

Cluster Based Personalized Search

Hyun Chul Lee¹ and Allan Borodin²

¹ Thooracom, Toronto, ON, M4W 0A1 leehyun@thooracom

² DCS, University of Toronto, Toronto, ON, M5S 3G4 bor@cs.toronto.edu

Abstract. We study personalized web ranking algorithms based on the existence of document clusterings. Motivated by the topic sensitive page ranking of Haveliwala [20], we develop and implement an efficient “local-cluster” algorithm by extending the web search algorithm of Achlioptas et al. [10]. We propose some formal criteria for evaluating such personalized ranking algorithms and provide some preliminary experiments in support of our analysis. Both theoretically and experimentally, our algorithm differs significantly from Topic Sensitive Page Rank.

1 Introduction

Due to the size of the current Web and the diversity of user groups using it, the current algorithmic search engines are not completely ideal for dealing with queries generated by a large number of users with different interests and preferences. For instance, it is possible that some users might input the query “Star Wars” with their main topic of interest being “movie” and therefore expecting pages about the popular movie as results of their query. On the other hand, others might input the query “Star Wars” with their main topic of interest being “politics” and therefore expecting pages about proposals for deployment of a missile defense system. (Of course, in this example, the user could easily disambiguate the query by adding say “movie” or “missile” to the query terms.) To both expedite simple searches as well as to try to accommodate more complex searches, *web search personalization* has recently gained significant attention for handling queries produced by diverse users with very different search intentions. The goal of web search personalization is to allow the user to expedite web search according to ones personal search preference or context.

There is no general consensus on exactly what web search personalization means, and moreover, there has been no general criteria for evaluating personalized search algorithms. The goal of this paper is to propose a framework, which is general enough to cover many real application scenarios, and yet is amenable to analysis with respect to correctness in the spirit of Achlioptas et al [10] and with respect to stability properties in the spirit of Ng et al. [26] and Lee and Borodin [24] (see also [12, 17]). We achieve this goal by assuming that the targeted web service has an underlying cluster structure. Given a set of clusters over the intended documents in which we want to perform personalized search, our framework assumes that a user’s preference is represented as a preference vector over these clusters. A user’s preference over clusters can be collected either on-line or off-line using various techniques [27, 15, 29, 19]. We do not address how to collect the user’s search preference but we simply assume that the user’s search preference (possibly with respect to various search features) is already available and can be translated into his/her search preference(s) over given cluster structures of targeted documents. We define a class of personalized search algorithms called “local-cluster” algorithms that compute each page’s ranking with respect to each cluster containing the page rather than with respect to every cluster. We propose a specific local-cluster algorithm by extending the approach taken by Achlioptas et al. [10]. Our proposed local-cluster algorithm considers linkage structure and content generation of cluster structures to produce a ranking of the underlying clusters with respect to a user’s given search query and preference. The rank of each document is then obtained through the relation of the given document with respect to its relevant clusters and the respective preference of these clusters. Our algorithm is particularly suitable for equipping already existing web services with a personalized search capability without affecting the original ranking system.

Our framework allows us to propose a set of evaluation criteria for personalized search algorithms. We observe that Topic-Sensitive PageRank [20], which is probably the best known personalized search algorithm

in the literature, is not a local-cluster algorithm and does not satisfy some of the criteria that we propose. In contrast, we show that our local-cluster algorithm satisfies the suggested properties.

Our main contributions are the following.

- We define a personalized search algorithm which provides a more practical implementation of the web search model and algorithm proposed by Achlioptas et al [10].
- We propose some formal criteria for evaluating personalized search algorithms and then compare our proposed algorithm and the Topic-Sensitive PageRank algorithm based on such formal criteria.
- We experimentally evaluate the performance of our proposed algorithm against that of the Topic-Sensitive PageRank algorithm.

2 Motivation

We believe that our assumption that the web service to be personalized admits cluster structures is well justified. For example, we mention:

- *Human generated web directories*: In web sites like Yahoo [7] and Open Directory Project[5], web pages are classified into human edited categories (possibly machine generated as well) and then organized in a taxonomy. In order to personalize such systems, we can simply take the leaf nodes in any pruning of the taxonomy tree as our clusters.
- *Geographically sensitive search engines*: Sites like Yahoo Local [8], Google Local [3] and Citysearch [2] classify reviews, web pages and business information of local businesses into different categories and locations (e.g., city level). Therefore, in this particular case, a cluster would correspond to a set of data items or web pages related to the specific geographic location (e.g. web pages about restaurants in Houston, TX).

We note that the same corpus can admit several cluster structures using different features. For instance, web documents can be clustered according to features such as topic [7, 5, 6, 4, 1], whether commercial or educational oriented [9], domain type, language, etc. Our framework allows incorporating various search features into web search personalization as it works at the abstract level of clustering structures.

3 Preliminaries

Let G_N (or simply G) be a web page collection (with content and hyperlinks) of node size N , and let q denote a query string represented as a term-vector. Let $\mathcal{C} = \mathcal{C}(G) = \{C_1, \dots, C_m\}$ be a clustering (not necessarily a partition) for G (i.e. each $x \in G$ is in $C_{i_1} \cap \dots \cap C_{i_r}$ for some i_1, \dots, i_r). For simplicity we will assume there is a single clustering of the data but there are a number of ways that we can extend the development when there are several clusterings. We define a **cluster-sensitive page ranking algorithm** μ as a function with values in $[0, 1]$ where $\mu(C_j, x, q)$ will denote the ranking value of page x relative to ³ cluster C_j with respect to query q . We define a user’s preference as a $[0, 1]$ valued function P where $P(C_j, q)$ denotes the preference of the user for cluster C_j (with respect to query q). We call $(G, \mathcal{C}, \mu, P, q)$ an *instance of personalized search*; that is, a personalized search scenario where there exist a user having a search preference function P over a clustering $\mathcal{C}(G)$, a query q , and a cluster-sensitive page ranking function μ . Note that either μ or P can be query-independent.

Definition 1. Let $(G, \mathcal{C}, \mu, P, q)$ be an instance of personalized search. A **personalized search ranking PSR** is a function that maps G_N to an N -dimensional real vector by composing μ and P through a function F ; that is, $PSR(x) = F(\mu(C_1, x, q), \dots, \mu(C_m, x, q), P(C_1, q), \dots, P(C_m, q))$. For instance, F might be defined as a weighted sum of μ and P values.

³ Our definition allows and even assumes a ranking value for a page x relative to C_j even if $x \notin C_j$. Most content based ranking algorithms provide such a ranking and if not, we can then assume x has rank value 0.

4 Previous Algorithms

4.1 Modifying the PageRank algorithm

Due to the popularity of the PageRank algorithm [14], the first generation of personalized web search algorithms are based on the original PageRank algorithm by manipulating the *teleportation factor* of the PageRank algorithm. In the PageRank algorithm, the rank of a page is determined by the stationary distribution of a modified uniform random walk on the web graph. Namely, with some small probability $\epsilon > 0$, a user at page i uniformly jumps to a random page, and otherwise with probability $(1 - \epsilon)$ jumps uniformly to one of its neighboring pages⁴. That is, the transition probability matrix $P_\epsilon^A = \epsilon \cdot U + (1 - \epsilon) \cdot A$; $U = ev^T$ is the teleportation factor matrix, with $e = (1, 1, \dots, 1)$ and v the uniform probability vector defined by $v_i = \frac{1}{N}$; and $A = (a_{ij})$ with $a_{ij} = \frac{1}{\text{outdeg}(i)}$ if (i, j) is an edge and 0 otherwise. The first generation of personalized web search algorithms introduce some bias, reflecting the user’s search preference, by using non-uniform probabilities on the teleportation factor (i.e. controlling v). Among these, we have Topic-Sensitive PageRank [20], Modular PageRank [21] and BlockRank [22]. In this paper, we restrict analysis to Topic-Sensitive PageRank.

