##### DataRank: A Personalized Online Recommendation system for Biomedical Datasets

Firstname A. Lastname, MD, MPH, Firstname B. Lastname, MD, PhD

Fri Mar 06 20:44:56 2015

## 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 paper from PubMed central) to enrich the features associated with the dataset (e.g., by aggregating MeSH terms from paper in the bipartite citation graph). For each search query, DataRank first maps the "free text query" to a “MeSH query” and yield an initial ranking of datasets for the MeSH query using a Bayesian approach, which the likelihood is proportional to Jaccard index and the prior is proportional to number of citations of that 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 dataset that has been discovered in more than 1 million papers. 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 [1,2], a unified and systematic way for retrieving them 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 [3]. 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, and rank 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 rank them, (3) **recommendating** other datasets besides current search results based on 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 data recommendation. 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 method section, we present our methods for offline and online ranking algorithms. We elaborate implementation remarks and experimental study in section 3, and finally make conclusions and state possible future works in section 4.

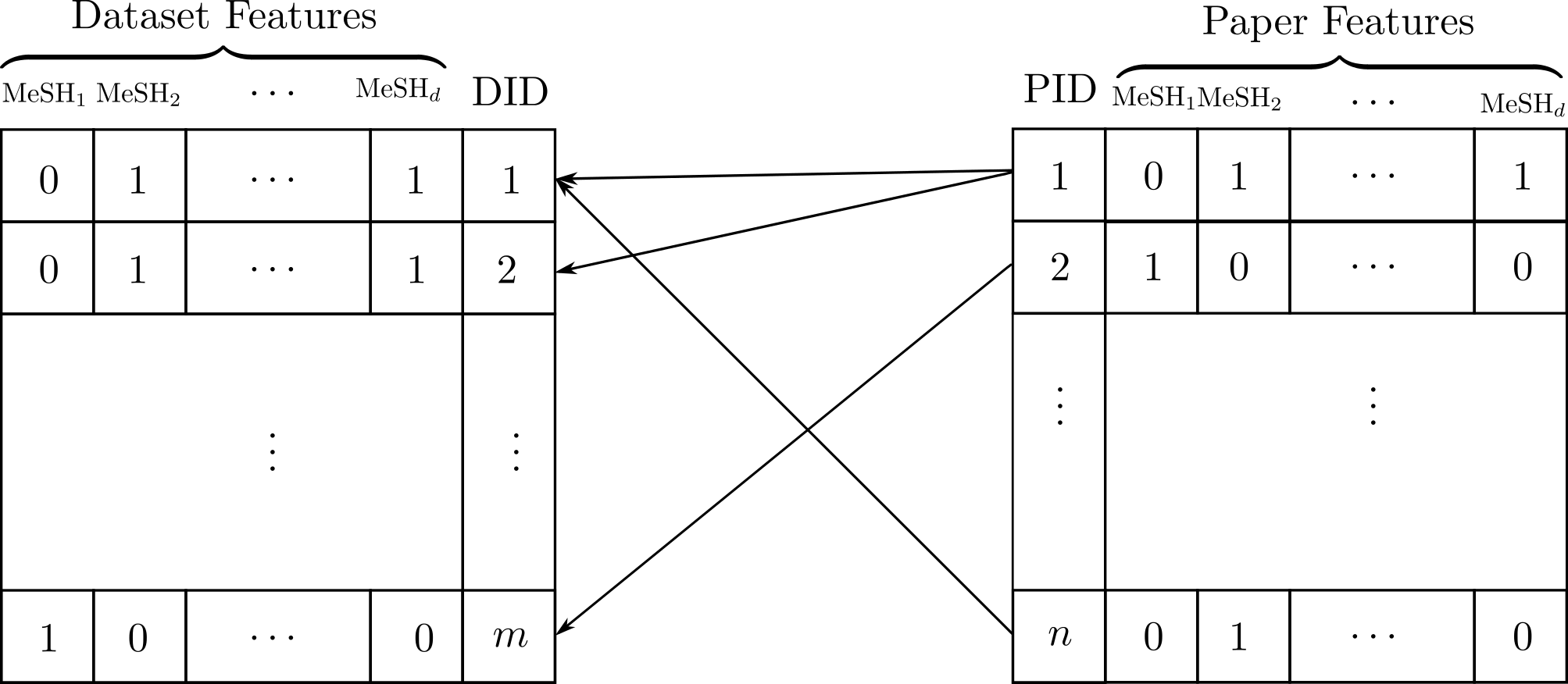


Figure 1: A bipartite graph between  datasets and  papers with MeSH terms as their features. PID and DID are Paper ID and Dataset ID, where existence of an edge between a paper and a dataset implies that dataset is cited in that paper. We propogate the MeSH terms from the paper to the dataset to build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

