DataRank: A Personalized Online Recommendation System for Biomedical Datasets

Arya Iramhr[1], Huan Wang[1], Shuang Wang[1],
Jaideep Vaidya[2], Zhaohui Qin[3], Hwanjo Yu[4], Xiaoqian Jiang[1]
[1]University of California, San Diego, La Jolla, CA 92093
[2]Rutgers, Newark, New Jersey, 07102
[3]Emory University, Atlanta, GA 30322
[4]POSTECH, Pohang, Gyoungbuk, Republic of Korea

Abstract

We propose a novel framework called DataRank to support personalized presentation of biomedical datasets to researchers. DataRank takes the bipartite citation graph (between datasets and papers from PubMed Central) to enrich the features associated with the dataset (e.g., by aggregating MeSH terms from papers in the bipartite citation graph). For each search query, DataRank first maps the "free text query" to a "MeSH query" and yields an initial ranking of datasets for the MeSH query using a Bayesian approach, where the likelihood is proportional to the Jaccard index and the prior is proportional to number of citations of the dataset. We further extended DataRank with an online algorithm by incorporating user-feedbacks regarding ranking relevance. The online algorithm uses the offline DataRank as its prior and computes its likelihood by estimating the user rating for unknown values using collaborative filtering. A demo web search engine has been developed to rank more than 20,000 datasets. http://biocaddieweb.ucsd-dbmi.org:8080/

Introduction

Over the past decades, a wide range of datasets including clinical, gnomic, imaging, behavioral, etc. are created for biomedical research. Although many repositories (e.g. DBGap, TCGA, GEO, etc.) have been established [1–3], a unified and systematic way for retrieving datasets (like querying articles on PubMed) still does not exist. Like many big-data applications, there is a "diminishing return" effect, that is, retrieving of biomedical datasets becomes harder as the number and category of datasets increase. Personalization is extremely important to the user experience for the information retrieval task. There are many success examples in the commercial world, for example, Netflix gains its reputation by suggesting movies to users and Amazon makes good recommendations of books and games to customers [4]. But there is no such a system for personalized recommendation system for biomedical datasets. The current biomedical information systems for data indexing and searching have a number of limitations, including:

1. **Indexing** of datasets rely on manual effort, which is a costly.
2. **Searching** between and within the repositories is not well developed, which often requires users' knowledge on specific datasets.
3. **Integration** of the repositories for consistent search is hard because indexing and maintenance of each repository is done separately.
4. **Ranking** of search results is often done naively based partial matching of the query to the dataset's limited metadata.

To address these challenges, we develop a novel framework called *DataRank* to support personalized presentation of biomedical datasets to researchers. DataRank takes the bipartite citation graph between datasets and articles to generate dataset features as inputs, followed by ranking datasets based on their relevancy to the searching query as well as users' feedback. The framework consists of the following steps, (1) crawling PMC articles to discover datasets citation from papers, (2) **generating** features for identified datasets, e.g., MeSH, number of citations, etc., and (3) **ranking** datasets based on generated features and users' feedback.

The crawling step is non-trivial but it is not the focus of this paper and thus will not be discussed in details. We concentrate on feature generation and dataset ranking. The feature generation step begins with creating

regular expressions for citation rules (learned from NIH data repositories), which discovers more than 20,000 datasets from approximately 1 million PMC full text articles. Based on identfied paper-to-dataset pairs, we created a bipartite graph between dataset and paper identifiers and use them to build a feature vector for each dataset, as illustrated in Figure 1.

The rest of this paper is organized as follows. In the background section, we discuss related work. in the method section, we present our methods for offline and online ranking algorithms. We elaborate implementation remarks and experimental study in sections experiment setup and results, and finally make conclusions and state possible future works in the last section..

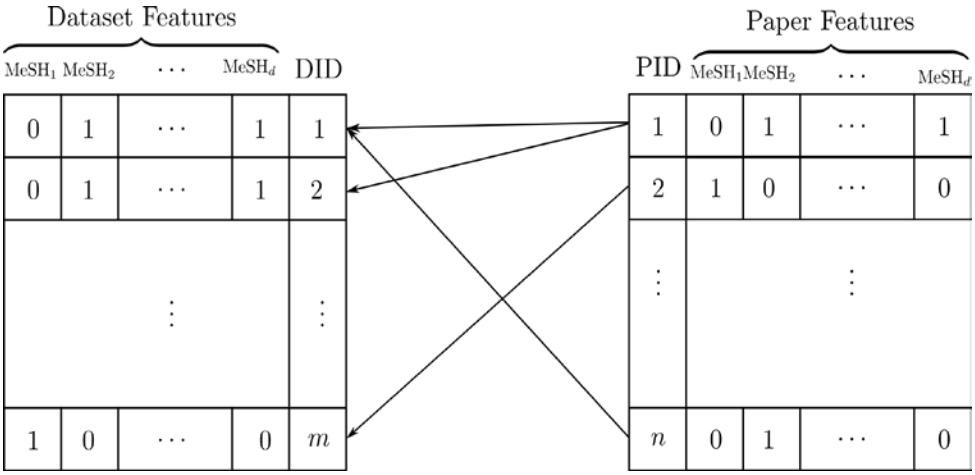


Figure 1: A bipartite graph between m datasets and n papers with MeSH terms as their features. PID and DID are Paper ID and Dataset ID, where existence of an edge between a paper and a dataset implies that dataset is cited in that paper. We propogate the MeSH terms from the paper to the dataset to build a binary vector of feature variales for each dataset.

