities and term similarities to score potential search terms. The linear combination of the dynamic-based scoring and the multi-collaborative filtering-based scoring is able to produce high quality recommendations that are most relevant to the patients and that are most interested to physicians.

4. LOCAL SPARSE LINEAR MODEL ENSEMBLE FOR TOP-N RECOMMENDATION

4.1 Introduction

Top-N Recommender Systems (RS) have been widely used in E-commerce applications. However, two typical issues still challenge the current top-N RS development: 1). data sparsity, when there are not sufficient data to train a good RS model, and 2). user/item heterogeneity, when a global model (e.g., the popular matrix factorization models) trained from all the users/items fail for certain users/items. Existing methods that tackle the first issue include factorized models [27] and implicit feedback based models [28], etc. Emerging methods dealing with the second issue include the most recent local model based approaches [29, 30].

In this paper, we develop local sparse linear model ensemble to tackle both the data sparsity and the user/item heterogeneity issues. We learn multiple local Sparse LInear Models (`SLIM`) [11] for all the users and items in the system. These models are then combined in various ways to produce top-N recommendations. `SLIM` is strong in learning relations among items, while localizing `SLIM` with respective to certain users and items better reveal localized item relations among a certain group of users. By combining multiple `SLIM` models, signals from multiple models are aggregated so as to enable better results for sparse data. Our experiments over datasets of different sparsity levels demonstrate the superior performance of the model ensemble method.

4.2 Related Work

4.2.1 Sparse Linear Method for top-N Recommendation

Ning and Karypis proposed a state-of-the-art Sparse LInear Method (`SLIM`) for top-N recommendation [11]. In `SLIM`, the user u's preference over an item i is modeled as a linear aggregation over the items that the user purchased before, that is,

$$\tilde{r}_{u,i} = R(u,\cdot)W(\cdot,i), \tag{4.1}$$

where $\tilde{r}_{u,i}$ is the estimated user preference of user u on item i, $R(u,\cdot)$ is the user preference over other items, and $W(\cdot,i)$ is coefficient with respect to item i. To solve for W, `SLIM` solves the following optimization problem,

$$\begin{aligned} \min_{W} \quad & \frac{1}{2}\|R - RW\|_F^2 + \frac{\beta}{2}\|W\|_F^2 + \lambda\|W\|_{\ell_1} \\ \text{s.t.} \quad & W \geq 0, \mathrm{diag}(W) = 0. \end{aligned} \tag{4.2}$$

where $\|\cdot\|_F$ is the Frobenious norm of a matrix; $\|\cdot\|_{\ell_1}$ is the ℓ_1 norm of a matrix; the first term $\|R - RW\|_F^2$ measures the error of re-constructing R using the linear model; the $\|W\|_F^2$ term regularizes W so that values in W will not become too large; the $\|W\|_{\ell_1}$ term introduces sparsity in the solution; the non-negativity constraint over W enforces only non-negativity relations; the zero diagonal constraint over W excludes the cases in which items are used to generate preferences over themselves.

4.2.2 Local Low-Rank Matrix Approximation

Lee *et al.* [29] developed a Local Low-Rank Matrix Approximation (`LLORMA`) method. `LLORMA` first randomly selects a set of K anchor pairs $\{(u_k^*, i_k^*)\}$ ($k = 1, \cdots, K$). With respect to each anchor pair (u_k^*, i_k^*), a local model is learned us-

ing the training data $\{(u,i)\}$ that are selected based on user-item kernel values $\mathcal{K}((u,i),(u_k^*,i_k^*))$. The local models are low-rank matrix factorization models, that is,

$$\tilde{r}_{u,i}^k = \mathbf{u}_u^{k\mathsf{T}} \mathbf{v}_i^k, \tag{4.3}$$

where $\mathbf{u}_u^k$ and $\mathbf{v}_i^k$ are the latent factors for user u and item i from the k-th model, respectively. The global model prediction $\tilde{\tilde{r}}_{u,i}$ of user u's preference over item i is a weighted combination of the predictions from multiple local models as follows:

$$\tilde{\tilde{r}}_{u,i} = \sum_{k=1}^{K} \frac{\mathcal{K}((u,i),(u_k^*,i_k^*))}{\sum_{k'=1}^{K} \mathcal{K}((u,i),(u_{k'}^*,i_{k'}^*))} \tilde{r}_{u,i}^k \tag{4.4}$$