Topic-Sensitive PageRank One of the first proposed personalized search ranking algorithms is *Topic-Sensitive* PageRank[20]. It computes a topic-sensitive ranking (i.e. cluster-sensitive in our terminology) by constraining the uniform jumping factor of a random surfer to each cluster. More precisely, let T_j be the set of pages in a cluster C_j . Then, when computing the PageRank vector with respect to cluster C_j , we use the personalization vector v^j where

$$v_i^j = \begin{cases} \frac{1}{|T_j|} & i \in T_j \\ 0 & i \notin T_j \end{cases}$$

The page rank of document x with respect to cluster C_j is then computed as the solution to $TR(x, C_j) = (1 - \epsilon) \cdot A^T \cdot TR(x, C_j) + \epsilon \cdot v^j$. We note that if there exists $y \in C_j$ with a link to x , then $TR(x, C_j) \neq 0$ whether or not $x \in C_j$. During query time, the cluster-sensitive ranking is combined with a user’s search preference. Given query q , using (for example) a multinomial naive-Bayes classifier we compute the class probabilities for each of the clusters, conditioned on q . Let q_i be the i^{th} term in the query q . Then, given the query q , we compute for each C_j the following: $Pr(C_j|q) = \frac{Pr(C_j) \cdot Pr(q|C_j)}{Pr(q)} \propto Pr(C_j) \cdot \prod_i Pr(q_i|C_j)$. $Pr(q_i|C_j)$ is easily computed from the class term-vector D_j . The quantity $Pr(C_j)$ is not as straightforward. In the original Topic-Sensitive PageRank, $Pr(C_j)$ is chosen to be uniform. Certainly, more advanced techniques can be used to better estimate $Pr(C_j)$. To compute the final rank, we retrieve all documents containing all of query terms using a text index. The final query-sensitive ranking of each of these pages is given as follows. For page x , we compute the final importance score $TSPR(x, q)$ as $TSPR(x, q) = \sum_{C_j \in \mathcal{C}} Pr(C_j|q) \cdot TR(x, C_j)$. Topic-Sensitive PageRank algorithm is then a personalized search ranking algorithm with $\mu(C_j, x, q) = TR(x, C_j)$, and $P(C_j, q) = Pr(C_j|q)$.

4.2 Other Personalized Systems

Aktas et al. [11] employ the Topic-Sensitive PageRank algorithm at the level of URL features such as Internet domain names. Chirita et al. [15], extend the Modular PageRank algorithm [21]. In [15], rather than using the arduous process for collecting the user profile as in Modular PageRank[21], the user’s bookmarks are used to derive the user profile. They augment the pages obtained in this way by finding their related pages using Modified PageRank and the HITS algorithms.

Most content based web search personalization methods are based on the idea of re-ranking the returned pages in the collection using the content of pages (represented as snippet, title, full content, etc) with respect to the user profile. Some content analysis based personalization methods consider how to collect user profiles as part of its personalization framework. Liu et al.[25] propose a technique to map a user query to a set of categories, which represent the user’s search intention for the web search personalization. A user profile

⁴ When page i has no hyperlinks (i.e. $\text{outdeg}(i) = 0$) it is customary to let $\epsilon = 1$.

and a general profile are learned from the user’s search history and a category hierarchy respectively. Later, these two profiles are combined to map a user query into a set of categories. Chirita et al. [15] propose a way of performing web search using the ODP (open directory project) metadata. First, the user has to specify a search preference by selecting a set of topics (hierarchical) from the ODP taxonomy. Then, at run-time, the web pages returned by the ordinary search engine can be re-sorted according to the distance between the URL of a page and the user profile. Sun et al. [28] proposed an approach called CubeSVD (motivated by HOSVD, High-Order Singular Value Decomposition) which focuses on utilizing the click-through data to personalize the web search. Note that the click-through data is highly sparse data containing relations among user, query, and clicked web page.

5 Our Algorithm

We propose a personalized search algorithm for computing cluster-sensitive page ranking based on a linear model capturing correlations between cluster content, cluster linkage, and user preference. Our model borrows heavily from the Latent Semantic Analysis (LSA) of Deerwester et al. [16], which captures term-usage information based on a (low-dimensional) linear model, and the SP algorithm of Achlioptas et al. [10], which captures correlations between 3 components (i.e. links, page content, user query) of web search in terms of proximity in a shared latent semantic space. For a given clustering \mathcal{C} , let $CS(x) = \{C_j \in \mathcal{C} | x \in C_j\}$. Given an instance $(G, \mathcal{C}, \mu, P, q)$ of personalized search, a **local-cluster algorithm** is a personalized search ranking such that F is given by $F(\mu(C_1, x, q), \dots, \mu(C_m, x, q), P(C_1, q), \dots, P(C_m, q)) = \sum_{C_j \in CS(x)} P(C_j, q) \cdot \mu(C_j, x, q)$. That is, only preferences for clusters containing a site x will effect the ranking of x .

Our algorithm personalizes existing web services utilizing existing ranking algorithms. Our model assumes that there is a generic page ranking $R(x, q)$ for ranking page x given query q . Using an algorithm to compute the ranking for clusters (described in the next section), we compute the cluster-sensitive ranking $\mu(C_i, x, q)$ as

$$\mu(C_i, x, q) = \begin{cases} R(x, q) \cdot CR(C_i, q) & \text{if } x \in C_i \\ 0 & \text{Otherwise} \end{cases}$$

where $CR(C_i, q)$ refers to the ranking of cluster C_i with respect to query q . Finally $PSR(x, q)$ will be computed as $PSR(x, q) = \sum_{C_j \in CS(x)} P(C_j, q) \cdot \mu(C_j, x, q)$. We call our algorithm PSP (for Personalized SP algorithm) and note that it is a local-cluster algorithm.

5.1 Ranking Clusters

The algorithm for ranking clusters is the direct analogy of the SP algorithm [10] where now clusters play the role of pages. That is, we will be interested in the aggregation of links between clusters and the term content of clusters. We also modify the generative model of [10], so as to apply to clusters. This generative model motivates the algorithm and also allows us to formulate a correctness result for the PSP algorithm analogous to the correctness result of [10]. We note that like the SP algorithm, PSP is defined without any reference to the generative model. Let $\{C_1, \dots, C_m\}$ be a clustering for the targeted corpus. Now following [16] and [10], we assume that there exists a set of k unknown (latent) basic concepts whose combinations represent every topic of the web. Given such a set of k concepts, a *topic* is a k -dimensional vector λ , describing the contribution of each of the basic concepts to this topic.

Authority and Hub values for clusters We first review the notion of a page’s authority and hub values as introduced in Kleinberg [23] and utilized in [10]. Two vectors are associated with a web page x :

- There is a k -tuple $A(x) \in [0, 1]^k$ reflecting the topic on which x is an authority. The i -th entry in $A(x)$ expresses the degree to which x concerns the concept associated with the i -th entry in $A(x)$. This topic vector captures the content on which this page is an authority.
- The second vector associated with x is a k -tuple $H(x) \in [0, 1]^k$ reflecting the topic on which x is a hub. This vector is defined by the set of links from x to other pages.

Based on this notion of page hub and authority values, we introduce the concept of cluster hub and authority values. With each cluster $C_j \in \mathcal{C}$, we associate two vectors:

- The first vector associated with C_j is a k -tuple $\tilde{A}^{(j)}$ which represents the expected authority value that is accumulated in cluster C_j with respect to each concept. We define $\tilde{A}^{(j)}$ as $\tilde{A}^{(j)}(c) = \sum_{x \in C_j} A(x, c)$ where $A(x, c)$ is document x 's authority value with respect to the concept c .
- The second vector associated with C_j is a k -tuple $\tilde{H}^{(j)}$ representing the expected hub value accumulated in cluster C_j with respect to each concept. We define $\tilde{H}^{(j)}$ as $\tilde{H}^{(j)}(c) = \sum_{x \in C_j} H(x, c)$ where $H(x, c)$ is document x 's hub value with respect to concept c .

Link Generation over clusters In what follows, we assume all random variables have bounded range. Given clusters C_p and $C_r \in \mathcal{C}$, our model assumes that the total number of links from pages in C_p to pages in C_r is a random variable with expected value equal to $\langle \tilde{H}^{(p)}, \tilde{A}^{(r)} \rangle$. Note that the intuition is the same as in the link generation model for two arbitrary documents [10]. The more closely aligned the hub topic of the pages in C_p is with the authority topic of the pages in C_r , the more likely it is that there will be a link from a document in C_p to a document in C_r . Therefore, the link generation model among different clusters is described in terms of a $m \times m$ matrix $\tilde{W} = \tilde{H} \cdot \tilde{A}^T$ where the p -th row of \tilde{H} is $(H^{(p)})^T$ and the r -th row of A is $(A^{(r)})^T$. Each entry (p, r) of \tilde{W} represents the expected number of links from C_p to C_r . Let \widehat{W} be the actual link structure of documents for the targeted corpus and let Z be the $n \times m$ indicator matrix Z whose (x, j) entry indicates whether or not page x is part of cluster j . The assumption is that the actual number of links $\overline{W} = Z^T \widehat{W} Z$ from C_p to C_r is an instantiation of the link generation model for clusters.

