Exploiting social bookmarking services to build clustered user interest profile for personalized search

Harshit Kumar *, Sungin Lee, Hong-Gee Kim 1

BIKE Lab, Seoul National University, Yeongun Campus, Republic of Korea

A R T I C L E   I N F O

Article history:
Received 5 January 2013
Received in revised form 19 March 2014
Accepted 8 May 2014
Available online 24 May 2014

Keywords:
User profiling
Clustering
Personalized search
Social bookmarking service

A B S T R A C T

Search engine users tend to write short queries, generally comprising of two or three query words. As these queries are often ambiguous or incomplete, search engines tend to return results whose rankings reflect a community of intent. Moreover, search engines are designed to satisfy the needs of the general populace, not those of a specific searcher. To address these issues, we propose two methods that use Singular Value Decomposition (SVD) to build a Clustered User Interest Profile (CUIP), for each user, from the tags annotated by a community of users to web resources of interest. A CUIP consists of clusters of semantically or syntactically related tags, each cluster identifying a topic of the user’s interest. The matching cluster, to the given user’s query, aids in disambiguation of user search needs and assists the search engine to generate a set of personalized search results. A series of experiments was executed against two data sets to judge the clustering tendency of the cluster structure CUIP, and to evaluate the quality of personalized search. The experiment results indicate that the CUIP based personalized search outperforms the baseline search and is better than the other approaches that use social bookmarking services for building a user profile and use it for personalized search.

© 2014 Elsevier Inc. All rights reserved.

1. Introduction

The abundance of information available on the Web has made search engines (SEs) an indispensable tool. Higher availability of information means that there is a greater chance of finding sought-after information on the Web, but with increased complexity of discovering relevant information. While SEs do a good job of ranking results to maximize global happiness, they fail to do a very good job for specific individuals [39]; it appears that the rankings reflect a community of intent rather than the goals of individuals. There are many reasons for the ineffectiveness of SEs. First, user queries are of poor quality: the average length of user queries ranges between two to three words [36]; such short queries cannot effectively describe the user search intent or user information needs. Second, some queries are polysemous [33]: they have different meanings in different contexts; hence it is impossible for the SE to judge the user intent from the short polysemous queries.

The major shortcoming of SEs is the inability to incorporate user modeling with search and unadaptiveness to individual users. Personalization has emerged as an appealing approach when dealing with the issues caused by the variation of on-line behaviors and individual differences observed in user interests, information needs, search goals, query contexts, and others [5]. Many methods [1,4,8,15,16,26,28,41,44,45] are proposed to study user search behavior and use it to build a profile of
In this section, we first present the state-of-the-art in building a UIP, and discuss the most recent approaches to building a UIP from social bookmarking services for personalized search. Differences in approaches are tabulated in Table 1. Finally, we discuss two well-known approaches to obtaining personalized search results.

### Table 1

A comparison summary of the proposed approaches with the other similar approaches that uses folksonomy for personalized search. (a) Source of terms for constructing a UIP, (b) Web document representation, (c) similarity measure, (d) first-order co-occurrence, (e) second-order co-occurrence, (f) clustering of terms in a UIP, (g) UIP and resource length normalization factor, (h) adaptation/no adaptation/proposed.

<table>
<thead>
<tr>
<th>Method</th>
<th>tfUIP [28]</th>
<th>tfIdfUIP [45]</th>
<th>tfIdfCUIP [34]</th>
<th>svdCUIP</th>
<th>modSvdCUIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) Source of terms for constructing a UIP</td>
<td>Annotations by the community of users to Web documents</td>
<td>User annotations to Web documents</td>
<td>User annotations to Web documents on delicious</td>
<td>Annotations by the community of users to Web documents clicked by the user</td>
<td>Annotations by the community of users to Web documents clicked by the user</td>
</tr>
<tr>
<td>(b) Web document representation</td>
<td>User annotations to Web documents</td>
<td>Resource profile (folksonomy based)</td>
<td>Resource profile (folksonomy based)</td>
<td>Document contents</td>
<td>Document contents</td>
</tr>
<tr>
<td>(c) Similarity measure</td>
<td>Calculates the tfIdf cosine similarity between a UIP and a resource profile of the Web document, Eq. (1)</td>
<td>Calculate the cosine similarity between a global cluster matching a user query and the resource profile of the Web document</td>
<td>Calculate the cosine similarity between the matching cluster in the CUIP to the user query and the document contents, Eq. (8)</td>
<td>Calculates the cosine similarity between the matching cluster in the CUIP to the user query and the document contents, Eq. (8)</td>
<td>Calculates the cosine similarity between the matching cluster in the CUIP to the user query and the document contents, Eq. (8)</td>
</tr>
<tr>
<td>(d) First-order co-occurrence</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>(e) Second-order co-occurrence</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>(f) Clustering of terms in a UIP</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>(g) Adaptation</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>(h) Adaptation</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

This paper makes the following contributions:

1. We propose two methods to build a CUIP for personalized search: one that uses Singular Value Decomposition (SVD) to generate svdCUIP, and the other a variation of SVD, modSVD, to generate a modSvdCUIP. A set of pairs of the form \((t, tw)\), where \(t\) is a tag and \(tw\) is the accumulated weight of the tag \(t\), constitutes a User Interest Profile (UIP). A CUIP is defined as a set of term clusters, where each term cluster consists of semantically related tags of user interests and tag weights.

2. An automatic evaluation method is proposed to test the proposed methods with the baseline search and folksonomy based personalized search approaches.

3. We performed experiments to evaluate the proposed methods on two different data sets. The first data set, called custom data set, was created from the search histories of 12 volunteers. This data set was organized to establish the ground truth for the evaluation of clustering tendency and clustering accuracy of CUIPs generated by the proposed methods. The second data set is a much bigger data set harvested from the AOL search query log. This data set was used to test the improvement in personalized search for the two proposed methods, and their comparisons with other methods.

4. Our results show that personalized search using the modSvdCUIP is better than using the tfUIP (term frequency UIP) [28] and tfIdfUIP (term frequency Inverse Document Frequency UIP) [45], and exhibits modestly better performance than the tfIdfCUIP [34] and svdCUIP. Each cluster, in the cluster structure CUIP, identifies a topic, and the application of CUIP helps disambiguate the context of user query, which is particularly needed for vague queries.

The rest of the paper is organized as follows. Section 2 discusses the related work starting with the traditional approaches to user profiling for personalized search, followed by the current approaches to user profiling that involve social bookmarking services. Section 3 presents our proposed methods. The experiments are detailed in Section 4, and the paper is concluded in Section 5.

### 2. Background and related work

In this section, we first present the state-of-the-art in building a UIP, and discuss the most recent approaches to building a UIP from social bookmarking services for personalized search. Differences in approaches are tabulated in Table 1. Finally, we discuss two well-known approaches to obtaining personalized search results.
2.1. User interest profile – traditional

Google’s innovative page ranking search [6] revolutionized the use of SEs. PageRank uses the citation graph of the Web along with the introduction of link analysis in SE systems. SEs, such as Google, Yahoo, and MSN, do a commendable job for experienced users, but fail to satisfy the needs of naive users. Teevan et al. [39] reported that although SEs provide the best possible result set, they are not satisfactory at individual user levels. Search results can be improved by personalization [4,10,20,26,33,38], by, for example, recommending varying results to different users for the same query. The results are differentiated based on user interests, which are obtained from the user’s UIP. Automatic construction of a UIP usually deals with the observation of user browsing behavior. Kelly and Teevan [20] reviewed several possible approaches to inferring user preferences by categorizing user behavior across many dimensions such as examine, retain, and reference. Agichtein et al. [3] organized user interests as a set of features that are organized into three groups: Query-text, ClickThrough, and Browsing. The Query-text feature includes result title, URL, and summary. The ClickThrough feature includes Click-Frequency (number of clicks for the result), IsClickBelow (whether there was a click on a result below the current URL), and IsClickAbove (whether there was a click on a result above the current URL). The Browsing feature includes TimeOnPage, TimeOnDomain, and the deviation of the dwell time from the expected dwell time for a query. Shen et al. [33] collected user interests from clicked document summaries, titles, Click-Through histories, and query histories that were accumulated over a session. Teevan et al. [38], Chirita et al. [10] used the files on the user’s desktop to construct a UIP. A major limitation of these approaches is that there can be a lot of terms on the user’s desktop, which makes a UIP noisy or misleading. Das et al. [12] used collaborative filtering (CF) for personalization. The underlying assumption of the CF approach is that users who agreed in the past tend to agree again in the future.