Background

As repositories of data are getting bigger and diverse, effective information retrieval is getting more challenging. To cope with the growth of repositories, a number of techniques have been developed to improve biomedical information retrieval, including:

- relevance ranking [5], which documents are ranked according to Term-Frequency Inverse-Document-Frequency (TF-IDF) relevance to the free-text query.
- query expansion [6], which the (mappable) tokens in the query mapped to a MeSH term using Automatic Term Mapping (ATM) [7] and query is augmented with a set of MeSH terms,
- relevance feedback [8], which user feedbacks in previous ranking results are used to improve ranking results.

These techniques are limited as they do not personalize the presentaiton of information to users. In the machine learning community, tools and learning algorithms for ranking objects have been intensively studied [9]. For example, given a labeled set (i.e. specifying the rankings for every item of interest) RankSVM [10] can train a nonlinear model to predict the ranking of new items based on the training set. RefMed [8] uses RankSVM to learn a model for each query on PubMed articles by incorporating user feedbacks as "training dataset labels" into the training process. Unfortunately, like other ranking algorithms, RankSVM only works when enough labeled training samples (user feedbacks) are provided to the learning algorithm, which is very unlikely in real world situations, to expect a user to rate all the search results.

An alternative solution is to use data recommendation system [9], which was developed to infer unknwon user ratings from limited observations. For instance, collaborative filtering is a well-known algorithm for estimating user ratings for set of items based on their similarities to set of rated items [11]. Our framework will be the first to combine relevance ranking, query expansion and relevance feedback using a Bayesian approach without requiring a training process.

Methodology

The *DataRank* framework consists of an offline algorithm and an online algorithm. For a new user, the system works in an offline mode, which ranks datasets based on relevancy and popularity (e.g., number of citations). After receiving user feedbacks, it updates the model to present user-specific rankings. We incorporate user feedbacks explicitly by giving an option to rate search results. We will first describe the features, explain our offline ranking algorithm, and finally discuss the online ranking algorithm.

Features:

For each dataset, we generate its corresponding features using the set of MeSH terms from its citing articles. The corpus of n articles is represented by a $d \times n$ binary matrix, $M \in \{0,1\}^{d \times n}$, which d is the number of MeSH terms and n is the number of articles. An element M_{ij} of the matrix has a value one if the i^{th} MeSH term is associated with the j^{th} article, otherwise, the value is zero. Similarly, we define a matrix of $X \in \{0,1\}^{d \times m}$ to represent m datasets via bag-of-words of MeSH terms, where $\mathbf{x}_i$ is the i^{th} column of X. We use $\mathbf{y} \in \{1, \dots, m\}^m$ to denote the labels (identifiers) of datasets, which is a m-dimensional vector with elements permuted from labels $1 \dots m$.

We store the bipartite graph between articles and datasets using an adjacency matrix $A \in \{0,1\}^{n \times m}$, which 1 indicates the j^{th} article cited the k^{th} dataset, and 0 indicates there is not a citation relationship. For simplicity, we consider binary features for both articles and datasets. Therefore, dataset features can be readily obtained from the article features and adjacency matrix:

$$X = \min(M * A, 1) \quad (1)$$

where $\min(\cdot)$ is a elementwise operator over its matrix argument. This is just to propagate MeSH terms from articles to datasets.

Offline Ranking:

We propose a probabilistic approach and use a graphical model to specify dependencies between random variables, as illustrated in Figure 2. Noet that $\mathbf{x}$ and $\mathbf{y}$ represent random variables corresponding to dataset features and dataset labels, respectively. We also introduce another random variable $\mathbf{q}$ for search queries over the same sample space as $\mathbf{x}$, i.e., MeSH terms $\mathbf{q} \in \{0,1\}^d$. Finally, $\mathbf{c}$ is an (observed) random variable that defines a prior over the labels $\mathbf{y}$. The graphical model is shown in Figure 2-(a). It should be noted that variables $\mathbf{x}$ (dataset features) and $\mathbf{c}$ (datasets prior) are observed and we do not need their marginal distributions and therefore we only consider the query $\mathbf{q}$ as the evidence.

In this model, the problems of ranking for a given query $\mathbf{q}$ is to find the posterior distribution for the dataset labels given evidences. More precisely, the posterior distribution

$$Pr(\mathbf{y}|\mathbf{q}, \mathbf{c}, \mathbf{x}) \propto Pr(\mathbf{y}|\mathbf{c}) Pr(\mathbf{q}|\mathbf{x}) \quad (2)$$

specifies the ranking, where the posterior distribution is expanded according to the graphical model, see Figure 2-(a). Here, we can consider $Pr(\mathbf{y}|\mathbf{c})$ as dataset-prior and consider $Pr(\mathbf{q}|\mathbf{x})$ as the query-likelihood. The next step is to specify dataset-prior and query-likelihood distributions.

