# Towards Intelligent Search Assistance for Inquiry-Based Learning

**Weijian Xuan**
MHRI
University of Michigan
Ann Arbor, MI 48109
wxuan@umich.edu

**Meilan Zhang**
School of Education
University of Michigan
Ann Arbor, MI 48109
meilanz@umich.edu

## Abstract

In Online Inquiry-Based Learning (OIBL) learners search for information to answer driving questions. While learners conduct sequential related searches, the search engines interpret each query in isolation, and thus are unable to utilize task context. Consequently, learners usually get less relevant search results. We are developing a NLP-based search agent to bridge the gap between learners and search engines. Our algorithms utilize contextual features to provide user with search term suggestions and results re-ranking. Our pilot study indicates that our method can effectively enhance the quality of OIBL.

## 1 Introduction

Major science education standards call on students to engage in Online Inquiry-Based Learning where they pose scientific Driving Questions (DQ), plan their search, collect and analyze online information, and synthesize their findings into an argument. In collaboration with National Science Digital Library (NSDL), we are developing an integrated Online Inquiry-Based Learning Environment (OIBLE), called IdeaKeeper (Quintana and Zhang, 2004), to help learners fulfill the promise of OIBL. IdeaKeeper is among the first reported OIBLE that integrates various online search engines with support for inquiry planning, information search, analysis and synthesis.

Our observation reveals that searching is one of the bottlenecks impeding students' learning experience. Students demonstrate various problems in search. First, they repeatedly search for very similar keywords on search engines. Second, they are usually unable to develop effective search terms. Many search keywords students generate are either too broad or too narrow. Although learners have specific search purposes, many times they are unable to express the purposes in keyword-based queries. In fact, by analyzing the search logs, we found that the average query length is only about 2 words. In such typical cases in OIBL, informative contexts are not presented in queries, and thus the requests become ambiguous. As a result, the search engines may not interpret the query as the learners intended to. Therefore, the results are usually not satisfactory. Given the self-regulated nature of OIBL and limited self-control skills of K-12 students, the problem is even more serious, as students may shift their focus off the task if they constantly fail to find relevant information for their DQ.

## 2 Related Work

In Information Retrieval field, many algorithms based on relevance feedback are proposed (Buckley, et al., 1994; Salton and Buckley, 1990). However, current general web search engines are still unable to interactively improve research results. In NLP domain, there are considerable efforts on Question Answering systems that attempt to answer a question by returning concise facts. While some QA systems are promising (Harabagiu, et al., 2000; Ravichandran and Hovy, 2002), they can only handle factual questions as in TREC (Voorhees, 2001), and the context for the whole task is largely not considered. There are proposals on using context in search. Huang et al (2001) proposed a term suggestion method for interactive web search. More existing systems that utilize contextual information in search are reviewed by Lawrence (2000). However, one problem is that "context" is defined differently in each



*Proceedings of the 2nd Workshop on Building Educational Applications Using NLP,*
pages 25–28, Ann Arbor, June 2005. 

study. Few attempts target at inquiry-based learning, which has some unique features, e.g., DQ/SQ.

We are developing an OnLine Inquiry Search Assistance (OLISA). OLISA applies Natural Language Processing (NLP) and Information Retrieval (IR) techniques to provide students query term suggestions and re-rank results returned from search engines by the relevance to the current query as well as to the DQ. OLISA is not a built-in component of IdeaKeeper, but can be very easily plugged into IdeaKeeper or other OIBL systems as a value-added search agent. The main advantage of OLISA is that it utilizes the context of the whole learning task. Our pilot study demonstrated that it is a simple and effective initiative toward automatically improving the quality of web search in OIBLE.

## 3 Method

### 3.1 Utilizing Learning Context

OLISA acquires search context by parsing OIBL logs and by monitoring search history. For example, in the planning phase of a learning task, IdeaKeeper asks students to input DQ, Sub-Questions (SQs), potential keywords, and to answer some questions such as "what do I know", "what do I want to know", etc.