A rather simplistic approach to construct a UIP is to explicitly ask a user for his/her topics of interest. The UIP is then used for filtering search results by checking content similarity between the returned Web documents and the UIP. Early versions of Google personalization asked the user to choose the categories of interest. The Google SE applied this information to filter search results. An inherent limitation of this approach is that user interests are subject to changes over time. Moreover, it is shown [9] that users are quite reluctant to provide explicit information about their interests or any explicit feedback on search results. Other important methods using ontologies emerged as well in which a UIP is constructed by classifying Web pages in the user’s web browser cache into appropriate concepts in the reference ontology [17,35] or ODP [10].

2.2. User interest profile – folksonomy based methods

Recently, some research works have investigated social bookmarking services for building and applying a UIP for personalized search [22–24,28,41,45] and resource recommendation [1,2,34,40].

The approaches by Noll and Meinel [28], Xu et al. [45], Vallet et al. [41] for personalized search builds a UIP from the tags that the user uses to annotate resources. A Resource Profile (RP) for a resource is constructed from the tags that the community has used to annotate it. A resource clicked by a user manifests the user’s interest in it and possibly the tags associated with it. Tags assigned by a user to a resource can hardly be a complete description of the resource. However, collective tagging of a resource by a community of users provides a more complete description of it. We believe that there are syntactical differences between the search terms that a user uses and the terms found in a search result document. Each user has a specific vocabulary of terms that he/she uses to formulate a query. And each author of a document has his/her own vocabulary of terms too. Chances are that the vocabularies are different. The rift effectively results in the low similarity score or re-ranking score between the search result and the UIP. Note also that there can exist similarity in semantics among the terms in the user’s UIP and the RP of the result document. If a UIP consists of all the tags, used by a community of users, to annotate the resources of user interests, it is very likely to have a greater correspondence between the UIP and the RPs of result documents. Hence, it is our proposal that a UIP should consist of all the tags used by a community of users to annotate the documents or resources clicked by the user. We have adapted the approaches presented in [28,34,45] to construct a UIP by amalgamation of tags from the RPs of the resources or Web documents clicked by the user. We are of the opinion that any of these approaches can be benefited by the application of SVD, an approach proposed by us to construct a CUIP.

2.3. Personalized search

Pitkow et al. [30] described two approaches to personalizing Web search results: query expansion [10,17,35] and re-ranking of search results [16,21,28,41,44]. In query expansion, user interests are conflated with a given query, and the expanded query is used for searching the Web. For re-ranking of search results, the SE results are re-ranked by computing the similarity between the document contents and the terms in the UIP. The approach we follow is the former one.

The method by Noll and Meinel [28], referred to as tfUIP in our work, re-ranks a document by computing the dimension-less cosine similarity between the tags in the RP of the document and the UIP.

\[
\text{tfUIP}(\text{UIP}, d) = \sum_{t \in \text{UIP}, tf_d(t) > 0} \text{tf}_{\text{UIP}}(t)
\]

(1)
The method by Xu et al. [45], referred to as \( \text{tfIdfUIP} \), re-ranks a document by computing the cosine similarity between the tags in the RP of the document and the terms in the \( \text{UIP} \).

\[
\text{tfIdfUIP}(\text{UIP}, d) = \frac{\sum_i (\text{tf}_d(t) \cdot \text{idf}_d(t) \cdot \text{tfIdf}_d(t))}{\sqrt{\sum_i (\text{tf}_d(t) \cdot \text{idf}_d(t))^2} \cdot \sqrt{\sum_i (\text{tfIdf}_d(t))^2}}
\]  

(2)

The method by Vallet et al. [41], an adapted approach of Xu et al. [45], referred to as \( \text{tf-iuf} \), excludes length normalization factors of the \( \text{UIP} \) and documents from the similarity score computation, and includes the inverse user frequency and inverse document frequency.

\[
\text{tf-iuf}(\text{UIP}, d) = \sum_i (\text{tf}_d(t) \cdot \text{iuf}_d(t) \cdot \text{tfIdf}_d(t))
\]  

(3)

The justification for exclusion of document length normalization factor is similar to that of \( \text{tfIdfUIP} \) that using the document length normalization factor would penalize the score of popular documents. The reason for exclusion of \( \text{UIP} \) length normalization factor is that in all computations of similarity scores, the \( \text{UIP} \) length normalization factor is constant. Similar to \( \text{tfIdfUIP} \), \( \text{tfIdfUIP} \) and \( \text{tf-iuf} \) use all terms in the user’s \( \text{UIP} \) for computation of similarity scores to re-rank search result documents.

Shepitsen et al. [34] presented a personalization algorithm for recommendation in folksonomies, referred to as \( \text{svdCUIP} \) in our work, which relies on hierarchical tag clusters. Their approach clusters the entire tag space of a folksonomy system to obtain one common, global cluster structure available to those users who are registered with the folksonomy system. This restrains the outreach of the approach. Further, they gauge user interest in each tag cluster based on the user usage of tags for resources’ annotations. A set of matching clusters extracted from the overall clustered tag space makes up a \( \text{CUIP} \) to be used for personalized resource recommendation. And, both \( \text{tf-idf} \) and \( \text{tf} \) are used to compute the similarity score of resources and a \( \text{CUIP} \).

Our proposed methods, for personalized search based on \( \text{svdCUIP} \) and \( \text{modSvdCUIP} \), use a \( \text{UIP} \) length normalization factor during similarity score computation because the methods expand the user query with the tags from the matching cluster in the user \( \text{CUIP} \), and compute the similarity score between the expanded query and the document contents. The \( \text{UIP} \) length normalization factor varies in accordance with queries because each query may match to a different tag cluster. Because RPs can only be constructed for a small subset of documents, we refrain from using RPs of documents for ranking them. The methods calculate the similarity between the expanded query and document contents. In fact, we have found that it is only possible to construct RPs for approximately 50% of Web documents when using social bookmarking services. This seriously jeopardizes the outreach or acceptability of personalized search systems.

In a nutshell, the \( \text{tfIdfUIP} \) and \( \text{tfIdfUIP} \) re-rank the search result set by computing the similarity scores between the terms in the \( \text{UIP} \) and RPs of documents in the result set, whereas the proposed approaches are based on query expansion and use document contents for ranking search results.

3. Personalized search based on \( \text{CUIP} \)

This section explains (1) how a \( \text{CUIP} \) is built from a user search history by applying matrix factorization and a clustering algorithm, and how the \( \text{CUIP} \) is used for personalized search.

3.1. Aggregating tags from user search history

When a user clicks on a Web document, it indicates the user interest in that document [3]. A user search history provides a collection of the Web documents clicked by the user. Let’s call the collection set \( U \). For each Web document \( u \in U \), its annotations (tags) are extracted from a social bookmarking service. The tags are stemmed during extraction. Let \( T \) be a set of stemmed tags extracted from the social bookmarking service. Note that it is not necessary for the user to have previously used these tags for annotation. The extracted tags were annotated to the documents by the users of the social bookmarking service. Let \( R \) be a binary relation between \( U \) and \( T \). In order to express that a web document \( u \in U \) is in a relationship with a tag \( t \in T \), we write \( \langle t, u \rangle \in R \), which can be read as “the tag \( t \) is a topic of the web document \( u \)”. Each tag, \( t \), annotated to a Web document, \( d \), has a tag-value \( w(t, d) \) representing the number of times \( d \) has been annotated with \( t \). For example, \( w(\text{java}, d) = 1 \) means the tag \( \text{java} \) has been used to annotate the document \( d \) once. A tag weight, \( w(t) \), is an aggregated value of \( t \) originating from the resource profiles (RPs) of multiple documents. It is very likely that the same tag may originate from multiple documents, each with a potentially different tag-value for the tag. We use the standard result set fusion technique, shown in Eq. (4), to aggregate the tag weight, \( w(t) \), from the Web document collection \( |U| \).

\[
w(t) = \sum_{i=1}^{|U|} w(t, d_i)
\]  

(4)

A \( \text{UIP} \) is constructed by collecting all the tags along with their tag weights.