Query-likelihood

Since the binary feature vector $\mathbf{x}$ and query $\mathbf{q}$ are representation of binary sets, using Vann diagram, we can easily compute the likelihood

$$Pr(\mathbf{q}|\mathbf{x}) = \frac{Pr(\mathbf{q},\mathbf{x})}{Pr(\mathbf{x})} = \frac{|\mathbf{q} \cap \mathbf{x}|}{|\mathbf{x}|} \quad (3)$$

where the set operation $\cap$ are applied to the set representation of the binary vectors.

However, this probability does not account for mismatches in partial matching of terms in the query and features of each dataset. This leads to the phenomena that two queries with the same number of matches but different number of mismatches have the same probability. To rectify this problem, we can include the number of mismatches to the denominator, which is equivalent to condition on $\mathbf{q} \cup \mathbf{x}$

$$\widehat{Pr}(\mathbf{q}|\mathbf{x}) \propto Pr(\mathbf{q}|\mathbf{q} \cup \mathbf{x}) = \frac{|\mathbf{q} \cap \mathbf{x}|}{|\mathbf{q} \cup \mathbf{x}|} \quad (4)$$

the value of (5) is known as Jaccard index or Tanimoto coefficient [12]. For the query likelihood, we have

$$Pr(\mathbf{q}|\mathbf{x}) = \frac{\widehat{Pr}(\mathbf{q}|\mathbf{x})}{\sum_{i=1}^{m} \widehat{Pr}(\mathbf{q}|\mathbf{x}_i)} \quad (5)$$

Dataset-prior:
We propose a prior for the datasets. A simple way to think about this is to imagine the posterior distribution in a scenario where there is no evidence available. This is equivalent to say what is the best ranking of datasets when there is no queries. There are many answers to this question, and one way is to sort them based on their general popularity. Interestingly, we can quantitatively specify the popularity for the datasets by taking into account of how many citations they have, i.e., $\mathbf{c} = \mathbf{1}^T A$. More precisely, we can write the prior as

$$Pr(\mathbf{y}|\mathbf{c}) = \frac{\mathbf{c}}{\mathbf{c}^T\mathbf{1}} = \frac{\mathbf{1}^T A}{\mathbf{1}^T A \mathbf{1}} \quad (6)$$

where $\mathbf{1}$ is a vector of one. Note that $\mathbf{1}^T A$ implies $\mathbf{1}^T$ has n dimensions and $\mathbf{1}^T A \mathbf{1}$ implies $\mathbf{1}$ has n dismensions.

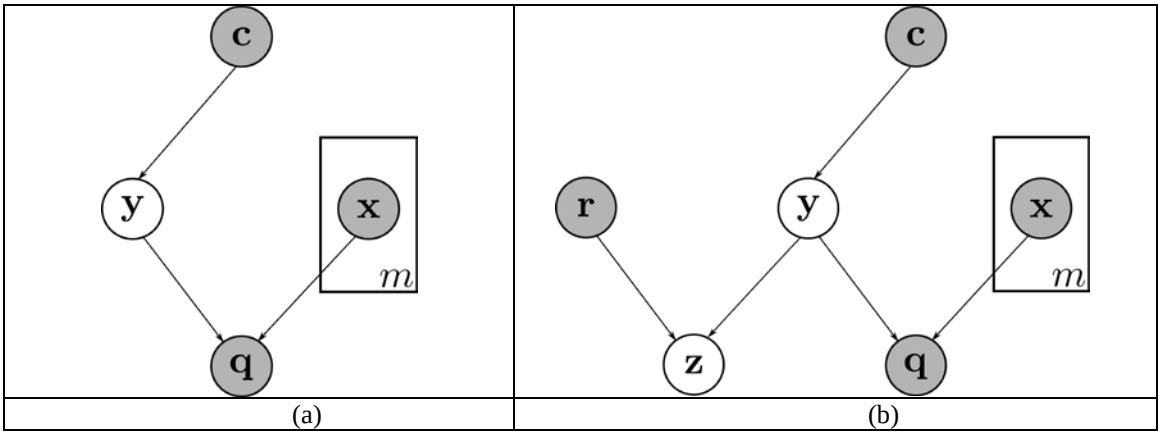


Figure 2: Probabilistic graphical models for offline (a) and online (b) methods for ranking datasets. The shaded nodes are observed variable, arrow implies conditional dependence and rectangles are short hand for replication of the inside nodes. For details please see text.

So far, we have specified the prior, the likelihood, and hence the posterior for the offline model. The ranking of datasets is based on sorting datasets decreasingly according to their posterior distribution. Next, we will introduce how to develop an online ranking algorithm.

