Personalization of Web Search Using Social Signals

Ali Khodaei, Sina Sohangir and Cyrus Shahabi

Abstract Over the last few years, Web has changed significantly. Emergence of social networks and Web 2.0 have enabled people to interact with Web document in new ways not possible before. In this paper, we present *PERSOSE* a new search engine that personalizes the search results based on users' social actions. Although the users' social actions may sometimes seem irrelevant to the search, we show that they are actually useful for personalization. We propose a new relevance model called persocial relevance model utilizing three levels of social signals to improve the Web search. We show how each level of persocial model (users' social actions, friends' social actions and social expansion) can be built on top of the previous level and how each level improves the search results. Furthermore, we develop several approaches to integrate persocial relevance model into the textual Web search process. We show how *PERSOSE* can run effectively on 14 million Wikipedia articles and social data from real Facebook users and generate accurate search results. Using PERSOSE, we performed a set of experiments and showed the superiority of our proposed approaches. We also showed how each level of our model improves the accuracy of search results.

Keywords Social search · Personalized search · Social network · Facebook · Information retrieval · Wikipedia

A. Khodaei
Yahoo! Corporation, 701 1st Ave., Sunnyvale, CA 94089, USA
e-mail: alik@yahoo-inc.com

S. Sohangir (✉)
GraphDive Company, 1295 El Camino Real, Suite B, Menlo Park, CA 94025, USA
e-mail: sohangir@yahoo.com

C. Shahabi
Department of Computer Science, University of Southern California,
Los Angeles, CA 90089, USA
e-mail: shahabi@usc.edu

Ö. Ulusoy et al. (eds.), *Recommendation and Search in Social Networks*,
Lecture Notes in Social Networks, DOI 10.1007/978-3-319-14379-8_8

1 Introduction

While in early stages, search engines' focus was mainly on searching and retrieving relevant document based on their content (e.g., textual keywords), new search engines, and new studies start to focus on context alongside content as well. For instance, [1] proposed a search engine that combines traditional content-based search with context information gathered from users' activities. More recently, search engines started to make the search results more personalized. With personalized searches, search engines consider the searchers' preferences, interests, behavior, and history. The final goal of personalized search as well as other techniques studying users' preferences and interests is to make the returned results more relevant to what the user is actually looking for.

Emergence of social networks on the Web (e.g., Facebook and Google Plus) have caused the following key changes on the Web. First, social networks reconstruct friendship networks in the virtual world of the Web. Many of these virtual relationships are good representatives of their actual (friendship) networks in the real world. Second, social networks provide a medium for users to express themselves and freely write about their opinions and experiences. The social data generated for each user is a valuable source of information about that users' preferences and interests. Third, social networks create user identifiers (identities) for people on the Web. Users of a social network such as Facebook will have a unique identity that can be used in many places on the Web. Not only such users can use their Facebook identities on the social network itself but they can also use that identity to connect and interact with many other web sites and applications on the Web. Along the same lines, social networks such as Facebook and Google Plus provide utilities for other web sites to get integrated with them directly, enabling users of the social network to interact directly with those web sites and Web documents using their social network identity. For instance, a Web document can be integrated into Facebook (using either *Facebook Connect* or *instant personalization*[1]) allowing every Facebook user to perform several actions (e.g., *LIKE*, *RECOMMEND*, *SHARE*) on that document. Finally, many search engines are starting to connect to social networks and allows users of such social networks to be the users of the search engine. For instance, the Bing search engine is connected to Facebook and hence users with their Facebook identities can log in into Bing to perform their searches.

The above developments inspired us to study a new framework for search personalization. In this paper, we propose a new approach for performing personalized search using users' social actions (activities) on the Web. We utilize the new social information mentioned above (users' social activities, friendships, user identities, and interaction of users on Web documents) to personalize the search results generated for each user. We call this new approach to personalization of search, *persocialized search* since it uses *social* signals to *personalize* the search. While a traditional

[1] https://developers.facebook.com/docs/guides/web/.

personalized search maintains information about the users and the history of their interactions with the system (search history, query logs), a persocialized search system maintains information about the users, their friendships (relations) with other users and their social interactions with the documents (via social actions).

Recently, McDonnell and Ali [2] conducted a complete survey on the topic of *social search* and various existing approaches to conduct social search. As mentioned in McDonnell and Ali [2] there exist several definitions for social search: One definition is the way individuals make use of peers and other available social resources during search tasks [3]. Similarly, Vuorikari et al. [4] defines social search as using the behavior of other people to help navigate online, driven by the tendency of people to follow other people's footprints when they feel lost. A third definition is by Amitay et al. [5] and is defined as searching for similar-minded users based on similarity of bookmarks. Finally, Evans et al. [6]'s definition of social search includes a range of possible social interactions that may facilitate information seeking and sense-making tasks: utilizing social and expertise networks; employing shared social work spaces; or involving social data mining or collective intelligence processes to improve the search process. For us, social search focuses on utilizing querying users' as well as her friends' **social actions** to improve the conventional textual search. By integrating these social actions/signals into the textual search process, we define a new search mechanism: *persocialized search*. Our main goal in this paper is to prove our hypothesis that these social actions (from the querying user and his friends) are relevant and useful to improve the quality of the search results.

Toward this end, we propose a new relevance model called the *persocial* relevance model to determine the social relevance between a user and a document. Persocial model is developed in three levels, where each level complements the previous level. First, we are using social actions of a user on documents as implicit judgment/rating of those documents by the user. For instance if a Facebook user u, performs any type of social action (e.g., LIKES, SHARES) on document d, she implicitly expresses her positive opinion about d. As a result, d should get a slightly higher score for queries relevant to d and issued by u. In Sect. 4, we show that using social actions from each user and boosting documents' score with such actions (level 1), by itself improves the accuracy of search results. Second, it is both intuitive and proven [7] that people have very similar interests with their friends. Also, people tend to trust the opinions and judgements of their friends more than strangers. As a result, not only the documents with direct social actions by user u are relevant to u, but also those documents with social actions performed by u's friends are also relevant to user u. Hence, we adjust (increase) the weights given to those documents for relevant queries issued by u. As we discuss in more details in Sect. 3, many parameters such as the strength of social connections between users as well as the influence of each user must be incorporated in to the model for generating the most accurate results. In Sect. 4, we show that using the social signals from the friends will improve the search results significantly. Furthermore, we show that using a combination of user data and his/her friends data generates the best results. Finally, the Web documents are often well-connected to each other. We argue that social features of each document should be dynamic, meaning that social actions/signals of the document can and should be

propagated to other adjacent documents. A user's interest for a document—shown by a social action such as *LIKE*—can often imply the users' interests in other relevant documents—often connected to the original document. Thus, we use connections among documents to let social scores flow among documents, hence generating a larger document set with more accurate persocial relevance scores for each user.

In sum, the major contribution of this paper is to propose a model and build a system for utilizing users' social actions to personalize the Web search. We propose a new relevance model to capture relevance between documents and users based on users' social activities. We model three levels of personalization based on three sets of social signals and show how each level improves the Web search personalization. In addition, we propose three new ranking approaches to combine the textual and social features of documents and users. Furthermore, we develop a persocialized search engine dubbed *persocialized search engine* ('PERSOSE' for short) to perform persocialized search on real data using real users. Using *PERSOSE*, we conduct a comprehensive set of experiments using 14 million documents of Wikipedia as our document set and real Facebook users as our users. As a result of the experiments, we show that social actions of a user, his friends' social actions and social expansion of documents (all three levels of social signals) improve the accuracy of search results.