Similar to the well-known term frequency * inverse document frequency for documents in IR, the same can be modeled in constructing a \( \text{UIP} \). The \( \text{tf*idf} \) multiplies the normalized tag frequency \( \text{tfIdf}_d(t)/|d| \) by the relative distinctness of the tag \( t[i] \) in the
Web document corpus. The distinctness is measured by the log of the total number of Web documents, $|U|$, divided by the number of Web documents, $|td[i]|$, to which the tag $t[i]$ was annotated to. We define the $tf * idf$ as follows.

$$\text{tfidf}(i,j) = \frac{td[i][j]}{|td[i]|} \log_2 \left( \frac{|U|}{|td[i]|} \right)$$

(5)

### 3.2. Latent semantics in UIP

Latent semantics connotes hidden relationships among terms that may exist, but are not explicitly visible. The latent semantics between terms can be discovered by observing the patterns between them such as co-occurrence. Extracting latent semantics between terms helps improve the usefulness of the UIP. Co-occurrence between tags can be classified into two types:

1. Two or more tags that annotate the same document: there exist first-order co-occurrences between the tags.
2. Two or more tags that do not annotate the same document; however, there is some hidden relationship between them because they may be related to similar topics: there exist second-order co-occurrences between the tags.

We propose a system that discovers semantically related tags and groups them together, even though they are not identical or do not annotate the same document. The approaches to establishing latent structures in a UIP are based on the assumption that the more similar tags are, the more closely related they are.

#### 3.2.1. Computing the tag-tag similarity matrix

Co-occurrence similarity derives similarity between two or more tags that annotate the same document. The degree of similarity is calculated using the co-occurrence frequency, called first-order co-occurrence similarity. Another type of co-occurrence similarity is second-order co-occurrence similarity that derives similarity between two tags that do not annotate the same document, but both are related to at least one other tag that annotates the document. It is analogous to finding a friend of a friend and quantifies the degree of friendship relationship. A straightforward approach to measuring the similarity between two tags is to use the Jaccard coefficient between their tag vectors. An alternative approach is to employ matrix factorization on the tfidf matrix.

We use two matrix-factorization-based methods to calculate the tag-tag similarity matrices. Latent Semantic Analysis (LSA) [13] uses a matrix factorization technique, Singular Value Decomposition (SVD), to create a new abstract representation of a document corpus in the latent squares sense. The SVD decomposes the tfidf matrix into three matrices, $A = USV^T : U$, a tag by dimension matrix; $S$, a diagonal matrix of singular values; and $V$, a document by dimension matrix. One advantage of the SVD is that it is possible to find a low-rank approximation of the original matrix that removes noise. When we select the $k$ largest singular values from $S$ and their corresponding singular vectors from $U$ and $V$, we get the rank $k$ approximation, $A_k = U_kS_kV_k^T$, where $k$ is the dimension reduction parameter. The left singular vectors provide a mapping from the tag space to a newly generated abstract space, while the right singular vectors provide a mapping from the document space to a newly generated space. To compute the tag-tag similarity matrix, we compute $U_k$, a low-rank approximation of $U$ matrix. After the dimensionality reduction step, the term-term similarity matrix, $Sim_k$, is computed by using Eq. (6).

$$Sim_k = U_kS_k(U_kS_k)^T = U_kS_kS_k^T U_k^T = U_kS_k^T U_k^T$$

(6)

Dimensionality reduction reduces noise in the tag-tag similarity matrix, resulting in richer relationships between tags that reveals the hidden semantics present in the document corpus. The value of $Sim_{ij}$ in $Sim_k$ represents the similarity between tags $t_i$ and $t_j$. The higher the value, the higher the relatedness is between the tags. In theory, the value of $Sim_{ij}$ captures both orders of co-occurrence similarities between $t_i$ and $t_j$ across the corpus. That is, the value is based on the transitive relation between terms due to a chain of intermediate terms that link the terms $t_i$ and $t_j$. Note that it is not necessary for $t_i$ and $t_j$ to belong to the same document, but there should be a chain of terms that link them. Two factors influence the magnitude of similarity value $Sim_{ij}$: (1) the number of intermediate tags, or the length of the chain that connects $t_i$ and $t_j$; and (2) the tag-weights of the intermediate tags. The following observations were made in respect to the similarity matrix, $Sim_k$

1. For smaller values of $k$, the $Sim_k$ successfully captured the first-order and second-order co-occurrence relationships between terms. However, the values suggested a stronger second-order co-occurrence similarity compared to first-order co-occurrence similarity.
2. For higher values of $k$, close to the number of dimensions of the document space, the $Sim_k$ successfully captured the first-order co-occurrence similarity between terms but failed to capture the second-order co-occurrence similarity.
3. An intermediate value of $k$ worked out to be a good solution but not the best. This is because the second-order similarity values were too small to be identified by the clustering algorithm to generate crisp cluster structures.
This seriously jeopardizes the effectiveness of the clustering algorithm to generate clearly separated clusters. In real scenarios, sparseness of a similarity matrix, \( \text{Sim}_{\text{m}} \), could be as high as 90%, which seriously affects the ability of the SVD to correctly discover the second-order co-occurrences. We show in the experiment section the effect of sparseness of \( \text{Sim} \) matrices on clustering tendency and clustering accuracy.

To circumvent the limitation, we propose an approach called modified SVD (\text{modSVD}). It constructs a tag-tag similarity matrix \( \text{modSim} \), which calculates the cosine similarity between tag vectors of similarity matrix \( \text{Sim} \) using Eq. (7). Each tag vector represents the projection of a tag in the tag space. For instance, each tag \( t_i \) in the similarity matrix, \( \text{Sim}_{\text{m}} \), has a non-zero value for each term \( t_j \) that co-occurs with it. Calculating the similarity between two tag vectors requires computing the overlap between them that discovers second-order co-occurrence relations between the tags.

\[
\text{modSim}(t_1, t_2) = \frac{\sum_{i=1}^{n} t_{1i} t_{2i}}{\sqrt{\sum_{i=1}^{n} t_{1i}^2 \sum_{i=1}^{n} t_{2i}^2}}
\]  

The tag-tag similarity matrix, \( \text{modSim} \), captures the similarity between all pairs of tag vectors to discover second-order co-occurrence relations. Higher values of \( \text{modSim}_{ij} \) signify a greater overlap between the two vectors across \( n \) dimensions. Thus, it aids in demarcating clusters boundaries, resulting in fine clusters, and also helps induce sense from contextual similarity.

3.2.2. Tag clustering to generate \text{svdCUIP} and \text{modSvdCUIP}

Deerwester et al. [13] urged the necessity of clustering in Information Retrieval (IR) tasks. The authors state that IR systems treat each term as independent from others. Treating a term independently may lose the latent contextual information that can make substantial difference in information retrieval tasks. This has motivated us to use clustering in our work.

Term Clustering algorithms generally consist of two phases. The first phase requires computing a term-term similarity matrix, and the second phase uses the matrix to generate clusters of coherent terms. Two major types of clustering algorithms are available: partitioning and hierarchical. The partitioning clustering generates topic clusters, whereas the hierarchical clustering generates cluster hierarchies. Topic clusters are created by grouping similar and closely related terms together into a unified topic. In a cluster hierarchy, terms are placed in the leaves at the bottom of the hierarchy with more specialized topics immediately above them, and so on. Hierarchies are very large and complex in nature. We want hierarchies but not too specific terms. We are, on the other hand, interested in crisp clusters. Therefore, we adapted a hybrid approach that generates a hierarchy, which is further dissected to generate crisp term clusters. We used the Hierarchical Agglomerative Clustering Algorithm (HAC) [18] because it fits best when the number of clusters is unknown beforehand.