4.2.3 Combining Local Models for Recommendation

The idea of combining local models for recommendation has attracted increasing attention recently. For example, Xu *et al.* [31] proposed a co-clustering method which is based on graph cut. The user-item matrix is viewed as a bipartite graph. And it is assumed that when one user and one item belong to one or more co-clusters together, this item will be given a high rating by this user. This method also allows for soft memberships which means user or item doesn't have to belong to one cluster. The prediction comes from the largest interest group shared by user and item.

Beutel *et al.* [32] has adopted a different method to approximate matrix first cluster users and items, respectively. Then they fit models on the user-item clusters using some mean values. Every co-clustering model aims at fitting the residual of previous co-clustering models. The residuals from this clustering-mean fitting process are forwarded to the next iteration of same processes. For each co-clustering model, a simple K-means like method is used to find local structures. The prediction of each local structure is based on the average rating in that local structure.

Christakopoulou and Karypis [30] combine local models and a global model. First, an initial user clustering is made, and during the learning process, the cluster of user is reassgined and weights of each local model is updated so as to optimize the loss

function. For each local model and global model, SLIM is used to model the item-item relationship in local and global structures. The prediction is made by summing over the local predictions and global prediction.

4.3 Methods

We developed a model ensemble over local sparse linear models for top-N recommendation. Following the idea from Lee [29], a set of anchor pairs is first selected. With respect to each of the anchor pairs, a local `SLIM` model is trained. The local models are then ensembled via combining their results or combining the models directly.

4.3.1 Anchor Pair Selection

We first randomly select a user u^* out of m users as the anchor user. Then from u^*, we randomly select an item i^* that u^* has purchased. This item will be the anchor item. The user-item pair (u^*, i^*) will be the anchor pair with respect to which each local model will be built.

4.3.2 Training Data Selection for Local Models

Training data for each local model with respect to each anchor pair (u^*, i^*) are selected according to: 1). user and item similarities, and 2). user and item popularities.

Similarity-based Training Data Selection

In this method, we use a radial basis function (RBF) kernel to measure the similarity between a user-item pair (u', i') and the anchor pair (u^*, i^*), and apply two

different similarity-based schemes to select training data. The RBF kernel over the two pairs is defined as follows [29]:

$$\mathcal{K}_{ui}((u', i'), (u^*, i^*)) = \mathcal{K}_u(u', u^*) \times \mathcal{K}_i(i', i^*) \tag{4.5}$$

where both $\mathcal{K}_u$ and $\mathcal{K}_i$ are RBF kernels:

$$\mathcal{K}_u(u', u^*) = \exp(-\gamma \|R(u', \cdot) - R(u^*, \cdot)\|^2), \tag{4.6}$$

$$\mathcal{K}_i(i', i^*) = \exp(-\gamma \|R(\cdot, i') - R(\cdot, i^*)\|^2). \tag{4.7}$$

In Equation 4.6 and Equation 4.7, $R(u, \cdot)$ ($R(\cdot, i)$) is purchase profile of by user u (of item i). Since both $\mathcal{K}_u$ and $\mathcal{K}_i$ are valid kernels, $\mathcal{K}_{ui}$ is also a valid kernel. The definition in Equation 4.6 and Equation 4.7 follow the idea in user-based and item-based Collaborative Filtering (CF), respectively, that calculates user and item similarities directly from user-item matrix R. This is different from the idea in Lee *et. al.* [29], where the user and item similarities are calculated from their latent factors that are obtained via Matrix Factorization (MF) over R. By doing the CF-based user and item similarities, we can avoid unnecessary complications related to the non-convex properties of typical MF approaches and their high computational costs.

