Utilizing User-input Contextual Terms for Query Disambiguation

Byron J. Gao
Texas State University
bgao@txstate.edu

David C. Anastasiu
Texas State University
da1143@txstate.edu

Xing Jiang
Nanyang Technological University
jian0008@ntu.edu.sg

Abstract

Precision-oriented search results such as those typically returned by the major search engines are vulnerable to issues of polysemy. When the same term refers to different things, the dominant sense is preferred in the rankings of search results. In this paper, we propose a novel technique in the context of web search that utilizes contextual terms provided by users for query disambiguation, making it possible to prefer other senses without altering the original query.

1 Introduction

World Wide Web and search engines have become an indispensable part of everyone's everyday life. While web search has come a long way over the past 10 years, it still has a long way to go to respond to the ever-increasing size of the web and needs of web surfers. Today, web search is under intensive and active research, drawing unparalleled attention from both industry and academia.

Need of disambiguation. One of the major challenges in web search lies in unsatisfactory relevance of results caused by ambiguity. Query terms are inherently ambiguous due to polysemy, and most queries are short containing 1 to 3 terms only (Jansen et al., 2000). Thus queries are in general prone to ambiguity of user intent or information needs, resulting in retrieval of enormous irrelevant pages. As the web increases in size at an increasing rate, ambiguity becomes ubiquitous and users are in increasing need of effective means of disambiguation. The ambiguity issue and its consequences are demonstrated in Example 1.

Example 1 *There are 17 entries in Wikipedia for different renown individuals under the same name of "Jim Gray", including a computer scientist, a sportscaster, a zoologist, a politician, a film director, a cricketer, and so on. Suppose we intend to find information about* `Jim Gray`*, the Turing award winner, we can issue a query of "Jim Gray" in Yahoo!For this extremely famous name in computer science, only 3 are relevant in the top 10 results. They are his Wikipedia entry, homepage at Microsoft Research, and DBLP entry.*

Straightforward approach. One intuitive way of disambiguation would be to apply available domain knowledge and refine the query by adding some confining contextual terms. This would generally improve precision. However, there are several inevitable problems in this approach. First, the improvement on precision is at the sacrifice of recall. For example, many `Jim Gray` pages may not contain the added contextual terms and are thus excluded from the search results.

Second, the query is altered, leading to unfavorable ranking of results. Term proximity matters significantly in ranking (Manning et al., 2008). Some good pages w.r.t. the original query may be ranked low in the new search results because of worsened term proximity and relevance w.r.t. the new query. Thus, with this straightforward approach only limited success can be expected at best, as demonstrated in Example 2.

Example 2 *Suppose we know that* `Jim Gray` *is a computer scientist, we can issue a query of "Jim Gray computer". All the top 10 results are about* `Jim Gray` *and relevant. However, many of them are trivial pages, failing to include 2 of the 3 most important ones. His DBLP entry appears as the*

Coling 2010: Poster Volume, pages 329–337,
Beijing, August 2010

27th result, and his homepage at Microsoft Research appears as the 51st result.

This limited success is achieved by using a carefully selected contextual term. "Computer" is a very general term appearing on most of the `Jim Gray` pages. Also, there are no other competitively known computer people with the same name. Most other contextual terms would perform much worse. Thus a third problem of this straightforward query refinement approach is that only few contextual terms, which may not be available to users, would possibly achieve the limited success. Often, much of our domain knowledge would cause more damage than repair and is practically unusable, as demonstrated in Example 3.

Example 3 *Suppose we know that* `Jim Gray` *has David DeWitt as a close friend and colleague, we can issue a query of "Jim Gray David DeWitt". Again, all the top 10 results are about* `Jim Gray` *and relevant. However, the theme of the query is almost completely altered. Evidently, the 1st result "Database Pioneer Joins Microsoft to Start New Database Research Lab", among many others, talks about David DeWitt. It is relevant to* `Jim Gray` *only because the lab is named "Jim Gray Systems Lab" in honor of him.*

The `Bobo` approach. Can we freely apply our domain knowledge to effectively disambiguate search intent and improve relevance of results without altering the original query? For this purpose, we propose and implement `Bobo`.[1]

For conceptual clarity, the `Bobo` interface features two boxes. Besides a regular query box, an additional box is used to take *contextual terms* from users that capture helpful domain knowledge. Contextual terms are used for disambiguation purposes. They do not alter the original query defined by query terms. Particularly, unlike in the straightforward approach, positive contextual terms are not required to be included in search results and negative contextual terms are not required to be excluded from search results. Contextual terms help estimate relevance of search results, routing them towards a user intended domain, filtering out those not-in-domain, or irrelevant, results.