## Methodology

The *DataRank* framework consists of an offline algorithm and an online algorithm. For a new user, the algorithm works in an offline mode to rank datasets based on relevancy and popularity (e.g., number of citation). After receiving user feedbacks, it updates the model to present user-specific rankings. We incorporate user feedbacks explicitly by giving an option to rate the 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 articles is represented by a binary matrix, , which is the number of MeSH terms and is the number of articles. An element of the matrix has a value one if the MeSH term is associated with the article, otherwise, the value is zero. Similarly, we define a matrix of to represent datasets via bag-of-words of MeSH terms, where is the column of . We use a vector to denote the identifiers of datasets. We store the bipartite graph between articles and datasets using an adjacency matrix , which 1 indicates the article cited the dataset, and vice versa. We consider binary features for both articles and datasets. Therefore, dataset features can be readily obtained from the article features and adjacency matrix:

(1)

where is a elementwise operator over its matrix argument.

### Offline Ranking:

We propose a probabilistic approach and use a graphical model to specify dependencies between random variables, as illustrated in Figure 2. Here should be understood as realizations of corresponding random variables. We also introduce another random variable for search queries over the same sample space as , i.e., MeSH terms . Finally, is an (observed) random variable that defines a prior over the labels . The graphical model is shown in Figure 2-(a). It should be noted that, in any case, variables (dataset features) and (datasets prior) are observed and we do not need their marginal distributions and therefore we only consider the query as evidence.

In this model, the problems of ranking for a given query is to find the posterior distribution for the dataset labels given evidences. More precisely, the posterior distribution

(2)

specifies the ranking, where the posterior distribution is expanded according to the graphical model Figure 2-(a). Here, we can consider as dataset-prior and consider as the query-likelihood. The next step is to specify dataset-prior and query-likelihood distributions.

Query-likelihood

Since the binary feature vector and query are representation of binary sets, using Vann diagram, we can easily compute the likelihood

(3)

where the set operation 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

(4)

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

(5)

Dataset-prior:

We propose a prior for the datasets. A simple way to think about the prior is to imagine the posterior distribution in a scenario where there is no evidence available. This is equivalent to say that what is the best ranking of datasets when there is no queries. There are many answers to this question, and one reasonable 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., . More precisely, we can write prior as

(6)

where is a vector of one. Note that implies has dimensions and implies has dismensions.

|  |  |
| --- | --- |
|  |  |
| (a) | (b) |

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: In this part we extend the offline-DataRank to the online setting by incorporating user feedback into ranking. For this porous, we propose a new model, Figure 1-(b), which introduces *incomplete* user ratings , and the dataset online-labels , where is the number of different state of ratings for each result in the ranking and 0 is used to denote unknown values in the ratings. Thus, in the ranking process, ratings are initialized with zero value and at each epoch users rates the search results and updates the . As shown in the graphical model, the online-labels depend on user feedback but, (offline) dataset labels do not.

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

(7)

where the factorization induced by the graphical model Figure 1-(b) and in this model evidence is incomplete user rating . Interestingly the offline posterior can be regarded 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, which makes a perfect sense.

However, the ratings is incomplete, i.e. contain zero values, and in order to specify the likelihood we first need to estimate the unknown values of user ratings. Once we estimate all the unknown values in the user rating vector we can readily compute the likelihood by normalizing

We use collaborative filtering [?] to estimate unknown values of user ratings for the datasets that has not been rated yet. Collaborative filtering methods generally work by constructing a similarity matrix between items (dataset), and defining a method for finding a set of neighbours of each item, and then computes the rating of undated items as weighted linear combination its rated neighbours, where the weights are proportional to similarity between items.

Since the rating vectors is extremely sparse we choose to define the method for finding neighbours to return all the rated items, to use all the user rating information. Thus, the key step remains to perform collaborative filtering is to define a similarity measure between datasets. Regarding, the binary MeSH representation of datasets, we opt to use Tanimoto kernel [?] between datasets for measuring similarity between datasets

(8)

where is a symmetric positive definite matrix, is intersection kernel [?] between and , and is a general nonlinear feature function.

Having a similarity matrix between datasets, using collaborative filtering we can readily fill the unknown values in the completed rating vector

(9)

where is the column of the kernel matrix. Having computed the completed rating vector, to compute the likelihood, we only need to normalize

(10)

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

## Experimental Setup

### Implementation:

DataEngine 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 more extensible for future recommendation algorithm research with assistance of numpy and scipy.

#### System Architecture:

A brief system architecture is shown below. The abstract architecture is a standard MVC model.