Term Content Generation over Clusters Once again, our term-content generation model heavily borrows from that introduced in [10]. We assume that there are l terms and the term distributions over clusters are given by the following two distributions:

- The first distribution expresses the expected number of occurrences of terms as authoritative terms within all documents. More precisely, we assume a k -tuple $\tilde{S}_A^{(u)}$ whose c -th entry describes the expected number of occurrences of the term u in the set of *all pure authority documents* in the concept c which are not hubs on anything.
- The second distribution expresses the expected number of occurrences of terms as hub terms within all documents. More precisely, we assume a k -tuple $\tilde{S}_H^{(u)}$ whose c -th entry describes the expected number of occurrences of the term u in the set of *all pure hub documents* in the concept c which are not authorities on anything.

The above distributions can be expressed in terms of two matrices, namely \tilde{S}_A , the $l \times k$ matrix whose rows are indexed by terms, where row u is the vector $(\tilde{S}_A^{(u)})^T$, and \tilde{S}_H , the $l \times k$ matrix, whose rows are indexed by terms, where row u is the vector $(\tilde{S}_H^{(u)})^T$. Our model assumes that terms within cluster C_p having authority value $\tilde{A}^{(p)}$ and hub value $\tilde{H}^{(p)}$ are generated from a distribution of bounded range where the expected number of occurrences of term u is $\langle \tilde{A}^{(p)}, \tilde{S}_A^{(u)} \rangle + \langle \tilde{H}^{(p)}, \tilde{S}_H^{(u)} \rangle$. We describe the term generation model of clusters with a m by l matrix \tilde{S} , where again m is the number of underlying clusters and l is the total number of possible terms, $\tilde{S} = \tilde{H} \cdot \tilde{S}_H^T + \tilde{A} \cdot \tilde{S}_A^T$. The (j, i) entry in \tilde{S} represents the expected number of occurrences of term i within all documents in cluster j . Let \widehat{S} be the actual term-document matrix of all documents in the targeted corpus. Analogous to the previous link generation model of clusters, we assume that $\overline{S} = Z^T \widehat{S}$ is an instantiation of the term generation model of clusters described by \tilde{S} .

User Query The user has in mind some topic on which he wants to find the most authoritative cluster of documents on the topic when he performs the search. The terms that the user presents to the search engine should be the terms that a perfect hub on this topic would use, and then these terms would potentially lead

to the discovery of the most authoritative cluster of documents on the set of topics closely related to these terms. The query generation process in our model is given as follows:

- The user chooses the k -tuple \tilde{v} describing the topic he wishes to search for in terms of the underlying k concepts.
- The user computes the vector $\tilde{q}^T = \tilde{v}^T \tilde{S}_H^T$ where the u -th entry of \tilde{q} is the expected number of occurrences of the term u in a cluster.
- The user then decides whether or not to include term u among his search terms by sampling from a distribution with expectation $\tilde{q}[u]$. We denote the instantiation of the random process by $\bar{q}[u]$.

The input to the search engine consists of the terms with non-zero coordinates in the vector \bar{q} .

Algorithm Description Given this generative model that incorporates link structure, content generation, user preference, and query, we can rank clusters of documents using a spectral method. While the basic idea and analysis for our algorithm follows from [10], our PSP algorithm is different from the original SP algorithm in one substantial aspect. *In contrast to the original SP algorithm which works at the document level, our algorithm works at the cluster level making our algorithm computationally more attractive and consequently more practical⁵.* For our algorithm, in addition to the SVD computation of \bar{M} and \bar{W} matrices, the SVD computation of \bar{S} is also required. This additional computation is not very expensive because of the size of matrix \bar{S} . We need some additional notation. For two matrices A and B with an equal number of rows, let $[A|B]$ denote the matrix whose rows are the concatenations of the rows of A and B . Let $\sigma_i(A)$ denote the i -th largest singular value of a matrix A and let $r_i(A) = \sigma_1(A)/\sigma_i(A) \geq 1$ denote the ratio between the primary singular value and the i -th singular value. Using standard notation for the singular value decomposition (SVD) of matrix $B \in \Re^{n \times m}$, $B = U\Sigma V^T$ where U is a matrix of dimensions $n \times \text{rank}(B)$ whose columns are orthonormal, Σ is a diagonal matrix of dimensions $\text{rank}(B) \times \text{rank}(B)$, and V^T is a matrix of dimensions $\text{rank}(B) \times m$ whose rows are orthonormal. The (i, i) entry of Σ is $\sigma_i(B)$.

The cluster ranking algorithm pre-processes the entire corpus of documents independent of the query.

Pre-processing Step

1. Let $\bar{M} = [\bar{W}^T | \bar{S}]$. Recall that $\bar{M} \in \Re^{m \times (m+l)}$ (m is the number of clusters and l is the number of terms). Compute the SVD of the matrix as $\bar{M}^* = U_{\bar{M}} \Sigma_{\bar{M}} V_{\bar{M}}^T$
2. Choose the largest index r such that the difference $|\sigma_r(\bar{M}^*) - \sigma_{r+1}(\bar{M}^*)|$ is sufficiently large (we require $\omega(\sqrt{(m+l)})$). Let $\bar{M}_r^* = (U_{\bar{M}})_r (\Sigma_{\bar{M}})_r (V_{\bar{M}}^T)_r$ be the rank r -SVD approximation to \bar{M} .
3. Compute the SVD of the matrix \bar{W} as $\bar{W}^* = U_{\bar{W}} \Sigma_{\bar{W}} V_{\bar{W}}^T$
4. Choose the largest index t such that the difference $|\sigma_t(\bar{W}^*) - \sigma_{t+1}(\bar{W}^*)|$ is sufficiently large (we require $\omega(\sqrt{(t)})$). Let $\bar{W}_t^* = (U_{\bar{W}})_t (\Sigma_{\bar{W}})_t (V_{\bar{W}}^T)_t$ be the rank t -SVD approximation to \bar{W} .
5. Compute the SVD of the matrix \bar{S} as $\bar{S}^* = U_{\bar{S}} \Sigma_{\bar{S}} V_{\bar{S}}^T$
6. Choose the largest index o such that the difference $|\sigma_o(\bar{S}^*) - \sigma_{o+1}(\bar{S}^*)|$ is sufficiently large (we require $\omega(\sqrt{(o)})$). Let $\bar{S}_o^* = (U_{\bar{S}})_o (\Sigma_{\bar{S}})_o (V_{\bar{S}}^T)_o$ be the rank o -SVD approximation to \bar{S} .

Query Step

Once a query vector $\bar{q}^T \in \Re^l$ is presented, let $\bar{q}^T = [0^m | \bar{q}^T] \in \Re^{m+l}$. Then, we compute the cluster authority vector $a^T = \bar{q}^T \bar{M}_r^{*-1} \bar{W}_t^*$ where $\bar{M}_r^{*-1} = (V_{\bar{M}}^T)_r (\Sigma_{\bar{M}})_r^{-1} (U_{\bar{M}})_r$ is the pseudo-inverse of \bar{M}_r .

⁵ To the best of our knowledge, the SP algorithm was never implemented. Ignoring any personalization aspects (i.e. setting the preference P to be a constant function), the cluster framework provides a significant computational benefit.

5.2 Final Ranking

Once we have computed the ranking for clusters, we proceed with the actual computation of cluster-sensitive page ranking. Let $a(C_j, q)$ denote the authority value of cluster C_j for query q as computed in the previous section. The cluster-sensitive page rank for page x with respect to cluster C_j is computed as

$$\mu(x, C_j, q) = \begin{cases} R(x, q) \cdot a(C_j, q) & \text{if } x \in C_j \\ 0 & \text{Otherwise} \end{cases}$$

where again $R(x, q)$ is the generic rank of page x with respect to query q .