`Bobo` works in two rounds. In round I, a query is issued using by default the combination of query terms and contextual terms, or just the contextual terms if the query returns too few results. Then from the results, some top-ranked high-quality pages are (automatically) selected as *seeds*. In round II, a query is issued using the query terms. Then the results are compared with the seeds and their similarities are computed. The similarity values reflect the degree of relevance of search results to the user intent, based on which the results are re-ranked.

Example 4 reports the `Bobo` experiment using the same contextual terms as in Example 3.

Example 4 *As in Example 3, suppose we know* `Jim Gray` *has David DeWitt as a colleague. Then with* `Bobo`*, we can enter "Jim Gray" in the query box and "David DeWitt" in the auxiliary box. As a result with default preferences, all the top 10 results are relevant including all the top 3 important* `Jim Gray` *pages. From the top 10, only 1 page, the DBLP entry, contains "David DeWitt" as they coauthored papers. The theme of the query is not altered whereas in Example 3, all the top 10 results contain "David DeWitt".*

In Example 4, the selected seeds are relevant to `Jim Gray`. Observe that seeds can be useful if they are relevant to the user-intended domain, not only the user-intended query. `Bobo` works effectively with such seeds and thus can utilize a much expanded range of domain knowledge. Helpful contextual terms do not even need to co-occur with query terms on any page. They only need to occur, possibly separately, on some pages of the same domain, as demonstrated in Example 5.

Example 5 *Using the criteria of being in the same community as* `Jim Gray` *but co-occuring on no web pages, we randomly chose a student name, Flavia Moser. In* `Bobo`*, we entered "Jim Gray" in the query box, "Flavia Moser" in the auxiliary box, and used only the contextual terms for the round I query. As a result, 11 of the top 12 results were relevant including all the top 3 important* `Jim Gray` *pages. Of course, none of the returned pages contains "Flavia Moser".*

[1] `Bobo` has been implemented using Yahoo! web search API and maintained at http://dmlab.cs.txstate.edu/bobo/.

2 Related Work

Disambiguating search intent, capturing information needs and improving search performance have been a fundamental research objective in information retrieval and studied from different perspectives. Voorhees (1993) shows that disambiguation cannot be easily resolved using thesauruses. The filtering problem (Baeza-Yates and Ribeiro-Neto, 1999; Schapire et al., 1998) views disambiguation as a binary text classification task assigning documents into one of the two categories, relevant and irrelevant. The routing problem (Schutze et al., 1995; Singhal et al., 1997) differs from text classification in that search results need to be ranked instead of just classified (Gkanogiannis and Kalamboukis, 2008).

Contextual search (Lawrence, 2000; Finkelstein et al., 2002; Kraft et al., 2006), personalized search (Haveliwala, 2002; Teevan et al., 2005; Zhu et al., 2008), and implicit relevance feedback (Kelly and Teevan, 2003; Joachims et al., 2005; White et al., 2004) generally utilize long-term search history to build user profiles. These profiles are used on a regular basis to guide *many* queries. Such approaches entail little extra user involvement in search, but need to manage profiles, face the privacy issue, and swallow the inflexibility in context switch.

Explicit and pseudo relevance feedback (RF) techniques (Ruthven and Lalmas, 2003; Baeza-Yates and Ribeiro-Neto, 1999; Manning et al., 2008) are more related to Bobo in the sense that they do not build long-term profiles. Instead, they construct a one-time search context that are used only once to guide a *single* query each time. Such approaches enjoy the flexibility to switch spontaneously from one domain to another in response to different information needs.

