Improving User’s Web Search Experience by Interactive Re-ranking and Zooming Interfaces

Jianquan Liu, Hanxiong Chen, Kazutaka Furuse, Nobuo Ohbo
Department of Computer Science, Graduate School of Systems and Information Engineering,
University of Tsukuba, 1-1-1 Tennodai, Tsukuba-shi, Ibaraki-ken, 305-8577, Japan
{ljq, chx, furuse, ohbo}@dblab.is.tsukuba.ac.jp

Abstract: The well-known search engines such as Google, Yahoo! and MSN, are not enough for the advanced users who require user experience, for instance, much more correlative words, or interactive operations applied to returned results such as re-ranking and zooming. To focus on this idea of interactivity, we proposed and implemented interactive interfaces to improve user experience for supporting Web search. They are nice interfaces supporting dynamically changing re-ranking, related word zooming, and related word emphasizing and deemphasizing. It has been served as online search service recently. Our experimental evaluations confirm their convenience and good online performance.

Introduction
Web search has been taking an important role of our ordinary life due to its convenience, with rapidly developing of the Internet. You cannot be too hard to say that people are using the online Web search service everyday. However, only a large number of search result pages are returned to users, or sometimes correlative words are suggested for users to search again. Moreover, in most of them, such as Google, Yahoo, MSN, and so on, re-ranking function, visualization view, zooming operation and other interactive motions cannot be supported for advanced searching users. In these cases, most of these search engines will become powerless.

To satisfy with the requirements from advanced users as much as possible, many researches have been focusing on improvement based on the original searching results. For example, (Ding & Chi, 2003), (Yamamoto, Nakamura, & Tanaka, 2007a) and (Yamamoto, Nakamura, & Tanaka, 2007b) presented the improvement by ranking or re-ranking method. (Liu, Chen, Yu, & Ohbo, 1998) firstly used data mining technique to generate related words, and provided them to users as refinement candidates. Similarly, (Otsuka, Toyoda, & Kitsuregawa, 2005) and (Sato & Sasaki, 2003) gave a finding model and a collection method of related words to refine search keyword specification for users.

Besides, in order to make searching more convenient and visualizable, there was a study for comparing textual and zoomable user interfaces published in (Rivadeneira & Bederson, 2003). It attempted to provide a controlled comparison among three interfaces, Grokker1, Grokker Text2 and Vivisimo3, and concluded a better understanding of the potential that a ZUI (Zoomable User Interface) based on visualization may offer clustered search results. As the result of ZUI’s better understandable potential, we considered to apply ZUI in our research objective.

In spite of such many researches concerned with improvement, they ignore the fact that combining some techniques interactively could provide a more convenient and faster advanced search support and improve user experience. In this paper, we are interested in aggregation of some techniques to create an interactive interface to support Web search for improving user experience, including re-ranking function, visualization view, zooming operation, and special emphasizing and deemphasizing. We also present the implementation of our Web Search Supporting System (ZmSearch4). It can refine the search results fetched from Yahoo!JAPAN Web- Search API5, and provide a candidate set of related words with input keywords, via an interactive interface. It can also support an aggregation of interface operations — search results re-ranking, dynamically related words zooming, target words emphasizing in highlight color, slider for easily page changing, and response to search again.

The remainder of the paper is organized as follows. In Section 2, we discuss some related applications. Section 3 presents our approach, and then the system design and its implementation come to Section 4 and Section 5. Following these, we give our experimental evaluations of our system and interactive interface in Section 6. Finally, Section 7 concludes this paper, together with the future work.

1 http://www.groxis.com/
2 http://www.vivisimo.com/
3 http://zmsearch.dblab.is.tsukuba.ac.jp/
4 http://www.dblab.is.tsukuba.ac.jp/~ljq/zmsearch/
5 http://developer.yahoo.co.jp/search/
Related works and applications

Interface applications

In this section, we discuss several applications about efficient interface techniques, followed by related work with simple introduction of each application. Meanwhile we will pick up their convenience and key techniques as interface application to information retrieval.

Google Maps\(^6\) is probably the first application of zooming interface in the field of Information Retrieval, since it was firstly announced on the Google Blog\(^7\) on February 8 2005. Google Maps features a map that users can pan (by dragging the mouse) and zoom (by using the mouse wheel). Users may enter an address, intersection or general area to quickly find it on the map. Its zooming bar is a very nice interface to change the map view according to different zooming levels. It is not too hard to say that the zoomable interface applied in Google Maps can be considered efficient and excellent.

Grokker\(^8\) is a visual search engine, which is also using zoomable interface to present search results after clustering processing. It is implemented under FLASH components and its look is much different from the zooming interface of Google Maps. It can also divide related keywords into several clusters, and then it will provide users with detailed information according to different zooming levels and chosen cluster. Its efficiency was proved, but with a little pity that its online performance is too slow for practical use.

