Personalized Social & Real-Time Collaborative Search

Mukesh Dalal
1533 Rio Grande St., Davis, CA 95616 USA
Mukesh.Dalal@gmail.com

ABSTRACT

This paper presents Adaptive Web Search (AWS), a novel search technique that combines personalized, social, and real-time collaborative search. Preliminary empirical results from a small sample suggest that an AWS prototype built on WAMP platform using Yahoo! Web Search API generates more relevant results and allows faster discovery of information.

Categories and Subject Descriptors:
H.3.3 [**Information Search and Retrieval**]: Search process.

General Terms: Algorithms, Management, Design.

Keywords: Search, Personalized, Social, Collaboration, Real-Time, Web, Metasearch, Pagerank, Context, WWW.

1. INTRODUCTION

A standard search engine such as Google typically returns identical results for the same query, independent of the user or the context. In contrast, a *personalized search engine* [2] would return different results for the same query, depending on the user and the context. For example, if a user has recently been querying about "animals" and then queries about "Jaguar", then automobile results would be ranked lower than animal results. In addition to such short-term context, personalization may also use information such as long-term context, preferences and geographical location of the user, and querying time. While various approaches for personalized web search have been developed and even deployed, their inherent limitations have prevented widespread usage and satisfaction. These limitations include violation of privacy, need for explicit feedback, cumbersome customization such as manual sliders, and delayed and poor personalization.

A *social search engine* extends personalized search by also incorporating the preferences of other users. These users could be explicitly identified as friends (e.g. MySpace), members of the same group (e.g. Yahoo! Groups), or automatically identified as similar or correlated users (e.g. user based collaborative filtering). A typical example of social search is that the results of my book searches should be influenced by the preferences of my trusted friends and members of my book clubs. Currently popular social search tools are either vote-driven (e.g. Digg and Del.icio.us) or require manual entry (e.g. Rollyo, Google custom search engines). While the former require explicit feedback and are easy to game, the latter require cumbersome setup and maintenance. Further, none of them combines personalization with social search.

A *collaborative search engine* further extends social search by allowing real-time feedback among multiple users. For example, I could collaborate in real-time with three remote friends to find a common destination for the next camping trip. The collaboration could be reinforcing (where my results are biased towards their preferences), complementary (biased against), or hybrid. Such real-time collaborative search is not possible with current search engines.

This short paper presents *Adaptive Web Search (AWS)*, which combines personalized, social and collaborative search. A prototype has been implemented on the WAMP (Windows, Apache, MySQL, and PHP) platform leveraging Yahoo! Web Search API. Preliminary empirical results from on a small set of users suggest that AWS generates more relevant results and allows faster discovery of information.

2. PRINCIPLES OF ADAPTIVE SEARCH

Our approach for AWS is motivated by several desirable design principles. A user should be able to perform most searches by interacting directly with just one convenient search engine, in contrast to approaches such as Rollyo and Google custom search engines that require users to explicitly manage multiple search engines and choose one of them for each query. AWS should be fully automated, similar to the currently popular commercial search engines (CPCSEs). AWS should provide relevant results across all realistic query scenarios, in contrast to a vertical search engine limited to a narrow domain, e.g. Retrevo for consumer electronics. The user should not be required to explicitly customize the search engine or provide explicit feedback on search results, in contrast to approaches such as Collarity that require manual setting of a slider between local to global or Eurekster swickis that require users to explicitly promote or demote sites. However, AWS should leverage all explicit contributions made by users, for example, tags and bookmarks. AWS should not rely on private information such as age and gender, and any transfer of information between users should require their explicit consent. AWS should be as efficient and scalable as the CPCSEs, and should similarly minimize spam or noise in its results.

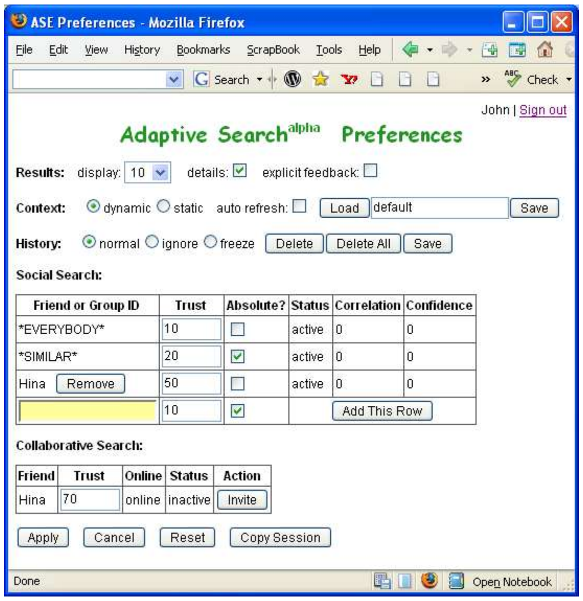
3. PERSONALIZED SEARCH

Personalization in AWS is based on meta-searching [4]: the search query is sent to multiple concurrent *basic search engines* whose results are then aggregated using a *preference vector* over those search engines. AWS ideally requires several orthogonal basic search engines such that their linear combinations cover a large space of custom search engines. The preference vector determines the specific custom search engine to be used.

While the current AWS prototype uses country specific searches allowed by Yahoo's public API, search engines based on topic-sensitive pageranks [1] would work better. Another possibility is to use the various custom search engines, say, built using Rollyo, Google, or MSN Live Search.