As discussed in Section 1, we assume that the user provides his search preference having in mind certain clusters (types of documents that he/she is interested). If the user exactly knows what the given clusters are, then he might directly express his search preference over these clusters. However, such explicit preferences will not generally be available. Instead, we consider a more general scenario in which the user expresses his search interests through a set of keywords (terms). More precisely, our user search preference is given by:

- The user expresses his search preference by providing a vector p over terms whose i -th entry indicates his/her degree of preference over the term i .
- Given the vector p , the preference vector over clusters is obtained as $p^T \cdot \tilde{S}^T$.

Let $P(C_j) = \tilde{p}^T \tilde{S}^T(j)$ denote this preference for cluster C_j . The final personalized rank for page x is computed as $PSP(x, q) = \sum_{C_i \in CS(x)} R(x, q) \cdot a(C_i, q) \cdot P(C_i)$

The next theorem formalizes the correctness of our PSP algorithm with respect to the generative model.

Theorem 1. *Assume that the link structure for clusters, term content for clusters and search query are generated as described in our model: \overline{W} is an instantiation of $\tilde{W} = \tilde{H}\tilde{A}^T$, \overline{S} is an instantiation of $\tilde{S} = \tilde{A}\tilde{S}_A^T + \tilde{H}\tilde{S}_H^T$, \overline{q} is an instantiation of $\tilde{q} = v^T \tilde{S}_H^T$, the user's preference is provided by p^T , and $R(q)$ is a vector whose entries correspond to the generic ranks of pages (i.e. $R(x, q)$ corresponds to the generic rank of page x with respect to query q). Additionally, we have*

1. \overline{q} has $\omega(k \cdot r_k(\overline{W})^2 r_{2k}(\overline{M})^2 r_k(G^T))$ terms.
2. $\sigma_k(\overline{W}) \in \omega(r_{2k}(\overline{M}) r_k(G^T) \sqrt{m})$ and $\sigma_{2k}(\overline{M}) \in \omega(r_k(\overline{W}) r_{2k}(\overline{M}) r_k(G^T) \sqrt{m})$,
3. \overline{W} , \overline{HS}_A^T and \overline{S}_H^T are rank k , $\overline{M} = [\overline{W}^T | \overline{S}]$ is rank $2k$, $l = O(m)$, and $m = O(k)$.

then the PSP algorithm computes a vector of personalized ranks that is very close to the correct ranking. More precisely, we have $\frac{\|Z\overline{q}^T \overline{M}^{-1} \overline{W}_i^ \cdot p^T \cdot \overline{S}_o^* R(q) - Zv^T \tilde{A}^T p^T \tilde{S}^T R(q)\|_2}{\|Zv^T \tilde{A}^T p^T \tilde{S}^T R(q)\|_2} \in O(1)$. The proof of this theorem is similar to that of Achlioptas et al. [10].*

6 Personalized Search Criteria

We present a series of results comparing Topic-Sensitive PageRank algorithm and our PSP algorithm with respect to a set of personalized search algorithm criteria that we propose. (Some proofs can be found in the Appendix.) Our criteria are all of the form “small changes in the input imply small changes in the computed ranking”. We believe such criteria have immediate practical relevance as well as being of theoretical interest. Since our ranking of documents produces real authority values in $[0, 1]$, one natural approach is to study the effect of small continuous changes in the input information as in the rank stability studies of [12, 17, 24, 26].

One basic property shared by both Topic-Sensitive PageRank and our PSP algorithm is continuity.

Theorem 2. *Both TSPR and our PSP ranking algorithms are continuous; i.e. small changes in any μ value or preference value will result in a small change in the ranking value of all pages.*

Our first distinguishing criteria is a rather minimal *monotonicity property* that we claim any personalized search should satisfy. Namely, since a (cluster based) personalized ranking function depends on the ranking of pages within their relevant clusters as well as the preference of clusters, when these rankings for a page and cluster preferences are increased, we expect the personalized rating can only improve. More precisely, we have the following definition:

Definition 2. Let $(G, \mathcal{C}, \mu, P, q)$ and $(G, \mathcal{C}, \mu, \tilde{P}, q)$ be two instances of personalized search. Let χ and ψ be the set of ranked pages produced by $(G, \mathcal{C}, \mu, P, q)$ and $(G, \mathcal{C}, \mu, \tilde{P}, q)$ respectively. Suppose that $x \in \chi$, $y \in \psi$ share the same set of clusters (i.e. $CS(x) = CS(y)$), and suppose that $\mu(C_j, x, q) \leq \mu(C_j, y, q)$ and $P(C_j, q) \leq \tilde{P}(C_j, q)$ hold for every C_j that they share. We say that a personalized ranking algorithm is **monotone** if $PSR(x) \leq P\tilde{S}R(y)$ for every such $x \in \chi$ and $y \in \psi$.

We now introduce the idea of “locality”. The idea behind locality is that (small) discrete changes in the cluster preferences should have only a minimal impact on the ranking of pages. The notion of locality justifies our use of the terminology “local-cluster algorithm”. A **perturbation ∂_α of size α** changes a cluster preference vector P to a new preference vector $\tilde{P} = \partial_\alpha(P)$ such that P and \tilde{P} differ in at most α components. Let $P\tilde{S}R$ denote the new personalized ranking vector produced under the new search preference vector \tilde{P} .

Definition 3. Let $(G, \mathcal{C}, \mu, P, q)$ and $(G, \mathcal{C}, \mu, \tilde{P}, q)$ be the original personalized search instance and its perturbed personalized search instance respectively. Let $AC(\partial_\alpha)$, the active clusters, be the set of clusters that are affected by the perturbation ∂_α (i.e., $P(C_j, q) \neq \tilde{P}(C_j, q)$ for every cluster C_j in $AC(\partial_\alpha)$). We say that a personalized ranking algorithm is **local** if for every $x, y \notin AC(\partial_\alpha)$, $PSR(x, q) \leq PSR(y, q) \Leftrightarrow P\tilde{S}R(x, q) \leq P\tilde{S}R(y, q)$ where PSR refers to the original personalized ranking vector while $P\tilde{S}R$ refers to the personalized ranking vector after the perturbation.

Theorem 3. Topic-Sensitive PageRank algorithm is not monotone and not local.

In contrast we show that our PSP algorithm does enjoy the monotone and local properties.

Theorem 4. Any linear local-cluster algorithm (and hence PSP) is monotone and local.

We next consider a notion of stability (with respect to cluster movement) in the spirit of [26, 24]. Our definition reflects the extent to which small changes in the clustering can change the resulting rankings. We consider the following page movement changes to the clusters:

- A **migration** $migr(x, C_i, C_j)$ moves page x from cluster C_i to cluster C_j .
- A **replication** $repl(x, C_i, C_j)$ adds page x to cluster C_j (assuming x was not already in C_j) while keeping x in C_i .
- A **deletion** $del(x, C_j)$ is the deletion of page x from cluster C_j (assuming there exists a cluster C_i in which x is still present).

We define the *size* of these three page movement operations to be $\mu(C_i, x, q) + \mu(C_j, x, q)$ for migration/replication, and $\mu(C_j, x, q)$ for deletion. We measure the size of a collection M of page movements to be the sum of the individual page movement costs. Our definition of stability then is that the resulting ranking does not change significantly when the clustering is changed by page movements of small size.

We recall that each cluster is a set of pages and its induced subgraph, induced from the graph on all pages. We will assume that the μ ranking algorithm is a stable algorithm in the sense of [26, 24]. Roughly speaking, locality of a μ ranking algorithm means that there will be a relatively small change in the ranking vector if we add or delete links to a web graph. Namely, the change in the ranking vector will be proportional the ranking values of the pages adjacent to the new or removed edges.

Definition 4. Let $(G, \mathcal{C}, \mu, P, q)$ and $(G, \mathcal{C}, \mu, \tilde{P}, q)$ be a personalized search instance. A personalized ranking function PSR is **cluster movement stable** if for every set of page movements M there is a β , independent of G , such that

$$\|PSR - P\tilde{S}R\|_2 \leq \beta \cdot \text{size}(M)$$

where PSR refers to the original personalized ranking vector while $P\tilde{S}R$ refers to the personalized ranking vector produced when the set of page movements M has been applied to a given personalized search instance.