We use distinctness parameter, \( d \), to cut the single hierarchy of clusters to obtain a number of clusters. It is very important to choose the right value of \( d \) to generate appropriate term clusters matching the user’s perspective, thus achieving a high clustering accuracy. Fig. 1 shows a dendrogram output when the similarity matrix \( \text{modSim} \) is input to the HAC. With \( d \geq 1.4 \), one cluster is created, a hierarchy of all the terms; with \( d = 0.4 \), there are three clusters; and, with \( d < 0.3 \), there is a flat list of terms.

At the outset, HAC treats each term as a singleton cluster and then successively merges pairs of clusters until all the clusters have been merged into a single cluster that contains all the terms. Cluster proximity is used to merge clusters. There are three well known proximity measures: single linkage, complete linkage, and average linkage. The single linkage proximity measure is the distance between the closest two points that are in two different clusters, i.e., the maximum similarity between two terms. On the contrary, the complete linkage takes the distance between the farthest two points in two different clusters as the cluster proximity. The average linkage defines cluster proximity as the average pairwise proximity, an average length of edges of all the terms from two different clusters. We carried out experiments using the three proximity measures, but this paper reports on only the average linkage in the experiment section because it worked better than the others.
A CUIP that results from the application of HAC on a Sim matrix obtained by applying the SVD on a tfidf matrix is called SVD based CUIP (SvdCUIP). And, a CUIP that results from the application of HAC on a modSim matrix obtained by calculating the cosine similarity of every pair of tag vectors in the similarity matrix, Sim, is called modSVD based CUIP (modSvdCUIP).

We also generate a tfidfCUIP for each user, an adaptation of Shepitsen et al. [34] approach. A term-term similarity matrix is generated by computing the cosine similarity between tag vectors in the tfidf matrix, which is fed to HAC to generate the tfidfCUIP. The tfidfCUIP is a local cluster structure unlike the Shepitsen et al. [34] approach where the terms in the UIP are mapped to a global cluster structure to construct a CUIP.

3.3. Personalized search

This section explains how to use a CUIP for personalized search. The classic SEs compute the relevance between a query and a document using the similarity between the terms that match. They are “One-size-fits-all” in that the search results are the same irrespective of the user. However, a document relevant to a user might not be relevant to another user, though, they both have issued the same query. Thus, the user query as well as its context should be mapped to the term space of the document contents. A query conflated with the contextual terms is called expanded query.

The CUIP helps disambiguate a user query by suggesting a matching cluster. The terms in Web documents and the expanded query are represented as vectors in the space. By using the Vector Space Model (VSM) [32], we compute the similarity between the term spaces of the documents and that of the expanded query to compute the rank of the documents. Let $d = t_1, t_2, \ldots, t_n$ be the term vector for a document, where $n$ is the dimension of the term space. Let $q = q_1, q_2, \ldots, q_m$ be the expanded query. The similarity between a document $d$ and a query $q$ is calculated using Eq. (8).

$$\text{sim}(d, q) = \frac{d^T q}{||d|| ||q||}$$

Given a user query, two steps are executed in the following order: first, find a matching cluster $g_m$ in the user CUIP to the query; second, the query and the tags in the matching cluster are fed to the underlying search engine to generate a set of documents that are ranked using Eq. (8).

We employ the Normalized Google Distance (NGD) [11] to determine the closest cluster, for a given user query, in the user CUIP. This involves computing the semantic similarity between each cluster and the query, to choose the cluster that has the maximum similarity to the query (refer to Eq. (9)).

$$\text{CUIP} = \{g | g = \{t_1 : tw_1, t_2 : tw_2, \ldots, t_n : tw_n\}\}$$

$$g_m = \text{MIN}_i \text{NGD}(q, g_i)$$

$$\text{NGD}(t_1, t_2) = \frac{\max\{\log(f(t_1)), \log(f(t_2))\} - \log(f(t_1), t_2)}{\log N - \min\{\log(f(t_1)), \log(f(t_2))\}}$$

$$q_e = q \cup g_m$$

where $l$ is the number of clusters in the user CUIP. The variables, $f(t_1)$, $f(t_2)$, and $f(t_1, t_2)$, are the numbers of search results for terms $t_1$, $t_2$, and ($t_1$ and $t_2$), respectively. The value $\text{NGD}(t_1, t_2)$ lies between 0 and $\infty$. If $\text{NGD}(t_1, t_2) = 0$, it signifies high relatedness between $t_1$ and $t_2$. The greater the value of $\text{NGD}(t_1, t_2)$, the lesser the relatedness between the terms.

It is important to note that semantic distance is a more general concept than similarity [7]; similar entities are usually assumed to be related by virtue of their likeness (bank - trust company), but dissimilar entities may also be semantically related by lexical relationships such as meronmy (car – wheel) and antonymy (hot – cold), or just by any kind of functional relationship or frequent association (pencil – paper, penguin – Antarctica). The idea behind using NGD is to compute semantic distance between two entities that might appear dissimilar on the surface but are semantically related to each other.

4. Experimental evaluation

4.1. Data set and experiment methodology

To examine the effectiveness of the proposed methods, we conducted a series of experiments on two different data sets. First, to evaluate the clustering tendency and clustering accuracy of the CUIP, we recruited 12 users whose search histories were harvested to construct the first data set, referred as Custom Data Set. Second, to evaluate the quality of personalized search using the proposed methods, we constructed another data set from the AOL search query log.\(^2\) For both data sets, the URL-tag annotations were harvested from the Delicious Server using the Delicious API.\(^3\)

\(^2\) http://www.gregsadetsky.com/aol-data/.
\(^3\) http://www.delicious.com.
4.1.1. Custom data set and evaluation metrics

This data set consists of data from 12 users, mostly master’s students, who have considerable experience using search engines. Each user’s log of search history for a period of 3 months or 13 weeks was harvested as an RSS feed from the individual’s Google Search History. The RSS feed consists of the following meta data: title of the query input by the user; title of the Web document clicked by the user; the address of the Web document clicked by the user; and, the dates and times at which the queries were submitted. The data set contains 2921 queries, and 6477 clicked Web documents. Of the documents, only 3617 (approximately 55%) were found to be annotated on Delicious.

In clustering, measuring its accuracy and correctness in any certainty is best left to the user’s judgement. Therefore, to establish the ground truth, we asked each user to group related terms extracted from the tag annotations of the Web documents clicked by the user. Each user was asked to manually group related terms together; they were instructed to group terms based on their own understanding rather than the general understanding. The grouping generated manually by the user is called user cluster structure. Generating ground truth manually for evaluation is a normal procedure used in many research works [14,25,27,29,43]. Since this process is subjective, we take the average of the scores from all the users as the final score. The whole process was a very labor intensive and time consuming task, which was the primary reason why we opted to experiment with a small set of users.