Given the similarities, all the user-item pairs are weighted by their similarities with the anchor pair (u^*, i^*) as follows, Each local model is trained using the selected data $R^{\mathcal{K}_{ui}}_{u^*,i^*}(u', i')$ defined as follows,

$$R^{\mathcal{K}_{ui}}_{u^*,i^*}(u', i') = \mathcal{K}_{ui}((u', i'), (u^*, i^*))R(u', i'), \tag{4.8}$$

where $R^{\mathcal{K}_{ui}}_{u^*,i^*}$ has values in $[0, 1]$ (i.e., any floating values between 0 and 1), and will be used for local model training. This data selection method is referred to as $\mathtt{S}_{\mathcal{K}_{ui}}$.

Popularity-based Training Data Selection

In this method, we select the user-item pairs such that the selected users/items have similar popularities as the anchor user/item. The user popularity is defined as the number of items that the user has purchased, and the item popularity is defined as the number of users who have purchased the item. For each anchor pair (u^*, i^*), we first select $\alpha\%$ of all the users who have the closest but lower or higher popularity than u^*, respectively. From the selected users, we select $\alpha\%$ of all the items that have the closest but lower or higher popularity than i^*, respectively. The interactions between the selected users and items will be used as training data. This training data selection method is referred to as $\mathtt{S}_{\mathcal{P}_{ui}}$.

4.3.3 Model Combination and Recommendation Generation

After training a `SLIM` model on each of K selected training datasets with respect to K anchor pairs, we ensemble the model results or the models themselves in order to produce recommendations.

Model Result Aggregation

Each local model first produces a recommendation list. Each of the items that has ever appeared in any of the K recommendation lists is then scored. These items are re-ranked using the scores into a new ranked list and the top-N items in the new list will be recommended. This result-aggregation based method is referred to as `MRA`.

To score the items, we use the following two approaches. The first one is the Borda [33] approach, which scores each item using the sum of their ranking positions from all the recommendation lists. The Borda scoring approach is denoted as $\mathtt{C}_{\mathtt{r}}$.

The second scoring approach is to use the weighted sum of recommendation scores from all the recommendation lists. In specific, the score of a user u on an item i,

denoted as $\tilde{\tilde{r}}_{u,i}$, is calculated using Equation 4.4 as in `LLORMA`. This scoring approach is denoted as $\texttt{C}_\texttt{s}$.

Linear `SLIM` Model Ensemble

Instead of combining recommendation results from local models, we can also combine models directly. In the case of `SLIM`, the coefficient matrices from local models are linearly combined as follows:

$$W^e_{i,j} = \frac{1}{|\{k|W^k_{i,j} \neq 0, k = 1, \cdots, K\}|} \sum_{k=1}^{K} W^k_{i,j}, \tag{4.9}$$

where W^k is the k-th model (coefficient matrix) with respect to the k-th anchor pair, $\{W_{i,j}|W^k_{i,j} \neq 0, k = 1, \cdots, K\}$ is the set of coefficients in which $W^k_{i,j} \neq 0$, $|\cdot|$ is the cardinality of a set , and W^e is the ensembled model (coefficient matrix). We use W^e to produce recommendations as in Equation 4.1.

Note that only a certain portion of W^k that corresponds to the selected training items can have non-zero values. Thus, the linear combination of multiple W^k's resembles using multiple small plates to approximate a manifold. Thus, it may represent non-linear relations among items. This method is referred to as `LSME`.

4.4 Materials

4.4.1 Datasets

We evaluated different methods on a benchmark dataset ML100K[1], and its sparsified versions. From the original dataset (referred to as ML100K-1), we generated three sparsified datasets (referred to as ML100K-2, ML100K-3 and ML100K-4, respectively). The first sparsified dataset MK100K-2 is generated by randomly selecting 50% of purchases from ML100K-1, The second/third sparsified dataset ML100K-

[1]https://grouplens.org/datasets/movielens/

3/ML100K-4 is generated by randomly selecting 50% of purchases from ML100K-2/ML100K-3. Table 4.1 represents the dataset summaries.