2 Overview

In this section, we present the problem statement without going into much details (we present some of definitions/formalizations in Sect. 3.1). We also provide the system overview of *PERSOSE*.

The objective of *PERSOSE* search engine can be stated as follows:

Suppose $D = \{d_1, d_2, \ldots, d_n\}$ is the set of documents that exist in our system. Each document is composed of a set of textual keywords. Also, there is a set $U = \{u_1, u_2, \ldots, u_m\}$ of users interacting with the system. Users can search for documents but more importantly users can also perform a set of defined *social actions* (e.g., *LIKE*, *RECOMMEND*, *SHARE*) on the documents. We also assume a social network modeled as a directed graph $G = (V, E)$ whose nodes V represent the users and edges E represent the ties (relationship) among the users. Finally, each query issued to the system has two parts: the textual part of the query which is presented by a set of textual keywords (terms), and social part of the query which is defined mainly as the user issuing the query. The goal of *PERSOSE* is to first identify and model the social dimension of the documents in the system, and next to score and rank the documents based on their relevance to both the textual and the social dimensions of the query. We call this type of search performed by *PERSOSE*, *PerSocialized Search* since search is *personalized* using *social* signals.

System Overview. A general overview of *PERSOSE* is displayed in Fig. 1. As shown in this figure, there exist two types of objects/modelus in *PERSOSE*: Modules that belong to the (existing) textual search models and modules that are new and are

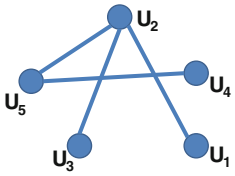

Fig. 1 Overview of *PERSOSE*

part of the new social model. In Fig. 1, textual modules are displayed by solid lines, social modules are depicted by dotted lines and modules with both textual and social features are shown by mixed lines.

Accordingly, *PERSOSE* has two engines: (1) the textual engine reads (crawls) the documents in the system and generates the necessary textual metadata for each document (e.g., textual vectors); there is nothing new about the textual engine, (2) the social engine has two inputs. One is the social network G with all its properties and relationships. The second data structure maintains a dataset of users' social activities. This dataset, for each user in the social network, contains all their social activities (feed) including their interaction with documents in the system. The social engine processes this dataset as well as graph G and generates multiple social vectors for documents and users. In addition to the social vectors, the social engine defines and calculates relevance scores between documents and users as well as among documents. Description of each vector as well as the detailed description of the new relevance model are discussed in Sect. 3.1.

Another major module in our system is the ranker module. Ranker which contains both the textual and persocial aspects, receives queries from each user and generates a ranked list of documents for each query and returns them back to the user. As we mentioned earlier, each query has two parts: the textual part of the query (set of terms) and the user issuing the query. Ranker gets both information as well as different vectors generated from the textual and social engines and using one of the approaches described in Sect. 3.2 ranks the documents based on their (textual and social) relevance to the query. Details of different ranking approaches are discussed in Sect. 3.2.

3 Persocialization

In this section, we show how to personalize the search results using social data or what we call *search persocialization*. First, we propose a new relevance model called persocial relevance model to capture and model the social information for both documents and users. In the second part, we show how to use the proposed persocial relevance model to perform persocialized search and propose various rankings.

3.1 Persocial Relevance Model

In this section, we model social relationships between users and documents as well as other social information about users, and propose a new weighting scheme to quantify the relevance of each user to each document.

We define the persocial relevance model at three levels, each level complementing the previous level. We develop the simplest model in level 1 using minimum amount of social data, i.e., social data from user himself. We extend our model significantly in level 2, creating the core of our persocial model. In this level, we also define multiple new social vectors in order to be able to model the persocial relevance more accurately. In the process of modeling level 2 persocial relevance, we create a new weighting scheme called *uf-ri* weighting scheme and define new weights and weight functions for several relationships in the system. Finally, in level 3, we extend our model even further using the concept of *social expansion.*

3.1.1 Persocial Relevance—Level 1

In the first level of the persocial model, we leverage each user's past social data to calculate the persocial relevance between that user and the documents.

Definition We formalize social interactions between users and documents by *social actions*. We define $A = \{a_1, a_2, \ldots, a_l\}$ as a set of all possible *social actions* available to the system. For each document d_j, a set A_{d_j} defines a set of valid (supported) actions for d_j. A_{d_j} is a subset of A ($A_{d_j} \subseteq A$) and contains all the social actions possible for document d_j. For each user u_i we define a set UDA_i as a set of all document action pairs performed by user u_i. To be more formal, $UDA_i = \{(d_j,a_k) \mid$ if there is an action a_k on document d_j by user $u_i\}$. Each social action is unique and can be applied only once by user u_i on document d_j (nevertheless, that action can be applied by the same user u_i on multiple documents and/or by multiple users on the same document d_j).

Social actions do not have equal importance. We define a weight function W: $A \rightarrow \mathbb{R}$ mapping social actions to real numbers in the range [0, 1]. Values generated by the weight function represent the importance of each social action in the system. The weight function should be designed by a domain expert with the following two constrains: (1) each weight should be between 0 and 1 (inclusive), and (2) the more important the action, the higher the value. The importance of actions are determined based on the domain/application.

Example Assume that our document set contains all the Web pages of a sports web site (e.g., ESPN). Web pages can include news articles, athlete profile pages, sports teams pages and so on. Also, this web site is already integrated (connected) with a social network platform. In this example, all Web pages in our document set are

connected to the Facebook social plug-ins[2] and support the following social actions: *LIKE*, *RECOMMEND* and *SHARE*.

So, $A = \{LIKE, RECOMMEND, SHARE\}$ and also

$A_{d_j} = \{LIKE, RECOMMEND, SHARE\}$ for each and every d_i in our document set (all documents support all actions).

Each user u_i in the system, can *LIKE*, *RECOMMEND* or *SHARE* any document d_j on the web site. With this example, we define weight function W as follows: $W(\text{RECOMMEND}) = 0.6$, $W(\text{LIKE}) = 0.6$, and $W(\text{SHARE}) = 0.8$. These weights indicate that in this domain, *SHARE* is the most important action and *LIKE* and *RECOMMEND* actions have the same importance.

Definition *Persocial relevance—level 1* between document d_j and user u_i is defined based on the number and type of social actions between user u_i and document d_j, and as follows:

$$psRel_{L1}(u_i, d_j) = \sum_{a_k | (d_j, a_k) \in UDA_i} W(a_k)$$

where $psRel_{L1}(u_i, d_j)$ is the persocial relevance level 1 between user u_i and document d_j.

Example In our running example, assume we have two documents d_1 and d_2 and user u_1. User u_1 has *LIKED* and *SHARED* d_1 and he also has *RECOMMENDED* document d_2. Hence, $prRel_{L1}(u_1, d_1) = W(LIKE) + W(SHARE) = 1.4$ and $prRel_{L1}(u_1, d_2) = W(RECOMMEND) = 0.6$.