For each user, two sets of CUIPs are generated: one set consists of svdCUIPs, and the other of modSvdCUIPs. These CUIPs are called system generated cluster structures. In each set, a CUIP is generated for each combination of dimension reduction parameter k and distinctness parameter d. To construct a svdCUIP and a modSvdCUIP, the similarity matrices simk and modSimk are generated, respectively. The value for k is initialized to 10, and it increases in an increment of 10 until it reaches 110. This creates 11 simk and 11 modSimk similarity matrices. Similarly, the distinctness parameter d is initialized to 0.03, and it increases in an increment of 0.02 until 0.13, after which it increases in an increment of 0.1 until 0.93 (a total of 14 values). For each user, 154 svdCUIPs and an equal number of modSvdCUIPs were created. Let the user generated cluster be C = {c1, c2, ..., cn}, and the system generated cluster be D = {d1, d2, ..., dm}. We chose the Silhouette Coefficient [31] evaluation metric (unsupervised evaluation) to judge the cluster tendency, and the Fscore (supervised evaluation) evaluation metric to compare the clustering accuracy. The Silhouette Coefficient is a popular method that combines cohesion and separation. Eq. (10) computes the Silhouette Coefficient for each tag ti in the system cluster structure.

\[ s(i) = \frac{(b_i - a_i)}{\max(a_i, b_i)} \]  

(10)

where bi is the minimum of all the average distances between term ti and all the terms in other clusters that do not contain ti (separation); and, ai is the average distance between term ti and all other terms in the same cluster (cohesion). Eq. (11) computes the average Silhouette Coefficient, s, which is the average of the Silhouette Coefficients for all the terms (N) in the cluster structure.

\[ s = \frac{1}{N} \sum_{i=1}^{N} s(i) \]  

(11)

An average Silhouette Coefficient is a very useful overall quality measure to measure the clustering tendency of a cluster structure. Kaufman and Rousseeuw [19] provided an interpretation of the average Silhouette Coefficient, s, as a measure of evidence in support of a cluster structure: the value of the average Silhouette Coefficient between [0.7, 1.0] suggests strong evidence; between [0.5, 0.7] reasonable evidence; between [0.25, 0.5] weak evidence; and between [−1, 0.25] no evidence.

We also compare the clustering accuracy of the system generated cluster structure with the user generated cluster structure. Fscore [25] measures the extent to which a system generated cluster contains only tags of a particular user generated cluster and all objects of that user generated cluster. Eq. (12) computes an Fscore by combining precision and recall. Precision, pi, is the proportion of the tags of user generated cluster ci in the system generated cluster di; Recall, ri, is the fraction of matching tags in the system generated cluster di that match the tags in the user generated cluster ci.

\[ Fscore_i = \frac{2 * p_i * r_i}{p_i + r_i} \]  

(12)

4.1.2. AOL query data set and evaluation metrics

The AOL search query log has 20 million web queries collected from 650,000 users. Each row in the data set contains five attributes: (1) AnonID, an anonymous user id; (2) Query, the query issued by the user; (3) Query Time, the time at which the query was submitted to the AOL search engine; (4) Item Rank, the rank of the Web document clicked by the user; and (5) ClickURL, the address of Web document clicked by the user. We created a dataset of 2000 users, a sub set of the total data set. This dataset contains 1,244,714 Web documents, out of which 829,285 documents (approximately 66%) were found to be annotated on the Delicious server. The documents have 212,011 tags annotated to them.

Our experiment methodology is geared towards measuring the effectiveness of the proposed personalized search methods and evaluating the improvement they offer in comparison to other methods.
4.1.3. Experiment set up to estimate the value of $k$ and $d$

The complete data set is split into two equal parts: the first part is called as the training, or development, data set; and the second part is called as the evaluation data set. The training data set is used to estimate the value of parameters $k$ and $d$ for $svdCUIP$ and $modSvdCUIP$, which are directly used in the evaluation dataset to compare the performance of the proposed approaches with the other personalized search approaches. The evaluation data set helps guard against both under fitting and over fitting.

From the training data set, we construct $UIPs$ and $CUIPs$, and pairs of query and associated Web document (referred as target Web document) are extracted from the user search history. For each pair, the query is submitted to the base search engine to calculate the rank of the target Web document, called $r_b$. Next, the query is expanded with the tags in the matching cluster from the $CUIP$. The expanded query is submitted to the search engine to calculate the new rank of the target Web document, called $r_e$. The difference in the inverse ranks of the personalized search method and the baseline method is the improvement [42] of the personalized search method, calculated using Eq. (13).

$$\text{improvement} = \frac{1}{r_a} - \frac{1}{r_b}$$

The values of $k$ and $d$, for which the improvement of the proposed methods over baseline search is maximum, are used directly for the further stage of evaluation.

4.1.4. Experiment set up to compare the proposed approaches with other approaches

The following steps execute on the evaluation data set:

1. **Indexing**: The contents of each document in the dataset is indexed using Lucene API. Lucene is our base search engine, and search using it is referred to in this paper as baseline search method.

2. **User profile**: The search history of each user is randomly divided into two parts: the first part, which makes 90% of the entire history, is used for building $UIPs$ and $CUIPs$; and the second part, the remaining 10%, is used for generating pairs of queries and URLs, called test collection, to automatically evaluate our methods.

3. **Evaluation**: For each document in the second part, we create a pair that consists of the document itself and the query associated with it. Each pair constitutes a test case against which the tasks (a), (b), (c), and (d) below are executed. A test case designates a query and its target Web document.

   (a) For each query and Web document combination in a test case, submit the query to the base search engine to obtain a ranked list of search results. Let the rank of the target Web document in the search result set be $r_b$. This is the rank of the target document produced by the baseline search method.

   (b) For both $tfUIP$ and $tfIdfUIP$, the Web documents in the search result set are re-ranked by calculating the similarity between the $RP$ of the Web documents and tags in the $UIP$ using Eqs. (1) and (2), respectively. Let the new ranks of the target document in the re-ranked search result set designated as $r_{tfUIP}$ and $r_{tfIdfUIP}$, respectively. Eq. (13) computes the improvement as the difference between the inverse ranks of the personalized search method and the baseline method.

   (c) Search results are not re-ranked for the $svdCUIP$, $modSvdCUIP$, and $tfIdfCUIP$ methods, rather, the query is expanded with the tags in the matching cluster from the $CUIP$. The expanded query is submitted to the search engine to determine a new rank of the target document. The search engine generated the ranking of documents by calculating the similarity between the expanded query and the document contents using the Eq. (8). The difference in the inverse ranks determined for the personalized search method and the baseline method is the improvement of the personalized search method.

4.2. Experiment results

Sections 4.2.1, 4.2.2, and 4.2.3 determine, for both $svdCUIP$ and $modSvdCUIP$, the value(s) of dimensionality reduction parameter $k$ and distinctness parameter $d$ that show(s) strong, or at least reasonable, clustering tendency and clustering accuracy. Section 4.2.4 presents an exemplary $modSvdCUIP$. The Sections 4.2.5 and 4.2.6 determine, for both $svdCUIP$ and $modSvdCUIP$, the value(s) of dimensionality reduction parameter $k$ and distinctness parameter $d$ using the Improvement as an evaluation metric. And, Sections 4.2.7 and 4.2.8 compare the proposed methods with the other methods using the evaluation metric Improvement.

4.2.1. Clustering tendency

Assessing the presence of clusters in a data set is an important step in cluster analysis. The plot in Fig. 2 helps visualize clustering tendency in system generated clusters, if any, and also approximates the correct number of clusters in the cluster structure.

It is clear that the cluster structure $modSvdCUIP$ has stronger evidence of cluster tendency, whereas the $svdCUIP$ shows reasonable or weak evidence of clustering tendency. We observed that the clustering tendency in a $CUIP$ was affected by

---

5 lucene.apache.org/core/.
the ratio of number of zero values to the number of positive values in the tag-tag similarity matrix; the lower the better. The average ratio for the tag-tag similarity matrix modSim is 0.9, and 1.68 for the tag-tag similarity matrix sim. The maximum and minimum ratios for the modSim are 3.2 and 0.6, respectively, and for the sim, 6.2 and 1.0, respectively. This evidence explains why the cluster structure, svdCUIP, exhibits weak cluster tendency.

Fig. 2 also indicates that the average Silhouette Coefficient (S) decreases as the number of clusters exceeds over 50, which suggests that the best cluster structure was obtained when the number of clusters was around 50. This was acceptable because the average number of tags in a UIP was 594, which could possibly result in 50–70 clusters. However, what is surprising is that, even with less than 10 clusters in the modSvdCUIP, the plot shows strong clustering tendency. To try to find the natural number of clusters in a cluster structure, one should look for a knee, a peak, or dip in the plot [37]. The plot for the modSvdCUIP shows a rise followed by a dip and a peak occurring around when the number of clusters falls between 40 and 60. For the svdCUIP, the plot clearly shows a peak when the number of clusters reaches 50.