Table 4.1.: Datasets Used in Evaluation

dataset	#users	#items	#ratings	rsize	csize	density
ML100K-1	943	1,682	100,000	106.05	59.45	6.30%
ML100K-2	943	1,682	49,760	52.77	29.58	3.14%
ML100K-3	943	1,682	24,647	26.14	14.65	1.55%
ML100K-4	943	1,682	12,086	12.82	7.19	0.76%

Columns of "#users", "#items" and "#ratings" represent the number of users, items and ratings in the datasets, respectively. Columns of "rsize" and "csize" represent the average number of ratings for each user and each item, respectively. Column of "density" represents the density of each dataset (i.e., density = #ratings/(#users × #items)).

4.4.2 Evaluation Methodology and Metrics

We applied 5-time Leave-One-Out cross validation (LOOCV) to evaluate the performance of different methods. In each run, one of the purchases of each user is randomly selected into the testing set, and the remaining purchases are used in the training set. For SLIM, we search the parameters in the following ranges. For ML100K-1, the range of λ is {0.00001,0.0001,0.001,0.01,0.1,1,2,5}, the range of β is {0.1,1,2,5,10,15}. For ML100K-2, the range of λ is {0.001,0.01,0.1,1,2,5}, the range of β is {5,10,15,20,25}. For ML100K-3, the range of λ is {0.001,0.01,0.1,1,2,5}, the range of β is {5,10,15,20,25}. For Local SLIM with combined Models(LSM) based on popularity, we find the best performance in the following parameters. For ML100K-1, the range of λ is {2,5,10}, the range of β is {0.01,0.1,1}, the range of n (number of local models) is {10,20,50,60}, the range of bandwidth is {0.3,0.4}. For ML100K-2, the range of λ is {0.0001,0.001,0.01,0.1,1,2,5}, the range of β is {5,10,15,20}, the range of n (number of local models) is {50,60,70}, the range of bandwidth is {0.3,0.4}. 'For ML100K-3, the range of λ is {0.0001,0.001,0.01,0.1,1}, the range of β is {2,5,10,15}, the range of n (number of local models) is {50,60,70}, the range of bandwidth is {0.3,0.4}. The performance of LSM based on popularity and LSIM on the above datasets is shown in Table 4.2. For each user, a size-N ($N = 10$ by default) ranked

list of items is recommended from the ensemble model trained using the training set. The evaluation is performed by comparing the recommendations for each user and the left-out item of that user in the testing set. We use Hit Rate (HR) and the Average Reciprocal Hit-Rank (ARHR) [11] as the evaluation metrics. HR is defined as the rate of correctly recommended items, that is,

$$\text{HR} = \frac{\#\text{hits}}{\#\text{users}}, \tag{4.10}$$

where #users is the total number of users in the testing set, and #hits is the number of users who have their testing items correctly recommended (i.e., hit). ARHR is a weighted version of HR defined as follows:

$$\text{ARHR} = \frac{1}{\#\text{users}} \sum_{i=1}^{\#\text{hits}} \frac{1}{p_i} \tag{4.11}$$

where if an item of a user is hit, p_i is the position of the item in the ranked recommendation list. Thus, ARHR measures how strongly an item is recommended, in which the weight is the reciprocal of the hit position in the recommendation list. Higher HR and ARHR values indicate better performance.

4.5 Experimental Results

4.5.1 Overall Performance

Table 4.2 presents the best performance of the methods on the four datasets. `SLIM` outperforms other methods on ML100K-1 in HR. However, when the datasets become sparser, `LSME` and `MRA` outperform `SLIM`. In specific, `LSME` with $\mathtt{S}_{\mathcal{P}_{ui}}$ as the training data selection method outperforms `SLIM` on ML100K-2 at 1.89%, and on ML100K-3 at 2.97%. `MRA` with $\mathtt{S}_{\mathcal{K}_{ui}}$ and $\mathtt{C}_\mathtt{s}$ outperforms `SLIM` on ML100K-2 at 1.26%, on ML100K-3 at 14.9% and on MK100K-4 at 18.4%. This demonstrates that the