Query Used	Categories	Query Used	Categories	Query Used	Categories	Query Used	Categories
middle east	Society/Issues News/Current Events Recreation/Travel	northern light	Science/Astronomy Kids and Teens/School Time Science/Software	popular blog	Arts/Weblogs Arts/Chats and Forums News/Weblogs	jaguar	Recreation/Autos Sports/Football Science/Biology
planning	Home/Personal Finance Shopping/Weddings Recreation/Parties	star wars	Arts/Movies Games/Video Games Recreation/Models	common tricks	Home/Do It Yourself Arts/Writers Resources Games/Video Games	technique	Science/Methods and Techniques Arts/Visual Arts Shopping/Crafts
integration	Computers/Software Health/Alternative Society/Issues	strong man	Sports/Strength Sports World/Deutch Recreation/Drugs	chaos	Science/Math Society/Religion and Spirituality Games/Video Games	vision	Health/Senses Computers/Artificial Intelligence Business/Consumer Goods
proverb	Society/Folklore Reference/Quotations Home/Homemaking	conservative	Society/Politics Society/Religion and Spirituality News/Analysis and Opinion	english	Arts/Education Kids and Teens/School Time Society/Ethnicity	graphic design	Business/Publishing and Printing Computers/Graphics Arts/Graphic Design
fishing expedition	Recreation/Camps Recreation/Outdoors Sports/Adventure Racing	liberal	Society/Politics Society/Religion and Spirituality News/Analysis and Opinion	war	Society/History Games/Board Games Reference/Museums	environment	Business/Energy and Environment Science/Environment Arts/Genres

Table 1. Sample queries and the preferred categories for search used in our experiments

Query	PSP	TSPR	Query	PSP	TSPR	Query	PSP	TSPR	Query	PSP	TSPR
middle east	0.76	0.8	northern lights	0.7	0.8	popular blog	0.93	0.7	jaguar	0.96	0.46
planning	0.96	0.56	star wars	0.6	0.66	common tricks	0.66	0.9	technique	0.96	0.7
integration	0.6	0.16	strong man	0.9	0.86	chaos	0.56	0.56	vision	0.43	0
proverb	0.9	0.83	conservative	0.86	0.76	english	0.8	0.26	graphic design	1	0.73
fishing expedition	0.86	0.66	liberal	0.76	0.73	war	0.83	0.16	environment	0.93	0.5
Average		0.80						0.59			

Table 2. Top 10 precision scores for PSP and Topic-Sensitive PageRank

Theorem 5. *Topic-Sensitive PageRank algorithm is not cluster movement stable.*

Theorem 6. *The PSP algorithm is cluster movement stable.*

7 Experiments

As a proof of concept, we implemented the PSP algorithm and the Topic-Sensitive PageRank algorithm for comparison. In section 7.1, we consider the retrieval effectiveness of our PSP algorithm versus that of the Topic-Sensitive PageRank algorithm. In section 7.2, we briefly discuss experiments regarding monotonicity and locality. A more complete reporting of experimental results can be found on this web site.

As a source of data, we used the Open Directory Project (ODP) ⁶ data, which is the largest and most comprehensive human-edited directory in the Web. We first obtained a list of pages and their respective categories from the ODP site. Next, we fetched all pages in the list, and parsed each downloaded page to extract its pure text and links (without nepotistic links). We treat the set of categories in the ODP that are at distance two from the root category (i.e. the “Top” category) as the cluster set for our algorithms. In this way, we constructed 549 categories (or clusters) in total. The categorization of pages using these categories did not constitute a partition as some pages (5.4% of ODP data) belong to more than one category.

⁶ <http://www.dmoz.com>

7.1 Comparison of Algorithms

To produce rankings, we first retrieved all the pages that contained all terms in a query, and then computed rankings taking into account the specified categories (as explained below). The PSP algorithm assumes that there is already an underlying page ranking for the given web service. Since we were not aware of the ranking used by the ODP search, we simply used the pure PageRank as the generic page ranking for our PSP algorithm. The Topic-Sensitive PageRank was implemented as described in Section 4.1. We used the same $\alpha = 0.25$ value used in [20].

We devised 20 sample queries and their respective search preferences (in terms of categories) as shown in Table 1. The “preferred” categories were chosen as follows: for each query in Table 1, we chose a random subset of the categories given for the top ranked pages returned by the ODP search. For the Topic-Sensitive PageRank algorithm, we did not use the approach for automatically discovering the search preference (See Eq. 4.1) from a given query since we found that the most probable categories discovered in this way were heavily biased toward “News” related categories. Instead, we computed both Topic-Sensitive PageRank and PSP rankings by equally weighting all categories listed in Table 1.

The evaluation of ranking results was done by three individuals: two having CS degrees and one with an engineering degree, all with extensive web search experience. We used the precision over the top-10 ($p@10$) as the evaluation measure using the methodology employed by Tsaparas [30]. That is, for each query we merged the top 10 results returned by both algorithms into a single list. Without any prior knowledge about what algorithm was used to produce the corresponding result, each person was asked to carefully evaluate each page from the list as “relevant” if in their judgment the corresponding page should be treated as a relevant page with respect to the given query and one of the specified categories, or *non-relevant* otherwise. In Table 2, we summarize the evaluation results where the presented precision value is the average of all 3 precision values. These evaluation results suggest that our PSP algorithm outperforms the Topic-Sensitive PageRank algorithm. We also report on the actual produced results in *experimental result 1* of our web page.

To gain further insight, we analyzed the distribution of categories associated with each produced ranking. An ideal personalized search algorithm should retrieve pages in clusters representing the user’s specified categories as the top ranked pages. Therefore, in the list of top 100 pages associated with each query, we computed how many pages were associated with those categories specified in each search preference. Each page p in the list of top 100 pages was counted as $1/|nc(p)|$ where $nc(p)$ is the total number of categories associated with page p . We report on these results in *experimental result 2* of our web page. The results here exclude four queries (strong man popular blog, common tricks, and vision) which did not retrieve a sufficient number of relevant pages in their lists of top 100 pages. Note that using the $1/|nc(p)|$ scoring, the total sum of all three preferred categories for each query was always less than 100 since several pages pertain to more than one category. For several queries in our web page, one can observe that each algorithm’s favored category is substantially different. For instance, for the query “star wars”, the PSP algorithm prefers “Games/Video Games” category while the Topic-Sensitive PageRank prefers “Recreation/Models” category. Furthermore, for the queries “liberal”, “conservative”, “technique”, “english” and “planning” the PSP algorithm and the Topic-Sensitive PageRank algorithm share a very different view on what the most important context associated with “liberal”, “conservative”, “technique”, “english” and “planning” is. One should also observe that when there is a highly dominant query context (e.g. “Society/ Relationships” category for “Society/Issues” category for “integration, and “Arts/Graphic Design” for “graphic design”) over other query contexts, then for both algorithms the rankings are dominated by this strongly dominant category with PSP being somewhat more focused on the dominant category. Finally, averaging over all queries, 86.38% of pages in the PSP list of top 100 pages were found to be in the specified preferred categories while for Topic-Sensitive PageRank, 69.05% of pages in the list of top 100 pages were found to be in the specified preferred categories.

We personally considered a number of queries altering the preferred categories. For “integration”, we considered the single category “Science/Math” and the precision over the top 20 was .5 for PSP and .35 for TSPR. (Both algorithms suffered from uses of the term integration not applying to the calculus meaning.) For the “star wars” query, we added the category “Society/Politics” to our preferred categories. We note that the ODP search does not return any pages within this category. Looking at the top 100 pages returned, PSP returned 3 pages in society/politics not relevant to our query while TSPR returned 33 non relevant pages in

this category. We also considered the query “middle east” (using the single category “Recreation/Travel”), the query “conservative” (using the single category “News/Religion and Spirituality”), and the query “jaguar” (using the single category “Sports/Football”) with regard to precision over the top 10 and observe that PSP performed qualitatively much better than TSPR. We report on these results in the *experimental result 3* of our web page.

