DataRank: A Framework for Ranking Biomedical Datasets

Arya Iranmehr[1], Huan Wang[2], Hannah Chen[2], Xiaoqian Jiang[3]

[1]Department of Electrical and Computer Engineering, UC San Diego,
[2]Department of Computer Science, UC San Diego
[3]Department of Biomedical Informatics, UC San Diego

Abstract

Due to the advent of high-performance technologies for the generation, storage and processing of data, research in biomedical sciences is increasingly moving towards "data-driven research" via the manipulation and analysis of biomedical datasets. Unfortunately, the vast majority of these datasets are underutilized after their initial publications due to an increased rate of dataset generation and difficulty in searching for relevant datasets. In this paper, we develop the DataRank framework to rank biomedical datasets by incorporating into our model three different criteria: query-relevance, research-importance and user-preference. We implement query-relevance via a multi-label classification approach using binary relevance schemes. Research-importance of datasets is incorporated into ranking by utilizing a citation network. Preference ranking is implemented in an online setting where DataRank re-ranks searching results according to user feedback. Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR) performance measures on DataRank query results are compared against search results from the Gene Expression Omnibus (GEO) data repository and Jaccard index method. The comparisons conducted on five different held-out validation set of queries and show that DataRank achieves higher MAP and MRR. Finally, we assess the effect of incorporating user feedback by measuring Regret for five different subjects and show that this measure is non-increasing.

1. Introduction

New advancements in technologies have resulted in the proliferation of data production at an unprecedented scale. But the capability of finding or understanding data, lags behind our ability to generate data. Shortly after the popularization of next-generation sequencing in 2006, it was found that the volume of genome data doubles every year and they are becoming increasingly complex[1]. Using nucleotide sequencing as an example, it was found in 2009 that submission rates to a global database were growing at 200% per annum; conservation of this data is guarded by three global repositories which exchange new data on a daily basis [2].

Alongside this data proliferation, which is commonly funded by the public sector, follows an expectation of freely accessible data for global biomedical research. The scientific communities have been concerned with scalability and collaborative issues associated with data management, including but not limited to necessary standardization and infrastructure to support useful data processing and work-flow methods. Increased commitments to data sharing is reinforced by efforts dedicated to building databases, repositories and biobanks [3]. Multidisciplinary databases also led to the collection and analysis of data found from multiple repositories, which may be deposited elsewhere for shared use. In a generation of data-intensive and data-driven computing, scientists are engaging in records as a corpus of text and collections of interlinked data resources that may identify papers of interest, suggest hypotheses to explore, and even the production of new data [4]. Existing entities have been addressing issues to facilitate data use: the European Bioinformatics Institute has developed open standards and search systems indexing data, and the National Center for Biotechnology Information (NCBI) has invested greatly in the PubMed [5] search engine for accessing biomedical literature citations of the MEDLINE database. Despite these efforts, the growing size and number of partially coordinated and incompatible dataset formats make data discovery and integration a notable challenge.

In the past, people may have explicitly cited a dataset's identification number, cited the publication connecting to the dataset's first use, merely mentioned it within a paper, or not cited the dataset at all. Authors and publishers are now encouraged to include data availability information in publications. While much work has focused on

PubMed article querying, there still lacks a unified, systematic way to retrieve datasets in a personalized manner.

A direct search requires background knowledge on the part of the requester and can be done by directly searching within a repository for the data or authors connected to the dataset. The National Institute of Health (NIH) currently manages 65 biomedical data sharing repositories [6]. NIH has also invested resources to provide user-friendly mechanisms for query and analysis, such as with GEO. GEO records can be accessed directly using an accession number or queries can be made to study-level or gene-level databases. Queries can be "effectively performed by simply entering appropriate keywords and phrases into the search box. However, given the large volumes of data stored in these databases, it is often useful to perform more fine-grained queries" [7]. In other words, efficient querying requires knowing what a dataset may encompasses.

Citation analysis and bibliometrics establish precedence of ideas and hint at the quality or significance of a scientific work [27,28]. We exploit an article's bibliometric information to infer features (information) about datasets by using related articles and to learn a dataset's popularity. The goal of our system, DataRank, is to develop a biomedical dataset search engine that takes into account the dataset's research context reflected by its cited articles, its significance within the biomedical community, and its relevancy to the end users.

This paper provides a framework for searching biomedical datasets by utilizing the relationship between articles and the relationship between datasets and articles. More precisely, a dataset-article bipartite network augments datasets with features from related papers, such as Medical Subject Heading (MeSH) or topics. Then we apply multi-label classification, citation analysis and nearest neighbor to data in order to provide a ranking model that accounts for different dataset aspects, such as *query-relevance*, *research-importance* and *user-preference*.

The organization of this paper is as follows: Section 2 reviews related work. Section 3 describes The DataRank model and its extensions. We evaluate our approach on five different validation sets of queries and compare our results to existing GEO search results in Section 4.. Finally, Section 5 concludes and discusses possible extensions of our model.

