2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining

Towards a Personalized, Scalable, and Exploratory Academic Recommendation Service

Onur Küçüktunç[1,2], Erik Saule[1], Kamer Kaya[1], and Ümit V. Çatalyürek[1,3]
[1]Dept. of Biomedical Informatics, The Ohio State University
[2]Dept. of Computer Science and Engineering, The Ohio State University
[3]Dept. of Electrical and Computer Engineering, The Ohio State University
kucuktunc.1@osu.edu, {esaule,kamer,umit}@bmi.osu.edu

***Abstract*—Literature search is an integral part of the academic research. Academic recommendation services have been developed to help researchers with their literature search, many of which only provide a text-based search functionality. Such services are suitable for a first-level bibliographic search; however, they lack the benefits of today's recommendation engines. In this paper, we identify three important properties that an academic recommendation service could provide for better literature search: *personalization*, *scalability*, and *exploratory search*. With these objectives in mind, we present a web service called theadvisor which helps the users build a strong bibliography by extending the document set obtained after a first-level search. Along with an efficient and personalized recommendation algorithm, the service also features result diversification, relevance feedback, visualization for exploratory search. We explain the design criteria and rationale we employed to make the theadvisor a useful and scalable web service with a thorough evaluation.**

I. Introduction

We argue that an academic recommendation system must allow a researcher to execute a complex **personalized** query and to obtain the highest accuracy, the query should be processed at a conceptual level. Yet, even the best algorithms may not be able to pinpoint the important documents precisely, only the user will recognize them. Therefore, the tool should also allow the user to **explore** its database in multiple ways to enable discovering interesting documents. Furthermore, such a system needs to be **efficient** to keep the response time short enough to encourage its users to issue more queries, and since the amount of data will increase, the system must also be **scalable** to stay efficient in the future.

Existing academic services: One of the many academic services, *DBLP* is publicly available and references more than two million papers in computer science with complete bibliographic information, venue of publication, and author disambiguation (proper handling of homonyms). *CiteSeer$^\chi$* is another service which harvests the web for publications in computer and information sciences. The service analyses the documents and extracts their text, titles, authors, and reference lists. The data is used to feed multiple services including citation analysis within *CiteSeer$^\chi$* and automatic paper recommendation based on text similarity in *RefSeer* [1].

CiteULike is a social tagging application where researchers manage their bibliography, tag them with keywords, and share it with other researchers. The service can then provide a recommendation based on the common interests between researchers. Services similar to *CiteULike* help researchers to select and group papers and let them specify different areas of interests, but the recommendation part is commonly very limited. *Microsoft Academic Search* is a text-based search engine which also features filtering by areas, topics, co-authorship information, and temporal trends. Similarly, *ArnetMiner* also allows to search with respect to trends, rankings, and topics of researchers/conferences.

Google Scholar is a popular and generic academic text-based search engine which exposes citations of papers. As new documents are indexed, the service also provides personalized suggestions, however, only based on the current publications of the user. This strategy is beneficial to stay up-to-date on published topics, but when a researcher starts working in a new discipline, such a system cannot provide relevant information.

Motivation and contributions: All these existing services allow to perform simple queries on the data they host such as "which documents are popular in a given topic or relevant to a keyword?", "which researchers are close to a topic or another researcher?", or "are there any documents that match a given set of keywords?". While writing an article, these questions are good starting points, which we refer to as *first-level bibliographic search*. A recommender system employing features like personalization, exploratory search, etc., could provide a better literature search. Existing web services generally address the scalability concerns; however, none of these services are personalized and exploratory to a satisfactory extent.

We present the**advisor** (http://theadvisor.osu.edu), a web service to improve the literature search process with a personalized and exploratory search. We previously evaluated the direction awareness feature [2], presented preliminary results on result diversification [3], and the details of various efficient implementations [4], [5]. The contributions of this paper can be summarized as: (1) we describe each component of the framework in detail, (2) we show how relevance feedback and result diversification improve the service in practice, (3) we present a novel and efficient hybrid implementation of the recommendation algorithm, and (4) we show how different techniques complement each other to provide a powerful document discovery engine.

II. Framework Overview

The framework has four main components: (a) **paper mapper** finds the entities in our citation graph which correspond the input from the user, (b) **recommendation engine** recommends a diversified set of papers to the user, (c) **visualization** uses graph drawing techniques on the recommended paper set to visualize the relations between them, and (d) **relevance feedback** gets the comments from the user on the recommendations and refines the search accordingly. Figure 1 gives an overview of the**advisor**'s framework and the relationship between its components.

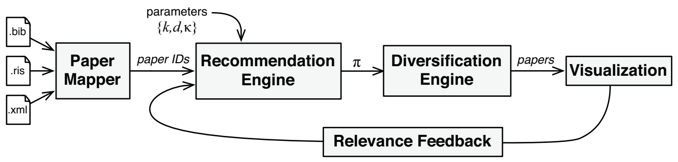


Fig. 1. Overview of the the**advisor** framework.