Online Ranking: We will extend the offline *DataRank* to the online setting by incorporating user feedback into the final ranking. For this purpose, we propose a new model, Figure 2-(b), which introduces *incomplete* user ratings $\mathbf{r} \in \{\mathbf{0} \cdots \boldsymbol{k}\}^{\boldsymbol{m}}$, and the dataset online-labels $\mathbf{z}$, where $\boldsymbol{k}$ is the number of different state of the ratings (e.g., 1-5 scale in our case) and 0 is used to denote an unknown value in the rating. Note that the word "incomplete" must means that a user only needs to provide ratings to some search results isntead of all of them. In the ranking process, ratings $\mathbf{r}$ are initialized with zeros, and at each epoch $\boldsymbol{t}$, users can rate the search results and update the $\mathbf{r}$. As shown in the graphical model, the online-labels depend on user feedback but (offline) dataset labels $\mathbf{y}$ do not.

Similar to offline method, the task is to compute the posterior

$$Pr(\mathbf{z}|\mathbf{r}, \mathbf{y}, \mathbf{c}, \mathbf{q}, \mathbf{x}) \propto Pr(\mathbf{z}|\mathbf{r}) Pr(\mathbf{y}|\mathbf{c}, \mathbf{q}, \mathbf{x}) \quad (7)$$

where the factorization induced by the graphical model Figure 2-(b) and the evidence is incomplete user rating $\mathbf{r}$. We can treat the offline posterior $Pr(\mathbf{y}|\mathbf{c}, \mathbf{q}, \mathbf{x})$ as prior in this model, which implies that without any evidence, user ratings, the online posterior is exactly equal to its prior, i.e., offline posterior.

Because the ratings $\mathbf{r}$ is incomplete (i.e., containing zero values), we first need to estimate the unknown values of user ratings in order to specify the likelihood $Pr(\mathbf{z}|\mathbf{r})$. Once we estimated all the unknown values in the user rating vector $\mathbf{r}$, we can readily compute the likelihood $Pr(\mathbf{z}|\mathbf{r})$ by normalizing $\mathbf{r}$. We use collaborative filtering [13] to estimate unknown values of user ratings for datasets that have not yet been rated. Collaborative filtering methods work by constructing a similarity matrix between items (datasets), and defining a method for finding a set of neighbors for each item, and then computes the rating of undated

items as weighted linear combination of its rated neighbours, for which the weights are proportional to the similarity between items.
Because the rating vectors are extremely sparse, we choose to define the method for finding neighbors to return all rated items, to use all the user rating information. The key step in collaborative filtering is to define a similarity measure between datasets. Regarding, the binary MeSH representation of datasets, we opt to use Tanimoto kernel [14] between datasets for measuring similarity between datasets

$$K_{i,j} = K(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{k}_\cap(\mathbf{x}_i, \mathbf{x}_j)}{\mathbf{k}_\cap(\mathbf{x}_i, \mathbf{x}_i) + \mathbf{k}_\cap(\mathbf{x}_j, \mathbf{x}_j) - \mathbf{k}_\cap(\mathbf{x}_i, \mathbf{x}_j)} \qquad (8)$$

where K is a $m \times m$ symmetric positive definite matrix, $\mathbf{k}_\cap(\mathbf{x}_i, \mathbf{x}_j) = \frac{|\phi(\mathbf{x}_i) \cap \phi(\mathbf{x}_j)|}{|\phi(\mathbf{x}_i) \cup \phi(\mathbf{x}_j)|}$ is intersection kernel [15] between two datasets $\mathbf{x}_i$ and $\mathbf{x}_j$, and $\phi(\cdot)$ is a general nonlinear feature function.

With a similarity matrix between datasets, we can readily fill the unknown values in the completed rating vector $\mathbf{R}$ using collaborative filtering,

$$\mathbf{R}_i = \begin{cases} \mathbf{r}_i, & \mathbf{r}_i \neq 0 \\ \mathbf{r}^T K_i, & \mathbf{r}_i = 0 \end{cases} \qquad (9)$$

where $\mathbf{r}_i$ stands for the rating to the ith dataset and K_i is the i^{th} column of the kernel matrix. After computed the completed rating vector, to compute the likelihood, we only need to normalize $\mathbf{R}$,

$$Pr(\mathbf{z}|\mathbf{r}) = \frac{\mathbf{R}}{\mathbf{1}^T\mathbf{R}} \qquad (10)$$

Using (5) and (10) we can fully specify the online prediction posterior (7) and therefore, prroduce the online ranking for the *DataRank* system.

Experimental Setup

Implementation:

DataRank is deployed on an Ubuntu 13.04 system with Intel Xeon X5650 2.67GHz and 4 GB RAM. Nginx 1.4.6 is used for static request and django 1.7.4 is used for dynamic request. Uwsgi is used as an application server hosting django models, which is also used as an interconnector between nginx and django. Nginx is chosen for a consideration of future high traffic. A django/python combination is highly extensible for future recommendation algorithm research with assistance of buildin pacakges like numpy and scipy.