2. Background

Ranking is a main issue in many information retrieval, machine learning, natural language processing, data mining and advertisement problems. Given a query q, the task of ranking involves computing the ranking score $s_i = f(q, d_i)$ for all the items in the collection $D = \{d_1, \ldots, d_n\}$. The function f is called the *ranking model* and takes two arguments: the searching query and an item from the collection. The ranking model computes the relevance of the examining item to the query based on the ranking model.

Query-Relevance Ranking. In general, relevance ranking models can be divided into two major groups: *traditional* ranking models and *learning-to-rank* models. Traditional ranking models compute the relevance of each item directly from their arguments, (q, d_i), without a training process. For example, in document retrieval, popular methods such as BM25 [12], LMIR [13], VSM [14,15], and BM25F [16] compute their ranking score for each document based only on Term-Frequency (TF) for each term of the query in the examining document and Inverse Document Frequency (IDF) for the query terms. On the other hand, learning-to-rank models operate by learning f using training data, i.e., a set of queries with known relevant items. RankSVM [11, 17], RankBoost [18,21], RankNet [19,20] GBRank [26] adapt Support Vector Machines, AdaBoost, Neural Network and Regression are all examples of well known learning-to-rank models. Although learning-to-rank approaches generally outperform traditional methods [22], they are only feasible when a representative training corpus of queries with corresponding relevant targets are available. However, collecting training data is a costly and timely task requiring human judgment in labeling a set of queries via crowd-sourcing [23,24] or existing real-world query logs [25], i.e. click-through data. In the context of biomedical informatics there has been a lot of efforts in improving ranking. For example, ATM [9] proposed to map the free-text query to a set of MeSH terms. In [8], authors modeled relevance ranking for PubMed documents using TF-IDF features.

Item-Importance Ranking. PubFocus [27] takes into account item popularity, i.e., publication quality, by incorporating bibliometrics. Similarly, [33,34] prioritized important articles in the search results by running a PageRank [29] algorithm on the article citation network. In addition, CoRank [31] and MultiRank [32] introduce the joint use of collaboration networks between authors and article citation networks to estimate popularity of items in the corpus, i.e. importance ranking of objects without any query..

User-Preference Ranking. In an effort to incorporate user feedback into ranking, RefMed [10] uses RankSVM to learn a model for each query on PubMed articles. But, RankSVM only works when enough labeled training samples (e.g. a large number of user feedbacks) are provided to the learning algorithm - an unlikely real world situation where users all rate their search results. Other authors [30,35] have also exploited online learning techniques to improve the ranking results for the user.

In the next section we first describe our indexing procedure, and explain different aspects of our DataRank algorithm: relevance ranking, importance ranking, and preference ranking.

3. Methods

We first specify the training corpus used in this paper. Forming a training corpus amounted to indexing datasets and articles, creating a set of queries, and collecting their associated relevant targets.

3.1 Collection Training Corpus

The first step in creating a search engine is to index items of interest. For DataRank, it means indexing biomedical datasets used in scientific publications and forming a bipartite graph between datasets and articles. The many different use cases and dataset origins prove to be a nontrivial complication for dataset indexing. For example, a repository may have strict guidelines to only accept primary datasets, not reanalysis, meta-analysis, or comparisons. Datasets may also be collected over a range of time, published at various times, or being a subset or derivative of larger, dynamic databases. There are three ways we attempt to perform indexing. First, we can search full text documents and matching regular expression rules for accession numbers of each repository. In our experience, this method is restrictive not only because of the many differing ways an author may cite a dataset, but also because of the small fraction of PubMed articles which are publicly available via bulk download through the API of PubMed Central (PMC) webservice. Second, we can find datasets via the secondary source identifier (SI) descriptor used in MEDLINE citations. Beginning in 2005, many data types discussed within MEDLINE articles, such as sequencing data, gene expression data, and clinical trial data, could be included in the SI element. Unfortunately, this MEDLINE meta-data does not consistently reflect paper-dataset relationship. For example, for the 58,950 GEO repository datasets[1] the SI descriptor yielded only 14,700 referenced datasets out of 25 million MEDLINE records, which is unrealistic. Third, we can find the corresponding Original Publications (OP) associated with datasets, which are the articles describing the generation of data. We obtained all the Citing Publications (CP) that have cited the OPs. Therefore, CPs are publications that used or are semantically related to the datasets. After examining all three alternatives, we find that the third approach gives the largest set for indexing dataset-papers. Therefore, CPs are publications that used or are semantically related to the datasets. After examining all three alternatives, we find that the third approach gives the largest set for indexing dataset-papers.