Dataset collection: We retrieved information on 1.9M computer science articles from DBLP (http://dblp.uni-trier.de/), 740K technical reports on physics, mathematics, and computer science from arXiv (http://arxiv.org/), and 40K publications from HAL-Inria (http://hal.inria.fr/) open access library. This data is well-formatted and disambiguated; however, it contains only a few citation information (less than 470K edges). We obtained the data from CiteSeer$^{\mathcal{X}}$ (http://citeseerx.ist.psu.edu/) using the OAI interface in March 2012 to increase the number of paper-to-paper relations of computer science publications. After the papers from four datasets are merged, and the papers without any references or incoming citations are discarded, the final citation graph has about 1M papers and 6M references, which is currently being used in our service.

Paper mapper: The recommendation process starts with converting the entries within user's bibliography file (in BibTeX, RIS, or EndNote XML format) into the internal IDs of the papers. This set will be used as the seed papers. The rest of the components are discussed in the following sections.

III. Personalization and Accuracy

There are several citation recommendation approaches in the literature, many of which employ textual similarity between documents. However, it has been shown that text-based similarity is not sufficient for this task and that most of the relevant information is contained within the references [6]. Besides, it is plausible that there is already a correlation between citation similarities and text similarities of the papers [7]. In the**advisor**, we use the *citation graph* of known bibliography [2]. In other words, we do not take the textual data into account because our aim is to find all conceptually related, high quality documents even if they use a different terminology.

A. Citation recommendation

To rank the papers, we use a PageRank variant on the citation graph which is based on the Random Walk with Restarts algorithm (RWR). As PageRank models a random surfer, in the**advisor**, our algorithm models a random researcher who picks a paper, reads it, and continue with a random reference.

Let $G = (V, E)$ be the *citation graph*, with n papers $V = \{v_1, \ldots, v_n\}$. In G, each directed edge $e = (v_i, v_j) \in E$ represents a *citation* from v_i to v_j. For the rest of the paper, we use the phrases *"references of v"* and *"citations to v"* to describe the graph around vertex v; $\delta^-(v)$ and $\delta^+(v)$ to denote the number of references of and citations to v, respectively.

The paper recommendation problem is defined as follows: Given a set of m seed papers $\mathcal{Q} = \{q_1, \ldots, q_m\}$ s.t. $\mathcal{Q} \subset V$ and a parameter k, return the top-k papers which are relevant to the ones in $\mathcal{Q}$. We define a random walk on G arising from following the edges (links) with equal probability and a random restart at an arbitrary vertex with $(1-d)$ teleportation probability. The probability distribution over the states follows the discrete time evolution equation $\mathbf{p}_{t+1} = \mathbf{P}\ \mathbf{p}_t$, where $\mathbf{p}_t$ is the vector of probabilities of being on a certain state at iteration t, and $\mathbf{P}$ is the transition matrix defined as:

$$\mathbf{P}(u,v) = \begin{cases} (1-d)\frac{1}{n} + d\frac{1}{\delta(v)}, & \text{if } (u,v) \in E \\ (1-d)\frac{1}{n}, & \text{otherwise,} \end{cases} \tag{1}$$

where $\delta(v) = \delta^-(v) + \delta^+(v)$ is the total degree of the vertex $v \in V$. If the network is ergodic (i.e., irreducible and non-periodic), the process converges to a stationary distribution $\pi = \mathbf{P}\pi$ after a number of iterations. The final distribution π gives the PageRank scores [8] of the nodes based on *centrality*.

In our problem, a set of nodes $\mathcal{Q}$ was given as a query, and we want the random walks to teleport to only those given nodes. Let us define a prior distribution $p^*(v)$ which is $1/m$ for $v \in \mathcal{Q}$ and 0, otherwise. If we substitute the $(1/n)$s in (1) with p^*, we get a variant of PageRank, which is known as *personalized PageRank* (PPR) or *topic-sensitive PageRank* [9]. PPR scores can be used as the relevance scores of the items in the graph. The rank of each seed node is reseted after the system reaches to a steady state, i.e., $\forall q \in \mathcal{Q}$, $\pi_q \leftarrow 0$, since the objective is to extend $\mathcal{Q}$ with the results.

We introduce the *direction awareness* parameter $\kappa \in [0, 1]$ to obtain either more recent or more traditional results in the top-k documents [2]. Given a query with a *seed* paper set $\mathcal{Q}$, a damping factor d, and a direction awareness parameter κ, *direction-aware random walk with restart* (DaRWR) computes the steady-state probability vector π. The rank vector at iteration t is computed with the linear equation $\mathbf{p}_{t+1} = p^* + \mathbf{A}\mathbf{p}_t$, where p^* is an $n \times 1$ restart probability vector defined above, and $\mathbf{A}$ is a structurally-symmetric $n \times n$ matrix of edge weights, such that a nonzero a_{ij} is equal to $\frac{d(1-\kappa)}{\delta^+(i)}$ or $\frac{d\kappa}{\delta^-(i)}$ for $(i,j) \in E$ and $(j,i) \in E$, respectively.