The preference vector for a user is obtained from a search profile that's updated after each search. The search profile incorporates information such as short-term and long-term contexts, automatically-identified query goal [3], and geographic location. In contrast to Google personalized search and other similar approaches [5], AWS is able to personalize results even using a single click. For example, after clicking on just one typical UK site, the search for "football" resulted in top ranking of soccer related sites, while standard search continued to rank American football sites at the top.

As shown in the adjoining figure of Adaptive Search preferences, the user may choose to freeze or ignore search and click history. The user may also choose to enable explicit feedback (for each search result) or use a fixed search context (that is, preference vector). The user may also save and load named search contexts, for example, to repeat an earlier search providing same results as before. The user may also save or delete current session's history, or even delete her entire history (including previously saved history). In our experiments, most users used the default settings: no explicit feedback, dynamic context, and normal history.


ASE Preferences - Mozilla Firefox
File Edit View History Bookmarks ScrapBook Tools Help
Search
Check
John | Sign out
Adaptive Search alpha Preferences
Done
Open Notebook

Results: display: 10 details: ✓ explicit feedback: ☐

Context: [● dynamic] ○ static auto refresh: ☐ Load default Save

History: [● normal] ○ ignore ○ freeze Delete Delete All Save

Social Search:

Friend or Group ID	Trust	Absolute?	Status	Correlation	Confidence
EVERYBODY	10	☐	active	0	0
SIMILAR	20	✓	active	0	0
Hina Remove	50	☐	active	0	0
	10	✓	Add This Row		

Collaborative Search:

Friend	Trust	Online	Status	Action
Hina	70	online	inactive	Invite

Apply Cancel Reset Copy Session

4. SOCIAL SEARCH

Social search is enabled by having the preference vector also depend on the saved search profiles of other users. The exact dependence is defined by the trust settings on the preferences page. For example, John's absolute trust value of 50 for Hina means that Hina's saved profile is given half the weight of John's own profile in computing John's preference vector. Note that both John and Hina need to explicitly agree before any such sharing, as indicated by the "active" status. The preference page also displays the correlation and its significance between the two profiles. The trust value can also be set relative (not absolute), where it gets scaled by the correlation. For example, a relative trust of (+50) with correlation (-0.5) will yield an absolute trust of (-25). Negative absolute trust in another user leads to demotion of results preferred by that user – this non-standard treatment of negative trust seems more appropriate in the absence of trust propagation.

Apart from individual users, trust values can also be set for groups. For example, John can set a trust value of 10 in a book club, where the profile for the book club is created by aggregating the profiles of all its users. A group owner can set the weighting used in such an aggregation, which is set by default to be uniform. There are two special groups: *EVERYBODY* includes all users, while *SIMILAR* for a user includes only those users whose profiles have significantly positive correlations with that user.

5. COLLABORATIVE SEARCH

Real-time search collaboration is enabled by making the preference vector to also depend of the current session profiles (saved and unsaved) of other users. As indicated in the figure, you can invite any of your friends (with "active" social status) to collaborate and assign an absolute trust value for that collaboration. Collaboration is active only if your friend explicitly approves your invitation or independently invites you for collaboration. Sharing of profiles can be fine-tuned by appropriately setting the two trust values – social trust for long-term context and collaborative trust for short-term context.

AWS allows a user to spawn multiple concurrent sessions, where each session may concurrently use multiple search windows. Each session has its own history and collaborations, for example, while John may collaborate in one session with Hina and Michael in searching for a camping site, he may concurrently collaborate in another session with Mary in searching for a wedding dress. "Copy Session" spawns another session with exactly the same history and collaborations.

6. RESULTS & CONCLUSIONS

The current prototype of AWS was tested using about 150 queries. In this small sample, 32% more relevant results were found on average within the top 10 results as compared to Yahoo!'s standard search results. With social search, using only default settings, the relevance increased by about 45%. Collaborative search allowed discovering information in about 43% fewer queries than independent searches. AWS also seem to satisfy the design principles listed earlier. In particular, it added only 2% overhead to Yahoo!'s search time.

AWS seems to be a promising approach for personalized, social and collaborative web search. However, its utility needs to be formally validated using carefully designed larger experiments. Successful deployment may lead to a truly viral and sticky search engine, a challenging goal even for Google.

7. REFERENCES

[1] Haveliwala, T.H. Topic-Sensitive Pagerank: A Context-Sensitive Ranking Algorithm for Web Search. IEEE Transactions on KDE, 15 (4). 784-796, 2003.

[2] Keenoy, K. and Levene, M. Personalisation of Web Search. Springer-Verlag Lecture Notes in Computer Science, 3169. 201-228, 2005.

[3] Lee, U., Liu, Z. and Cho, J., Automatic Identification of User Goals in Web Search. Proceedings of the 14th international conference on World Wide Web, (2005), 391-400.

[4] Meng, W., Yu, C. and Liu, K.L. Building Efficient and Effective Metasearch Engines. ACM Computing Surveys (CSUR), 34 (1). 48-89, 2002.

[5] Qiu, F. and Cho, J., Automatic Identification of User Interest for Personalized Search. Proceedings of the 15th international conference on World Wide Web, (2006), 727-736.