3.1.2 Persocial Relevance—Level 2

The amount of data generated from one user's social actions is typically insignificant. If we only consider the users' own social actions, many documents will end up having persocial relevance of zero for that user. In addition, as we discussed earlier people have very similar interests with their friends trust the opinions of their friends more than others. Hence, in the second level of persocial model, we utilize friendship relationships between users to improve and extend the level 1 model.

Definition A weight $w_{i,j} > 0$ is associated with each user u_i and document d_j. The term $w_{i,j}$ represents the social importance/relevance of user i to document d and its value is equal to $prRel_{L1}(u_i, d_j)$ defined earlier. For user u_i with no social action on document d_j, $w_{i,j} = 0$.

[2] https://developers.facebook.com/docs/plugins/.

We define *document social vector* to represent the social dimension of the document d_j and represent it as S_{d_j}, defined as bellow:

$$S_{d_j} = (w_{1,j}, w_{2,j}, \ldots, w_{m,j})$$

where m is total number of users.

The concept of social vector for a document is analogous (and inspired by) the concept of the textual vector of a document. While textual vector represents the textual dimension of the documents, social vector characterizes the social dimension of the documents. Moreover, our weights $w_{i,j}$ are analogous to term frequency weights ($tf_{i,j}$) in the context of textual search. While each $tf_{i,j}$ indicates the relevance between term (keyword) i and document j, each $w_{i,j}$ represents the relevance between user i and document j. Traditionally (and in the context of textual search), such term frequency is referred as *tf* (*term frequency*) *factor* and offers a measure of how well that term describes the document's textual content. Similarly, we name our social weights ($w_{i,j}$) *uf* (*user frequency*) *factor*. The *uf* factor provides a measure of how well a user describes a document's *social* content.

Example Continuing with our running example, let's add users u_2 and u_3 to the system. Suppose u_2 has *LIKED* document d_1 and u_3 has no social action on d_1. Given this information and previous information about u_1, the social vector for d_1 is as follows.

$$S_{d_1} = (w_{1,1}, w_{2,1}, w_{3,1}) = (1.4, 0.6, 0).$$

Definition We measure $w'_{i,p}$ or the weight between user u_i and user u_p based on the *user relatedness function* between user u_i and u_p. User relatedness function is denoted by $W'(u_i, u_p)$ and measures the relatedness/closeness of two users. There are several existing measures to calculate the relatedness/closeness of two nodes in a graph/social network. Some of the approaches consider the distance between nodes, some look at the behaviors of users in a social network and some take into consideration number of mutual neighbors of two nodes. While the required data is available, any of the above methods or any other existing method can be used for the user relatedness function as long as the following three constraints are satisfied: (1) $W'(u_i, u_i) = 1$, (2) $0 \leq W'(u_i, u_p) \leq 1$ and the more relevant the users, the higher the value, and (3) $W'(u_i, u_p) = 0$ when $W'(u_i, u_p) < \delta$. The first constraint states that each user is the most related user to himself. The second constraint normalizes this measure and also ensures that the more related users are assigned higher scores. Finally, the third constraint filters out all relationships that their significance is below a certain threshold (δ).

Now, we define *user social vector* to represent the social dimension of the user u_i and present it as S'_{u_i}, defined it as below:

$$S'_{u_i} = (w'_{1,i}, w'_{2,i}, \ldots, w'_{m,i}).$$

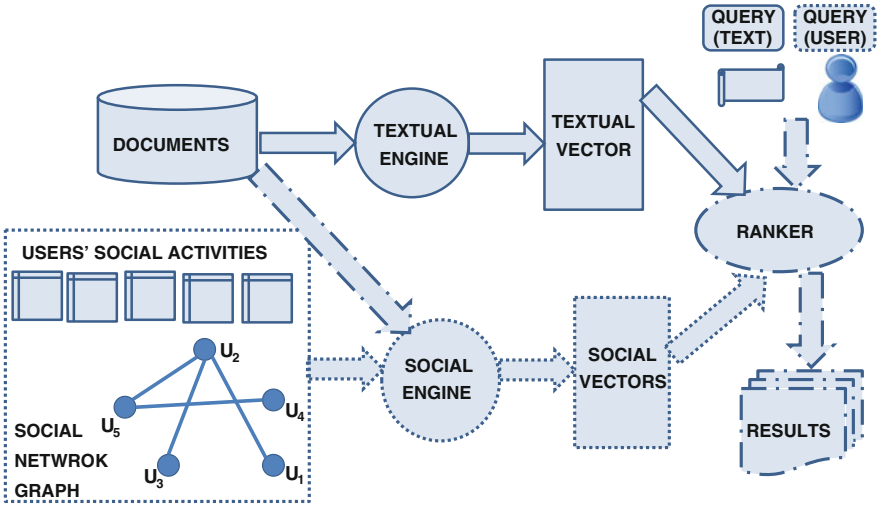

Fig. 2 Friendship structure for the running example

Example Let's add users u_4 and u_5 to the running example. Friendship structure among all five users of our system is depicted in Fig. 2.

In the following, we calculate the user social vector for user u_1 using two different user relatedness functions.

As case 1, we use an inverse of distance between two users (in the network) to capture their relatedness. We also set the threshold value δ equal to 0.3. More formally, $W'(u_i, u_p) = \frac{1}{dist(u_i,u_p)+1}$ where $\delta = 0.3$ and $dist(u_i, u_p)$ is the number of edges in a shortest path connecting u_i and u_p ($dist(u_i, u_i) = 0$). Using this function for user $u1$:

$$\begin{aligned} S'_{u_1} &= (W'(u_1, u_1), W'(u_2, u_1), W'(u_3, u_1), W'(u_4, u_1), W'(u_5, u_1)) \\ &= (1, 0.5, 0.33, 0, 0.33). \end{aligned}$$

Note that $W'(u_4, u_1) = 1/(1 + 3) = 0.25$ but since $0.25 < 0.3$, this value becomes zero.

In addition to relatedness between users, knowing the overall importance/influence of each user also can help us in detecting (and thus giving more weight) to social actions with higher quality and more reliability. Often, when a high profile user (super user) performs a social action on a document, that action and consequently that document are of higher value/quality compared to the case when the same action is performed on the same document by a less influential user.

We quantify the overall (global) importance of each user by the *user weight function* $W''(u_i)$. This measure quantifies the significance of a user in the social network. For instance, with Twitter, a user with many followers will be assigned a higher weight than a user with only few followers, or with Facebook, a user with

more friends is often more important to the social network than a user with fewer friends. In the field of graph theory and social networks, this value is called *centrality* and there exist several approaches to measure it. Four popular methods to compute the centrality value are: degree centrality, betweenness, closeness, and eigenvector centrality [8]. Similar to the user relatedness function, the user weight function is also generic enough and most of the existing approaches can be applied to obtain W''.

Definition We define a weight function $W'': U \rightarrow \mathbb{R}$ mapping users to real numbers in the range [0,1]. Each value $w''(i)$ generated by this weight function represents the overall importance of each user i in the system. The weight function should satisfy the following two constrains: (1) each $w''(i)$ should be between 0 and 1 (inclusive), and (2) the more important the user, the higher the value. The importance of users are determined by user weight function[3] W".

In the context of textual search, there is the *idf* (*inverse document frequency*) *factor* for each term in the system offering a measure to how important (distinctive) is that term in the system. Analogously, we name the weights generated by the weight function W", *ui* (*user influence*) *factor*. The value of *ui* for a user provides a measure of how important that user is in the system.