RF is regarded as the most popular query reformation strategy (Baeza-Yates and Ribeiro-Neto, 1999). It iterates in multiple rounds, typically two, to modify a query step by step. Explicit RF asks explicit feedback from users, whereas pseudo (or blind) RF assumes relevance of top-ranked results. The problem of explicit RF is that it requires too much user involvement. Users are often reluctant to provide explicit feedback, or do not wish to prolong the search interaction. Web search engines of today do not provide this facility. Excite.com initially included but dropped it due to the lack of use (Manning et al., 2008).

Pseudo RF, first suggested by Croft and Harper (1979) and since widely investigated, automates the manual part of RF, so that users get improved search performance without extended interactions. Psuedo RF has been found to improve performance in the TREC ad hoc task and Cornell SMART system at TREC 4 (Buckley et al., 1995). Unfortunately, pseudo RF suffers from a major flaw, the so-called *query drift* problem. Query drift occurs when the feedback documents contain few or no relevant ones. In this case, search results will be routed farther away from the search intent, resulting in even worse performance. Different approaches (Mitra et al., 1998; Yu et al., 2003; Lee et al., 2008)have been proposed to alleviate query drift but with little success. Some queries will be improved, others will be harmed (Ruthven and Lalmas, 2003).

Similarly to RF, Bobo works in two rounds. Similarly to pseudo RF, it makes use of top-ranked round I results. However, Bobo and RF differ fundamentally in various aspects.

Firstly, Bobo is not a query reformation technique as RF. In RF, the *automatically generated* additional terms become part of the reformed query to be issued in round II, while in Bobo, the *user-input* contextual terms are *not* used in round II. The terms generated by RF may work well as contextual terms for Bobo but not the other way around. In general, effective contextual terms form a much larger set.

In query reformation, it is often hard to understand why a particular document was retrieved after applying the technique (Manning et al., 2008). In Bobo, the original query is kept intact and only the ranking of search results is changed.

Secondly, in RF, only query terms are used in round I queries. In Bobo, by default the combination of query terms and contextual terms, both entered by users, is used, leading to much more relevant seeds that are comparable to explicit RF. In this sense, Bobo provides a novel and effective remedy for query drift.

Beyond that, Bobo can use contextual terms

only to obtain seeds that are relevant to the user-intended domain and not necessarily the user-intended query, leading to effective utilization of a largely expanded range of domain knowledge.

Thirdly, RF can have practical problems. The typically long queries (usually more than 20 terms) generated by RF techniques are inefficient for IR systems, resulting in high computing cost and long response time (Manning et al., 2008). In Bobo, however, both query terms (1 to 3) and contextual terms (1 to 2) are short. A round I query combining the two would typically contain 2 to 5 terms only.

3 Overview

Bobo uses the vector space model, where both documents and queries are represented as vectors in a discretized vector space. Documents used in similarity comparison can be in the form of either full pages or snippets. Documents are preprocessed and transformed into vectors based on a chosen term weighting scheme, e.g., TF-IDF.

The architecture of Bobo is shown in Figure 1. Without input of contextual terms, Bobo works exactly like a mainstream search engine and the dashed modules will not be executed. Input of contextual terms is optional in need of disambiguation of user intent. Domain knowledge, directly or indirectly associated with the query, can be used as "pilot light" to guide the search towards a user-intended domain.

With input of contextual terms, Bobo works in two rounds. In round I, a query is issued using by default the combination of query terms and contextual terms, or just the contextual terms if they are unlikely to co-occur much with the query terms. Then from the results, the top k documents (full pages or snippets) satisfying certain quality conditions, e.g., number of terms contained in each seed, are selected as seeds. Optionally, seeds can be *cleaned* by removing the contained query terms to reduce background noise of individual seeds, or *purified* by removing possibly irrelevant seeds to improve overall concentration. Contextual terms themselves can be used as an *elf seed*, which is a special document allowing negative terms, functioning as an explicit feedback.