System Architecture:

A brief system architecture is shown below. The abstract architecture is a standard model-view-controller (MVC) model.

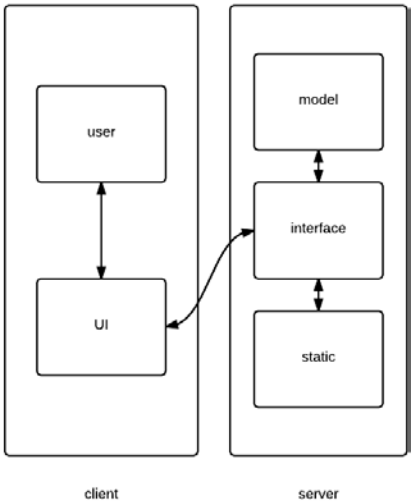


Figure 3: High level system components

All user interaction with UI is passed to interface layer for further response. All static request is passed to nginx and will be responsed immediately. Dynamic request is passed to model layer. After processing and computation, a response will be given back to user via same path.

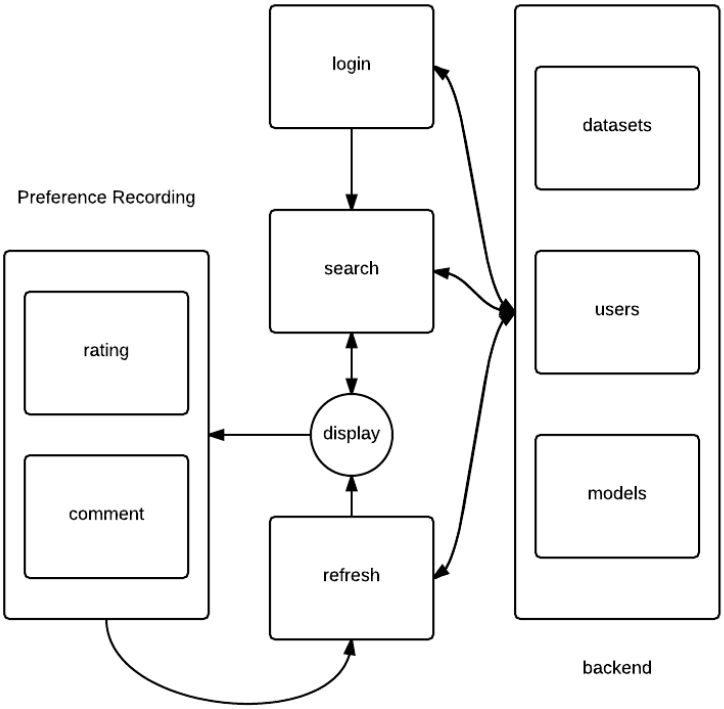


Figure 4: More detailed workflow of the system

At system entry, there is a login management component (see Figure 5). Users could register, login or stay anonymously. When user either registered or logined, this user will be given a unique identifier. For registered user, all history action could be retrieved for recommendation and all new actions will be recorded as preferences. The search component stays active. Users can insert one or a combination of keywords, a list of datasets will be shown on display via an offline ranking algorithm. Keywords will be passed to backend and be processed by a set of function and algorithms, and then a list of datasets is given with a keywords related order. The algorithm was introduced in the method section.