4.2.2. Determining the value for dimension parameter, k, for the custom data set

Figs. 3 and 4 present 3-dimensional plots that show how the average Silhouette Coefficient changes in response to the changes of k and d. The figures help determine the values of k and d for each method. The svdCUIP in Fig. 3 exhibits a clear pattern: for low values of k regardless of d, there is no evidence of clustering tendency; however, for high values of k, between 90 and 100 and low values of d, there is a reasonable evidence of clustering tendency. The weak clustering tendency of the svdCUIP is due to the fact that the magnitude of relationship between tags is low. This jeopardizes the ability of clustering algorithms to discern cluster boundaries.

The average Silhouette Coefficient vs. k and d plot in Fig. 4 for the cluster structure modSvdCUIP also exhibits a distinct pattern: unlike the svdCUIP, the plot for the modSvdCUIP shows a strong evidence of clustering tendency for values of k = 30 and 40 and middle values of d. It ascertains the fact that increasing the value of d decreases clustering tendency. The modSvdCUIP exhibits a strong clustering tendency because the modSim overcomes the limitation of the sim by capturing the information present in second order co-occurrence. Moreover, the information in the modSim matrix is less sparse and more robust than the sim matrix.

4.2.3. Determining the value of distinctness parameter, d, for the custom data set

The experiment, in this section, focuses on determining the appropriate value of d for the highest accuracy cluster structure. Fscore is used as an evaluation metric to measure and compare the accuracy of the system generated cluster structure with the user generated cluster structure. Fig. 5 shows the accuracy obtained by each method, and demonstrates that the modSvdCUIP has better clustering accuracy than the svdCUIP.

The average clustering accuracy for the modSvdCUIP and svdCUIP is 0.58 and 0.16, respectively; there is a 244% increase in average clustering accuracy. This indicates that the modSvdCUIP produced by the modSvd is more accurate than the svdCUIP produced by the Svd. With the modSvd, the dimension reduction parameter k = 30 has higher clustering accuracy than k = 40. Also, the difference in clustering accuracy between k = 30 and k = 40 is marginal. Moreover, both of the curves follow the same pattern, signifying that the clustering accuracies of the modSvdCUIP for k = 30 and k = 40 are nearly identical with a slightly better performance at k = 30. The highest clustering accuracy for the modSvdCUIP is 0.75, obtained with k = 30 and d = 0.07.

Another identical accuracy was exhibited when k = 90 and k = 100 in the Svd. A careful observation, however, reveals that the svdCUIP for k = 100 shows a marginal improvement over k = 90, with d = 0.03 and d = 0.05. This suggests that either value
of the dimension reduction parameter can be used for constructing the svdCUIP. The highest clustering accuracy for the svdCUIP is 0.55, with \( k = 100 \) and \( d = 0.03 \).

These results suggest that the accuracy of the modSvdCUIP produced by the mod Svd is superior to the cluster structure svdCUIP produced by the SVD.

4.2.4. CUIP visualization

We developed our own implementation of Hierarchical Agglomerative Clustering (HAC) in Java. Table 2 shows the snapshot of the modSvdCUIP, the output of HAC for \( d = 0.53 \), for one of the users. For interested readers, a complete modSvdCUIP, svdCUIP, and tfidf/CUIP is provided in Appendix C.

The quality of clusters hinges on the level of term coherency, each cluster representing a distinct topic area. Table 2 shows a high level of term coherency in clusters, each of which shows user interests such as finance, religion, porn, law, automotive, and entertainment. Moreover, the terms in each cluster are contextually related, which aids to disambiguate context, synonym terms, and polysemous terms. For instance, Cluster 1 captures the notion of the user’s interests in finance, and
disambiguates the context of the polysemous term “bank”, which in Cluster 1 refers to a financial institution, not to other meanings such as bank as in a river bank.

Cluster 2 indicates that the user is interested in Judaism religion. Synonym terms are clustered together such as “Jewish” and “Judaism” in Cluster 2, “auto” and “automotive” in Cluster 5, “movies” and “film” in Cluster 6. Cluster 5 can be interpreted as that the user is interested in the automotive, in particular cars. She/he might also be interested in the electronic parts of the car. Cluster 6 represents the user’s entertainment options; the user prefers to watch movies or soccer games. The term video is rightly disambiguated by being associated with the term “movie”.

These results show clear evidence of emergence of topics and contexts that would otherwise be latent in a UIP. A CUIP is an important source of information that can be effectively used for query suggestion, query classification, Web page recommendation, personalized search, or Web search result ranking.

4.2.5. Determining the value of the dimension reduction parameter k for the AOL data set

Since the personalization algorithm relies on the user CUIP to personalize search results, the selection of a proper dimension value is integral to the success of the personalization algorithm. The goal of tuning the dimension parameter is to discover the second order co-occurrence similarity between tags. Fig. 6 plots the improvement of proposed methods in reference to the baseline search when the value of \( k \) changes from 10 to 110 in an increment of 10. It indicates that the \( modSvdCUIP \) based personalized search shows greater improvement than the \( svdCUIP \) based personalized search. In this experiment, the most improvement was obtained when the value of \( k \) for the \( svdCUIP \) and \( modSvdCUIP \) was 90 and 100, respectively. Note that in a reduced space, the performance of the \( modSvdCUIP \) based personalized search degraded below 0; this means that it performed worse than the baseline search. However, when \( k \) was set to 50 and above, it showed improved performance.

These results show that both methods benefited from the dimensional reductional step. In the following experiments, the value of \( k \) for the \( svdCUIP \) and \( modSvdCUIP \) was set to 90 and 100, respectively.

4.2.6. Determining the value of distinctness parameter, d, for the AOL data set

The distinctness parameter \( d \), controls how distinct or well separated the clusters are. As the value decreases, we get closer to a single cluster or a few large clusters; hence, grouping unrelated terms together or spanning multiple topic areas. On the contrary, as the value increases, we end up with lots of clusters of a single term or lots of small-sized clusters, thus rendering the information in the clusters inadequate to represent topics. The parabolic graph in Fig. 7 supports this idea. Note that there is no dimension reduction applied to the \( tfIdfCUIP \) method.

Fig. 7 also shows that the \( modSvdCUIP \) based personalized search outperformed the \( tfIdfCUIP \) and \( svdCUIP \). The maximum improvement was obtained when \( d \) was set to 0.09, 0.13, and 0.63 for the \( tfIdfCUIP \), \( svdCUIP \), and \( modSvdCUIP \), respectively. Performance of each CUIP is related to the number of clusters and the size of each cluster. The number of clusters for the

---

**Table 2**

Example of cluster structure.

<table>
<thead>
<tr>
<th>Cluster 1</th>
<th>Cluster 2</th>
<th>Cluster 3</th>
<th>Cluster 4</th>
<th>Cluster 5</th>
<th>Cluster 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bank, banking, finance,</td>
<td>Religion, culture,</td>
<td>Amateur, sex, adult,</td>
<td>Government, patent,</td>
<td>Auto, automotive,</td>
<td>Video, movies,</td>
</tr>
<tr>
<td>business, supplier</td>
<td>judaism, jewish, israel</td>
<td>toys, girls, porn, voyeur</td>
<td>trademark, law, legal</td>
<td>parts, electronics, car</td>
<td>film, soccer, game</td>
</tr>
</tbody>
</table>

---

Fig. 5. A comparison of different value combinations \( k \) and \( d \) vs AverageFscores for the \( modSvdCUIP \) (when \( k = 30, 40 \)) and the \( svdCUIP \) (when \( k = 90, 100 \)) for average linkage.
tfIdfCUIP with $d = 0.09$ is 54, 89 for the svdCUIP@90 with $d = 0.13$, and 76 for the modSvdCUIP@100 with $d = 0.63$. Also, the average number of tags in each cluster, average cluster size, for the tfIdfCUIP with $d = 0.09$ is 6, 3 for the svdCUIP@90 with $d = 0.13$, and 4 for the modSvdCUIP@100 with $d = 0.63$. In short, having too many clusters, with only a few tags in each cluster, does not help disambiguate topics; this justifies why the tfIdfCUIP and the modSvdCUIP performed better than the svdCUIP.