In round II, a query is issued using the query

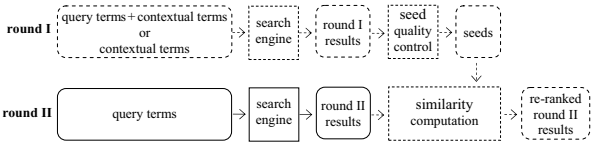

Figure 1: Architecture of Bobo.

terms. Then, each returned result (full page or snippet) is compared to the seeds to compute a similarity using a designated similarity measure, Jaccard coefficient or Cosine coefficient. In the computation, seeds can be *combined* to form a prototype as in Rocchio, or not combined using none generalization as in instance-based lazy learning to better capture locality and handle polymorphic domains. Based on the assumption that seeds are highly relevant, the similarity values estimate the closeness of search results to the user intent, based on which the results are re-ranked.

Bobo was implemented using Yahoo! web search API. For each query, Bobo retrieves 30 HTML pages from the API. If snippets are used for seeding and comparison, the response time of Bobo is sufficiently fast. If full pages are used, page downloading and preprocessing are prohibitively time-consuming. However, the goal of Bobo is to illustrate the promise of the novel disambiguation approach. If Bobo were implemented at the server (search engine) side, response time would not be an issue.

4 Principles and Preferences

In this section, we introduce in detail the design principles and preferences of Bobo regarding the various key issues. We also discuss possible improvements in these aspects.

4.1 Use of Contextual Terms

How to use contextual terms has a fundamental impact on the behavior and performance of Bobo.

In round I. By default, the combination of query terms and contextual terms are used in round I queries. This produces seeds that are relevant to the user-intended query. For instance, in Example 4, the seeds are relevant to Jim Gray. This usage of contextual terms actually provides a novel and effective remedy for query drift, thanks to the input of domain knowledge.

Generally, a large portion of domain knowledge cannot be utilized in a straightforward manner, due to the fact that contextual terms may co-occur with query terms in very few or none web pages. However, as shown in Example 5, Bobo allows using only contextual terms for round I queries, enabling utilization of indirectly associated domain knowledge.

As elf seed. Contextual terms can be considered forming a pseudo document, which can be optionally used as a seed. We call such a seed *elf seed* as it is actually a piece of explicit relevance feedback. Unlike normal seeds, an elf seed may contain positive as well as negative terms, providing a way of collecting positive as well as negative explicit feedback.

Discussion. The option of combing query terms and contextual terms in round I queries can be automated. The idea is to combine the terms first, then test the k^{th} result to see whether it contains all the terms. If not, only the contextual terms should be used in the query.

4.2 Quality of Seeds

As in pseudo relevance feedback, quality of seeds plays an critical role in search performance. The difference is that in Bobo, input of contextual terms is largely responsible for the much improved relevance of seeds. To provide further quality control, Bobo accepts several user-input thresholds, e.g., number of seeds and number of terms contained in each seed. Beyond that, Bobo also provides the following options.

Removing query terms. By default, Bobo uses a combination of contextual terms and query terms in round I queries. Thus usually all the seeds contain the query terms. Round II results contain the query terms as well. Then, in similarity computation against the seeds, those query terms contribute almost equally to each round II result. This amount of contribution then becomes background noise, reducing the sensitivity in differentiating round II results.

By default, Bobo removes query terms from seeds. Although a simple step, this option significantly improves performance in our experiments.

Purifying seeds. Different approaches have been proposed to alleviate query drift by improving relevance of pseudo feedback, but with limited success (Ruthven and Lalmas, 2003). In Bobo, due to the input of domain knowledge, we can well assume that the majority of seeds are relevant, based on which, we can design simple mechanisms to purify seeds. Briefly, we first calculate the centroid of seeds. Then, we compute the similarity of each seed against the centroid, and remove those outlying seeds with poor similarities.

Discussion. Current search engines take into account link-based popularity scores in ranking search results. In Bobo, round I search results are not used to directly meet information needs of users. They are never browsed by users. Thus, different search engines with alternative ranking schemes may be used to better fulfill the purpose of round I queries.

Round I queries do not need to be issued to the same region as round II queries either. Working in a more quality region may help avoid spamming and retrieve better candidates for seed selection.