We define *influence social vector* to represent the importance/influence of all the users, and present it as S''. S'' is defined as follows:

$$S'' = (w''_1, w''_2, \ldots, w''_m).$$

Example For the network depicted in Fig. 2, we use the degree centrality of nodes (users) as an indication of their importance as follows:

$$W''(u_i) = \frac{deg(u_i)}{m-1}$$

where $deg(u_i)$ is the number of edges of node u_i and m is number of nodes (users).

Using the above user weight function, the following weights are generated for the five users:

$$w''(u_1) = 0.5, w''(u_2) = 0.75, w''(u_3) = 0.5, w''(u_4) = 0.25, w''(u_5) = 0.5.$$

Thus, $S'' = (0.5, 0.75, 0.5, 0.25, 0.5)$

Definition *Persocial relevance—level 2* between document d_j and user u_i is defined based on the number and type of social actions between user u_i and document d_j, the relationships between user u_i and other users, the overall importance of each user and the number and type of social actions between user u_i's friends[4] and document d_j, as follows:

[3] Commercialized and more complicated examples of this measure include Klout (klout.com) and PeerIndex (peerindex.com).

[4] To be more precise, set U′ of users such that $\forall u'_l \in U' | W'(u'_l, u_i) > \delta$.

$$psRel_{L2}(u_i, d_j) = \sum_{k=1}^{m} (w(k, j) \times w'(k, i) * \times w''(k)) \quad (1)$$

where $w(k, j)$ is the user frequency (*uf*) factor, $w'(k, i)$ is the user relatedness (*ur*) factor, and $w''(k)$ is the user influence (*ui*) factor. We call this weighting scheme *uf-ri* (user frequency-relatedness influence) weighting scheme. While in classical textual weighting schemes such as tf-idf, for given terms, more weight is given to the documents with (1) more occurrences of terms (tf) , and (2) more important terms (idf), in our *uf-ri* weighting scheme, for a given user, more weight, is given to the documents with (1) more important actions (2) performed by more important users (3) whom are more related (closer) to the given user.

Example Given the values we have so far (using case 2 for W'), the persocial relevance level 2 between $u1$ and document d_1 is calculated as follows:

$$prRel_{L2}(u_1, d_1) = \sum_{k=1}^{5} (w(k, 1) \times w'(1, k) * \times w''(k)) = 1.4 \times 0.5 + 0.6 \times 0.5 \times 0.75 + 0 + 0 + 0 = 0.7 + 0.225 = 0.925$$

3.1.3 Persocial Relevance—Level 3

In this section, we present the concept of *social expansion* and discuss how it can be useful in generating more accurate persocial relevance scores. We show how to define level 3 of persocial relevance by integrating social expansion to the persocial relevance level 2.

Each document on the Web is often well connected to other documents, most commonly using hyperlinks. We argue that social features of each document should be dynamic, meaning that social actions/signals of the document can and should be propagated to other adjacent documents. A user's interest for a document—shown by a social action such as *LIKE*—can often imply the users' interests in other relevant documents—often connected to the original document. In simpler words, we enable social signals to *flow* in the network of documents.[5] We propose to propagate social actions from one document—with some social action—to all documents connected to that document. As an example, imagine a user *LIKES* ESPN's *Los Angeles Lakers* page. Using this signal (action) alone can help us deriving the fact that this document is socially relevant to this user. However, we can do much better by taking into consideration the adjacent documents to the *Los Angeles Lakers* document.

By looking at documents that the original document links to, we can retrieve a new set of documents that are also socially relevant to the user. For our example, the *Los Angeles Lakers* document has outgoing links to document on *NBA* and *Kobe*

[5] Many existing approaches and definitions can be used to measure *connections* between documents. Here, we do not go into details of such approaches.

Bryant. Assuming there is one outgoing link for each of the two documents, half of the original social score can be given to each of these two new documents. As a result, documents on *NBA* and *Kobe Bryant* become socially relevant to the user as well (note that the original *Los Angeles Lakers* document is still more socially relevant to the user than the other two documents.). If we continue this propagation, many new documents will get adjusted social scores from the same social action.

We define *persocial relevance level 3* ($psRel_{L3}$) between document d_j and user u_i as follows:

$$psRel_{L3}(u_i, d_j) = psRel_{L2}(u_i, d_j) + \sum_{d_k \in D_{d_j}} V'(d_k, d_j) \times psRel_{L2}(u_i, d_k) \quad (2)$$

where $psRel_{L2}(u_i, d_j)$ is the persocial relevance between document d_j and user u_i (level 2) as defined in Eq. 1, D_{d_j} is a set of documents connected to the document d_j, and $V'(d_k, d_j)$ is value of *document relatedness function* between document d_k to document d_j. *Document relatedness function* is measuring the connectivity of two documents. Again, we intentionally define this function as generic as possible and do not limit our model by any particular implementation. Simple models like number of hyperlinks between two documents or more sophisticated models such as those that calculate the textual and/or topical similarities between two documents can be used.

The main advantage of using social expansion is to find more *socially relevant* documents for each user. Social expansion also helps in adjusting documents' scores and assigning more accurate relevance scores to each document. Imagine a user who has two explicit *LIKES* on *Google* and *Microsoft*. The same user also has other social actions on *XBOX* and *Bing*. Without using expansion, both *Google* and *Microsoft* generate the same social weight for this user, while using expansion will propagate some weight from both *XBOX* and *Bing* to *Microsoft* and hence gives *Microsoft* a slight advantage (assuming there are links from *XBOX* and *Bing* to *Microsoft*). Using social expansion is also very practical for the current state of the Web where social actions are not very common yet and many documents do not have any social action. Social expansion will help more documents to get scored and hence it will improve the overall social search experience.

3.2 Persocialized Ranking

As described earlier, goal of the ranker module in *PERSOSE* is to personalize and rank the search results using both the social and textual features of the documents. In this section, we discuss three different approaches to rank the documents based on the combination of the textual relevance and persocial relevance scores. In any of the discussed approaches, persocial relevance model of any level (1 through 3) can be applied. Hence, for instance, if friends' information do not exist in the system and

only querying users' own actions are available, we can use persocial relevance level 1 as the persocial relevance model in the proposed approaches. We also incorporate textual relevance in the proposed approaches. Any existing textual model (e.g., tf-idf [9], BM25 [10]) can be used to calculate the textual relevance scores. Furthermore, we have to note that most of the existing search optimization techniques (e.g., pageRank [11]) or other personalized approaches are orthogonal to our approaches and can be added to textual relevance model part (for instance combination of the tf-idf and pageRank can be used as the textual model).

3.2.1 Textual Filtering, Persocial Ranking

In *textual filtering, persocial ranking* (TP) approach, first a regular textual filtering is conducted and all the documents with textual relevance larger than 0 are returned (in the simplest case, documents that have at least one of the query keywords). Next, the remaining documents are scored and ranked using their persocial relevance to the querying user. This is a two-step process in which filtering is based on the textual dimension of the documents and ranking is based on the social aspect of the documents.