Table 4.2.: Performance Comparison

dataset	SLIM				LSME-S$_{\mathcal{P}_{ui}}$					
	β	λ	HR	ARHR	$\alpha(\%)$	n	β	λ	HR	ARHR
ML100K-1	10	1e-3	**0.339**	0.154	40.0	50	1e-1	5e-0	0.326	0.142
ML100K-2	20	1e-0	0.159	0.056	30.0	60	10	1e-0	**0.162**	0.057
ML100K-3	20	1e-1	0.101	0.036	40.0	50	10	1e-1	0.104	0.036
ML100K-4	25	1e-3	0.049	0.016	40.0	60	10	1e-4	0.047	0.014

dataset	MRA-S$_{\mathcal{K}_{ui}}$-C$_\mathtt{r}$					MRA-S$_{\mathcal{K}_{ui}}$-C$_\mathtt{s}$				
	n	β	λ	HR	ARHR	n	β	λ	HR	ARHR
ML100K-1	50	1e-1	1e-7	0.255	0.097	20	1	1e-5	0.273	0.122
ML100K-2	50	5	1e-2	0.106	0.034	80	2	1e-5	0.161	0.057
ML100K-3	50	15	1e-1	0.093	0.029	20	10	1e-2	**0.116**	0.040
ML100K-4	70	20	1e-1	0.046	0.013	5	25	1e-4	**0.058**	0.020

Columns of "β" and "λ" present the parameters for the local SLIM models. Column of "n" represents the number of local models. Column of "$\alpha(\%)$" represents the percentage of users/items selected for training. Columns of "HR" and "ARHR" present the hit rate and average reciprocal hit-rank, respectively. LSME-S$_{\mathcal{P}_{ui}}$, MRA-S$_{\mathcal{K}_{ui}}$-C$_\mathtt{r}$ and MRA-S$_{\mathcal{K}_{ui}}$-C$_\mathtt{s}$ represent the combinations of different model ensemble, training data selection and recommendation combination schemes. **Bold** numbers are the best performance in terms of HR for each dataset.

ensembled based methods are superior in learning from sparser datasets for top-N recommendation.

Among the four methods, MRA-S$_{\mathcal{K}_{ui}}$-C$_\mathtt{r}$ has the worst performance overall but MRA-S$_{\mathcal{K}_{ui}}$-C$_\mathtt{s}$ has the best. The difference may stem from the recommendation result scoring and combination scheme C$_\mathtt{r}$ and C$_\mathtt{s}$. The C$_\mathtt{r}$ method scores recommendations based on their positions in multiple ranked lists, and thus treats the multiple local models and their recommendations equally. The C$_\mathtt{s}$ scores recommendations using a weighted sum of their respective recommendation scores from local models, and therefore is able to differentiate local models based on their recommendation qualities.

The LSME does not perform as well as MRA based methods. This may be due to the fact that when LSME combines multiple local models as in Equation 4.9, the qualities of local models and their respective significance are not considered. We will investigate this aspect in our future work.

4.6 Discussions and Conclusions

4.6.1 Computational Consideration

Table 4.3 presents the running time of SLIM and LSME with respect to their best performance on the four datasets. For LSME and MRA based methods, training multiple local models is computationally expensive. However, the training process can be trivially paralleled, that is, each local model can be trained independently and in parallel with others. In addition, the training process for each local model is in principle faster than the baseline SLIM model over a same dataset. This is because each local model either has smaller values (e.g., selected by $\mathtt{S}_{\mathcal{K}_{ui}}$) or has fewer training data (e.g., selected by $\mathtt{S}_{\mathcal{P}_{ui}}$). Thus, the model training for LSME and MRA based models can be even faster than SLIM. For example, for ML100K-1, SLIM takes 75.47 seconds for model training, but a parallel implementation of MRA-$\mathtt{S}_{\mathcal{K}_{ui}}$-$\mathtt{C}_{\mathtt{s}}$ could take 26.72 seconds.