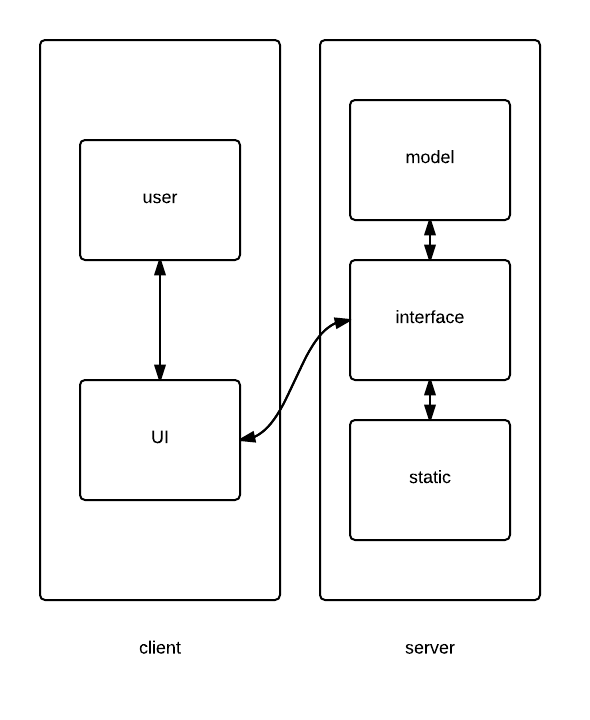


Figure 3: High level system components

All user interaction with UI is passed to interface layer for futuer 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.

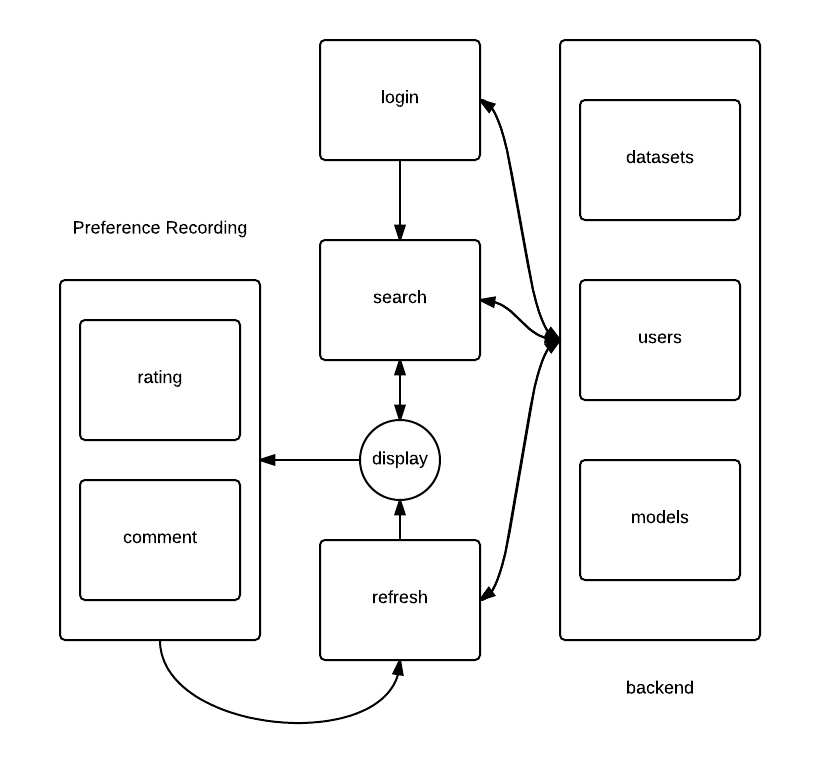


Figure 4: Detailed workflow of the system

At system entry, there is a login management component. User could register, login or stay anonymous. When user either registered or logined, this user will be given a unique identifier. For identified user, all history action could be retrieved for recommendation and all new actions will be recorded as preferences. A search component works at next step. User 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 will be introduced on section ??.

Display component plays as an important connector among different models and functions. User gives rates and comments on this step. When new user related information is added, a refresh component is activated. It will pass all new infomation to backend, and a backend online learning algorithm will collect all user historic and current preferrences and then give a new ranking based on updated information. This infomation will be shown on display component.

#### Searching Pattern:

Searching bar is the first function component faced by user. It could support several basic pattern. Current preffered keywords 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 model for further processing.

Now, dataset specific searching is supported. User may have a destination data source. Thus supporting a data resource fixed searching pattern is necessary. We use an at sign (@) on the end of keywords, and dataset name is given after it. This dataset name is stored, and all future refreshing will only occur under this scale. A simple example is (DNA;Genes@GeneBank).

#### Refresh:

Refreshing is one of the most important feature in this system. Refreshing is majorly 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, user could give ratings and update preferences unlimitedly. Everytime, when new information is given, an update of recommendation would be activated. A standard session work flow is: a user input keywords, and a list of datasets are given. After considering, user gives ratings to some of them and trigger refresh component. Relearning user preferences will be executed and a list of new ordered data will be given.