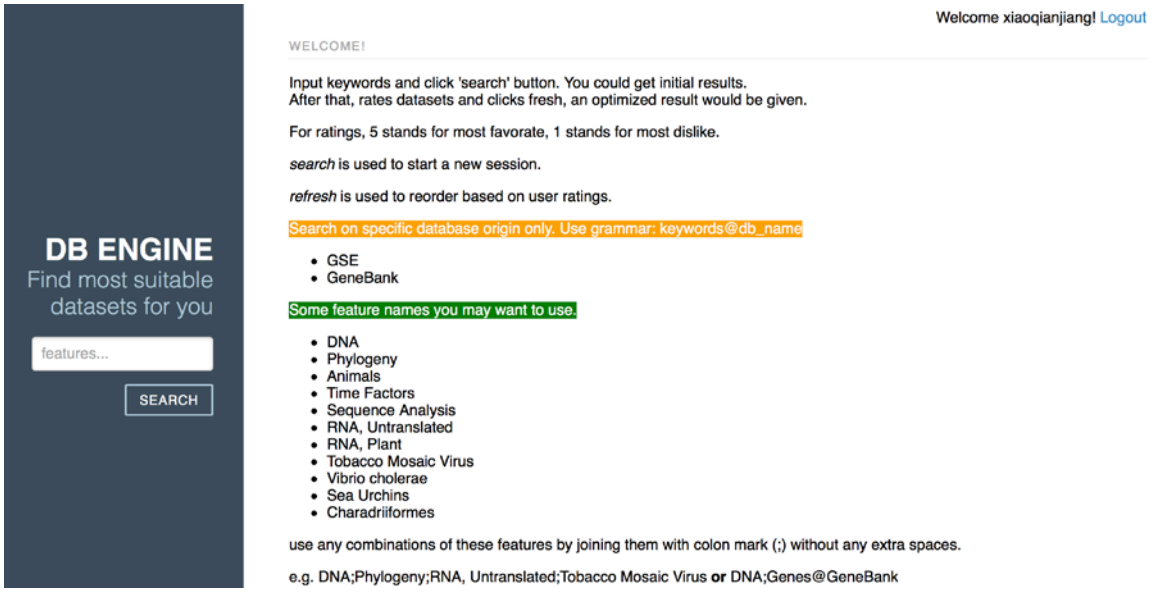


Figure 5: An illustration of the DataRank interface

The display component serves as an important connector between different models and functions. Users give rates and make comments through this component to interact with the system. When new user-related information is added, a re-ranking component is activated. It will pass all new infomation to the backend, and an online learning algorithm will collect user historic and current preferrences and give a new ranking based on updated information. This infomation will be shown on display component. We will elaborate on searching, re-ranking, and rating components of the DataRank system, as illustrated in Figure 6.

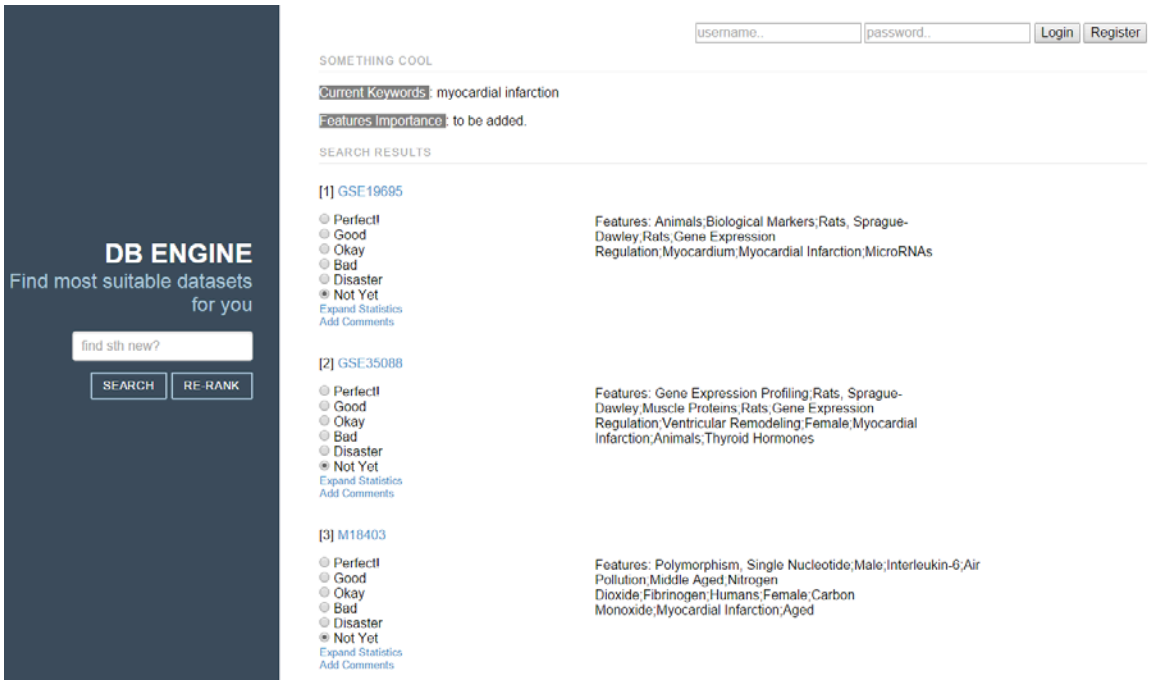


Figure 6: An illustration of the DataRank display components

Searching Pattern:

The searching bar is the first function component faced by users. It could support several basic patterns. Current prefered keywords by the system are MeSH terms. However, there are punctuations included in existing MeSH terms text. We choose semicolon as splitter. When a string of keywords is passed to backend component, a grammar parser component is activated. it first changes keywords to lower letters, splits keywords by semicolons, and then remove travial characters for each keyword. A set of cleaned keywords will be passed to next phase for further processing.

Respostory specific searching is supported. For example, users may have a target data source. Therefore, supporting a data resource fixed searching pattern is necessary. We use an at sign (@) on the end of keywords, and dataset name is given before it. This dataset name is cached, and all future re-ranking will only occur under this scope. A simple example is (DNA;Genes@GeneBank).

Re-ranking:

Re-ranking is one of the most important features in this system. Re-ranking is mostly session based. A session is define as the a period of actions with one fixed query. If a new query is requested, a new session will be created. In a session, users could give ratings and update preferences for unlimited time. Everytime, when new information is given, an update of recommendation will be activated. A standard session work flow is: a user input keywords, and a list of datasets are given. After some judgement, a user gives ratings to some of them and trigger the ranking component. After relearning user's preferences, a list of re-ranked data will be returned.