4.6.2 Parameter Selection

In principle, each local model should have its own optimal parameters. However, this will lead to a huge set of parameters that each LSME and MRA based models need to identify. To avoid this complexity, we use the same parameters for all the local models. We will investigate heuristics to identify optimal parameters for local models and thus further improve the performance of LSME and MRA.

4.6.3 Conclusions

We developed multiple LSME and MRA based methods to build local SLIM models and ensemble local models for better top-N recommendation. To select training data for each local model, we developed $\mathtt{S}_{\mathcal{K}_{ui}}$ and $\mathtt{S}_{\mathcal{P}_{ui}}$ methods, which select training data based on user-item similarities and popularities with respect to anchor pairs, respective. To combine local models, we developed LSME, which combines models (co-

efficient matrices in local SLIM models) in a linear fashion, and MRA, which combines local model recommendations based on recommendation orders and scores, respectively. Our experiments demonstrate significant improvement from such ensemble models particularly on sparse datasets.

Table 4.3.: Running Time SLIM and LSM

dataset	SLIM			LSME				
	β	λ	Time(second)	β	λ	n	$\alpha\%$	Time(second)
ML100K-1	10	1e-3	75.47	1e-1	5e-0	50	40.0%	7802.01/156.0402
ML100K-2	20	1e-0	14.79	10	1e-0	60	30.0%	281.53/4.6921
ML100K-3	20	1e-1	6.06	10	1e-1	50	40.0%	170.91/3.4182
ML100K-4	25	1e-3	1.82	10	1e-4	60	40.0%	68.98/1.1496

Columns corresponding to β and λ present the parameters for the corresponding method (i.e., ℓ_1-norm regularization parameter β and ℓ_1-norm regularization parameter λ for the underling SLIM models). Column corresponding to n represents the number of local models. Column corresponding to $\alpha\%$ represents the percentage of users/items selected for training. Columns corresponding to Time(second) is the total cpu time in seconds.

5. SUMMARY

In this dissertation, I have developed a novel ***D***ynamic and ***m***ulti-***C***ollaborative ***F***iltering method (*DmCF*) to improve information retrieval from electronic health records. I also have developed local sparse linear model ensemble to tackle both the data sparsity and the user/item heterogeneity issues for top-N recommendation.

When physicians review patient information in health information technology systems, most of them suffer from information overload because of the huge amount of available information about individual patients. To help improve the information retrieval from electronic health records, I have developed a novel hybrid dynamic and multi-collaborative filtering method. I tackled the problem of recommending the next search term to a physician while the physician is searching for information about a patient. The developed hybrid method considers search dynamics and multiple similarities for the next search term recommendation. It consists of two scoring components. The first component is designed to address search dynamics through a first-order Markov Chain. The second component is to score search terms based on similarities via multi-collaborative filtering. *Multi-collaborative filtering* (*mCF*) refers to that multiple types of similarities (e.g., physician similarities, patient similarities and information similarities) are integrated. I have developed ph***y***sician-***p***atient-similarity-based ***C***ollaborative ***F***iltering, and denoted as *ypCF*. Two approaches to identify the set of similar physicians and the set of similar patients have been developed for *ypCF*. The first approach is Patient-First Similarity Identification (*simP2Y*). In this approach, a set of similar patients is first identified and then the set of similar physicians is identified. The second approach is Physician-First Similarity Identification (*simY2P*). In this approach, a set of similar physicians is first identified and then the set of similar patients is identified. I also have developed ***T***ransition-involved ***p***atient-***t***erm-similarity-based ***C***ollaborative ***F***iltering (*TptCF*).

TptCF aggregates from all similar patients the transitions from the last search term in a sequence to another search term. The underlying assumption is that similar patients stimulate similar patterns of search sequences. I tested this new method using real electronic health record data from the Indiana Network for Patient Care. The experimental results demonstrated that for 46.7% of testing cases, this new method is able to correctly prioritize relevant information among top-5 recommendations that physicians are truly interested in.