In the following experiments that will execute on the evaluation data set, the value of $d$ was set to 0.09 for the tfIdfCUIP, $k = 90$ and $d = 0.13$ for the svdCUIP, and $k = 100$, $d = 0.63$ for the modSvdCUIP.

4.2.7. Comparison of the svdCUIP, modSvdCUIP, and tfIdfCUIP for different classes of queries

The purpose of using the modSvdCUIP for personalized search is to identify the query context that we supposed the tfIdfCUIP would not be able to provide. However, the results presented in the previous sections indicate that the personalized search based on the modSvdCUIP and tfIdfCUIP delivered comparable effectiveness in improving the ranks of target Web documents. To further look into the effect that clusters have on personalized search, we analyzed the test collection, and found that self-evident queries did not require disambiguation, and some vague queries received benefit when contextual tags were conflated with them. We identified 40 vague queries and 50 self-evident queries (refer to Appendix A). Appendix B shows some examples of expanded queries and how query disambiguation is useful to personalized search.

Fig. 8 shows that the modSvdCUIP performed significantly better than both methods for the vague queries. Paired sample t-test was used to test the significance, and $p$-values were found to be $< 0.05$.

And, any modification of the self-evident queries by query expansion degraded the performance of the CUIP based personalized search methods. The tfIdfCUIP had the worst negative effect when used for disambiguating self-evident queries because the average cluster size is larger compared to other methods, thus degrading the ranks of the target Web documents.
4.2.8. Comparing all five methods – improvement

This experiment aims to compare our proposed two methods with the others: (1) tf based personalized search, tfUIP; (2) tfIdf based personalized search, tfIdfUIP; and (3) tfIdfCUIP based personalized search.

As shown in Fig. 9, the worst performer is the tfIdfUIP, similar to as reported by Vallet et al. [41]; results of both this study and Vallet et al. [41] contradict those of Xu et al. [45] that the tfIdfUIP performed better than the tfUIP. A possible reason for the contradiction between ours and Xu et al. [45] approach is the total size of the result set; Xu et al. [45] re-ranked the top 100 Web documents, whereas our methods calculated the re-rank of the target URL in the top 600 documents. We suppose that the tfUIP showed better improvement than the tfIdfUIP because of the exclusion of two factors from the similarity score computation: document length and user profile length normalization factors. The user profile length normalization factor is dominant in the tfIdfUIP, and this penalizes the re-ranking score extensively.

The maximum improvement of the modSvdCUIP was 0.176766, whereas for the svdCUIP and the tfIdfCUIP was 0.132146 and 0.155571, respectively.

We performed significance test to determine if the difference between observed values from each approach are significant when compared with the baseline search. We used paired sample t-test and compared the average MRR values. Table 3

Table 3
Comparing the MRRs of tfIdfUIP, tfUIP, tfIdfCUIP, svdCUIP, and modSvdCUIP.

<table>
<thead>
<tr>
<th>Method</th>
<th>MRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>tfIdfUIP</td>
<td>0.3434</td>
</tr>
<tr>
<td>tfUIP</td>
<td>0.3625</td>
</tr>
<tr>
<td>tfIdfCUIP</td>
<td>0.4118</td>
</tr>
<tr>
<td>svdCUIP</td>
<td>0.3946</td>
</tr>
<tr>
<td>modSvdCUIP</td>
<td>0.4243</td>
</tr>
</tbody>
</table>
shows that the differences between the values from the \textit{tfIdfUIP}, \textit{tfUIP}, \textit{tfIdfCUIP}, \textit{svdCUIP}, and \textit{modSvdCUIP} are significantly better than the baseline search. The MRR values were confirmed to be significantly different using the paired t-test with 95% confidence interval: \textit{tfIdfUIP} ($p$-value = 1.87E–09), \textit{tfUIP} ($p$-value = 1.67E–10), \textit{tfIdfCUIP} ($p$-value = 4.1E–11), \textit{svdCUIP} ($p$-value = 4.2E–10), \textit{modSvdCUIP} ($p$-value = 2.31E–12). Thus, we can confidently conclude that the improvement of our proposed approaches is better than the baseline search.

5. Conclusions and future work

In this paper, we proposed two novel methods that exploited user search history and social bookmarking services for building a Clustered User Interest Profile (\textit{CUIP}) that consists of term clusters of user interests. The first method is based on the Singular Value Decomposition (SVD) to compute a tag-tag similarity matrix and use the Hierarchical Agglomerative Clustering (HAC) on the matrix to generate a cluster structure, \textit{svdCUIP}. The second method is an extension of the first method, called modified Singular Value Decomposition (\textit{modSVD}), that aims to group related tags based on their second-order co-occurrence similarity. This method is based on the assumption that related tags are often expressed together by similar sets of tags. These semantically related tags are bound to co-occur with similar neighbors. The objective of the \textit{modSVD} is to discover and group these semantically related tags into clusters to generate a \textit{modSvdCUIP}, each cluster of which identifies a unified topic.

To evaluate the effectiveness of the proposed approaches, we compared them with the baseline search and the three other methods that use folksonomy for constructing \textit{UIP} and Resource Profile (\textit{RP}): \textit{tfUIP} Noll and Meinel [28], \textit{tfIdfUIP} Xu et al. [45], \textit{tfIdfCUIP} (an adapted method Shepitsen et al. [34]). Our methods are more realistic as they make no assumption about the tagging activity of the user, and can be easily put to practice for any user who uses a search engine for his/her daily search needs. In our evaluations, we found that the improvement in the ranking scores of the target URLs for the \textit{modSvdCUIP} based personalized search were better than all the other methods; the \textit{modSvdCUIP} approach showed improvement of 71.6%, 27.8%, 12%, 6.6%, and 8.1% over the baseline (Lucene Search), \textit{tfIdfUIP}, \textit{tfUIP}, \textit{tfIdfCUIP}, and \textit{svdCUIP} approaches, respectively.

There are several issues that need to be addressed in modeling a \textit{UIP}. All the approaches, including ours, have reflected rather long-term, not short-term, user interests. For example, a user, who has before shown interest in comedy movies, may take interest in horror movies, a problem known as concept drift in IR. The system therefore should account for temporal effects reflecting the dynamic, time-drifting nature of the user search click through. Modeling temporal effects into a \textit{UIP} comes with new challenges; for instance, a change in user interests can be seasonal or permanent. Some changes are gradual and some are impulsive or drastic. To address these variations, a careful balance must be drawn between long term and current user interests. We will in the future try to incorporate concept drifts into the user’s \textit{UIP}. We will also look into more advanced methods such as probabilistic LSI and Latent Dirichlet Allocation (LDA) for discovering and building a more efficient \textit{CUIP}.

Acknowledgement

This research was supported by MSIP (the Ministry of Science, ICT and Future Planning), Korea, under the IT-CRSP (IT Convergence Research Support Program) (NIPA-2013-H0401-13-1001) supervised by the NIPA (National IT Industry Promotion Agency).

Appendix A. Pairs of query and target URL

List of self-evident query and target URL

Tables A.4 and A.5.