For this paper, we focus only on the GEO repository but our method can be applied to other NIH data repositories. After removing datasets without any OP citations in the GEO repository, we end up with 35,497 datasets connecting to 26,685 OP, in a many-to-one relation, i.e. each dataset has exactly one OP but each OP can be referred to by multiple datasets. Using Web of Science database, we managed to index bibliographic information of 12,679 OPs, which connects to 16,957 datasets, in order to find the related CPs. This results in finding 197,020 unique CPs that cited an OPs, with 401,153 total citations, see Figure 1 left. After constructing a dataset-citation-paper network, we removed OPs and papers connected directly to the related datasets, see Figure 1 right .

Having obtained an indexing on datasets and related papers, we then create a training corpus. Consider each paper as a query and all the related datasets as the relevant datasets for that query. To keep computations simple, we disregard full article, abstract, and title; only the set of MeSH terms associated with each paper is considered as the search query for that paper. For instance, a query-target pair for the article with PMID 25525874 is:
Query = {Animals, Apoptosis, genetics, metabolism, classification, metabolism, Crosses, Genetic DNA, Mitochondrial, Female, Hematopoietic Stem Cells, Interferon Type I, Interferon Type I, Membrane Proteins,

[1] In fact, GEO repository consist of 58,950 series and 3,848 dataset which each dataset consist of one or more series. Each series is itself is a collection of samples that is submitted to GEO by its creator. In this paper, by GEO datasets we refer to GEO series because the series are referred and cited in the PubMed articles more often than GEO datasets.

Mice, Inbred C57BL, Signal Transduction}
and **Targets** = {GSE57934, GSE59972}.

We use this corpus for training and validation of relevance ranking and importance ranking tasks. In addition, we run these queries on the GEO search engine to provide a baseline performance for DataRank. In the next subsections, we outline the DataRank methods that operate on this corpus.

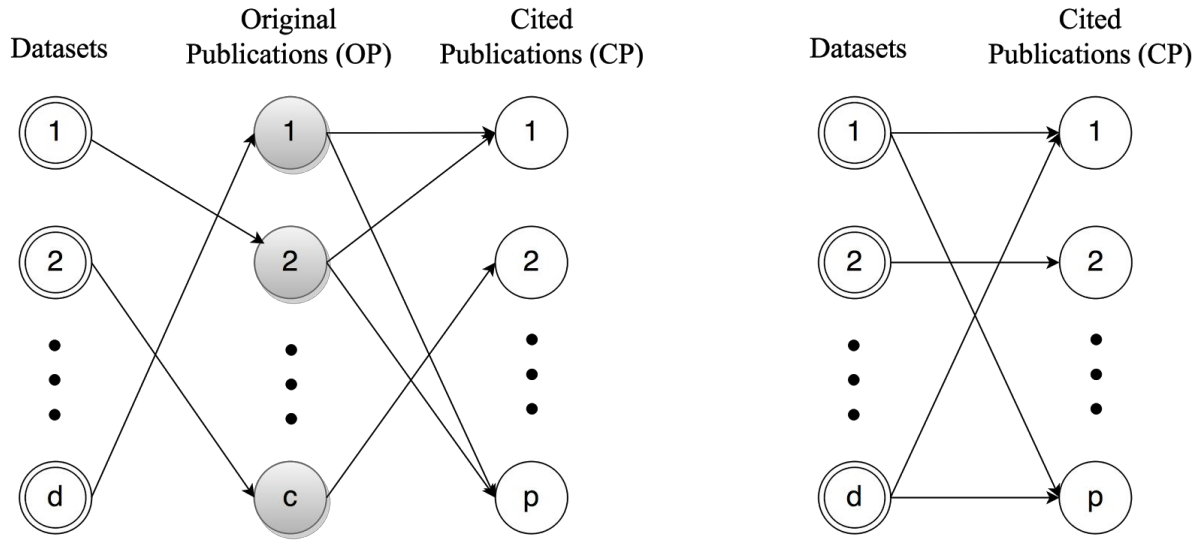

Figure 1. (Left) Illustration of network between datasets, OP and CP. After cleaning and ignoring datasets with small number of citations we reach a network with d=16,957; c=12,679; and p=197,020. The number of edges in the left and right network is 16,957 and is 367,623, respectively. (Right) the same network after removing OPs and connecting CPs to datasets directly.

3.2 Relevance Ranking

The training corpus does not contain all rankings of all items for each query, but instead only contain relevant items for each query. This simplifies the learning-to-rank problem, since the standard learning-to-rank algorithms assume that targets each sample should be ranked in a specific total order. As shown in Figure 1 right, each CP (sample, or query) is connected to one or more datasets (targets or labels) and for each sample there is not a total order ranking of datasets, but only a set of relevant items.

In this case, we model the ranking problem using *multi-label classification* [36,37]. Multi-label classification algorithms apply to supervised learning and ranking tasks in which each sample corresponds to one or more targets (labels). To formulate the ranking problem as a multi-label classification problem, it is sufficient to considering each query as a sample and its related datasets as labels.