3.2.2 Textual Ranking, Persocial Filtering

In *persocial filtering, textual ranking* approach, any document d_j with no persocial relevance to the querying user u_i (i.e., $psRel(u_i, d_j) = 0$) is pruned first. The result of this step is a set of documents with at least one social action from the querying user or her friends (related users). Next, the regular textual search is performed on the remaining documents and documents are scored and ranked based on the textual relevance model. This is also a two-step process with filtering step based on social dimension of the documents and ranking step based on the textual features of the documents.

3.2.3 Persocial–Textual Ranking

With *persocial–textual ranking* approach, both textual and social features of the documents are used simultaneously to calculate the final relevance of the query to each document. We define $Rel(q, d_j)$ as the overall (textual plus persocial) relevance of document d_j with query q. The value of $Rel(q, d_j)$ is defined by a monotonic scoring function F of the textual relevance and persocial relevance values. In *PERSOSE*, F is the weighted sum of the persocial relevance and textual relevance scores:

$$\begin{aligned} Rel(q, d_j) &= F(psRel(u_q, d_j), texRel(T_q, d_j)) \\ &= \alpha.psRel(u_q, d_j) + (1 - \alpha) \times texRel(T_q, d_j) \end{aligned}$$

where T_q is the textual part of the query, u_q is the querying user (social part of the query), $texRel(T_q, d_j)$ is a textual relevance model to calculate the textual relevance between T_q and document d_j, and α is a parameter set by the querying user, assigning relative weights to persocial and textual relevance values.

In this approach and using the above formula, ranking is calculated using both the textual and social features of documents and the query. This is a one-step process with no filtering step.

4 Experimental Evaluation

In this section, we evaluate the effectiveness of *PERSOSE* using data from Facebook and Wikipedia. We first discuss the dataset, approaches and other settings used for the experiments, and then present the results.

Data. For a complete and accurate set of experiments, we need a dataset that contains the following data: (1) a large set of documents with textual information, (2) link structure between documents, (3) real users with friendship relationships, and (4) social actions from users on documents.

Unfortunately no such dataset exists. As a result, we built such a dataset to be used in *PERSOSE* and to evaluate our approaches.

As outlined in Sect. 2, two main data types are fed into *PERSOSE*. One is a set of documents and the other is the social data containing social actions from users as well as relationships among users. We used Wikipedia articles as our document set and Facebook as our social platform. We developed a Web crawler to crawl around 14 million Wikipedia articles and extract textual information from those documents. While crawling, we also captured the relationships among documents and built a (partial) Wikipedia graph. In this graph, each node represents a Wikipedia article. Node d_1 has a directed edge to node d_2 if their Wikipedia articles are related to each other, either explicitly when article d_1 has a link to article d_2, or implicitly when article d_2 is mentioned several times by article d_1. The weight of each connection is based on the frequency and the position of the mentions of one article inside another. The total weight of all outgoing edges for each node of the graph always adds up to one.

As far as the social data, we integrated *PERSOSE* to Facebook using *Facebook Connect*, hence allowing users to log in into *PERSOSE* using their Facebook account and information. When a user connects to *PERSOSE*, our system asks for the permission to read and access users' facebook data. The Facebook data that our system read include users's Facebook activities (e.g., *STATUS*, *LIKES*, *PHOTOS*) as well as users' friendship information. We also read all public data from the users' friends.

Finally, we map users' Facebook activities to social actions on Wikipedia documents. In order to perform this step, we utilized the technology developed at GraphDive[6] to link Facebook data to Wikipedia articles. With GraphDive API, each

[6] http://graphdive.com/.

Facebook activity/post (e.g., *STATUS*, *CHECK-IN*, *LIKE*) can be mapped to one or more than one Wikipedia article. GraphDive algorithm works as follows. GraphDive API receives a Facebook post, parses the text to all possible word-level n-grams ($1 \leq n \leq$ total number of words in the post) and then looks for a Wikipedia article with the same title for each n-gram. For instance, for a *status update* of "I love Los Angeles and Southern California", GraphDive API, will match Wikipedia articles on *Los Angeles*, *California*, and *Southern California* to the post. There are other optimizations taken place by GraphDive API (e.g. disambiguation, varying weights for each n-gram, etc.) that are not the focus of this paper. We only use GraphDive API to map Facebook actions to Wikipedia articles and hence generating a rich set of documents with social actions from real users.

Actions. From the data that Facebook provides via its graph API,[7] we considered the following six actions: *LIKE*, *check-in*, *STATUS*, *PHOTO*, *WORK* and *SCHOOL*. *LIKE* is when a user *likes* a page/topic on Facebook or a document on the Web. *check-in* is when a user *check-ins* his/her location using Facebook. *STATUS* is a free format text usually describing users' activities and feelings. *PHOTO* is a text associated with each photo a user uploads to Facebook. Finally, *WORK* and *SCHOOL* are information about users' workplace and school/university, respectively. Each of the above six actions contain some textual content. As described above, using GraphDive technology, we map those textual content to a set of Wikipedia articles—when possible. For instance, when a user check-ins at *Peet's Coffee and Tea*, using GraphDive, we extract action *check-in* between the user and the Wikipedia article on *Peet's Coffee and Tea,* between the user and the Wikipedia article on *coffee,* and between the user and the Wikipedia article on *tea*.

Approaches. We use three main approaches described in Sect. 3.2 to generate the results: *textual filtering, persocial ranking* (TP), *persocial filtering, textual ranking* (PT) and *persocial–textual ranking* (HB).[8] We also use a baseline approach called *BS*. The *BS* approach generates the results based on the combination of tf-idf and PageRank models.

The same baseline approach is used as the textual model in our existing approaches (whenever textual model needed). The default setting is as follows. The social actions have the same weight (all equal to 0.5) and number of results returned for each approach is 5. When using friends data, we only access data from the top 25 friends (ranked by their user relatedness score to the user) of the user. Also, all four approaches use *expansion* as described in Sect. 3.1.3. Finally, α is set to 0.7 for the *HB* approach (to give more importance to the social part of the search and hence evaluate the impact of social signals more thoroughly). In addition to the main approaches and the baseline approach, we also implemented three levels of the persocial mode on the *hybrid* approach to study and evaluate the impact of each level. Three variations are called: *HB-Level1*, *HB-Level2*, and *HB-Level3*.

Queries. We generate two set of queries for our experiments. The first set called *qset1*, is generated from Google top 100 queries in 2009. For each user, five queries

[7] https://developers.facebook.com/tools/explorer/.

[8] *HB* stands for hybrid.

are randomly generated from that list. The second set of queries called *qset2* is generated from each user's social data. With *qset2*, we randomly generate 5 queries from users' own Facebook data (e.g., pages they liked, city they live, school they attended). We believe *qset2* is of higher quality since these are the queries that users are very familiar with and hence can understand and evaluate better. (For instance, user living in *Irvine, California* can evaluate the results for query *Irvine, California* very well.). Another benefit of choosing queries from users' Facebook profile is a higher chance of having social actions from the user on the query topic.

As a result, using *qset2* provides us with a better evaluation of our system. Note that in the absence of any social signal, our approaches will perform the same as the baseline approach and hence will not provide many new insights. For the above reasons, we only use *qset1* for the first set of experiments (comparing the main approaches) and use *qset2* for other experiments.