To tackle both the data sparsity and the user/item heterogeneity issues for top-N recommendation, I have developed local sparse linear method ensemble. To determine the local training data of each local model, an anchor user-item pair is selected for each local model, and I have developed similarity-based training data selection method and popularity-based training data selection based on the selected anchor user-item pair. The similarity-based selection method weights all training user-item pairs by their similarites with respect to the anchor user-item pair. The popularity-based selection method selects the training user-item pairs such that the selected users/items have similar popularities as the anchor user/item. I have employed sparse linear method `SLIM` model in each selected local training dataset. Three approaches to ensemble the model results or the models themselves have been developed. In the first approach, I have aggregated the model prediction results directly by using Borda list combination method because each local model prediction result is a recommendation list of items. In the second approach, I have used the weighted sum of recommendation scores from all local model recommendation lists. In the third approach, I have combined the local models in linear approach directly. Each local model is a coefficient matrix, which contains the item-item coefficients. I have evaluated different methods in a benchmark dataset ML100K and its sparsified versions. The experiments demonstrate 18.4% improvement from such ensemble models particularly on sparse datasets.

The hybrid of dynamics and multi-collaborative filtering method is able to produce high quality recommendations that are most relevant to the patients and that are most interested to physicians. The local sparse linear method ensemble achieves

significant improvement particularly on sparse datasets. Both of them demonstrate the effectiveness of using ensemble idea in top-n recommendation.

REFERENCES

REFERENCES

[1] M. Deshpande and G. Karypis, "Item-based top-n recommendation algorithms," *ACM Transactions on Information Systems (TOIS)*, vol. 22, no. 1, pp. 143–177, 2004.

[2] L. Guo, B. Jin, C. Yao, H. Yang, D. Huang, and F. Wang, "Which doctor to trust: a recommender system for identifying the right doctors," *Journal of medical Internet research*, vol. 18, no. 7, 2016.

[3] H. Jiang and W. Xu, "How to find your appropriate doctor: an integrated recommendation framework in big data context," in *Computational Intelligence in Healthcare and e-health (CICARE), 2014 IEEE Symposium on*. IEEE, 2014, pp. 154–158.

[4] Q. Zhang, G. Zhang, J. Lu, and D. Wu, "A framework of hybrid recommender system for personalized clinical prescription," in *2015 10th International Conference on Intelligent Systems and Knowledge Engineering (ISKE)*, Nov 2015, pp. 189 – 195.

[5] L. Duan, W. N. Street, and E. Xu, "Healthcare information systems: data mining methods in the creation of a clinical recommender system," *Enterprise Information Systems*, vol. 5, no. 2, pp. 169–181, 2011.

[6] C. C. Aggarwal *et al.*, *Recommender systems*. Springer, 2016.

[7] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," in *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. AUAI Press, 2009, pp. 452–461.

[8] F. Ricci, L. Rokach, and B. Shapira, *Recommender systems handbook*, 2nd ed. Springer Publishing Company, Incorporated, 2015.

[9] X. Ning, C. Desrosiers, and G. Karypis, "A comprehensive survey of neighborhood-based recommendation methods," in *Recommender Systems Handbook*, F. Ricci, L. Rokach, and B. Shapira, Eds. Boston, MA: Springer US, 2015, pp. 37–76.

[10] Y. Koren, R. M. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *IEEE Computer*, vol. 42, no. 8, pp. 30–37, 2009. [Online]. Available: http://dx.doi.org/10.1109/MC.2009.263

[11] X. Ning and G. Karypis, "SLIM: sparse linear methods for top-n recommender systems," in *Data Mining (ICDM), 2011 IEEE 11th International Conference on*. IEEE, 2011, pp. 497–506.

[12] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, N. Oliver, and A. Hanjalic, "Climf: learning to maximize reciprocal rank with collaborative less-is-more filtering," in *Proceedings of the sixth ACM conference on Recommender systems*. ACM, 2012, pp. 139–146.

[13] S. Zhang, L. Yao, and A. Sun, "Deep learning based recommender system: a survey and new perspectives," 2017.

[14] C. Zhang, K. Wang, H. Yu, J. Sun, and E.-P. Lim, "Latent factor transition for dynamic collaborative filtering," in *Proceedings of the 2014 SIAM International Conference on Data Mining*. SIAM, 2014, pp. 452–460.