Figure 2 shows how the probability of a node is distributed among its references, citations, and query papers. Note that $\mathbf{P}$, the transition matrix of random walk-based methods, is built using $\mathbf{A}$ and p^*; however, the edge weights in rows can be stored and read more efficiently with $\mathbf{A}$ in practice [4], [5].

B. Experiments on citation recommendation

As stated above, the direction awareness feature was evaluated in [2]. We present the results here for a complete evaluation of the framework.

Parameter test: We study the impact of the damping factor d and the direction-awareness parameter κ on the recommended papers. We will show that changing these parameters obtains recommendations that are either closer to or farther from the seed papers $\mathcal{Q}$, and/or that are either recent or more traditional. To verify these effects, the references of a randomly selected source paper, published between 2005 and 2010, are used as the seed papers. We use the top-10 results as the recommended paper set $\mathcal{R}$. The test is repeated for 2500 distinct queries that satisfy the given constraints.

Figure 3 shows the impacts of d and κ on the average publication year in $\mathcal{R}$ and the average shortest distance in the citation graph between $\mathcal{R}$ and $\mathcal{Q}$. When we increase the damping factor d, the probability of a random walk jumping back to the nodes in $\mathcal{Q}$ reduces; hence, it allows reaching vertices distant from $\mathcal{Q}$ more often. Figure 3 (right) shows

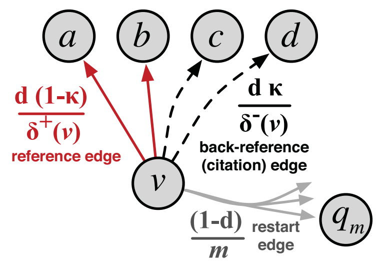


Fig. 2. At each iteration, DaRWR distributes the rank of paper v towards its references (a and b), citing papers (c and d), and the query papers ($q_i \in \mathcal{Q}$) weighted according to the parameters (d, κ).

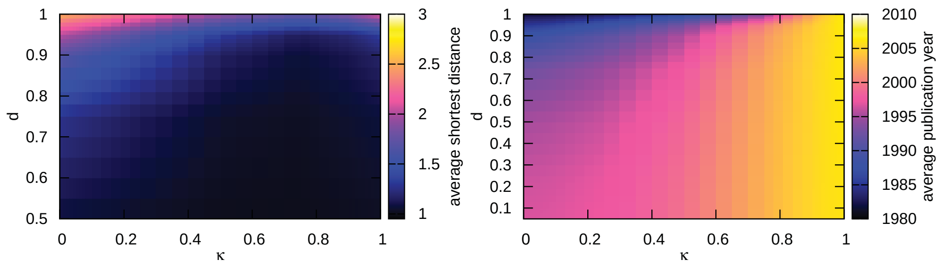


Fig. 3. The average shortest distance from seed papers (left) and publication year (right) of top-10 recommendations by DaRWR based on d and κ. Setting a higher value of d should allow to find relevant papers whose relations to the query are not obvious. κ parameter allows the user to obtain recent papers by setting it close to 1 or finding older papers by setting it close to 0

TABLE I. Results of the experiments with mean average precision (MAP@50) and 95% confidence intervals.

	hide random			hide recent			hide earlier		
	mean	interval		mean	interval		mean	interval	
DaRWR	48.00	46.80	49.20	**42.22**	**40.95**	**43.50**	**60.64**	**59.48**	**61.80**
PaperRank	**56.56**	**55.31**	**57.80**	38.75	37.50	40.00	58.93	57.76	60.10
Katz$_\beta$	46.33	45.16	47.50	34.56	33.42	35.70	44.19	42.97	45.40
Cocitation	44.60	43.39	45.80	14.22	13.25	15.20	55.97	54.64	57.30
Cocoupling	17.28	16.36	18.20	17.56	16.61	18.50	2.93	2.57	3.30
CCIDF	18.05	17.11	19.00	18.97	17.94	20.00	3.55	3.10	4.00

that increasing d leads to earlier papers since they tend to accumulate more citations. But for a given κ, varying the damping factor does not allow to reach a large diversity of time frames. The direction-awareness parameter κ can be adjusted to reach papers from different years with a range from late 1980's to 2010 for almost all values of d. In our online service, the parameter κ can be set to a value of user's preference.

Accuracy: We test the quality of the recommended citations by different methods in three different scenarios: The **hide random** scenario represents the typical use-case where a researcher is writing a paper and trying to find some more references. To simulate that, a source paper s with more than 20 references is randomly selected from the papers published between 2005 and 2010. Then we remove s and all the papers published after s from the graph. $\delta^+(s)/10$ references of s are randomly selected as the hidden set, and the rest is used as the query papers. We compute the citation recommendations on $\mathcal{Q}$ and report the mean average precision (MAP) of finding hidden papers within the top-50 recommendations for 2500 independent queries. In the **hide recent** scenario, the hidden paper are chosen as the most recent references. Similarly, we define **hide earlier** where the hidden papers are the oldest publications.