Relevance Assessment. After computing the top-5 results for each of our queries using all approaches, we ran a user study using Amazon Mechanical Turk.[9] One task (hit) was generated for each query. Users of our experiments were typical Mechanical Turk users that were willing to connect using Facebook Connect (and share their data with us) and had at least 50 Facebook friends. We asked workers to log in to our experiment setting using their Facebook account[10] via Facebook connect.[11] For each query and for each worker, top 5 results from all approaches were generated, mixed together (duplicates removed) and presented to the worker. Workers could choose whether each result (Wikipedia article) is *very relevant*, *relevant* or *nonrelevant*. User were not aware of different approaches and could not tell which results is for what approach. Moreover, for each query, we asked each user to provide us with an ordered list of top-5 most relevant documents (from the documents presented) based on his/her own preferences. We use this information to calculate nDCG for each query.

Each task (query assessment) was assessed by 12 workers for query set 1 and 8 workers for query set 2. Each worker was rewarded $0.25 by completion of each assessment.

User Relatedness. To capture the relatedness between two users, we used the total number of interactions between those users (on Facebook) as our metric. We retrieved and counted all the direct interactions (except private messages) between two users and used normalized value of this number as the value of our user relatedness function. Although we could use simpler metrics such as number of mutual friends, we believe that the number of actual interactions is a better representative of relatedness/closeness of two Facebook users than the number of mutual friends between them.[12]

---

[9] mturk.com.

[10] Each volunteer allowed us to read/access his/her Facebook data for this experiment.

[11] https://developers.facebook.com/docs/guides/web/.

[12] For instance, you may have a lot of mutual friends with your high school classmate, without being *close* or *related* to that person. On the other hand, you may not have a lot of mutual friends with your spouse or sister, and still be *close* to them.

Evaluation Metric. We evaluated the accuracy of the methods under comparison using popular nDCG@k and precision@k metrics. nDCG@k and precision@k are the two main metrics used for comparison of different ranking algorithms. Discounted Cumulative Gain (DCG) is a measure for ranking quality and measures the usefulness (gain) of an item based on its relevance and position in the provided list. For comparing different lists with various lengths, normalized Discounted Cumulative Gain (nDCG) is used. It is computed by dividing the DCG by the Ideal Discounted Cumulative Gain or IDCG. The higher the nDCG, the better ranked list. When computing nDCG@k, we considered the ordered list of top-5 results entered by the user as the ideal ordering (IDCG).

Another important metric is precision@k. What matters in many search engines is how many good results there are on the ïňĄrst page or the ïňĄrst three pages (vs. traditional metrics such as recall). This leads to measuring precision at ïňĄxed low levels of retrieved results, such as 10 or 30 documents. This is referred to as precision@k (precision at k), e.g., prec@10. It has the advantage of not requiring any estimate of the size of the set of relevant documents.

The relevance values used for *very relevant*, *somehow relevant* and *not relevant* are 2, 1, and 0, respectively. We calculate prec@k for two scenarios. For the first scenario *prec@k (rel)*, we considered the results evaluated as *somehow relevant* or *very relevant* as relevant. For the second scenario *prec@k* (*vrel*), we only considered the results evaluated as *very relevant* as relevant.

We calculated the final nDCG and precision values by averaging all nDCG and precision values for each query.

4.1 Main Approaches

In the first set of experiments, we evaluate the effectiveness of our three main approaches (rankers) and compare the results with the baseline approach.

The results—prec@5(rel), prec@5(vrel) and nDCG@5—of the four approaches and the two query sets are shown in Tables 1 and 2, respectively. The first observation is that for *qset2*, all our proposed approaches (*TP*, *PT* and *HB*) are noticeably better than the baseline (*BS*) approach. The second observation is that for *qset1*, while *HB* outperform *BS* with regards to all three metrics, the other two social approaches are

Table 1 Main approaches: *qset1*

Approach	prec@5(rel)	prec@5(vrel)	nDCG@5
BS	0.714	0.359	0.760
TP	0.630	0.329	0.652
PT	0.787	0.413	0.655
HB	0.760	0.420	0.815

Table 2 Main approaches: *qset2*

Approach	prec@5(rel)	prec@5(vrel)	nDCG@5
BS	0.787	0.491	0.689
TP	0.856	0.626	0.806
PT	0.890	0.628	0.777
HB	0.846	0.590	0.792

not as successful. This observation plus the first observation show that the hybrid (*HB*) approach is the best approach among all four approaches for all cases. We can also see that while the other two persocial approaches work pretty well for some queries (queries that users already have some related social actions), they may generate less accurate results for random/generic queries (although *PT* still outperforms *BS* for two of the three metrics). This shows that search persocialization works best for queries relevant to the querying user (queries such that the querying user has some social actions on documents relevant to those queries). The third observation is that for both datasets, the margin that our persocial approaches (except *TP* in *qset1*) are better than the baseline approach increases from prec@5(rel) to prec@5(vrel). This shows that if users are looking for *very relevant* results, our proposed approaches generate even better results.

4.2 Persocial Relevance Levels

In this set of experiments, we evaluate and compare the results generated from the three levels of persocial relevance with each other and also with the baseline approach. We use *HB* as our persocial ranker and *qset2* as the dataset. Results for three levels and the *BS* approach are shown in Table 3.

The first observation is that all three levels generate more (or equal for level 1 with regards to nDCG@5(rel) metric) accurate results than the baseline approach in regards to all the three metrics. This not only confirms the fact that our final proposed approach (level 3) generates more accurate results than the baseline, but also shows that even applying one or two levels of our persocial model can improve the search results. The second observation is that in regards to all three metrics, each level improves the accuracy of search results in comparison to the previous level. As we

Table 3 Levels

Approach	prec@5(rel)	prec@5(vrel)	nDCG@5
BS	0.787	0.491	0.689
HB-level1	0.787	0.506	0.730
HB-level2	0.809	0.548	0.744
HB-level3	0.846	0.590	0.792

discussed earlier, each level is built on top of the previous level and complements it by adding mode social signals to the persocial relevance model. in other words, this set of experiments proves our hypothesis and shows that (1) social actions improve the search results, (2) using friends social signals further improves the accuracy of the results, and (3) social expansion also adds to the accuracy of search personalization.

Overall, applying all three levels to the baseline approach will improve both the precision of nDCG of the results significantly. Metrics prec@5(vrel) and prec@5 (vrel) improve from 0.78 and 0.49 to 0.84 and 0.59 (6 and 20 % improvements), respectively. Also, the final ordering of the results in comparison to the ideal ordering (nDCG@5) improves significantly from 0.68 to 0.79 (16 % improvement) as well.

4.3 Friends Versus User

In this set of experiments, we compare the impact of using social data from friends-only, user (querying user) only, or a combination of both on our proposed model. We developed two new variations of *HB* called *HB-UO* (*User-Only*) and *HB-FO* (*Friends-Only*) and compare them with each other and also with the original *HB*. Again, *qset2* is used and social expansion is enabled for all the approaches. Results for the three approaches are shown in Table 4. The first and important observation is that the friends-only approach generates results as effective or even better than those of the user-only approach. This further proves the point that friends' interests and preferences are very similar to the users' own interests and preferences. This finding encourages using friends' actions in the search and ranking process.

The second observation from Table 4 is that *HB* is the best approach among all three (reconfirming the observation that level 2 results are better than level 1 results). As we also saw earlier (for the nonexpanded case), we can see that mixing data from both the querying user and his friends will generate the most accurate results.