4.3 Term Weighting

Bobo uses two term weighting schemes. The default one is the conventional TF-IDF. The other scheme, TF-IDF-TAI, uses term association to favor terms that show high co-occurrence with query terms. It is tailored to Bobo, where documents are not compared in isolation, but being "watched" by a query. While TF-IDF can be considered global weighting independent of queries, TF-IDF-TAI can be considered local weighting. Here we omit the details due to the page limit.

IDF estimation. To estimate the IDF values of terms, Bobo used 664, 103 documents in the Ad-hoc track of TREC dataset.[2] These documents can produce a reasonable approximation as they cover various domains such as newspapers, U.S. patents, financial reports, congressional records, federal registers, and computer related contents.

In particular, for a term A, $IDF(A) = \log_2 \frac{n}{DF(A)}$, where $DF(A)$ is the document frequency of A in the TREC data set and n = 664,103.

[2] http://trec.nist.gov/data/docs_eng.html.

4.4 Similarity Computation

By computing similarities between round II results and seeds, Bobo estimates how close different results are to the search intent.

Document type. Seeds can either be in type of snippets (including titles) or full pages. So it is with round II results. White et al. (2007) reported that snippets performed even better than full texts for the task of pseudo RF. In our experiments, snippets also performed comparably to full pages. Thus, Bobo uses "snippet" as the default option for fast response time.

Similarity measure. Bobo uses two standard similarity measures, Cosine coefficient (default) and Jaccard coefficient. Both performed very well in our experiments, with the default option slightly better.

Prototype-based similarity. Bobo implements two types of similarity computation methods, prototype-based or instance-based, with the latter as the default option.

The prototype-based method is actually a form of the well-known Rocchio algorithm (Rocchio, 1971; Salton and Buckley, 1997), which is efficient but would perform poorly in the presence of polymorphic domains. In this method, the seeds are combined and the centroid of seeds is used in similarity computation. Given a set S of seeds, the centroid $\vec{u}$ is calculated as $\vec{u} = \frac{1}{|S|}\sum_{s\in S}\vec{s}$, where $\vec{s}$ is the vector space representation of seed $s \in S$.

Recall that the original Rocchio algorithm for query reformation is defined as follows,

$$\vec{q_e} = \alpha\vec{q} + \beta\frac{1}{|D_r|}\sum_{\vec{d_j}\in D_r}\vec{d_j} - \gamma\frac{1}{|D_{ir}|}\sum_{\vec{d_j}\in D_{ir}}\vec{d_j}$$

where q is the original query vector, q_e is the modified query vector, and D_r and D_{ir} represent the sets of known relevant and irrelevant document vectors respectively. α, β, and γ are empirically-chosen tuning parameters.

If we assign $\alpha = 0$ and $\gamma = 0$, the Rocchio formula agrees with our definition of centroid of seeds. We assign $\alpha = 0$ because Bobo does not target query reformation. We assign $\gamma = 0$ not because of the lack of negative feedback, which is not hard to identify from low-ranked round I search results. The reason is that even in explicit RF, there is no evidence that negative feedback improves performance (Schutze et al., 1995).

Figure 2: A Polymorphic Domain.

Instance-based similarity. Rocchio is simple and efficient. However, it over-generalizes training data and is inaccurate in the presence of polymorphic, or disjunctive, domains. In Figure 2, the 10 seeds labeled by "+" are split into two separate and rather distant sub-domains. The centroid of seeds labeled by "⊕" is not local to any sub-domain. Search result 1 is close to the centroid whereas result 2 is not. Rocchio would give high relevance score to result 1 and poor score to result 2. However, result 2 actually belongs to one of the two sub-domains whereas result 1 does not.

To handle polymorphic domains and capture locality, Bobo uses an instance-based approach, where the similarity of a document against each individual seed is computed, weighted, and aggregated. Let $sim(d, S)$ denote the similarity between a document d and a set S of seeds, then,

$$sim(d, S) = \sum_{s\in S} sim(d, s) \times sim(d, s)$$

Using this approach, result 2 will receive much higher relevance score than result 1 in Figure 2.