To convert queries into features for the learning problem, the set of MeSH terms is transformed into a fixed length feature vector. The Bag-of-Word model is used to represent the set of mesh terms, i.e., each query is represented by the number of occurrences of each MeSH term. Since there are 27,455 possible MeSH terms, the feature vector is a 27,455 dimensional sparse binary vector which is efficiently represented as sparse feature vectors. For example, the feature vector for the below query is represented in both fixed length and sparse format as follows:

Query = {genetics, DNA, Female, }

Feature = $[0, \ldots, 0, 1, 0, \ldots, 1, 0, \ldots, 1, 0, \ldots, 0]^{\mathbf{T}}$

Sparse Feature = {1023:1, 9861:1, 22397:1}

A prevalent paradigm for multi-label classification is *problem transformation*; the multi-label classification problem is transformed into a set of classical single-label classification problems [36]. In particular, binary relevance methods [37] transform the multi-label classification problem with L labels into L different binary classification problems and takes the one-versus-the-rest approach in the training process. In the test phase,

binary relevance evaluates L decision function values for each test sample to predict a set of labels for that sample.

Table 1. Top 5 dataset ranked by the number of Original Paper Citation Count, OPCC, (Left) and ranked by Citing Publication Citation Count, CPCC, (Right).

Rank	Accession	OPCC	CPCC
1	GSE5259	7559	204379
2	GSE9164	4363	134104
3	GSE29611	2104	17684
4	GSE1133	1984	75243
5	GSE20846	1961	32257

Rank	Accession	OPCC	CPCC
1	GSE5259	7559	204379
2	GSE9164	4363	134104
3	GSE3425	1900	121390
4	GSE58	1273	114180
5	GSE55	1273	114180

The DataRank training phase is identical to the binary relevance learning procedure, but in the test phase we are not interested in predicting targets for each test case. Instead, the raw real-valued decision values are to be used in the relevance-ranking. More precisely, for each test query, DataRank sorts all the decision values and corresponding labels (datasets). It should be noted that the training process is time consuming and is performed offline. For each test case, DataRank computes L=16,957 decision functions, which is computed in parallel. Additionally, for linear classifiers, evaluating each decision function amounts to computing a dot product between the feature vector x and a weight vector w:

$$f_l(x) = w_l^T x + b_l \qquad for\ l = 1, ..., L$$

where scalar b is the offset in a linear transformation. The 27,455 dimensional feature vector has a sparsity density of 0.06% on average for all samples, so the relevance-ranking can rank new test queries in real-time without latency.

Standard soft-margin Linear Support Vector Machines (SVM) [41] is implemented using the sklearn package [38] to solve the binary classification problem. Linear SVM can effectively learn all the weight vectors w_l and offset terms b_l for the L decision functions. Finally, we used probability outputs of SVM decision function computed by sigmoid function as ranking score for relevance ranking

$$\sigma_l = \frac{1}{1+exp(-f_l(x))}$$

In this setting, each component of the weight vectors captures how predictive each MeSH term is for each dataset, i.e. each MeSH term in the query can be predictive of different datasets with different weights. Moreover, these weights are learned on a separate training corpus and evaluated on a held-out test corpus. In Section 4, we perform these experiments and compare it to the baseline GEO search engine on the same held-out queries.

3.3 Importance Ranking

"Importance" is a broad term and may encode a wide range of objectives, such as recentness or research impact of an item in the ranking, For simplicity, we only consider research impact as the factor encoding importance of a dataset.

A natural way to quantify a dataset's research impact is to evaluate its corresponding OP, for example, by computing the number of times that dataset is used and cited through its OP. We calculate Original Publication Citation Count (OPCC) as the number of outgoing edges for an OP, as seen in Figure 1 left.

A more comprehensive approach is to include the significance of CPs as well as those publications that cited CPs into this process. The importance of each dataset depends on its OP and the impact of each publication recursively depends on the significance of its citing publication; the importance of OPs and CPs depends on *how many times* it is cited and *by which publications* it is cited. When given a full citation network of papers, one

could apply the PageRank algorithm to obtain a ranking for articles (and therefore OPs) based on their importance.

Unfortunately, PageRank only works well if the full citation network is available, i.e. it requires finding all second-level CPs for the 197,020 first-level CPs in Figure 1 right, and finding all third-level CPs for the second level CPs, and so on. By only querying on 197,020 first-level descendant papers (Figure 1, right), we already found more than 1.5 million papers. This number is expected to grow exponentially as the number of hops increases until all papers are accounted for. Therefore, instead of building the full citation network of PubMed articles, we simply consider the number of times the first-level CP is cited as an index of their significance, Citing Publication Citation Count (CPCC). We then aggregate the number of citations of first-level CPs to the corresponding datasets. For example, the top 5 datasets ranked by OPCC and CPCC are shown in Table 1 left and right, respectively. Table 1 shows that datasets with GEO accession GSE5259 and GSE9164 are highly cited via both OPCC and CPCC, but the ranking is different for other datasets. For example, datasets with accessions GSE3425 is ranked 17 in OPCC ranking while is the third item in CPCC ranking.