Table A.4

<table>
<thead>
<tr>
<th>Query</th>
<th>Target URL</th>
<th>Query</th>
<th>Target URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Puzzle</td>
<td>zigzone.com</td>
<td>Math</td>
<td>mathlesson.com</td>
</tr>
<tr>
<td>Medicine</td>
<td>jmir.org</td>
<td>Estuer</td>
<td>en.wikipedia.org/George_Gurdjieff</td>
</tr>
<tr>
<td>Hostel</td>
<td>en.wikibooks.org/wiki/LaTeX/Tables</td>
<td>Radio</td>
<td>planetradiocity.com/internetradio/index.php</td>
</tr>
<tr>
<td>bollywood</td>
<td>bollywoodhungama.com/trade/releasedates/index.html</td>
<td>amazon</td>
<td>amazon.com</td>
</tr>
<tr>
<td>Basketball</td>
<td>nba.com</td>
<td>Pbs</td>
<td>wwwpbs.org</td>
</tr>
<tr>
<td>Islam</td>
<td>islamtoday.com</td>
<td>Boardgame</td>
<td>boardgamers.org</td>
</tr>
<tr>
<td>Columbia</td>
<td>columbia.edu</td>
<td>Redcross</td>
<td>redcross.org</td>
</tr>
<tr>
<td>Imdb</td>
<td>imdb.com</td>
<td>Thinkquest</td>
<td>library.thinkquest.org</td>
</tr>
<tr>
<td>Overstock</td>
<td>overstock.com</td>
<td>Gap</td>
<td>gap.com</td>
</tr>
<tr>
<td>Walmart</td>
<td>walmart.com</td>
<td>Citibank</td>
<td>citibank.com</td>
</tr>
<tr>
<td>Wikipedia</td>
<td>en.wikipedia.org</td>
<td>Mapquest</td>
<td>mapquest.com</td>
</tr>
</tbody>
</table>

(continued on next page)
**Table A.4 (continued)**

<table>
<thead>
<tr>
<th>Query</th>
<th>Target URL</th>
<th>Query</th>
<th>Target URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dictionary</td>
<td>dictionary.com</td>
<td>Costco</td>
<td>costco.com</td>
</tr>
<tr>
<td>Fbi</td>
<td>fbi.gov</td>
<td>Starbucks</td>
<td>starbucks.com</td>
</tr>
<tr>
<td>Mtv</td>
<td>mtv.com</td>
<td>Cisco</td>
<td>cisco.com</td>
</tr>
<tr>
<td>Marriott</td>
<td>marriott.com</td>
<td>Weather</td>
<td>weather.com</td>
</tr>
<tr>
<td>Hasbro</td>
<td>hasbro.com</td>
<td>Metlife</td>
<td>metlife.com</td>
</tr>
<tr>
<td>Bbc</td>
<td>bbc.co.uk</td>
<td>Playboy</td>
<td>playboy.com</td>
</tr>
<tr>
<td>Businessweek</td>
<td>businessweek.com</td>
<td>Washingtonpost</td>
<td>washingtonpost.com</td>
</tr>
<tr>
<td>Whitehouse</td>
<td>whitehouse.gov</td>
<td>Time</td>
<td>timeanddate.com</td>
</tr>
<tr>
<td>Carter</td>
<td>carters.com</td>
<td>Skype</td>
<td>skype.com</td>
</tr>
<tr>
<td>Microsoft</td>
<td>microsoft.com</td>
<td>Flickr</td>
<td>vflkrr.com</td>
</tr>
<tr>
<td>Oldnavy</td>
<td>oldnavy.com</td>
<td>Patent</td>
<td>freepatentsonline.com</td>
</tr>
<tr>
<td>Sports</td>
<td>qchbaseball.com</td>
<td>Princeton</td>
<td>princeton.edu</td>
</tr>
<tr>
<td>e-health</td>
<td>electronic-health.org/</td>
<td>jigsaw puzzle</td>
<td>jgzone.com</td>
</tr>
</tbody>
</table>

**Table A.5**

List of vague query and target URL pairs.

<table>
<thead>
<tr>
<th>Query</th>
<th>Target URL</th>
<th>Query</th>
<th>Target URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Magazine</td>
<td>automobilemagazine.com</td>
<td>Planet</td>
<td>solarspace.co.uk</td>
</tr>
<tr>
<td>Auction</td>
<td>ragoarts.com</td>
<td>Worksheet</td>
<td>abcteach.com</td>
</tr>
<tr>
<td>Latex</td>
<td>betweenthesheets.co.uk</td>
<td>Business</td>
<td>alibaba.com</td>
</tr>
<tr>
<td>History</td>
<td>onwar.com</td>
<td>latex</td>
<td>en.wikibooks.org/wiki/latex/mathematics</td>
</tr>
<tr>
<td>Telephone</td>
<td>skype.com</td>
<td>Keynote</td>
<td>apple.com/iwork/keynote/</td>
</tr>
<tr>
<td>Apple</td>
<td>kronenberg.org</td>
<td>Electronics</td>
<td>radioshack.com</td>
</tr>
<tr>
<td>divorce</td>
<td>divorcenet.com</td>
<td>Travel</td>
<td>chowbaby.com</td>
</tr>
<tr>
<td>Legal</td>
<td>womenslaw.com</td>
<td>Manufacture</td>
<td>alligator.org</td>
</tr>
<tr>
<td>Realtor</td>
<td>f prótons.com</td>
<td>Food</td>
<td>chinesefood.about.com</td>
</tr>
<tr>
<td>Quiz</td>
<td>iqtest.com</td>
<td>Queen</td>
<td>queenszoo.com</td>
</tr>
<tr>
<td>Price comparison</td>
<td>calibex.com</td>
<td>Gold</td>
<td>Taxfreegold.co.uk</td>
</tr>
<tr>
<td>History</td>
<td>bible-history.com</td>
<td>Music</td>
<td>traditionalmusic.com</td>
</tr>
<tr>
<td>Entertainment</td>
<td>playboy.com</td>
<td>Database</td>
<td>freepatentsonline.org</td>
</tr>
<tr>
<td>Religion</td>
<td>cyberhymnal.org</td>
<td>Bible</td>
<td>studylight.org</td>
</tr>
<tr>
<td>Sports</td>
<td>qchbaseball.com</td>
<td>Newspaper</td>
<td>alligator.org</td>
</tr>
<tr>
<td>Religion</td>
<td>tenets.zoroastrianism.com</td>
<td>Stories</td>
<td>skywriting.net</td>
</tr>
<tr>
<td>Music</td>
<td>hymnal.net</td>
<td>Philosophy</td>
<td>vbm-torah.org</td>
</tr>
<tr>
<td>Automobile</td>
<td>kbb.com</td>
<td>Pond</td>
<td>ponds.com</td>
</tr>
<tr>
<td>Worship</td>
<td>Textweek.com</td>
<td>Health</td>
<td>holisticjunctino.com</td>
</tr>
<tr>
<td>Assist</td>
<td>Natri.uky.edu</td>
<td>Travel</td>
<td>ryanair.com</td>
</tr>
</tbody>
</table>

List of vague query and target URL pairs.

**Appendix B. Examples of expanded queries**

1. The query pond was disambiguated by the cluster [beauty, products] thus pushing the www.ponds.com at the top of the result set.
2. The query religion is a very good example where cluster structure plays an important role. For one user who had interest in Christianity, the query religion was rightly disambiguated with the cluster [religion, Christian, church, catholic] resulting in URL www.cyberhymnal.org at higher rank. For another user, the same query religion was mapped to a cluster [moshiach, judaism, jewish, mysteri, mashiach, messiah] to disambiguate the context of term religion which resulted in the URL tenets.zoroastrianism.com promoted to the top position.
3. Another query latex was mapped to [latex, fetish, sheet, rubber, shop, house, satin, bed] pushing up the URL www.betweenthesheets.co.uk at the top position and lowering the rank of URLs related to Latex document markup language.

**Appendix C. An example of svdCUIP, modSvdCUIP, tfidfCUIP**

**tfidfCUIP (d = 0.09)**

```
[ngo, scuba, korea, dive, editplu, softwar, regex, bollywood, releas, movi, hindi, whitespac, tab, tip, format, data, excel, import, csv, financi, microsoft, fm, music, radio, dna, genealog, genet, scien, technolog, biologi, wp, wealth, wealthi, life, busi, mexico, philanthropi, person, slim, biographi, log, overview, classif, datamin, queri, video, divx, download, legenda, subtitl, film, free,
```

C.1. svdCUI (k = 90, d = 0.13)


C.2. modSvdCUI (k = 100, d = 0.63)

References