The methods are compared on three scenarios against widely-used citation based approaches: bibliographic coupling [10], Cocitation [11], CCIDF [12], PaperRank [13] and the Katz distance [14]. The parameters that lead to the best accuracy in different experiments are selected, i.e., $d = 0.75$ for both PaperRank and DaRWR, and $\kappa = 0.75, 0.95, 0.25$ for hide random, recent, and earlier scenarios, respectively.

Table I presents the MAP scores on there scenarios with their 95% confidence intervals. Cocoupling, CCIDF and Cocitation never perform best. Although PaperRank achieves the best results when the query is generic (on the hide random scenario); its direction-aware variant leads to a higher accuracy when the query is targeted. In each scenario, the confidence interval of the method that performs the best does not intersect with the intervals of other methods, indicating that their dominance is statistically significant.

IV. Exploratory Search

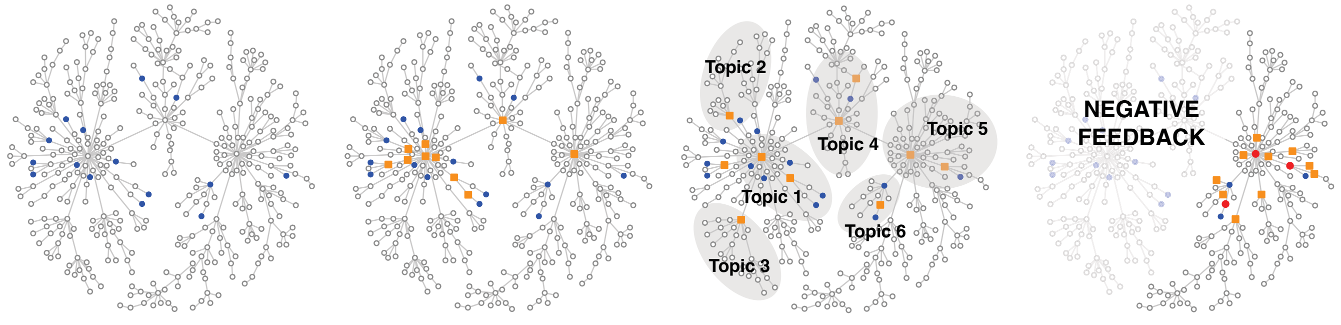
In our service, we provide the exploratory search functionality with (a) **result diversification** so that researchers may find papers from different aspects of the query, (b) **relevance feedback** so that researchers can focus on only the areas that they are interested in, and (c) **visualization** so that relevant papers that did not appear in the top-10 results are visible with their relationships to the seed papers as well as other recommendations. We give the details of each component and their experimental results below; however, since the exploratory search is more of a personalized and subjective matter, it is typically difficult to quantify its effects. We give an overview of how result diversification, relevance feedback, and visualization are useful in Figure 4.

A. *Result diversification*

Methods such as DaRWR tend to naturally return many recommendations from the same area of the graph, which leads to a poor coverage of the potential interests of the users. Diversifying the recommendations refers to the methods that increase the amount of distinct information one can reach via an automatized search. Diversification for the random-walk-based methods attracted attention recently: GrassHopper addresses diversified ranking on graphs by vertex selection with absorbing random walks [15]. DivRank uses a greedy vertex selection process and updates the transition matrix at each iteration with respect to the current node ranks by introducing a *rich-gets-richer* mechanism to the ranking [16]. Dragon approaches the problem from an optimization point, proposes the *goodness* measure to combine relevancy and diversity, and presents a near-optimal algorithm [17].

We can use any of these algorithms in the**advisor** since they can be easily enhanced with the direction-awareness property; however, the main criteria we need to consider here is the efficiency, and hence, whether or not the diversification algorithm can be used in a real-time service. We previously showed that GrassHopper has a high time complexity and it is not scalable to large graphs; DivRank updates the full transition matrix in each iteration, hence more iterations are needed for its convergence; and Dragon could not provide a diverse enough set of results [3].

We argue that finding the vertices which are locally maximum in the graph with respect to their ranks and returning the k most relevant ones will diversify the results and increase the coverage of citation graph. Once the ranks are computed, the straightforward approach for identifying the local maxima is to iterate over each node in the graph and check if its rank is greater than all of its neighbors' with a $\mathcal{O}(|E|)$ algorithm. The



(a) Query papers (b) DaRWR results (c) Diversified results (d) DaRWR results after feedback

Fig. 4. Exploratory properties on a sample graph. Seed papers in $\mathcal{Q}$ are highlighted in (a), top-k recommendations of DaRWR are dominated by the papers in the left part of the graph (b), diversified results with k-RLM introduce papers from different aspects of the query (c). When positive feedback is given to the papers in Topics 5 and 6, and negative feedback is given to the rest of the results, the recommendations are more focused on the right part of the graph (d).