We further compared the PSP and TSPR rankings using a variant of the Kendall-Tau similarity measure [20, 18]. Consider two partially ordered rankings σ_1 and σ_2 , each of length n and let U be the union of the elements in σ_1 and σ_2 . Let σ'_1 be the extension of σ_1 , where σ'_1 contains the pages in $\sigma_2 - \sigma_1$ appearing after all the URLs in σ_1 . We do the analogous σ'_2 extension of σ_2 . Using the measure $KTSim(\sigma_1, \sigma_2) = \frac{|\{(u,v): \sigma'_1, \sigma'_2 \text{ agree on order of } (u,v), u \neq v\}|}{|U|(|U|-1)}$, we computed the pairwise similarity between the PSP and TSPR rankings with respect to each query. Averaging over all queries, the KTSim value for the top 100 pages is 0.58 while the average KTSim value for the top 20 pages is 0.43, indicating a substantial difference in the rankings.

7.2 Monotonicity and Locality

How is the absence of monotonicity (as shown in Theorem 3) reflected in the ODP data? We searched our ODP dataset and randomly selected 19,640,761 pairs of sites (x, y) that share precisely one common cluster $C_{I(x,y)}$; i.e. $CS(x) \cap CS(y) = \{C_{I(x,y)}\}$ ⁷ and $TR(x, C_{I(x,y)}) < TR(y, C_{I(x,y)})$. We computed the final $TSPR(x)$ and $TSPR(y)$ by uniformly weighting all 549 categories (or clusters). We found that 431,116 (approximately 2%) of these pairs violated monotonicity; that is, the ranking in $C_{I(x,y)}$ was opposite to the ranking produced by TSPR without favoring any particular category (or clusters). This would lead to a violation of monotonicity in the sense of Theorem 3 if, for example, we generated a query using the intersection of common terms in x and y . We report on the distribution of pairs violating monotonicity in *experimental result 4* of our web page.

We also conducted a study on how sensitive the algorithms are to change in search preferences. We argued that such sensitivity is theoretically captured by the notion of locality in Section 6, and showed that the PSP algorithm is robust to the change in search preferences while Topic-Sensitive PageRank is not. Our experimental evidence indicates that the Topic-Sensitive PageRank algorithm is somewhat more sensitive to the change in search preferences. For each query we randomly chose 7 equally weighted categories so as to define a fixed preference vector. Let Δ_α^N refer to the class of perturbations induced by deleting some set of α categories. To compare the personalized ranking vectors produced under different perturbations, we again use the above KTSim measure [20, 18]. In particular, we varied α as 1, 3, and 5 and for each fixed α and for 5 random $\partial_i \in \Delta_\alpha^N$, we computed the resulting rankings and then all $\binom{5}{2}$ pairwise (KTSim) values considering the top 100 pages. We report on the average pairwise similarity⁸ across all queries for each fixed α in *experimental result 5* of our web page.

8 Online Considerations

Given the dynamic nature of the web, it is important to consider personalized search engines in an online scenario where pages are incrementally added, deleted or updated. As a consequence, clusters are updated and this may in turn result in a desired change of the clustering (i.e. where existing clusters are merged or split). Online considerations have not received much attention in this context.

The preprocessing phase of the PSP algorithm relies on the SVD computation of \overline{M} , representing the linkage and semantic relations between clusters. The online addition, deletion or update of pages would then correspond to the addition, deletion or update of fragments of rows and columns in \overline{M} and the consequent online updating of the SVD. There is a rich literature concerning online SVD updating. Recently, M. Brand [13] proposed a family of sequential update rules for adding data to a “thin” SVD data model, revising or

⁷ The pairs were selected in such way that they were reasonably distributed with respect to the common cluster.

⁸ Fagin et al [18] note that this KTSim variant has a mild personalization factor for items not common to both orderings and hence the rather large values.

removing data already incorporated into the model, and adjusting the model when the data-generating process exhibits non-stationarity. Moreover, he experimentally tested the practicability of the proposed approach in an interactive graphical movie recommender that predicts and displays ratings/rankings of thousands of movie titles in real-time as a user adjusts ratings of a small arbitrary set of movies. By applying such methods, the relevant aspects of the preprocessing phase becomes an online computation.

If the existing clustering structure is modified by (say) merging the existing clusters, (without affecting the existing pages), the online updating of our PSP algorithm can be efficiently implemented by merging ranks of affected clusters while re-normalizing ranks of unaffected clusters. illustrated in the following: We first define what a merging of clusters means. We briefly consider how the online PSP handles merges (and, analogously, splits). By a merging operation μ over a clustering \mathcal{C} , we refer to a transformation that modifies the given clustering, $C = \{C_1, \dots, C_m\}$, into a new clustering such that two clusters are merged. After renaming clusters, the new clustering of G is $\tilde{C} = \{\tilde{C}_1, \dots, \tilde{C}_{m-1}\}$ with $\tilde{C}_{m-1} = C_{m-1} \cup C_m$ and $\tilde{C}_s = C_s$ for every $s \leq m - 2$. When the existing clustering structure is modified by such a merging, the online updating of the PSP cluster ranking can be efficiently implemented by merging ranks of affected clusters while re-normalizing ranks of unaffected clusters as justified by the following:

Theorem 7. *Let*

$$A = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 & 0 & 0 \\ 0 & 0 & \cdots & 0 & 1 & 1 \end{bmatrix}$$

and let $\bar{q}^T = [0^m | \bar{q}^T]$, and $\bar{q}''^T = [0^{m-1} | \bar{q}^T]$. Let $\bar{M}' = [\bar{W}'^T | \bar{S}']$ and $\bar{M}'' = [\bar{W}''^T | \bar{S}''] = [A\bar{W}'^T A^T | A\bar{S}']$ where \bar{W}'^T, \bar{S}' are the original link and term/document matrices and \bar{W}''^T, \bar{S}'' are the newly created link and term/document matrices (as produced by merging). Then $\bar{q}''^T \bar{M}_r''^{-1} \bar{W}_t'' = \bar{q}^T \bar{M}_r'^{-1} \bar{W}_t' A^T$

To complete the merging operation, we define a preference vector $\tilde{P}(\tilde{C}_{m-1}, q) = \frac{P(C_{m-1}, q) + P(C_m, q)}{2}$ and $\tilde{P}(\tilde{C}_s, q) = P(C_s, q)$ for $s \leq m - 2$. Finally it is important to note that as new pages arrive and are placed into their relevant pages, the PSP ranking will only change gradually.

References

1. About.com. <http://www.about.com>.
2. Citysearch. <http://www.citysearch.com>.
3. Google local. <http://local.google.com>.
4. Google news. <http://news.google.com>.
5. Open directory project. <http://www.dmoz.org>.
6. Topix. <http://www.topix.net>.
7. Yahoo. <http://www.yahoo.com>.
8. Yahoo local. <http://local.yahoo.com>.
9. Yahoo! mindset. <http://mindset.research.yahoo.com>.
10. D. Achlioptas, A. Fiat, A. R. Karlin, and F. McSherry. Web search via hub synthesis. In *FOCS*, pages 500–509, 2001.
11. M. Aktas, M. Nacar, and F. Menczer. Personalizing pagerank based on domain profiles. In *WebKDD*, 2004.
12. A. Borodin, G. O. Roberts, J. S. Rosenthal, and P. Tsaparas. Link analysis ranking: algorithms, theory, and experiments. *ACM Trans. Internet Techn.*, 5(1):231–297, 2005.
13. M. Brand. Fast online svd revisions for lightweight recommender systems. In *SDM*, 2003.
14. S. Brin and L. Page. The anatomy of a large-scale hypertextual search engine. In *Computer Networks*, pages 107–117, 1998.
15. P. A. Chirita, W. Nejdl, R. Paiu, and C. Kohlschuetter. Using odp metadata to personalize search. In *SIGIR*, 2005.
16. S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the Society for Information Science*, 41(6):391–407, 1990.