#### Ratings and Comments:

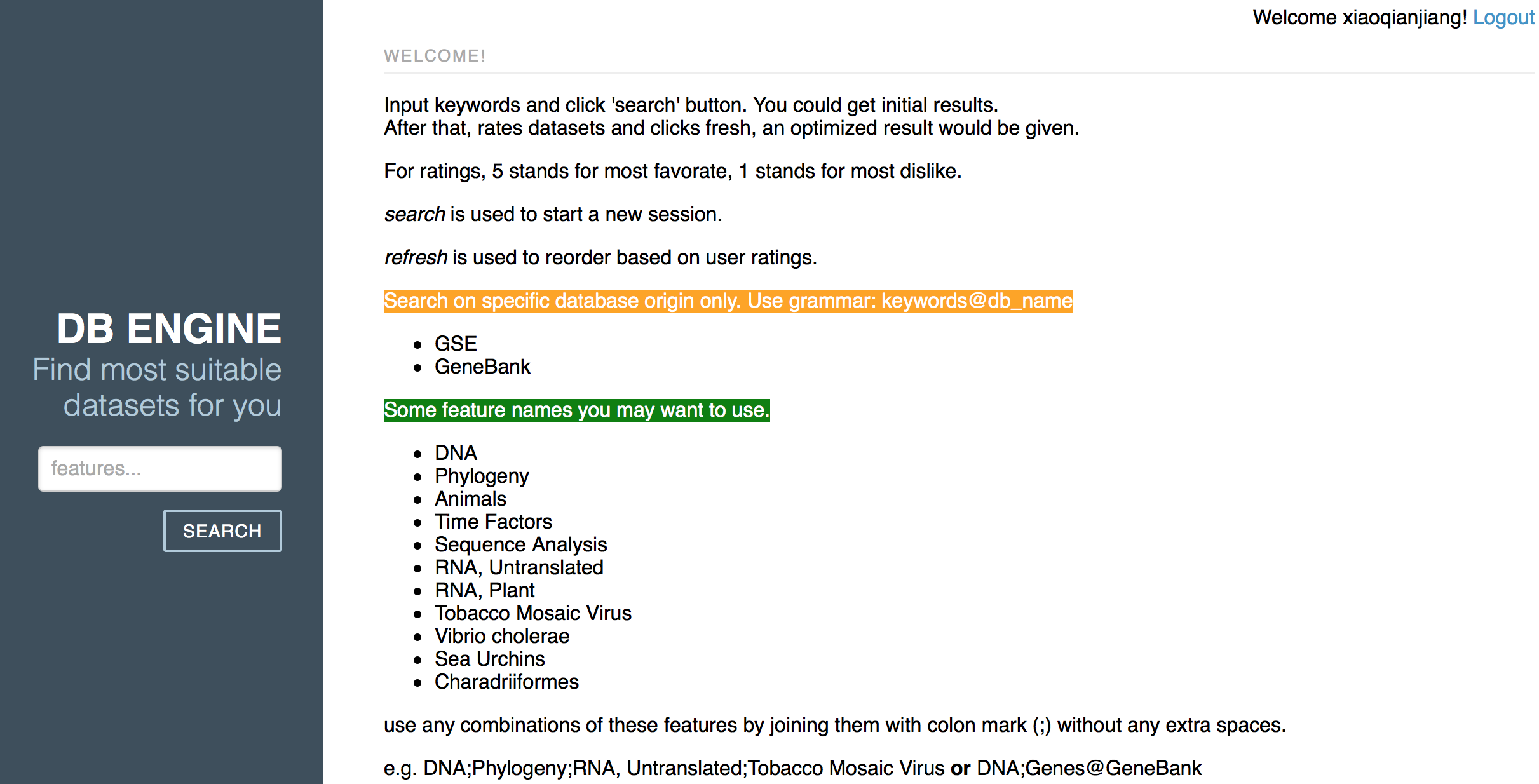
Ratings and comments are two important feature for online learning, which is introduced on section ??. As users may change their ideas frequently, ratings are stored on browser session, which provide better security than traditional cookies. Those ratings will be passed to server side and stored when refresh action is triggered. Each rating is a pair of dataset ID and rating score. Comments are also passed to server side with dataset id, and user related information will be added later. All stored ratings and comments include time information and user information and even keywords. All this information could be used for future ranking algorithm development.

#### Assistant Information:

In order to help user 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 in order to have a better understanding of datasets.

### Experiment Results

(Xiaoqian)



## Discussion and Conclusions

(Xiaoqian)

**Acknoledgement:** The authors are partially supported by

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