top-k results		**k-RLM diversified**	
paper	label	paper	label
Govan09	GM	Govan09	GM
Kourtis08	C	Kourtis08	C
Lao10	GM	Lao10	GM
Abbey10	GM	Bradley10	GM
Bradley10	GM	Hoemmen10	Sp
Hoemmen10	Sp	**Saak64**	GPU
Knight06	GM	**Guo10**	GPU
Davis97	P	**Lee10**	MC
Toledo97	Sp	**Im04**	GPU
Im00	Sp	**Kaiser10**	MC

topic	**query**	**top-k**	**k-RLM**
Multicore	2	0	2
GPU	2	0	3
Eigensolver	1	0	0
Graph Mining	3	5	3
Compression	1	1	1
Generic SpMV	1	3	1
Partitioning	1	1	0

Fig. 5. Comparison of the recommendations of DaRWR and diversified DaRWR for a given query related to SpMV optimization for emerging architectures. Without diversification, five out of ten recommendations are related to graph mining, where three of them are neighbors. Diversification allows to cover more topics by eliminating redundant results and including items from uncovered topics.

drawback of diversifying with local maxima is that for large k's (i.e., $k > 10$), the results of the recommendation algorithm are generally no longer related to the queried seed papers. Popular papers in unrelated fields can be returned, e.g., a set of well-cited physics papers for a computer science related query. Although this might improve the diversity, it hurts the relevancy, hence, the results will no longer be useful to the user. In order to keep the results within a reasonable relevancy threshold and to diversify them at the same time, we relax the algorithm by incrementally getting local maxima only within the top-γk results and removing the selected vertices from the subgraph for the next local maxima selection until $|S| = k$. We refer to this algorithm as parameterized relaxed local maxima (γ-RLM). Note that 1-RLM reduces to DaRWR.

Experiments: We have given the quantitative analysis of γ-RLM and compared against the listed diversification methods with various relevance and diversity measures in [3]. Here, we try to exemplify the effects of reranking the recommendations with a diversification method on a real world query[1]. The recommendations are diversified, visualized within the**advisor**, and manually clustered. The number of recommendations belonging to the clustered topics are given in Fig. 5.

The query is the bibliography of a submitted paper related to SpMV optimization for emerging architectures, hence a multidisciplinary paper. Although the query includes a couple of graph mining papers, 50% of top-k recommendations are related to graph mining. k-RLM diversification improves the results by eliminating redundant ones and covering other topics. Indeed, no results from the Multicore and GPU categories were returned before. After diversification, these two topics are now covered. If the user is interested in finding more GPU related papers she may give a positive feedback on the current set and refine the search without diversification. We believe that diversification is an essential part of the paper recommendation process, especially when the query is composed of multidisciplinary papers and/or papers from separate disciplines.

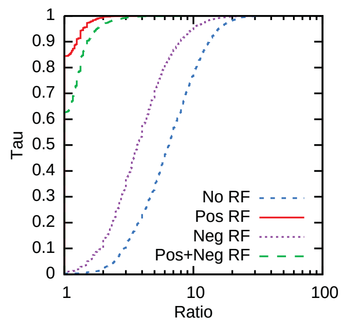
[1]The query is available at http://theadvisor.osu.edu/csfeedback.php?q=e302d9fea1f22310cbf64c39a0a20d4e.ris,0.75

B. *Relevance feedback*

Relevance feedback is an important part of the recommendation system since users may give positive and negative feedbacks on the results in order to reach to desired papers or topics. Users of the**advisor** are given the option of providing explicit relevance feedback to the set of recommended papers. The feedbacks can be either positive or negative, making a recommendation relevant or irrelevant for the query. When the user refines the query with the relevance feedback, the relevant results are added to $\mathcal{Q}$, and the irrelevant results are removed from the citation graph with all of their incident edges.

Experiments: In this test, 2500 source papers are randomly selected. For each source paper s, the graph is pruned by removing the papers published after s. Then, a target paper t is selected from the pruned graph, such that it is the most relevant paper at distance 3 from the source (i.e., $t = \text{argmax}_{t \in V} \mathbf{p}_t$, s.t. $d(s,t) = 3$ and year[t]$\leq$year[s]). Assuming that a user can only display 10 results at a time, we measure the number of pages that the user has to go through until she reaches t. We compare the feedback mechanism with the following behaviors: (1) **No feedback:** There is no feedback mechanism; the user should keep going to the next page until she finds the target paper. (2) **Only positive feedback (+RF):** Relevant results are added to $\mathcal{Q}$ in the next step, irrelevant results should not be displayed again. (3) **Only negative feedback (-RF):** Irrelevant results are removed from the graph, relevant results are kept but will not be displayed again. (4) **Both positive and negative ($\pm$RF):** Results are labeled as either relevant to be added to $\mathcal{Q}$ or irrelevant to be removed from the graph.