Note that, this approach resembles instance-based lazy learning such as k-nearest neighbor classification. Lazy learning generally has superior performance but would suffer from poor classification efficiency. This, however, is not a critical issue in our application because we do not have many seeds. The default number of seeds in Bobo is set to 10.

Discussion. While Bobo adopts rather standard approaches, we are aware of the many other approaches proposed in the literature for pairwise web page similarity computation. An interesting direction to investigate would be a link-based or hybrid approach. For example, Vassilvitskii and Brill (2006) uses web-graph distance for relevance feedback in web search.

5 Empirical Evaluation

We evaluated Bobo in comparison with regular Yahoo! search with and without using contextual terms. Results returned from Yahoo! may vary with time. This, however, will not change the general trends revealed by our empirical study. From these trends we conclude that, Bobo is a simple yet effective paradigm for query intent disambiguation without altering the original query and with maximized utilization of domain knowledge.

5.1 Experiment Setting and Methodology

Parameter setting. To emphasize the Bobo idea, unless otherwise specified, we used default options in the experiments that implement conventional approaches, e.g., TF-IDF for term weighting and Cosine coefficient for similarity computation. By default, number of seeds was set to 10 with each seed having at least 10 terms. The number of layers was set such that round II results were re-ranked in decreasing order of similarity. Cleaning seeds was set to yes. Purifying seeds, elf seed and weighting seeds were set to no.

Dataset. Finding information about people is one of the most common search activities. Around 30% of web queries include person names (Artiles et al., 2005). Person names, however, are highly ambiguous, e.g., only 90,000 different names are shared by 100 million people according to the U.S. Census Bureau (Guha and Garg, 2004).

To test the disambiguation effectiveness of Bobo, we constructed 60 ambiguous name queries and 180 test cases from the Wikipedia disambiguation pages.[3]

In Wikipedia, articles about two or more different topics could have the same natural page title. Disambiguation pages are then used to solve the conflicts. From the various categories, we used the human name category, containing disambiguation pages for multiple people of the same name. For each name, the disambiguation page lists all the different people together with their brief introductions. For example, an Alan Jackson is introduced as "born 1958, American country music singer and songwriter".

[3] en.wikipedia.org/wiki/Category:Disambiguation_pages.

Person names were chosen from the most common English first and last names for the year 2000 published on Wikipedia. The first 10 male and first 10 female given names were combined with the first 10 most common last names to make a list of 200 possible names. From this list, names were chosen based on the following criteria. For each name, there are at least 2 distinct people with the same name, each having at least 3 relevant pages in the returned 30 results.

In total 60 names were chosen as ambiguous queries. For each query, the actual information need was predetermined in a random manner. Then, for this predetermined person, 3 contextual terms were selected from her brief introduction, or her Wikipedia page in case the introduction was too short. For example, for the above Alan Jackson example, "music", "singer", or "songwriter" can be selected as contextual terms. Contextual terms were used one at a time, thus there are 3 test cases for each ambiguous query.

The identification of relevance of search results was done manually. For each query, let R_{30} be the set of relevant pages w.r.t. the information need contained in the 30 retrieved results. R_{30} can be considered containing the most important relevant pages for the original query.

Comparison partners and evaluation measures. To compare with Bobo, two types of regular search methods were used. The *Yahoo!* method uses the original query and performs the simplest Yahoo! web search, returning the same set of results as Bobo but without re-ranking.

To demonstrate the relevance improvement of Bobo over the *Yahoo!* method, we used a couple of standard ranking-aware evaluation measures, which were 11-point precision-recall graph, precision at k graph, Mean Average Precision (MAP) and R-precision.

The *Yahoo!-refined* method is the straightforward query refinement approach we previously discussed. It refines the original query by adding some contextual terms. The refined query alters the original query, leading to unfavorable ranking of results and failing to include many important relevant pages, i.e., R_{30} pages, in the top results.