17. D. Donato, S. Leonardi, and P. Tsaparas. Stability and similarity of link analysis ranking algorithms. In *ICALP*, pages 717–729, 2005.
18. R. Fagin, R. Kumar, and D. Sivakumar. Comparing top k lists. In *SODA*, pages 28–36, 2003.
19. P. Ferragina and A. Gulli. A personalized search engine based on web-snippet hierarchical clustering. In *WWW (Special interest tracks and posters)*, pages 801–810, 2005.
20. T. H. Haveliwala. Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):784–796, 2003.
21. G. Jeh and J. Widom. Scaling personalized web search. In *WWW*, pages 271–279, 2003.
22. S. Kamvar, T. Haveliwala, C. Manning, and G. Golub. Exploiting the block structure of the web for computing pagerank. Technical Report Stanford University Technical Report, Stanford University, March 2003.
23. J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, 1999.
24. H. C. Lee and A. Borodin. Perturbation of the hyper-linked environment. In *COCOON*, pages 272–283, 2003.
25. F. Liu, C. T. Yu, and W. Meng. Personalized web search by mapping user queries to categories. In *CIKM*, pages 558–565, 2002.
26. A. Y. Ng, A. X. Zheng, and M. I. Jordan. Link analysis, eigenvectors and stability. In *IJCAI*, pages 903–910, 2001.
27. F. Qiu and J. Cho. Automatic identification of user interest for personalized search. In *WWW*, 2006.
28. J.-T. Sun, H.-J. Zeng, H. Liu, Y. Lu, and Z. Chen. Cubesvd: a novel approach to personalized web search. In *WWW*, pages 382–390, 2005.
29. J. Teevan, S. T. Dumais, and E. Horvitz. Personalizing search via automated analysis of interests and activities. In *SIGIR*, pages 449–456, 2005.
30. P. Tsaparas. Application of non-linear dynamical systems to web searching and ranking. In *PODS*, pages 59–70, 2004.

9 Appendix

Proof-Sketch for Theorem 2

The continuity of Topic-Sensitive PageRank and PSP easily follow from the way how these algorithms produce the final ranking. Both algorithms linearly combine μ and P to produce the final ranking. That is, for both algorithms the final rank vector $FR(q)$ with respect to query q can be written as $FR(q) = \Gamma(q) \cdot P(q)$ where $\Gamma(q)$ is a $n \times m$ matrix whose (i, j) th-entry denotes $\mu(C_j, x_i, q)$, and $P(q)$ denotes the cluster preference vector.

We first prove the continuity of algorithms with respect to cluster preference vector. Given $\epsilon > 0$, we have $\|\Gamma(q) \cdot P(q) - \Gamma(q) \cdot \tilde{P}(q)\|_2 \leq \|\Gamma(q)\|_F \|P(q) - \tilde{P}(q)\|_2 < \epsilon$. Therefore, $\delta = \frac{\epsilon}{m}$ would be sufficient for achieving the continuity of algorithms with respect to cluster preference vector. The continuity with respect to μ can be proved in a similar fashion.

Proof of Theorem 3

Non-monotonicity of TSPR: Suppose that G is a graph that consists of 4 points $\{x_1, x_2, x_3, x_4\}$. Let $C = \{C_1, C_2, C_3\}$ be a clustering of G such that $x_1, x_2 \in C_1$, $x_3 \in C_2$, and $x_4 \in C_3$. Let assume that $x_3 \rightarrow x_1$, $x_4 \rightarrow x_1$, and $x_4 \rightarrow x_2$. In addition, we assume $\epsilon \geq 0.25$. We have $TR(x_1, C_1) = \frac{\epsilon}{2} + (1 - \epsilon)$, $TR(x_2, C_1) = \frac{\epsilon}{2} + (1 - \epsilon)$, $TR(x_1, C_2) = (1 - \epsilon)\epsilon$, $TR(x_1, C_3) = (1 - \epsilon)\frac{\epsilon}{2}$, and $TR(x_2, C_2) = 0$, $TR(x_2, C_3) = (1 - \epsilon)\frac{\epsilon}{2}$. Moreover, we assume $P(C_1, q) = \frac{2}{5}$, $P(C_2, q) = 1$, $P(C_3, q) = 1$, $\tilde{P}(C_1, q) = \frac{3}{5}$, $\tilde{P}(C_2, q) = 1$ and $\tilde{P}(C_3, q) = 1$. Therefore, all conditions of monotonicity are satisfied. However, we have

$$\begin{aligned}
TSPR(x_1) &= P(C_1, q)TR(x_1, C_1) + P(C_2, q)TR(x_1, C_2) + P(C_3, q) \cdot TR(x_1, C_3) = \frac{2}{5}(\frac{\epsilon}{2} + (1 - \epsilon)) + (1 - \epsilon)\epsilon + \\
(1 - \epsilon)\frac{\epsilon}{2} &> \frac{3}{5}(\frac{\epsilon}{2} + (1 - \epsilon)) + (1 - \epsilon)\frac{\epsilon}{2} = T\tilde{S}PR(x_2) = \tilde{P}(C_1, q)TR(x_2, C_1) + \tilde{P}(C_2, q)TR(x_2, C_2) \\
&+ \tilde{P}(C_3, q) \cdot TR(x_2, C_3)
\end{aligned}$$

Non-locality of TSPR: In particular, we show that a small perturbation in preference values can have considerably large impact on the overall ranking. Let $G = C_1 \sqcup C_2 \sqcup C_3 \sqcup C_4$, $|C_1| = |C_2| = N - \beta$ and $|C_3| = |C_4| = \beta$ where β is a fixed constant. Every page in $C_3 \sqcup C_4$ points to every page in $C_1 \sqcup C_2$.

One can verify that for each $x \in C_1$ and $y \in C_2$ we have $TSPR(x, C_1) = TSPR(y, C_2)$, and similarly we have $TSPR(x, C_2) = TSPR(y, C_1)$. Furthermore, $TSPR(x, C_3) = TSPR(x, C_4) = TSPR(y, C_3) = TSPR(y, C_4)$. Now, suppose that the original cluster preferences are altered from $P(C_1, q) = P(C_2, q)$, $P(C_3, q) < P(C_4, q)$ to $\tilde{P}(C_1, q) = \tilde{P}(C_2, q)$, $\tilde{P}(C_3, q) > \tilde{P}(C_4, q)$. From the original cluster preferences, we will have $TSPR(x) < TSPR(y)$ for $x \in C_1, y \in C_2$. On the other hand, from the modified cluster preferences, we will have $T\tilde{S}PR(x) > T\tilde{S}PR(y)$ for $x \in C_1, y \in C_2$. That is, we have shown non-locality. More precisely, $d_r(PSR, P\tilde{S}R) = (N - \beta)^2 \in o((2N)^2)$ as $2N \rightarrow \infty$.

Proof of Theorem 4

Monotonicity: Since by the assumption, for every $C_j \in CS(x) = CS(y)$, we have $P(C_j, q) \leq \tilde{P}(\tilde{C}_j, q)$, and $\mu(C_j, x, q) \leq \mu(C_j, y, q)$, we will have $PSR(x) = \sum_{C_j \in CS(x)} P(C_j, q) \mu(C_j, x, q) \leq \sum_{C_j \in CS(y)} P(C_j, q) \mu(C_j, y, q) = P\tilde{S}R(x)$.

Locality: It easily follows from the fact that the ranking produced local-cluster algorithms are only based on those clusters containing the point to be ranked. Therefore, the original ranking for points in UC is unaffected by the perturbation.

Proof of Theorem 5: We exhibit a counter-example to show that the Topic-Sensitive PageRank is not cluster movement stable. Let G be a graph that consists of $n + 1$ points and 3 clusters C_1, C_2 and C_3 . C_1 contains x_0 , C_2 contains $\{x_2, \dots, x_n\}$ and C_3 contains all points. We have $x_0 \rightarrow x_1, x_n \rightarrow x_1$ and $x_k \rightarrow x_{k+1}$ for every $1 < k < (n - 1)$. Furthermore, suppose that $P(C_1, q) = 1, P(C_2, q) = 0$, and $P(C_3, q) = 0$ ⁹. One can verify that $TSPR(x_0) = TR(x_0, C_1) = \epsilon, TSPR(x_n) = TR(x_n, C_1) = \delta$ and $TSPR(x_m) = TR(x_m, C_1) = (1 - \epsilon)^m(\epsilon + \delta)$ where $\delta = \frac{\epsilon(1-\epsilon)^n}{1-(1-\epsilon)^n}$ for every $1 \leq m < (n - 1)$. On the other hand, one can easily see that $TR(x_i, C_2) = 1/n$ for every $1 \leq i \leq n$. Now, we delete x_1 from C_2 . One can see that $T\tilde{S}PR(x_0, C_1) = \tilde{T}R(x_0, C_1) = 1$, and $T\tilde{S}PR(x_i, C_1) = \tilde{T}R(x_i, C_1) = 0$ for every $1 \leq i \leq n$. We have

$$\frac{\|TSPR - T\tilde{S}PR\|_2}{size(del(x_1, C_2))} = n \cdot \sqrt{((\epsilon - 1)^2 + \sum_{i=1}^{n-1} ((1 - \epsilon)^i(\epsilon + \delta))^2)} = n \sqrt{((\epsilon - 1)^2 + (\frac{1 - (1 - \epsilon)^{2n}}{1 - (1 - \epsilon)^2} - 1)(\epsilon + \delta)^2)} \geq n|(\epsilon - 1)|$$

which is unbounded with respect to n .