4.4 Number of Friends

In this set of experiments, we evaluate the impact of number of friends of the querying user on the accuracy of the results. We categorize users based on their number of friends into three groups: *popular*, *semipopular* and *nonpopular*. *Nonpopular* users are those with fewer that 200 friends (between 50 and 200). *Semipopular* users are

Table 4 User-only versus friends-only

Approach	prec@5(rel)	prec@5(vrel)	nDCG@5
HB	0.846	0.590	0.792
HB-UO	0.823	0.545	0.778
HB-FO	0.831	0.582	0.777

Table 5 Number of friends

Number of friends	prec@5(rel)	prec@5(vrel)	nDCG@5
Popular	0.889	0.626	0.826
Semipopular	0.821	0.564	0.782
Nonpopular	0.780	0.540	0.733

those with fewer than 500 friends and more than 200 friends. Finally, *popular* users are those with more than 500 friends (the most number of friends value among our workers is 1312). We present the prec@5(rel) results for the three groups in Table 5.

The main observation is that the accuracy of the results is directly correlated with the number of friends of the querying user. The *nonpopular* group generates the least accurate results and this is expected since not many social signals from friends and perhaps even from the user himself (users with fewer friends tend to be less active on their social network) are used to influence the search. The *popular* group generates the most accurate results, and *semipopular* group is in between. This observation shows that the larger the amount of data from a user's friends, the persocial relevance scores for that user is more accurate and hence the results generated for that user is improved.

To summarize, the main observations derived from our experimental evaluation are:

- Each level of our persocial model improves the accuracy of search results compared to the previous level. All levels generate more accurate results than the baseline approach.
- For *qset2*, all three proposed approaches generate more precise results and a better ranking than the baseline approach.
- For *qset1*, our proposed *HB* approach, generate more accurate results than the baseline approach (for all three metrics), while the results of the other two approaches vary.
- Results generated only from users' friends social data only is as good (if not better) than the results generated from users' own social actions. The best results are achieved when combining users' own and friends' social data.
- Accuracy of results for each user is directly correlated with the number of friends for that user.

5 Related Work

There are several groups of related studies on the application of social networks in search. With the first group, people through their social networks are identified and contacted directly to answer search queries. In other words, queries are directly sent to individuals and answers to the queries are coming from people themselves [12–14].

In this approach called *search services*, people and their networks are indexed and a search engine has to find the most relevant people to send the queries/questions to. An example of search services is the work in [12]. Except for the work in [12], there are not many academic studies regarding search services. There are also systems based on the synchronous collaboration of users in the search process. HeyStacks [15], as an example of such system, supports explicit/direct collaboration between users during the search. HeyStacks enables users to create *search tasks* and share it with others. HeyStacks is a complementary (and not comprehensive) search engine that needs to work a mainstream search engine to be useful.

In [16, 17], authors show how social platforms (such as Facebook, Linkedin) can be used for crowdsourcing search-related tasks. They propose a new search paradigm that embodies crowds as first class sources for the information seeking process. They present a model-driven approach for the specification of crowdsearch tasks. Crowdsourcing search tasks or *crowdsearching* is a fairly new topic focusing on the active and explicit participation of human beings in the search process. Some interesting models and applications of crowdsearching are presented in [14, 18].

Personalized search has been the topic of many studies in the research community. Search engine can either explicitly ask users for their preferences and interests [19, 20] or more commonly, use data sources related to users' search history such as query logs and click-through data. The most common data source used in search personalization is users' Web (query) log data. Some studies also look at different sources of personal data such as email and desktop files [21]. Recently, few studies started to exploit data from social online systems to infer users' interests and preferences. Xu et al. [22] and Noll and Meinel [23] exploit each user's bookmarks and tags on social bookmarking sites and proposes a framework to utilize such data for personalized search. In a similar paper [24], authors explore users' public social activities from multiple sources such as blogging and social bookmarking to derive users' interests and use those interests to personalize the search.

In [25], authors investigate a personalized social search engine based on users' relations. They study the effectiveness of three types of social networks: familiarity-based, similarity-based and both. In [26], which is a short paper, authors propose two search strategies for performing search on the Web: textual relevance (TR)-based search and social influence (SI)-based search. In the former, the search is first performed according to the classical tf-idf approach and then for each retrieved document the social influence between its publisher and querying user is computed. The final ranking is based on both scores. In the latter, first the social influence of the users to the querying user is calculated and users with high scores are selected. Then, for each document, the final ranking score is determined based on both TR and SI.

In a set of similar papers [27–29], authors propose several social network-based search ranking frameworks. The proposed frameworks consider both document contents and the similarity between a searcher and document owners in a social network. They also propose a new user similarity algorithm (MAS) to calculate user similarity in a social network. In this set of papers, the focus is mainly on user similarity functions and how to improve those algorithms. Majority of their experiments are limited

to a small number of queries on YouTube only. Also their definition of a *relevant* document is somehow ad hoc. A relevant (interesting) result is a result (video) whose category is similar/equal to the dominant category of videos that the searcher has uploaded.

With regards to commercial search engines, Bing and recently Google have started to integrate Facebook and Google+, respectively, into their search process. For some search results, they show the query issuer's friends (from his/her social network) that have *liked* or *+1ed* that result. Their algorithms are not public and it seems that they only show the *likes* and *+1s* and the actual ranking is not affected.

There exists a relevant but somehow different topic of *folksonomies*. Tags and other conceptual structures in social tagging networks are called folksonomies. A folksonomy is usually interpreted as a set of user-tag-resource triplets. Existing work for *social search* on folksonomies is mainly on improving search process over social data (tags and users) gathered from social tagging sites [30–32]. In this context, relationships between a user and a tag and also between two tags are of significant importance. Different ranking models proposed in the context of folksonomies include [33–35]. Studies on folksonomies and/or with focus on social tags/bookmarking face the same limitations of user-based tagging. The main issue with user tagging is that results are unreliable and inconsistent due the lack of control and consistency in user tags [2, 36]. Since there is no standard or limitation on the tags chosen by users are, many problems can arise that lower the quality of the results. As discussed in [36], examples of these issues include: synonymy (multiple tags for the same concept), homonymy (same tag used with different meaning), polysemy (same tag with multiple related meanings), and heterogeneity in interpretations and definitions of terms.

6 Conclusion and Future Work

In this paper, we introduced a novel way for personalization of Web search dubbed persocialized search. With persocialized search, we showed how social actions are relevant and useful to improve the quality of the search results. We proposed a model called persocial relevance model to incorporate three levels of social signals into the search process. In level 1, we showed how to utilize users' own social actions to improve the search results. With level 2, we added social data from users' friends to the proposed model. Finally, in level 3 we proposed social expansion to expand the effect of social action to more documents. Using the persocial relevance model, we proposed three ranking approaches to combine the existing textual relevance models with the persocial relevance models. Furthermore, we developed a system called PERSOSE as a prototype search engine capable of performing persocialized search. Employing PERSOSE, we conducted an extensive set of experiments using real documents from Wikipedia and real user and social properties from Facebook. With several set of experiments, we showed how different levels of our persocial model improve the accuracy of search results. We also evaluated the proposed ranking functions and compared them with each other and a baseline approach.