To demonstrate this point, we used the recall at k evaluation measure, which measures the frac-

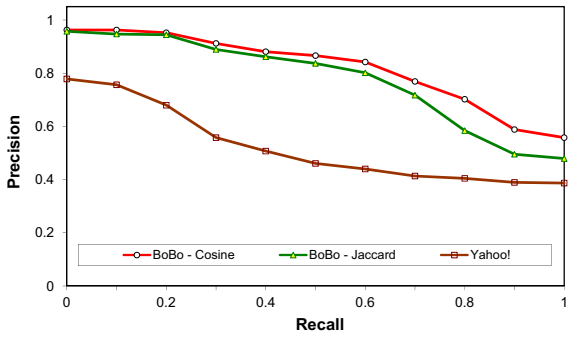

Figure 3: Bobo vs. Yahoo! on Averaged 11-point Precision-Recall.

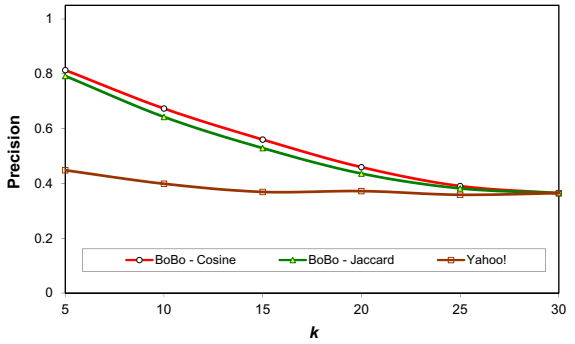

Figure 4: Bobo vs. Yahoo! on Averaged Precision at k.

tion of relevant pages (here, ones in R_{30}) contained in the top k results.

In the entire empirical study, the *Yahoo!* results were averaged over 60 queries, whereas all other results were averaged over 180 test cases.

5.2 Evaluation Results

In Figures 3, 4 and 5, Bobo results using both Cosine similarity and Jaccard coefficient are shown. The two performed similarly, with the former (default) slightly better.

Bobo vs. *Yahoo!*. The 11-point precision-recall graphs and precision at k graphs are presented in Figure 3 and Figure 4 respectively.

Web search users would typically browse a few top-ranked results. From Figure 4 we can see that for k =15, 10 and 5, the precision improvement of Bobo over *Yahoo!* is roughly 20% – 40%.

In addition, the MAP and R-precision values for Bobo are 0.812 and 0.740 respectively, whereas they are 0.479 and 0.405 for *Yahoo!* respectively. The improvement of Bobo over *Yahoo!* is about 33% for both measures.

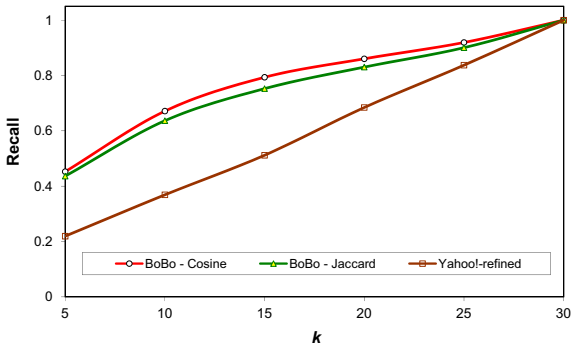

Figure 5: Bobo vs. Yahoo!-refined on Averaged Recall at k.

Bobo vs. *Yahoo!-refined*. The recall at k graphs are presented in Figure 5. From the figure we can see that for $k = 15$, $k = 10$ and $k = 5$, the recall (of important R_{30} pages) improvement of Bobo over *Yahoo!* is roughly 30%.

The results demonstrated that, although the straightforward query refinement approach can effectively improve relevance, it fails to rank those important relevant pages high, as it alters the original query and changes the query themes. Bobo, on the contrary, overcomes this problem by using the contextual terms "in the backstage", effectively improving relevance while keeping the original query intact.

Due to the page limit, here we omit other series of experiments that evaluated the flexibility of Bobo in choosing effective contextual terms and how the varied user preferences affect its performance. A user study was also conducted to test the usability and performance of Bobo.

6 Conclusions

As the web increases in size at an increasing rate, ambiguity becomes ubiquitous. In this paper, we introduced a novel Bobo approach to achieve simple yet effective search intent disambiguation without altering the original query and with maximized domain knowledge utilization.

Although we introduce Bobo in the context of web search, the idea can be applied to the settings of traditional archival information retrieval or multimedia information retrieval.