Ratings and Comments:

Ratings and comments are two important features for online learning. As users may change their ideas

frequently, ratings are stored in a browser session, which provides better security than traditional cookies. Those ratings will be sent to the server, when the re-ranking action is triggered. Each rating is a pair of dataset ID and rating score. Comments are also passed to server side with the dataset ID, and user related information will be included later. All stored ratings and comments are associated with time information, user information, and keywords that were searched. All this information could be used for future ranking algorithm development.

Assistant Information:
In order to help users evaluate and understanding datasets orders better, we offer several parameters, such as posterior likelihood value (online ranking results) and raw prior probability. Counts of citations are also provided. All these statistics could be used as an compliment of metadata to assist users.

Experiment Design and Results

We will perform two set of experiments to evaluate if: (1) *DataRank* learns about users' preference through soliciting ratings; (2) *DataRank* provides user-specialized results. In the first set of experiment, we aim to show how well *DataRank* predicts user ratings, by comparing *DataRank* with ranking of human subjects. We recruited five human subjects for this experiment. Because the ranking is linearly related to users' rating, we measure the discrepancy using the *loss* between the predicted value of rating and and the actual user rating. Suppose at the t^{th} step a user rates j^{th} dataset, we can compute the loss

$$\mathbf{l}^{(t)} = |\hat{\mathbf{r}}_j^{(t-1)} - \mathbf{r}_j^{(t)}| \qquad (11)$$

where $\hat{\mathbf{r}}$ is the predicted user rating in previous step. Since the loss is quite instantaneous we smooth it by defining *regret* [16] as average loss until know

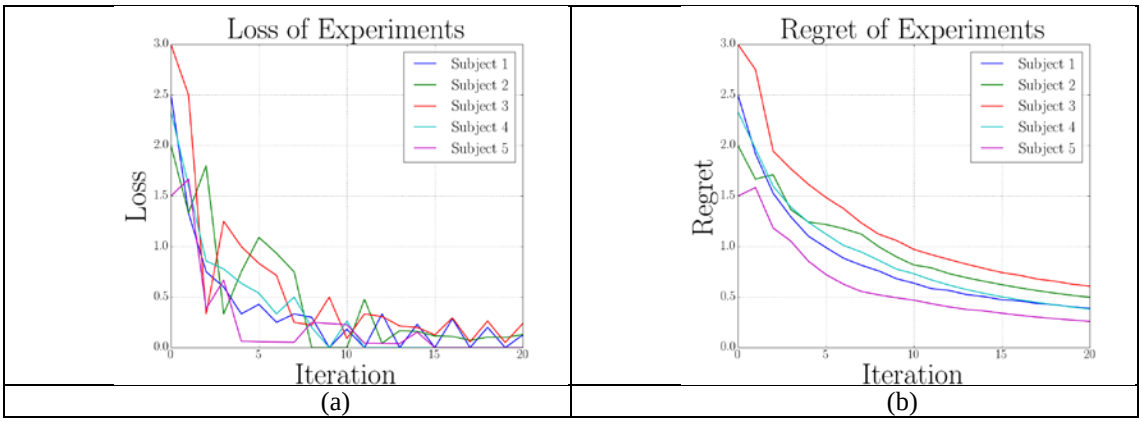
$$\mathbf{R}^{(t)} = \frac{1}{t}\sum_k^t |\hat{\mathbf{r}}_j^{(k-1)} - \mathbf{r}_j^{(k)}| \qquad (12)$$

We computed regret for each time for 5 different subjects and the result are shown in Figure 7. It can be seen that regret has a global decreasing meaning throughout time, which means *DataRank* makes more accurate predictions at each iteration.

For the second set of experiments, we ask different subjects search for the same query and rate the results according to their preferences to check if *DataRank* can capture different preferences of individual users. For this case user A and B search for the same query, e.g. "Mice", and $t = 1 \cdots T$ we compute the *Kappa statistic* [17] for measuring disagreement between users. For example we cam compute the ratings between users A and B at each time

$$\kappa_{A,B}^{(t)} = \kappa(\mathbf{r}_A^{(t)}, \mathbf{r}_B^{(t)}) \qquad (13)$$

We run this experiment for 5 subjects using the same query and computed parwise kappa score for all the users and depicted their average throughout time in Figure 8, which shows that even for the same query users tend to have disagreement in opinions.



(a) (b)

Figure 7: Loss and Regret of predicted ratings throughout time.