We normalize CPCC for all the datasets between zero and one to compute the importance score for each dataset, and use this in conjunction with relevance-ranking to yield the final ranking results:

$$\beta_i = \frac{CPCC_i}{\sum_{j=1}^{L} CPCC_j}$$

3.4 preference ranking

In practice, user preference is determined either by a user's explicit (integer) ratings of items or by a user's click-through data, which is referred to as an implicit (binary) rating [44].

Finally, we extend DataRank by incorporating explicit user feedback to capture account preference ranking. We introduce user rating $r_i \in \{0,1,2,3,4,5\}$ for each dataset d_i, where 0 represents the state of an unknown rating and user ratings take a value on a 1 to 5 scale. The user rating vector r is a vector of all ratings for a single user; user rating is *complete* if it does not contain any unknown (i.e. zero) values and is *incomplete* if includes unknown values.

Given a complete user rating, the problem of preference ranking is trivial because it amounts to merely sorting the items based on user ratings. However, the user rating is always incomplete and only a small number of items has been rated by the user. We are interested in estimating a complete rating vector z based on incomplete rating vector r. This problem can be regarded as collaborative filtering [42] for a single user. Collaborative filtering computes the rating of each unrated item by averaging the rating of k-most similar items.

In order to implement single-user collaborative filtering, we first need to define a similarity metric between datasets. As explained in Section 3.1, the datasets are represented by a set of MeSH terms corresponding to their related CPs. A natural similarity measure between two *sets* of items is computed by Jaccard index[43] $J(d_i,d_j)=\frac{|d_i \cap d_j|}{|d_i \cup d_j|}=J_{ij}$, i.e. the size of the intersection of sets divided by the size of the union of sets.

We fill the unknown user rating of the dataset d_i by computing the convex combination of the ratings of the all the rated items, i.e., the weighted average where weights are positive and sum to one:

$$z_i = \sum_{j|r_j \neq 0} \theta_{ij} r_j \qquad \theta_{ij} \geq 0, \quad \sum_j \theta_{ij} = 1 \ \forall \ i$$

We chose the weights to be proportional to Jaccard index which is always non-negative, in other words

$$\theta_{ij} = \frac{J_{ij}}{\sum_k J_{ik}}$$

By normalizing weights we guarantee that all estimated ratings take a value between 1 and 5. Similar to importance-score, the preference-score is computed by normalizing the completed user rating :

$$\alpha_i = \frac{z_i}{\sum_j z_j}$$

which will be used together with importance score and relevance score to compute the final score for each dataset. From the experimental study, we show that estimated user rating converges to the actual user rating as users provide more ratings.

3.5 DataRank Ranking Score

In the previous section, we proposed mechanisms for evaluating different criteria (relevance, importance and preference) for the ranking results. Combining different ranking score into one score is a nontrivial task. However, in order to avoid over-fitting and large number of hyperparamters, linear combination of different decision function is used in practice [47]. For DataRank, we choose to combine vectors of relevance, importance and preference score, σ, β, α respectively, by

$$S = \lambda_1 log(\sigma) + \lambda_2 log(\beta) + \lambda_3 log(\alpha)$$

where log is a component-wise natural logarithm and λ_1, λ_2, λ_3 are scalar model hyperparameters, to be chosen by cross-validation or heuristically. Without loss of generality one of the λ coefficients can be set to 1 which leaves DataRank with two hyperparamters.

4. Results

In this part we first compare the relevance ranking and importance ranking of DataRank with existing GEO web search engine, and then we analyze the performance of the preference ranking. The DataRank prototype search engine is implemented in python using Django, and is available at http://biocaddie.ucsd-dbmi.org:8888.

4.1 Relevance Ranking

Since in the problem of ranking datasets of users only the top results are important, we consider Mean Reciprocal Rank (MRR) [39] and Mean Average Precision (MAP) [40] performance measures for the relevance ranking experiments.

MAP is computed by averaging Average Precision at top-k (AP@k) for the all test query results. For each query, AP@k itself is computed by averaging Precision@k (P@k). For a set of m test queries, MAP is computed by

$$P@k = \frac{\#\ of\ relevant\ items\ in\ top\ k\ results}{k}, \qquad AP@k = \frac{\sum_{j=1}^{k} P@j\, I_j}{\#\ of\ relevant\ items\ in\ top\ k\ items}, \qquad MAP = \frac{1}{m}\sum_{i=1}^{m} AP@k_i$$

where I_j is one if item at rank j is relevant and zero otherwise. In all the experiments we set k=100.