Proof-Sketch for Theorem 6

We only consider replication and deletion as migration $migr(x_a, C_i, C_j)$ can be seen as a sequential application of $repl(x_a, C_i, C_j)$ followed by $del(x_a, C_i)$. To simplify our notation, we will simply use R^T to refer to $R^T I_n$ and P^T to refer to $P^T I_m$. Let $R^T Z P^T \omega$ be the ranking before the page movement. Let $\tilde{Z} = Z + E$ be the new matrix representing the page's membership in a cluster where E is given as $E_{a,j} = 1$ if it is $repl(x_a, C_i, C_j)$ and $E_{a,i} = -1$ if it is $del(x_a, C_i)$ while the rest of entries are all zero. Let $R^T \tilde{Z} P^T \tilde{\omega}$ be the ranking after the page movement. We will show that $\frac{\|RZP^T \omega - R\tilde{Z}P^T \tilde{\omega}\|_2}{\|\lambda^R R \lambda^\omega\|_1}$ is bounded by a constant where λ^R is the projection of R over the affected page (e.g. $\lambda_{ij}^R = 1$ for $i, j = a$ and $\lambda_{ij}^R = 0$ otherwise for $repl(x_a, C_i, C_j)$) while P_ω is the projection of ω over the affected clusters (e.g. $\lambda_{fd}^R = 1$ for $f = a, d = i, j$ for $repl(x_a, C_i, C_j)$). We have $\frac{\|RZP^T \omega - R\tilde{Z}P^T \tilde{\omega}\|_2}{\|\lambda^R R \lambda^\omega\|_1} \leq \frac{\|RZP^T \omega - R\tilde{Z}P^T \omega + R\tilde{Z}P^T \omega - R\tilde{Z}P^T \tilde{\omega}\|_2}{\|\lambda^R R \lambda^\omega\|_2} \leq \frac{\|RZP^T \omega - R\tilde{Z}P^T \omega\|_2}{\|\lambda^R R \lambda^\omega\|_2} + \frac{\|R\tilde{Z}P^T \omega - R\tilde{Z}P^T \tilde{\omega}\|_2}{\|\lambda^R R \lambda^\omega\|_2}$. The first term $\frac{\|RZP^T \omega - R\tilde{Z}P^T \omega\|_2}{\|\lambda^R R \lambda^\omega\|_2} = \frac{\|REP^T \omega\|_2}{\|\lambda^R R \lambda^\omega\|_2}$ is trivially bounded by $\frac{|R(x_a, q)P(C_i, q)\mu(C_i, a, q)|}{|R(x_a, q)\mu(C_i, a, q)|} \leq |P(C_i, q)| \leq 1$ for $del(x_a, C_i)$. For $repl(x_a, C_i, C_j)$ it requires some work. Note that we will have $\|REP^T \omega\|_2 = \sqrt{R(x_a)^2 P(C_i)^2 \omega_i^2} \leq \sqrt{R(x_a)^2 \omega_i^2 + R(x_a)^2 \omega_j^2} = \|\lambda^R R \lambda^\omega\|_2$ for $repl(x_a, C_i, C_j)$. Therefore, $\frac{\|REP^T \omega\|_2}{\|\lambda^R R \lambda^\omega\|_2} \leq 1$. The second term, $\frac{\|R\tilde{Z}P^T \omega - R\tilde{Z}P^T \tilde{\omega}\|_2}{\|\lambda^R R \lambda^\omega\|_2}$ is bounded as follows. One should note that $\|\lambda^R R \lambda^\omega\|_2 \geq \frac{1}{\sqrt{2}} \|\lambda^R R \lambda^\omega\|_F \|\tau\|_2 \|\omega\|_2$ where τ is the smallest possible cluster-ranking value (i.e. for a cluster having one page without no links). Therefore, we have $\frac{\|R\tilde{Z}P^T \omega - R\tilde{Z}P^T \tilde{\omega}\|_2}{\|\lambda^R R \lambda^\omega\|_2} \leq$

⁹ Since C_3 contains x_0 , it is not true that $P(C_3, q) = 0$ but when n is sufficiently large $P(C_3, q) \approx 0$. Therefore, we assume that $P(C_3, q) = 0$ for the sake of simplicity

$\sqrt{2} \frac{\|R\tilde{Z}P^T\|_F \|\omega - \tilde{\omega}\|_2}{\|\lambda^R R \lambda^\omega\|_F \|\tau\|_2 \|\omega\|_2}$. But, one can observe that

$$\frac{\|R\tilde{Z}P^T\|_F}{\|\lambda^R R \lambda^\omega\|_F} \leq \frac{\sqrt{\sum_{x_i} \sum_{x_i \in C_j} R(x_i, q)^2 P(C_j, x, q)^2}}{\sqrt{R(x_a, q)^2}} \leq \sqrt{\frac{\sum_{x_i} R^2(x_i, q) \sum_{x_i \in C_j} 1}{R^2(x_a, q)}} \leq \sqrt{\frac{\sum_{x_i} R^2(x_i, q) m}{R^2(x_a, q)}} \leq \sqrt{2m}$$

Moreover, we have $\frac{\|\omega - \tilde{\omega}\|_2}{\|\tau\|_2 \|\omega\|_2} \leq \frac{1}{\tau} (1 + \frac{\|\tilde{\omega}\|_2}{\|\omega\|_2}) \leq \frac{2}{\tau}$. Therefore, $\frac{\|R\tilde{Z}P^T \omega - R\tilde{Z}P^T \tilde{\omega}\|_2}{\|\lambda^R R \lambda^\omega\|_2} \leq \frac{2\sqrt{2m}}{\tau}$.

Once there is a bound for deletion and replication, it is easy to generalize the bound to other page movements as combinations of replications and deletions.

Proof for Theorem 7

One can first verify that $\overline{M}'' = \overline{M}' \lambda^*$ where λ^* is given in the form of

$$\lambda^* = \begin{bmatrix} \Lambda^T & | 0_{l \times l} \\ \hline 0_{l \times m-1} & | I_{l \times l} \end{bmatrix}$$

where $I_{l \times l}$ denotes the identity matrix of size $l \times l$, and $0_{l \times m-1}$ denotes a matrix of size $l \times m-1$ whose each entry's value is 0. Let $\overline{M}' = U_1 D_1 V_1^T$ and $\overline{M}'' = U_2 D_2 V_2^T$ be the SVD compositions for matrices \overline{M}' and \overline{M}'' . We have that the pseudo inverse \overline{M}''^{-1} is given as

$$\begin{aligned} \overline{M}''^{-1} &= V_2 D_2^{-1} U_2^T = V_2 (U_2^{-1} \Lambda U_1 D_1 V_1^T \Lambda^* V_2^{T-1})^{-1} U_2^T \\ &= V_2 V_2^T (\Lambda^{*-1} V_1 D_1^{-1} U_1^T \Lambda^{-1} U_2) U_2^T \\ &= \Lambda^{*-1} V_1 D_1^{-1} U_1^T \Lambda^{-1} = \Lambda^{*-1} \overline{M}'^{-1} \Lambda^{-1} \end{aligned}$$

Thus, one can verify that the Eq. 7 holds since

$$\overline{q}''^T \overline{M}_{r''}^{-1} \overline{W}_t = \overline{q}''^T \Lambda^{*-1} \overline{M}_r^{-1} \Lambda^{-1} \Lambda \overline{W}_t \Lambda^T = \overline{q}^T \overline{M}_r^{-1} \overline{W}_t \Lambda^T$$