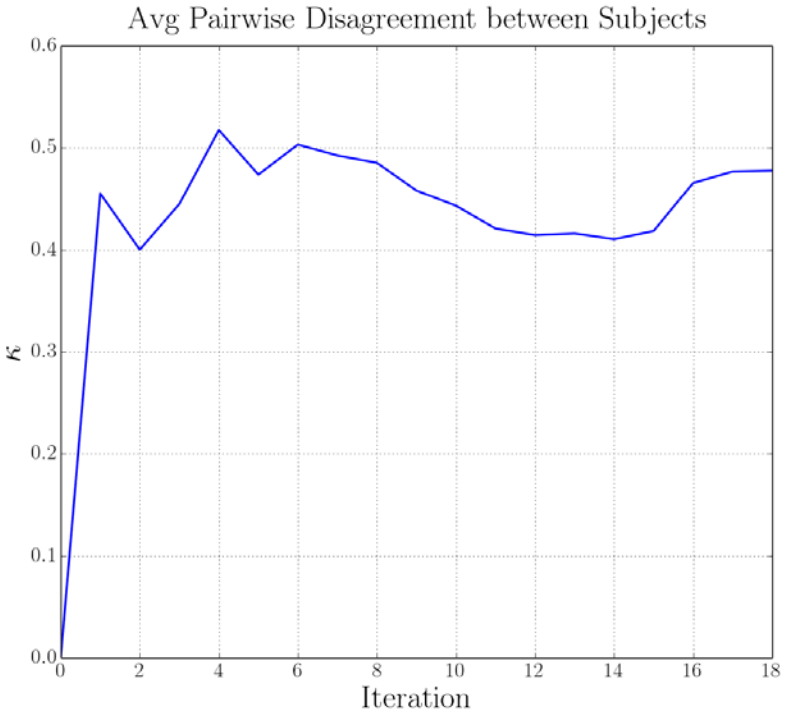


Figure 8: Average disagreement between users.

Discussion and Conclusions

We proposed a collaborative filtering framework *DataRank* using both offline and online algorithms to support personalized presentation of biomedical data. The system can adjust the ranking of individual datasets of interest based on peripheral information and users' feedback. It provides enough flexibility to incorporate information through the interaction with users and can be easily extended with additional features. We tested the system with a few experiements to check its convergence, improvement, and specialization, which demonstrated good usability. It is very encouraging but this pilot project has limitations. We did not consider user-level information, and therefore, the collaborative filtering model adopted here is a naive version. There are also important temporal dependency that are currently ignored in the system, which should be considered in the next version. We would also like to improve our query matching module through integrating MetaMap to identify more accurate information in the first place. Despite these limitations, we found it is very exciting to develop a real biomedical data recommandation system based on state-of-the-art techniques. We will keep the momentum to improve the system and make it a useful tool to the biomedical informatics community.

Acknoledgement: The authors are partially supported by NLM (R00LM011392, R21LM012060) and NIH NIH Big Data to Knowledge (U24AI117966).

References

1 Barrett T, Troup DB, Wilhite SE, *et al.* NCBI GEO: archive for functional genomics data sets--10 years on. *Nucleic Acids Res* 2011;**39**:D1005–10. doi:10.1093/nar/gkq1184

2 Mailman MD, Feolo M, Jin Y, *et al.* The NCBI dbGaP database of genotypes and phenotypes. *Nat Genet* 2007;**39**:1181–6. doi:10.1038/ng1007-1181

3 NIH. NIH Data Sharing Repositories. 2015.

4 Bennett J, Lanning S. The Netflix Prize. In: *KDD Cup and Workshop*. 2007. 35–8. doi:10.1145/1562764.1562769

5 Lu Z, Kim W, Wilbur WJ. Evaluating Relevance Ranking Strategies for MEDLINE Retrieval. *J Am Med Informatics Assoc* 2009;**16**:32–6. doi:10.1197/jamia.M2935

6 Lu Z, Kim W, Wilbur WJ. Evaluation of Query Expansion Using MeSH in PubMed. *Inf Retr Boston* 2009;**12**:69–80. doi:10.1007/s10791-008-9074-8

7 ATM. Automatic Term Mapping. 2015.

8 Yu H, Kim T, Oh J, *et al.* RefMed: relevance feedback retrieval system fo PubMed. In: *Proceedings of the 18th ACM conference on Information and knowledge Management*. 2009. 2099–100.

9 Adomavicius G, Tuzhilin A. Toward the Next generation of Recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Trans Knowl Data Eng* 2005;**17**.

10 Joachims T. Optimizing search engines using clickthrough data. In: *KDD*. 2002.

11 Koren Y. Collaborative filtering with temporal dynamics. In: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '09*. New York, New York, USA: : ACM Press 2009. 447. doi:10.1145/1557019.1557072

12 Manning CD, Raghavan P, Schütze H. *Introduction to information retrieval*. Cambridge University Press 2008.

13 Su X, Khoshgoftaar TM. A Survey of Collaborative Filtering Techniques. *Adv Artif Intell* 2009;**2009**:4:2–4:2.

14 Ralaivola L, Swamidass SJ, Saigo H, *et al.* Graph kernels for chemical informatics. *Neural Networks* 2005;**18**:1093–110.

15 Gärtner T, Le GQ V, Smola AJ. A Short Tour of Kernel Methods for Graphs. 2006.

16 Shai Shalev-shwartz PYS. *Online learning: Theory, algorithms, and applications*. 2007.

17 Viera AJ, Garrett JM. Understanding interobserver agreement: the kappa statistic. *Fam Med* 2005;**37**:360–3.