Similarly, MRR is the average of Reciprocal Rank for a ranking and is calculated

$$RR = \frac{1}{Rank\ of\ the\ first\ relevant\ item}, \qquad MRR = \frac{1}{m}\sum_{i=1}^{m} RR_i$$

To compute ranking of GEO search engine for each query, we used Entrez Programming Utilities [45] to search for queries. In addition to GEO search results, we compare DataRank with Jaccard index method. For the Jaccard index method, the ranking score is simply computed by calculating Jaccard index between query and every dataset features.

To provide a fair comparison between DataRank and GEO and Jaccard, we evaluated performance measures using 5-fold cross validation. In other words, the corpus of 197,020 samples is randomly split to five folds and in each run one split (39,404 queries) is selected for validation (test) and four splits (157,616 queries) is selected for training. For each method, the MRR and MAP are evaluated on the five validation set of queries and the results are illustrated in Figure 2.

We also combined the importance-ranking and relevance-ranking by computing the ranking score as product of relevance score and importance score: $s_l = \sigma_l + \lambda_1 \beta_l$ *for* $l = 1, ..., L$. The final ranking is then computed by sorting ranking scores. Average MAP and MRR of GEO search, relevance-ranking and relevance-ranking over five different held-out corpus is illustrated in the Figure 2.

Only-relevance ranking archives higher MRR on all validation sets and presents higher MAP, on average, compared to GEO. By incorporating importance scores into ranking, the MAP and MRR increases. The combination of relevance and importance ranking reveals absolutely higher MRR and MAP than GEO and on average higher MRR and MAP than only-relevance ranking. This results suggest that not only the

relevance-ranking performs better than the baseline GEO search results but also the importance-ranking seems to be complementary to the relevance ranking and it improves the predictive performance of DataRank

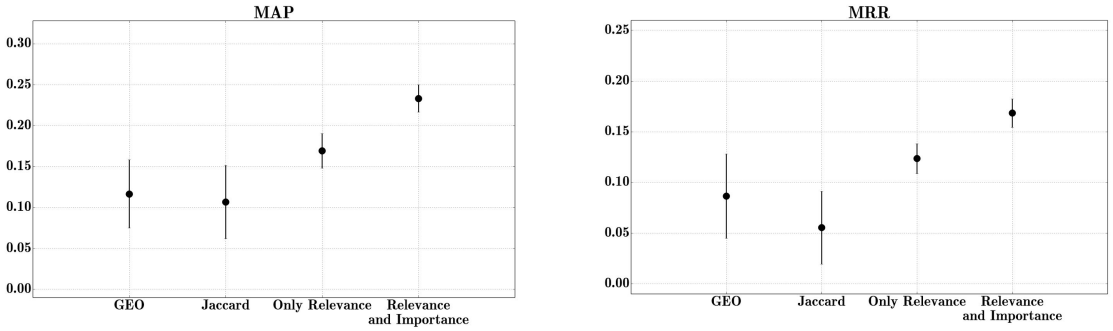

Figure 2. MAP (left) and MRR (right) of the algorithms computed on five different randomly selected validation corpus (See 4.1 for details). Dots represent average and bars represent standard deviation of the performance measures over five different folds.

4.3 Preference Ranking

Assessing the MAP and MRR performance of preference ranking requires a large amount of explicit user-rating to create train and test datasets. Unfortunately, such a data can only be collected if the system is actually used by a large population of researchers for a period of time. This chicken-and-egg problem is often referred as the "cold-start" problem in the recommender systems research community[42].

However, instead of evaluating the preference-ranking performance we measure the predictive performance of the rating estimation. We show that as user provide more ratings DataRank predicts the rating more correctly, which suggest that if the user provides enough feedbacks, DataRank will predicts user ratings and therefore provides user with more desirable results.

In this experiment, 5 different subjects, 2 professors and 3 PhD candidates, are chosen to examine DataRank. Each user aimed a dataset in GEO and searched for a query using DataRank. After showing the results, the subject is required to rate the most relevant item in the search result and submits his feedback to DataRank. Upon receiving the user feedback, DataRank estimates the rating for all the unrated datasets, computes preference score α, computes the ranking score $s_l = log(\sigma_l) + \lambda_1 log(\beta_l) + \lambda_2 log(\alpha_l)$ for all the datasets and re-ranks the search results for the user. The user again submits a new feedback for the newly generated search results and this process is repeated for 20 iterations. Note that, the user only searches for a query ones and in each iteration submits a rating for a dataset.