Figure 6-(left) presents the performance profile of the experiments. A feedback policy passing through point $(ratio, \tau)$ achieves a result at worse $ratio$ times the number of pages of the best policy in a fraction τ of the case. Figure 6-(right) presents the percentage of pages needed to find a target paper at distance-3 with positive and/or negative relevance feedback compared to not using feedback at all. Using negative feedback only reduces the number of pages one has to go through by 30.01% on average and using positive feedback allows to



Q	−RF	+RF	±RF
1	88.23	**17.64**	**17.64**
2	66.66	**33.33**	44.44
3	36.84	**10.52**	13.15
4	96.66	36.66	**33.33**
5	78.94	**31.57**	42.10
6	55	35	**30**
7	30.43	34.78	**26**
8	100	**15**	**15**
9	70	**30**	**30**
10	**22.22**	103.7	40.74
. . .			
AVG	**66.99**	**19.61**	**22.85**

Fig. 6. Relevance feedback experiments: performance profiles (left) and sample of the number of pages one has to go through expressed as a percentage of the number of pages without using any feedback (right), e.g., using only positive feedback allows to reduce the number of pages by 80.39% on average.

reduce the number of pages by 80.39% on average. Using both negative and positive feedback reduces the number of pages by 77.15% on average. The results show that the feedback mechanism, especially positive feedback, allows to speedup the process of searching for specific references.

C. Data visualization

Representation of the recommendations in a web service is crucial for user experience. Aside from displaying the full bibliographic entries for the list of suggested papers, we also visualize the results and their relationships to the seed papers and papers that are given positive feedback before. The sample graph (see Fig. 7) consist of the seed papers $\mathcal{Q}$ (*blue*), recommendations (*green*), and top-100 relevant papers (*white*). The subgraph is extracted from those vertices and all the edges within this subset. We then apply a force-directed layout algorithm [18] to improve the representation as well as expose the paper clusters.

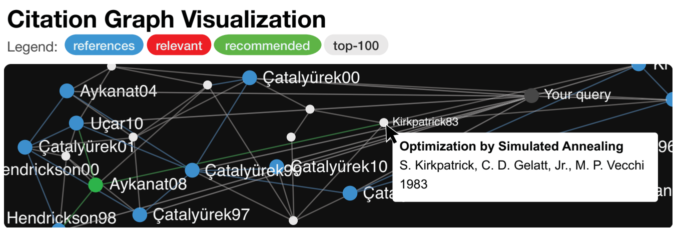

Citation Graph Visualization
Legend: references relevant recommended top-100
Aykanat04
Çatalyürek00
Your query
Uçar10
Kirkpatrick83
Çatalyürek01
Optimization by Simulated Annealing
S. Kirkpatrick, C. D. Gelatt, Jr., M. P. Vecchi
1983
endrickson00
Çatalyürek99
Çatalyürek10
Aykanat08
Hendrickson98
Çatalyürek97

Fig. 7. Visualization of a sample query.

V. Efficiency and Scalability

As described above, each iteration of the ranking algorithm DaRWR contains a sparse-matrix dense-vector multiplication (SpMV). This sparse linear algebra kernel is the building block of the**advisor**'s recommendation engine. With a straightforward and traditional implementation of SpMV, the ranking algorithm itself was taking around 3.5 seconds with full citation graph on a cutting edge CPU. Hence, the ranking algorithm needs to be optimized for an efficient web service.

We first observed that the nonzero pattern of the citation matrix is highly irregular and the computation suffers from this irregularity due to the high number of cache misses. Since there will be a penalty for each cache miss, we apply preprocessing steps, partitioning and ordering, to reduce the number of cache misses and make the ranking algorithm faster. Figure 8 shows the sparsity pattern of the matrix before (a) and after (b) the preprocessing obtained by using a hypergraph partitioning model and the Reverse Cuthill-McKee (RCM) [19] heuristic. In the original matrix, the nonzeros are distributed randomly. After the preprocessing, 83% of the nonzeros are placed in the diagonal blocks of the ordered matrix. Note that the preprocessing step is query oblivious, and it is only done once when the graph is updated. We also experimented with two other ordering heuristics, i.e., Approximate Minimum Degree (AMD) [20] and SlashBurn [21]. Some of these experiments with three different SpMV implementations can be found in [4], [5].

The first implementation, CRS-Full, is the straightforward implementation of DaRWR, assuming the citation matrix $\mathbf{A}$ is stored in compressed row storage (CRS) format. In CRS, the nonzeros in the same row are stored adjacently in two arrays which contain the column indices and the nonzero values. An additional array is used to mark the first nonzero for each row in these arrays. At each iteration, the probability of each vertex x is distributed among its neighbors unless its rank $\mathbf{p}_{t-1}(x)$ is zero. In the second implementation CRS-Half, we only use the reference edges (so we omit half of the nonzeros in the full matrix). For the remaining edges, the values of the nonzero elements are not explicitly stored. Instead, before iteration t, we scale the vector entry $\mathbf{p}_{t-1}(v)$ with $\frac{d(1-\kappa)}{\delta^+(v)}$ to find the nonzero values at row v. For contributions due to the citation edges of v (column v of the matrix), we use $\frac{d\kappa}{\delta^-(v)}$ to scale $\mathbf{p}_{t-1}(v)$. In the third implementation, COO-Half, we used the optimizations described for CRS-Half but store the row and column coordinates of the nonzeros explicitly.