The context information is represented as bag-of-words feature vectors. To calculate the vectors, we first remove common terms. We compiled a corpus of 30 million words from 6700 full-length documents collected from diverse resources. Word frequencies are calculated for 168K unique words in the corpus. A word is considered common if it is in the 1000 most frequent word list. Remaining words are stemmed using Porter's algorithm (Porter, 1980).

All contextual information are combined to form a main feature vector $(W_1^{(c)}, W_2^{(c)}, \cdots, W_n^{(c)})$, where $W_i^{(c)}$ is the weight of the *ith* term in combined context. It's defined by product of term frequency (*tf)* and inverse document frequency (*idf*).

Comparing with traditional *tf* measure, we do not assign a uniform weight to all words in context. Rather, we consider DQ/SQ and the current query more important than the rest of context. We define their *tf* differently from other context.

$$tf_i^{(dq)} = (1 + \ln(\#wordInContext / \#wordInDQ)) * tf_i^{(dq)} \quad (1)$$

The $tf_i^{(sq)}$ is calculated similarly. For the term frequency of current query $tf_i^{(q)}$, we assign it a larger weight as it represents the current information needs:

$$tf_i^{(q)} = (\#wordInContext / \#wordInQuery) * tf_i^{(q)} \quad (2)$$

Therefore,

$$tf_i^{(c)} = tf_i^{(q)} + tf_i^{(dq)} + tf_i^{(sq)} + tf_i^{(other)} \quad (3)$$

The inverse document frequency is defined by:

$$idf_i^{(c)} = \ln(N / n_i) \quad (4)$$

where $N$ is total number of documents in the corpus, and $n_i$ is the number of documents containing *ith* term. The term weight is defined by:

$$W_i^{(c)} = \frac{\ln(1 + tf_i^{(c)}) \times idf_i^{(c)}}{\sqrt{\sum \ln^2(1 + tf_i^{(c)}) \times \sum idf_i^{(c)2}}} \quad (5)$$

These context feature vectors are calculated for later use in re-ranking search results.

Meanwhile, we use Brill's tagger (Brill, 1995) to determine parts of speech (POS) of words in DQ/SQ. Heuristic rules (Zhang and Xuan, 2005) based on POS are used to extract noun phrases.

Noun phrases containing words with high term weight are considered as keyphrases. The keyphrase weight is defined by:

$$W_{P_i}^{(c)} = \sum_j W_j^{(c)} \quad (where\ W_j^{(c)} \in Phrase\ P_i) \quad (6)$$

### 3.2 Term Suggestion

When a user commits a query, OLISA will first search it on selected search engines (Google as default). If the total hit exceeds certain threshold (2 million as default), we consider the query potentially too general. In addition to the original query, we will call term suggestion component to narrow down the search concept by expanding the query. WordNet (Fellbaum, 1998) is used during the expansion. Below is the outline of our heuristic algorithm in generating term suggestion.

```
for each keyword in original query do
    if the keyword is part of a keyphrase then
        form queries by merging each phrase with the original query
    if multiple keyphrases are involved then
        select up to #maxPhrase keyphrases with highest weights
if #queries>0 then return queries
for each keyword that has hyponyms in WordNet do
    if some hyponym occur at least once in learning context then
        form queries by merging the hyponym with the original query
    else form suggestions by merging the hyponym with the original query
if #queries>0 or #suggestions> 0 then return queries and suggestions
for each keyword in original query that has synonyms in WordNet do
    if some synonym is part of a keyphrase then
        form suggestions by merging keywords in phrase with original query
    if multiple keyphrases are involved then
        select up to #maxPhrase keyphrases with highest weights
return suggestions
```

On the other hand, if the total hit is below certain threshold, the query is potentially too specific. Thus term suggestion component is called to generalize the query. The procedure is similar to the algorithm above, but will be done in the reverse direction. For example, keywords will replace phrases and hypernyms will replace hyponyms. Since there are cases where learners desire specific search terms, both original and expanded queries will be submitted, and results for the former will be presented at the top of the returned list.