We believe that in this paper, we defined the overall framework needed for the persocialized search. By design and whenever possible, we allowed for different implementations for the proposed methods. This enables an easier customization as well as optimization of *PERSOSE* for different settings and applications. For any given method, finding the best variations/implemantion for a given context is a general and orthogonal research topic that can and should be pursued by experts of that specific context (e.g., optimal user influence or action weight values should be determined based on a given application and by experts on that application.).

Also, there exist many opportunities to improve and extend the proposed framework in several other directions. Here, we briefly mention several directions of future work or easy extension to apply on our existing framework.

Query Categories. One promising direction to extend persocialized search is to study the effects of persocialization on different categories of queries. We have shown that in general, persocialized search improves the accuracy of the search results. As the next step, it is very reasonable to evaluate this improvement based on different query types and see for what type of queries, persocialized search works best and for what types it works the worst.

Intuitively, persocialized search should work very well with queries that explicitly or implicitly asking for opinions and evaluations. For instance, one can guess that queries on restaurants or movies will benefit significantly when social data from people's friends are integrated into the search process. On the other hand, when users know exactly what they want (e.g., navigational queries), persocialized search will probably have no significant effect.

Studying different query types can also help the system adjust the value of α in Eq. 3 (relative weight of textual and persocialized search relevance) automatically and on-the-fly. In the current system, users are in charge of determining the value of α based on their needs. By calculating value of α based on query categories, this process can be done automatically.

Recommendation and Discovery. With certain domains, the persocialized search described in Sect. 3 can be used to recommend items/documents to the users. For instance, for a online video web site/application, persocialized relevance scores can be used for discovery of interesting videos. When many friends of a user have interacted with a particular video, that video may become of an interest to that user. Recommendation based on our persocial relevance model (level 2) can discover and return such videos.

As another example, imagine a music service integrated with a social network. A song recommendation for such services can possibly benefit using our persocial relevance model. Songs can be suggested to a given user, based on what her friends has listened to or liked while considering friends' influence and closeness to the user.

This type of recommendation is very useful when the textual dimension of the documents do not provide much information about the documents (e.g., empty or few textual terms).

Results Interface. In any type of search, searcher usually needs to know why a returned result is relevant to his search. With textual search, this is often done using document (textual) snippets that contain one or more of the query keywords.

It will be an interesting research problem to study different designs that can add a *social snippet* to a qualified search result. For a given persocialized search result, a simple design would be to add the names of (top k) friends of the querying user who have some social actions on the resulting document plus the actual social actions, underneath the textual snippet. This will take only one extra line while providing to the querying user both the social actions and (close) friend names interacted with the returned document.

References

1. Craig AS, Gregory RG (2005) Connections: using context to enhance file search, 20th ACM symposium on operating systems principles. ACM Press, New York, pp 119–132
2. McDonnell M, Ali S (2011) Social search: a taxonomy of, and a user-centred approach to, social web search. Progr: Electron Libr Inf Syst 45.1, pp 6–28
3. Evans B et al (2009) Exploring the cognitive consequences of social search. In: Proceedings of computer human interaction
4. Vuorikari R et al (2009) Ecology of social search for learning resources. Campus-wide Inf Syst 26(4):272–286
5. Amitay E et al (2009) Social search and discovery using a unified approach. In: Proceedings of hyptertext and hypermedia conference
6. Evans B et al (2008) Towards a model of understanding social search. In: SSM
7. Konstas I et al (2009) On social networks and collaborative recommendation. In: SIGIR
8. Freeman LC et al (1979) Centrality in social networks: conceptual clarification. Soc Netw 1(3):215–239
9. Salton G et al (1987) Term weighting approaches in automatic text retrieval. Technical report. Cornell University
10. Robertson SE et al (1994) Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. In: SIGIR
11. Page L et al (1999) The PageRank citation ranking: bringing order to the web
12. Horowitz D et al (2010) The anatomy of a large-scale social search engine. In: WWW
13. Franklin MJ et al (2011) CrowdDB: answering queries with crowdsourcing. In: SIGMOD
14. Yan T, Vikas K, Deepak G (2010) Crowdsearch: exploiting crowds for accurate real-time image search on mobile phones. In: Proceedings of the 8th international conference on mobile systems, applications, and services. ACM
15. Smyth B, Briggs P, Coyle M, O'Mahony MP (2009) Google shared! a case-study in social search. In: User modeling, adaptation and personalization. Springer, pp 283–294
16. Bozzon A, Brambilla M, Ceri S (2012) Answering search queries with crowdsearcher. In: Proceedings of the 21st international conference on World Wide Web. ACM
17. Bozzon A et al (2012) Extending search to crowds: a model-driven approach. Search Comput 7538:207–222
18. Fraternali P et al (2012) CrowdSearch: Crowdsourcing Web search
19. Chirita PA, Nejdl W, Paiu R, Kohlschütter C (2005) Using odp metadata to personalize search. In: SIGIR'05: Proceedings of the 28th annual international ACM SIGIR conference on research and development in information retrieval. ACM, New York, pp 178–185
20. Ferragina P, Gulli A (2005) A personalized search engine based on web-snippet hierarchical clustering. In: WWW'05: special interest tracks and posters of the 14th international conference on World Wide Web. ACM, New York, pp 801–810
21. Chirita P-A, Firan CS, Nejdl W (2007) Personalized query expansion for the web. In: 30th annual international ACM SIGIR conference on research and development in information retrieval (SIGIR 2007). ACM, Amsterdam, pp 7–14

22. Xu S, Bao S, Fei B, Su Z, Yu Y (2008) Exploring folksonomy for personalized search. In: SIGIR'08: Proceedings of the 31st annual international ACM SIGIR conference on research and development in information retrieval. ACM, New York, pp 155–162
23. Noll M, Meinel C (2008) Web search personalization via social bookmarking and tagging. The semantic web, pp 367–380
24. Wang Q, Jin H (2010) Exploring online social activities for adaptive search personalization. In: Proceedings of the 19th ACM international conference on information and knowledge management. ACM
25. Carmel D et al (2009) Personalized social search based on the users social network. In: CIKM
26. Yin P et al (2010) On top-k social web search. In: CIKM
27. Gou L et al (2010) SNDocRank: document ranking based on social networks. In: WWW
28. Gou L et al (2010) SNDocRank: a social network-based video search ranking framework. In: MIR
29. Gou L et al (2010) Social network document ranking. In: JCDL
30. Rawashdeh M et al (2011) Folksonomy-boosted social media search and ranking. In: ICMR
31. Schenkel R et al (2008) Efficient top-k querying over social-tagging networks. In: SIGIR
32. Yahia SA et al (2008) Efficient network aware search in collaborative tagging sites. In: VLDB
33. Gulli A, Cataudella S, Foschini L (2009) Tc-socialrank: ranking the social web. Algorithms Models Web-graph 5427:143–154
34. Hotho A, Ja"schke R, Schmitz C, Stumme G (2006) Information retrieval in folksonomies: Search and ranking. In: Sure Y, Domingue J (eds) The semantic web: research and applications, volume 4011 of LNAI. Springer, Heidelberg, pp 411–426
35. Bao S, Xue G, Wu X, Yu Y, Fei B, Su Z (2007) Optimizing web search using social annotations. In: Proceedings of WWW. ACM, pp 501–510
36. Mizzaro S, Vassena L (2011) A social approach to context-aware retrieval. World Wide Web 14(4):377–405