The CRS-based algorithms can avoid some updates if they have no effect on $\mathbf{p}_t$, i.e., with a simple check $\mathbf{p}_{t-1}(v) = 0$ for each v. Since COO-Half stores and processes the nonzeros independently, it needs to do the check for each nonzero and this would be very expensive. CRS-Full can avoid roughly 12 million nonzeros/updates in the first iteration. This number is roughly 6 million for CRS-Half. COO-Half traverses all 12 million nonzero elements and does the corresponding updates even if most of them have no effect for the first couple of iterations. However, the number of positive values in $\mathbf{p}_{t-1}$ increases exponentially during the first iterations. As Figure 9 shows, the shortcuts in CRS-based algorithms are extremely useful for a couple of iterations. The figure also implies that the citation graph is highly connected since DaRWR and seems to traverse almost all the nonzeros in $\mathbf{A}$. That is, random walks and paths can reach to almost all vertices in the graph. We observed that 97% of the vertices of the citation graph G are in a single connected component.

Based on the observation that CRS-Full is very efficient for the first couple of iterations, and COO-Half is efficient overall, we propose to combine those methods to build a **Hybrid** algorithm that runs CRS-Full at the beginning, and switches to COO-Half when the CRS-Full iterations take longer than COO-Half. When the service starts, multiple recommendations are performed with CRS-Full and COO-Half. The time of each iteration is logged and used to decide when COO-Half becomes more efficient that CRS-Full. In the**advisor**, the first four iterations are performed with CRS-Full, and subsequent iterations are performed with COO-Half.

Experiments: We run the experiments on an architecture with a 2.27GHz quad-core *Intel Xeon* (Bloomfield) CPU and 48GB of main memory. Each core has 32KB L1 and 256KB L2

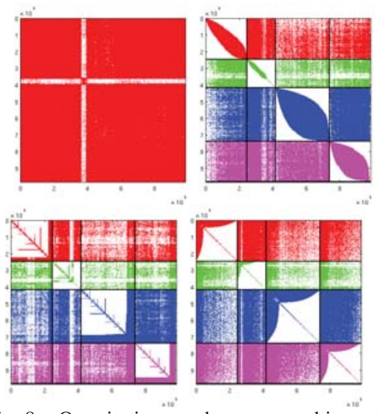


Fig. 8. Our citation graph represented in matrix form (a), the row/column ordered matrix with RCM (b), AMD (c), SlashBurn (d).

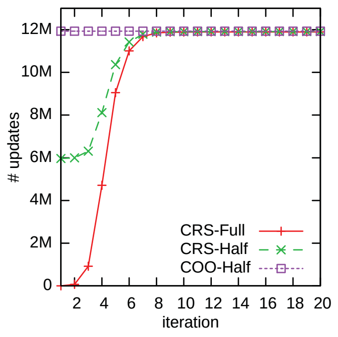


Fig. 9. Number of updates per iteration. CRS-Full and CRS-Half can avoid roughly all and half of updates in the first iteration, respectively. COO-Half does all the updates even if most of them have no effect for the first iterations.

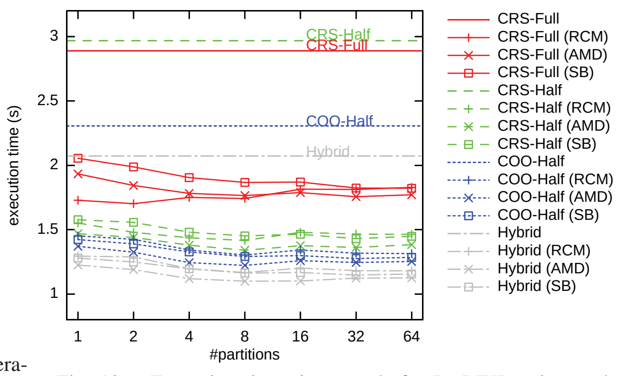


Fig. 10. Execution times in seconds for DARWR using each algorithm with different Ks and ordering heuristics. The values are the averages of the running times for 286 queries. For each query, DARWR performs 20 iterations.

caches and each socket has an 8MB L3 cache. All of the algorithms are implemented in C++. The compiler `gcc` and the `-O2` optimization flag are used. For the experiments, we use only one core from each processor. For DARWR, we use the default values of the service, which are $d = 0.8$, $\kappa = 0.75$.

Figure 10 shows the execution time of each algorithm on the citation graph partitioned with different Ks (i.e., $K = 1, 2, 4, 8, 16, 32, 64$) and ordered with different heuristics (i.e., RCM, AMD, SB). The simple implementation using CRS-Half takes 2.96 seconds. Using COO-Half drops the runtime to 2.3 seconds. Partitioning the matrix in 8 parts and ordering it with AMD allow to reach a runtime of 1.22 seconds [4].