References

Artiles, Javier, Julio Gonzalo, and Felisa Verdejo. 2005. A testbed for people searching strategies in the WWW. In *SIGIR*.

Baeza-Yates, R. and B. Ribeiro-Neto. 1999. *Modern Information Retrieval*. Addison-Wesley.

Buckley, Chris, Singhal Amit, , and Mitra Mandar. 1995. New retrieval approaches using smart: Trec 4. In *TREC*.

Croft, W. and D. Harper. 1979. Using probabilistic models of information retrieval without relevance information. *Journal of Documentation*, 35(4):285–295.

Finkelstein, Lev, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: the concept revisited. *ACM Transactions on Information Systems*, 20(1):116–131.

Gkanogiannis, Anestis and Theodore Kalamboukis. 2008. An algorithm for text categorization. In *SIGIR*.

Guha, R. V. and A. Garg. 2004. Disambiguating people in search. In *WWW*.

Haveliwala, Taher H. 2002. Topic-sensitive pagerank. In *WWW*.

Jansen, B. J., A. Spink, , and T. Saracevic. 2000. Real life, real users and real needs: A study and analysis of users queries on the web. *Information Processing and Management*, 36(2):207–227.

Joachims, Thorsten, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay. 2005. Accurately interpreting clickthrough data as implicit feedback. In *SIGIR*.

Kelly, Diane and Jaime Teevan. 2003. Implicit feedback for inferring user preference: a bibliography. *SIGIR Forum*, 37(2):18–28.

Kraft, Reiner, Chi Chao Chang, Farzin Maghoul, and Ravi Kumar. 2006. Searching with context. In *WWW*.

Lawrence, Steve. 2000. Context in web search. *IEEE Data Engineering Bulletin*, 23(3):25–32.

Lee, Kyung Soon, W. Bruce Croft, and James Allan. 2008. A cluster-based resampling method for pseudo-relevance feedback. In *SIGIR*.

Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schutze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

Mitra, M., A. Singhal, and C. Buckley. 1998. Improving automatic query expansion. In *SIGIR*.

Rocchio, J. 1971. *Relevance Feedback in Information Retrieval.* In The SMART Retrieval System – Experiments in Automatic Document Processing. Prentice-Hall.

Ruthven, Ian and Mounia Lalmas. 2003. A survey on the use of relevance feedback for information access systems. *Knowledge Engineering Review*, 18(1).

Salton, Gerard and Chris Buckley. 1997. *Improving retrieval performance by relevance feedback*. Morgan Kaufmann.

Schapire, Robert E., Yoram Singer, and Amit Singhal. 1998. Boosting and rocchio applied to text filtering. In *SIGIR*.

Schutze, Hinrich, David A. Hull, and Jan O. Pedersen. 1995. A comparison of classifiers and document representations for the routing problem. In *SIGIR*.

Singhal, Amit, Mandar Mitra, and Chris Buckley. 1997. Learning routing queries in a query zone. In *SIGIR*.

Teevan, Jaime, Susan T. Dumais, and Eric Horvitz. 2005. Personalizing search via automated analysis of interests and activities. In *SIGIR*.

Vassilvitskii, Sergei and Eric Brill. 2006. Using webgraph for relevance feedback in web search. In *SIGIR*.

Voorhees, Ellen M. 1993. Using wordnet to disambiguate word senses for text retrieval. In *SIGIR*.

White, Ryen W., Joemon M. Jose, C. J. Van Rijsbergen, and Ian Ruthven. 2004. A simulated study of implicit feedback models. In *ECIR*.

White, Ryen W., Charles L.A. Clarke, and Silviu Cucerzan. 2007. Comparing query logs and pseudo-relevance feedback for web search query refinement. In *SIGIR*.

Yu, Shipeng, Deng Cai, Ji-Rong Wen, and Wei-Ying Ma. 2003. Improving pseudo-relevance feedback in web information retrieval using web page segmentation. In *WWW*.

Zhu, Yangbo, Jamie Callan, and Jaime Carbonell. 2008. The impact of history length on personalized search. In *SIGIR*.