[15] N. Sahoo, P. V. Singh, and T. Mukhopadhyay, "A hidden markov model for collaborative filtering," *Mis Quarterly*, pp. 1329–1356, 2012.

[16] J. Z. Sun, K. R. Varshney, and K. Subbian, "Dynamic matrix factorization: a state space approach," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. IEEE, 2012, pp. 1897–1900.

[17] J. Z. Sun, D. Parthasarathy, and K. R. Varshney, "Collaborative kalman filtering for dynamic matrix factorization." *IEEE Trans. Signal Processing*, vol. 62, no. 14, pp. 3499–3509, 2014.

[18] D. Luo, H. Xu, Y. Zhen, X. Ning, H. Zha, X. Yang, and W. Zhang, "Multi-task multi-dimensional hawkes processes for modeling event sequences," in *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, ser. IJCAI'15, 2015, pp. 3685–3691.

[19] L. Xiong, X. Chen, T.-K. Huang, J. Schneider, and J. G. Carbonell, "Temporal collaborative filtering with bayesian probabilistic tensor factorization," in *Proceedings of the 2010 SIAM International Conference on Data Mining*. SIAM, 2010, pp. 211–222.

[20] M. Wiesner and D. Pfeifer, "Health recommender systems: concepts, requirements, technical basics and challenges," *International Journal of Environmental Research and Public Health*, vol. 11, no. 3, pp. 2580–2607, 2014.

[21] Y. Bao and X. Jiang, "An intelligent medicine recommender system framework," in *Industrial Electronics and Applications (ICIEA), 2016 IEEE 11th Conference on*. IEEE, 2016, pp. 1383–1388.

[22] F. Gräßer, H. Malberg, S. Zaunseder, S. Beckert, D. Küster, J. Schmitt, S. K. Abraham, and P. für Dermatologie, "Application of recommender system methods for therapy decision support," in *2016 IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom)*, Sept 2016, pp. 1–6.

[23] P. Moffett and G. Moore, "The standard of care: legal history and definitions: the bad and good news," *Western Journal of Emergency Medicine*, vol. 12, no. 1, p. 109, 2011.

[24] M. H. Lewis, J. K. Gohagan, and D. J. Merenstein, "The locality rule and the physician's dilemma: local medical practices vs the national standard of care," *JAMA*, vol. 297, no. 23, pp. 2633–2637, 2007.

[25] J. R. Norris, *Markov chains.* Cambridge university press, 1998, no. 2.

[26] C. C. Aggarwal and C. Zhai, *Mining text data.* Springer Science & Business Media, 2012.

[27] S. Kabbur, X. Ning, and G. Karypis, "Fism: Factored item similarity models for top-n recommender systems," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '13. New York, NY, USA: ACM, 2013, pp. 659–667. [Online]. Available: http://doi.acm.org/10.1145/2487575.2487589

[28] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on.* Ieee, 2008, pp. 263–272.

[29] J. Lee, S. Bengio, S. Kim, G. Lebanon, and Y. Singer, "Local collaborative ranking," in *Proceedings of the 23rd international conference on World wide web.* ACM, 2014, pp. 85–96.

[30] E. Christakopoulou and G. Karypis, "Local item-item models for top-n recommendation," in *Proceedings of the 10th ACM Conference on Recommender Systems.* ACM, 2016, pp. 67–74.

[31] B. Xu, J. Bu, C. Chen, and D. Cai, "An exploration of improving collaborative recommender systems via user-item subgroups," in *Proceedings of the 21st international conference on World Wide Web.* ACM, 2012, pp. 21–30.

[32] A. Beutel, A. Ahmed, and A. J. Smola, "Accams: Additive co-clustering to approximate matrices succinctly," in *Proceedings of the 24th International Conference on World Wide Web.* ACM, 2015, pp. 119–129.

[33] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar, "Rank aggregation methods for the web," in *Proceedings of the 10th international conference on World Wide Web.* ACM, 2001, pp. 613–622.