After collecting data for five subjects, we can retrospectively compute the standard performance measure for online learning, Regret [46], for the rating predictions. Regret is average loss over time

$$R^{(t)} = \frac{1}{t} \sum_{j=1}^{t} \phi^{(j)}$$

The loss at iteration t is defined as the difference between the most recent estimation $z_i^{(t-1)}$ and the actual rating that user submitted $r_i^{(t)}$ at iteration t:

$$\phi^{(t)} = |r_i^{(t)} - z_i^{(t-1)}|$$

which i is the index of the item that user is rated at iteration t. The Regret of the experiment for five different subjects is shown in the Figure 3 which demonstrates the non-increasing trend in of Regret for all the subjects. This implies that as the user provides more feedback about the items, DataRank brings similar datasets to the item that user rated, up and those new results more fits to the user preferences.

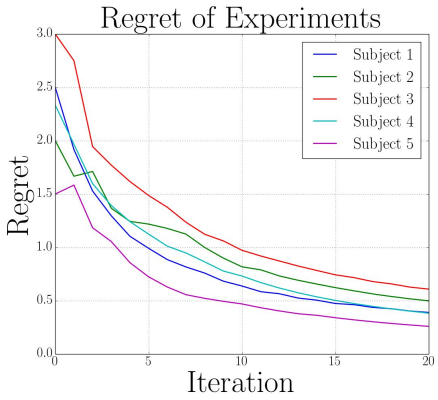

Figure 3. Regret of predicting user rating for 5 different expert subjects in 20 consecutive iteration

5. Conclusion

In this paper, a very general platform, DataRank, for ranking biomedical dataset is presented. DataRank infers a set of features for each dataset using its related publications. Using these features, DataRank is provides simple and efficient ranking models for relevance ranking, importance ranking, and preference ranking, and combine them in its final ranking model. Experimental study shows DataRank achieves a higher MAP and MRR than GEO Search Engine and Jaccard index on 16,957 datasets and 197,020 queries. We note four potential future works for DataRank. First, it can be easily extended to full the GEO repository and all the other 64 repositories. Second, more comprehensive kinds of features such as TF-IDF, Probabilistic Topics, etc. can be extract from publications and attributed to the related datasets to improve predictive performance of DataRank. Third, the preference ranking for each user can be boosted by taking into account of user's bibliography. Fourth, the importance ranking can be performed using PageRank algorithm on the full citation network.

Acknowledgments

The authors are supported by NIH grant U24AI117966.

References

1. A. Szalay and J. Gray. 2020 computing: Science in an exponential world. Nature, 440(7083):413–414, 2006.
2. C. Southan and G. Cameron. Beyond the tsunami: developing the infrastructure to deal with life sciences data., 2009.
3. S. Leonelli. Introduction: Making sense of data-driven research in the biological and biomedical sciences.Studies in History and Philosophy of Science Part C: Studies in History and Philosophy of Biological and Biomedical Sciences, 43(1):1–3, 2012.
4. C. A. Lynch. Jim gray's fourth paradigm and the construction of the scientific record., 2009.
5. N. C. for Biotechnology Information. Pubmed. http://www.ncbi.nlm.nih.gov/pubmed. [Online; queried 1-July-2015].
6. N.I. of Health. NIH Data Sharing Repositories. http://www.nlm.nih.gov/NIHbmic/nih_data_sharing_repositories.html, [Online; accessed 20-May-2015].
7. N. NCBI. Geo overview: Query and analysis. http://www.ncbi.nlm.nih.gov/geo/info/overview.html#query. [Online; queried 27-May-2015].
8. Lu Z, Kim W, Wilbur WJ. Evaluating Relevance Ranking Strategies for MEDLINE Retrieval. JAm Med Informatics Assoc 2009;16:32–6. doi:10.1197/jamia.M2935
9. Lu Z, Kim W, Wilbur WJ. Evaluation of Query Expansion Using MeSH in PubMed. Inf RetrBoston 2009;12:69–80. doi:10.1007/s10791-008-9074-8
10. Yu H, Kim T, Oh J, et al. RefMed: relevance feedback retrieval system of PubMed. In: Proceedings of the 18th ACM conference on Information and knowledge Management. 2009. 2099–100.
11. Joachims T. Optimizing search engines using clickthrough data. In: KDD. 2002.