If no new queries are constructed, OLISA will return the results from original query along with suggestions. Otherwise, OLISA will send requests for each expanded query to selected search engines. Since by default we return up to $R_T=100$ search engine results to user, we will extract the top $R_Q=R_T/(\#newQuery+1)$ entries from results of each new query and original query. These results will be re-ranked by an algorithm that we will describe later. Then the combined results will be presented to the user in IdeaKeeper along with a list of expanded queries and suggestions.

### 3.3 Query Reformulation

From our observation, in OIBLE students often submit questions in natural language. However, most of the time, such type of queries does not return desirable results. Therefore, we loosely follow Kwok (2001) to reformulate queries. We apply Link Grammar Parser (Sleator and Temperley, 1993) to parse sentence structure. For example, one student asked "What is fat good for". The parser generates the following linkage:

```
    +----------------Xp---------------+
    |       +----------Bsw---------+   |
    |       |    +----Paf----+     |   |
    +---Wq--+    +-SIs+      +-MVp-+   |
    |       |    |    |      |     |   |
LEFT-WALL what is.v fat.n good.a for.p ?
```

where "SI" is used in subject-verb inversion. By getting this linkage, we are able to reformulate the query as "fat is good for". Meanwhile, regular expressions are developed to eliminate interrogative words, e.g. "what" and "where".

Search engines may return very different results for the original query and the reformulated queries. For example, for the example above, Google returned 620 hits for the reformulated query, but only 2 hits for the quoted original question. By sending request in both original and reformulated forms, we can significantly improve recall ratio without losing much precision.

### 3.4 Integrating Multiple Search Engines

We enhanced the searching component of IdeaKeeper by integrating multiple search engines (e.g. Google, AskJeeves, NSDL, etc.). IdeaKeeper will parse and transform search results and present users with a uniform format of results from different search engines. A spelling check function for search keywords is built in OLISA, which combined spelling check results from Google as well as suggestions from our own program based on a local frequency-based dictionary.

### 3.5 Search Results Re-Ranking

After query reformulation OLISA will send requests to selected search engines. For performance issue, we only retrieve a total of 100 snippets ($R_Q$ snippets from each query) from web search engines. Feature vector is calculated for each snippet in the measure similar to (5), except that *tf* is actual frequency without assigning additional weight.

The similarity between learning context $C$ and each document $D$ (i.e. snippet) is calculated as:

$$Similarity(C, D) = \frac{\sum_i^n W_i^{(c)} W_i^{(d)}}{\sqrt{\sum_i^n W_i^{(c)2} \sum_i^n W_i^{(d)2}}} \qquad (7)$$

The higher the similarity score, the more relevant it will be to user's query as well as to the overall learning context.

OLISA re-ranks snippets by similarity scores. To avoid confusion to learners, the snippets from the original query and the expanded queries are re-ranked independently. $R_Q$ re-ranked results from original query appear at the top as default, followed by other re-ranked results with signs indicating corresponding queries. The expanded queries and further search term suggestions are shown in a dropdown list in IdeaKeeper.

## 4 Preliminary Results and Discussion

OLISA is under development. While thorough evaluation is needed, our preliminary results demonstrate its effectiveness. We conducted field studies with middle school students for OIBL projects using IdeaKeeper. Fig.1 shows a case of using OLISA search function in IdeaKeeper. By video

taping some students' search session, we found that enhanced search functions of OLISA significantly saved students' effort and improve their experience on search. The term suggestions were frequently used in these sessions.