Hybrid lines represent the algorithm we propose in Section V. The hybrid implementation was able to reduce the best execution time from 1.22 seconds to 1.09 seconds, which is more than 10% improvement. Note that in SpMV operations, it is very hard to obtain linear speedup with shared memory parallelization. Hence, to maximize the throughput we chose to use one processor per query. The proposed algorithm will further reduce the latency and improve the throughput of the service, as well as user experience.

VI. Conclusions

In this work, we identify the properties that an academic recommendation service should provide to its users as (1) personalized search, (2) exploration of the data, and (3) efficient and scalable methods. We argue that the existing academic services lack some of the mentioned properties. We introduce the**advisor**, a service that exhibits those properties. Personalization is achieved by the user indicating a set of relevant documents. Exploration is achieved by three techniques leading to an overall description of the area (diversification), guiding technique to reinforce interest (relevance feedback) and manual discovery (visualization). The efficiency of the service is ensured by classical HPC techniques. All the features of the system are based on sound algorithmic decisions and are shown to be very beneficial in practice. We believe that the**advisor** will find its place in the ecosystem of academic web services.

Currently, the service only uses textual information for displaying them. In the future, we will use such information to capture the intent of citations and to obtain topical information.

Acknowledgment

This work was partially supported by the U.S. Department of Energy SciDAC Grant DE-FC02-06ER2775 and NSF grants CNS-0643969, OCI-0904809 and OCI-0904802. The authors also would like to thank DBLP and CiteSeer for making their data available.

References

[1] Q. He, J. Pei, D. Kifer, P. Mitra, and L. Giles, “Context-aware citation recommendation,” in *Proc. WWW*, 2010.

[2] O. Kucuktunc, E. Saule, K. Kaya, and U. V. Catalyurek, “Direction awareness in citation recommendation,” in *Proc. DBRank’12*, 2012.

[3] ——, “Diversifying citation recommendation,” ArXiv, Tech. Rep. arXiv:1209.5809, Sep 2012. [Online]. Available: http://arxiv.org/abs/1209.5809

[4] O. Kucuktunc, K. Kaya, E. Saule, and U. V. Catalyurek, “Fast recommendation on bibliographic networks,” in *Proc. ASONAM*, 2012.

[5] ——, “Fast recommendation on bibliographic networks with sparse-matrix ordering and partitioning,” *Social Network Analysis and Mining (SNAM)*, 2013, (to appear).

[6] T. Strohman, W. B. Croft, and D. Jensen, “Recommending citations for academic papers,” in *Proc. SIGIR*, 2007.

[7] G. Salton, “Associative document retrieval techniques using bibliographic information,” *J. ACM*, vol. 10, pp. 440–457, 1963.

[8] S. Brin and L. Page, “The anatomy of a large-scale hypertextual web search engine,” in *Proc. WWW*, 1998.

[9] T. H. Haveliwala, “Topic-sensitive PageRank,” in *Proc. WWW*, 2002.

[10] M. M. Kessler, “Bibliographic coupling between scientific papers,” *American Documentation*, vol. 14, pp. 10–25, 1963.

[11] H. Small, “Co-citation in the scientific literature: A new measure of the relationship between two documents,” *J. Am. Soc. Inf. Sci.*, vol. 24, no. 4, pp. 265–269, 1973.

[12] S. Lawrence, C. L. Giles, and K. Bollacker, “Digital libraries and autonomous citation indexing,” *Computer*, vol. 32, pp. 67–71, 1999.

[13] M. Gori and A. Pucci, “Research paper recommender systems: A random-walk based approach,” in *Proc. IEEE/WIC/ACM Web Intelligence*, 2006.

[14] D. Liben-Nowell and J. M. Kleinberg, “The link-prediction problem for social networks,” *JASIST*, vol. 58, no. 7, pp. 1019–31, 2007.

[15] X. Zhu, A. B. Goldberg, J. V. Gael, and D. Andrzejewski, “Improving diversity in ranking using absorbing random walks,” in *Proc. HLT-NAACL*, 2007.

[16] Q. Mei, J. Guo, and D. Radev, “DivRank: the interplay of prestige and diversity in information networks,” in *Proc. KDD*, 2010.

[17] H. Tong, J. He, Z. Wen, R. Konuru, and C.-Y. Lin, “Diversified ranking on large graphs: an optimization viewpoint,” in *Proc. KDD*, 2011.

[18] I. G. Tollis, G. Di Battista, P. Eades, and R. Tamassia, *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, July 1998.

[19] E. Cuthill and J. McKee, “Reducing the bandwidth of sparse symmetric matrices,” in *Proc. ACM national conference*, 1969.

[20] P. R. Amestoy, T. A. Davis, and I. S. Duff, “An approximate minimum degree ordering algorithm,” *SIAM Journal on Matrix Analysis and Applications*, vol. 17, no. 4, pp. 886–905, 1996.

[21] U. Kang and C. Faloutsos, “Beyond ‘caveman communities’: Hubs and spokes for graph compression and mining,” in *Proc. ICDM*, 2011.