12. Robertson, S.E., Walker, S., Jones, S., Hancock-Beaulieu, M., Gatford, M.: Okapi at trec-3. In: Proceedings of the 3rd Text REtrieval Conference (1994).
13. Zhai, C., Lafferty, J.: A study of smoothing methods for language models applied to information retrieval. ACM Trans. Inf. Syst. 22(2), 179–214 (2004).
14. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Commun. ACM 18(11), 613–620 (1975)
15. Lee, Dik L., Huei Chuang, and Kent Seamons. "Document ranking and the vector-space model." Software, IEEE 14.2 (1997): 67-75.
16. Zaragoza, Hugo, et al. "Microsoft Cambridge at TREC 13: Web and Hard Tracks." TREC. Vol. 4. 2004.
17. Herbrich, R., Graepel, T., & Obermayer, K. (1999). Support vector learning for ordinal regression. Proceedings of ICANN 1999 (pp. 97–102).
18. Freund, Y., Iyer, R., Schapire, R. E., & Singer, Y. (1998). An efficient boosting algorithm for combining preferences. Proceedings of ICML 1998 (pp. 170–178).
19. Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., & Hullender, G. (2005). Learning to rank using gradient descent. Proceedings of ICML 2005 (pp. 89–96).
20. Quoc, C., and Viet Le. "Learning to rank with nonsmooth cost functions." Proceedings of the Advances in Neural Information Processing Systems 19 (2007): 193-200.
21. Zheng, Zhaohui, et al. "A general boosting method and its application to learning ranking functions for web search." Advances in neural information processing systems. 2008.
22. Chapelle, Olivier, and Yi Chang. "Yahoo! Learning to Rank Challenge Overview." Yahoo! Learning to Rank Challenge. 2011.
23. Hang, L. I. "A short introduction to learning to rank." IEICE TRANSACTIONS on Information and Systems 94.10 (2011): 1854-1862.
24. Hsueh, Pei-Yun, Prem Melville, and Vikas Sindhwani. "Data quality from crowdsourcing: a study of annotation selection criteria." Proceedings of the NAACL HLT 2009 workshop on active learning for natural language processing. Association for Computational Linguistics, 2009.
25. Li, Hang. "Learning to rank for information retrieval and natural language processing." Synthesis Lectures on Human Language Technologies 7.3 (2014): 1-121.
26. Zheng, Zhaohui, et al. "A regression framework for learning ranking functions using relative relevance judgments." Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 2007.
27. Plikus, Maksim V., Zina Zhang, and Cheng-Ming Chuong. "PubFocus: semantic MEDLINE/PubMed citations analytics through integration of controlled biomedical dictionaries and ranking algorithm." BMC bioinformatics 7.1 (2006): 424.
28. Dong, Peng, Marie Loh, and Adrian Mondry. "The" impact factor" revisited." Biomedical digital libraries 2.7 (2005): 1-8.
29. L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. 1999.
30. Moon, Taesup, et al. "Online learning for recency search ranking using real-time user feedback." Proceedings of the 19th ACM international conference on Information and knowledge management. ACM, 2010.
31. Zhou, Ding, et al. "Co-ranking authors and documents in a heterogeneous network." Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on. IEEE, 2007.
32. M. K.-P. Ng, X. Li, and Y. Ye. Multirank: co-ranking for objects and relations in multi-relational data. In Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 1217–1225. ACM, 2011.
33. Lin, Yongjing, et al. "A document clustering and ranking system for exploring MEDLINE citations." Journal of the American Medical Informatics Association 14.5 (2007): 651-661.
34. Bernstam, Elmer V., et al. "Using citation data to improve retrieval from MEDLINE." Journal of the American Medical Informatics Association 13.1 (2006): 96-105.
35. Ciaramita, Massimiliano, Vanessa Murdock, and Vassilis Plachouras. "Online learning from click data for sponsored search." Proceedings of the 17th international conference on World Wide Web. ACM, 2008.
36. Tsoumakas, Grigorios, and Ioannis Katakis. "Multi-label classification: An overview." Dept. of Informatics, Aristotle University of Thessaloniki, Greece (2006).
37. Elisseeff, André, and Jason Weston. "A kernel method for multi-labelled classification." Advances in neural information processing systems. 2001.

38. Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." The Journal of Machine Learning Research 12 (2011): 2825-2830.
39. McFee, Brian, and Gert R. Lanckriet. "Metric learning to rank." Proceedings of the 27th International Conference on Machine Learning (ICML-10). 2010.
40. Baeza-Yates, R., & Ribeiro-Neto, B. (1999). Modern information retrieval. Addison Wesley.
41. Cortes, Corinna, and Vladimir Vapnik. "Support-vector networks." Machine learning 20.3 (1995): 273-297.
42. Su X, Khoshgoftaar TM. A Survey of Collaborative Filtering Techniques. Adv Artif Intell 2009;2009:4:2–4:2.
43. Manning CD, Raghavan P, Schütze H. Introduction to information retrieval. Cambridge University Press 2008.
44. Oard, Douglas W., and Jinmook Kim. "Implicit feedback for recommender systems." Proceedings of the AAAI workshop on recommender systems. 1998.
45. Sayers E. E-utilities Quick Start. 2008 Dec 12 [Updated 2013 Aug 9]. In: Entrez Programming Utilities Help [Internet]. Bethesda (MD): National Center for Biotechnology Information (US); 2010-. Available from: http://www.ncbi.nlm.nih.gov/books/NBK25500/.
46. Shai Shalev-shwartz PYS. Online learning: Theory, algorithms, and applications. 2007.
47. Kittler, Josef, et al. "On combining classifiers." Pattern Analysis and Machine Intelligence, IEEE Transactions on 20.3 (1998): 226-239.