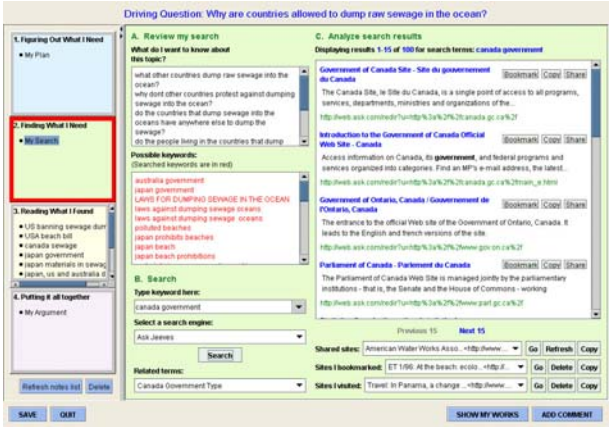


Fig. 1 Using OLISA function in IdeaKeeper

Our initials results also demonstrate that calculation on the snippets returned by search engines is simple and efficient. Therefore, we don't need to retrieve each full document behind. We want to point out that in our feature vector calculation each past query is combined into previous context. So the learning context is interactively changing.

Previous research has found that in OIBL projects, students often spend considerable time searching for sites due to their limited search skills. Consequently, students have little time on higher-order cognitive and metacognitive activities, such as evaluation, sense making, synthesis, and reflection. By supporting students' search, OLISA helps student focus more on higher-order activities, which provide rich opportunities for deep learning to occur.

Our future work includes fine-tuning the parameters in our algorithms and conducting more evaluation of each component of OLISA. We are also considering taking into account the snippets or documents users selected, because they also represent user feedback. How to determine the relative weight of words in selected documents, and how to disambiguate polysemies using WordNet or other resources are topics of future research.

## References


Brill, E. (1995). *Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part of Speech Tagging*, Computational Linguistics, 21, 543-565.

Buckley, C., Salton, G. and Allan, J. (1994). *The Effect of Adding Relevance Information in a Relevance Feedback Environment*. Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval. Dublin, Ireland, 292-300.

Fellbaum, C., Ed. (1998) *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA.

Harabagiu, S.M., Pasca, M.A. and Maiorano, S.J. (2000) *Experiments with Open-Domain Textual Question Answering*. Proceedings of the 17th conference on Computational linguistics. Saarbrucken, Germany, 292-298.

Huang, C.-K., Oyang, Y.-J. and Chien, L.-F. (2001) *A Contextual Term Suggestion Mechanism for Interactive Web Search*. Web Intelligence, 272-281

Kwok, C., Etzioni, O. and Weld, D. (2001) *Scaling Question answering to the Web*. ACM Transactions on Information Systems, 19, 242-262.

Lawrence, S. (2000) *Context in Web Search*, IEEE Data Engineering Bulletin, 23, 25–32.

Porter, M.F. (1980) *An algorithm for suffix stripping*. Program, 14, 130-137.

Quintana, C. and Zhang, M. (2004) *The Digital IdeaKeeper: Integrating Digital Libraries with a Scaffolded Environment for Online Inquiry*. JCDL'04. Tuscon, AZ, 388-388.

Ravichandran, D. and Hovy, E. (2002) *Learning Surface Text Patterns for a Question Answering System*. Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, Philadelphia, PA, 41-47.

Salton, G. and Buckley, C. (1990) *Improving retrieval performance by relevance feedback*. Journal of the American Society for Information Science, 41, 288-297.

Sleator, D. and Temperley, D. (1993) *Parsing English with a Link Grammar*. Proceedings of the Third International Workshop on Parsing Technologies.

Voorhees, E. (2001) *Overview of the TREC 2001 Question Answering Track*. Proceedings of the 10th Text Retrieval Conference (TREC10). Gaithersburg, MD, 157-165.

Zhang, M. and Xuan, W. (2005) *Towards Discovering Linguistic Features from Scientific Abstracts*. Proceedings of the 26th ICAME and the 6th AAACL conference. Ann Arbor, MI.