Vivisimo\(^9\) is another search engine, which also presents clustered search results. The difference is that it is applying textual interface to organize the search results into different clusters, according to their degree of similarity. It looks like the tree control as Windows Explorer. It is very fast and efficient because the detailed results will be loaded dynamically when the user click the “+” to spread the branches of the tree.

Recently, other studies for zoomable interface for information retrieval have been attracting much attention. For example, Araki (Araki, Miyamori, Minakuchi, Stejic, & Tanaka, 2005) proposed a new browsing method for information in Web environments by relating a continuous zooming operation to a search result window.

Mindset\(^10\) is served by Yahoo! Inc., which allows you to re-rank search results for your query into commercial or noncommercial (informational) results, based on whether you are shopping or seeking information (FAQ\(^11\)). Moreover, it applied slider as a nice interface for users to change their seeking purpose. But it is so commercial only towards the intent driven that it is not suitable to ordinary users, and it does not suggest any related words with query keywords for re-ranking either.

Rerank.jp\(^12\) introduced a re-ranking method, and then implemented a re-ranking interface supporting words deletion and emphasis by using multi-tag-cloud (Yamamoto et al., 2007b) interface. It is a nice system to support quick reranking according to the deletion or emphasis intended by user.

Related terms

In most of search engines, such as Google and Yahoo!, only frequently co-searched words can be suggested by analyzing a large number of user searching histories. So there are many studies which have focused on related words (or related terms) for information retrieval. Generally, the studies about related words can be divided into two kinds of measurement, co-occurrence and semantic similarity of words. For instance, an early work in (Takaki & Kitani, 1999) combines word co-occurrence with traditional word frequency to calculate relevance ranking of documents. On the other hand, Bollegala et al. (Bollegala, Matsuo, & Ishizuka, 2007) recently proposed a robust semantic similarity measure that uses the information available on the Web to measure similarity between words or entities.

Our approach

In order to provide a more convenient and faster support to Web search for improving user experience, we aggregated several efficient interface techniques to propose a key idea about interactive interface, and implemented an online Web search system applying our interactive interface idea.

We considered that our interactive interface can avoid annoying operations happening in most of modern search engines. For instance, users have to click “next page” to confirm whether the next several ten pages contain their seeking formation or target page. Instead, we applied a slider interface to avoid the heavy clicks and used asynchronous request technique to slower waiting time for the next page. To suggest users more correlated words to search again or re-rank the original search results, we provide a zoomable interface for users to drive re-ranking, displaying, emphasizing, deemphasizing and other operations. It looks like Google

\(^6\) http://maps.google.com/
\(^7\) http://en.wikipedia.org/wiki/GoogleMaps
\(^8\) http://www.groxis.com/
\(^9\) http://www.vivisimo.com/
\(^10\) http://mindset.research.yahoo.com/
\(^12\) http://rerank.jp/
Maps, of which related word list can be zoomed in or out according to the degree of similarity. At the same time, we provide check boxes embedding with related words, so that users can click to emphasize certain words in the highlight color amount the results. When emphasizing, search results are also re-ranked to enhance the order of documents which contain the target words. Due to this interactive re-ranking, the users can find out their seeking information more easily and quickly.

According to the main idea described above, our approach is to build a general Web search supporting system which can be applied quickly to support the current well-known search engines. To make implementation simple and typical, we use Yahoo!JAPAN WebSearch API in our implementation as the base. However, since the returned results are too huge, it is not impossible but very difficult and meaningless to analyze the whole results. For example, the Yahoo! Web search engine always returns about 1 billion results to the query term “NBA”. Instead, we treat the top 100 returned results as our analysis objective, a partial sample of the original results. That is enough to satisfy the general search task. The core design of our approach about two models, related words computation model and re-ranking model, is described in the following subsections.

Related words computation model

The vector model (Baeza-Yates & Ribeiro-Neto, 1999) is a simple, fast and popular retrieval model nowadays, being used to rank documents according to their degree of similarity to the query. In our study, we extended the vector model as our proposed computation model to rank related words according to their degree of similarity to the query.

**Definition 1** Let \( w_{q,j} \) be the weight associated a query (term) \( q \) with a document \( d_j \), where \( w_{q,j} \geq 0 \). Then the query vector \( \vec{q} \) is defined as \( \vec{q} = (w_{q,1}, w_{q,2}, \ldots, w_{q,N}) \) where \( N \) is the total number of documents in the returned result set. Similarly, let the weight \( w_{i,j} \) associated a term \( k_i \) with a document \( d_j \). Then the vector for \( k_i \) is represented by \( \vec{k}_i = (w_{i,1}, w_{i,2}, \ldots, w_{i,N}) \).

![Figure 1. The cosine of θ is adopted as sim(ki, q)](image)

Therefore, a term \( k_i \) and a query term \( q \) are represented as \( N \)-dimensional vectors as shown in Fig.1. Our extended vector model (shortly, EVM) proposes to evaluate the degree of similarity of the term \( k_i \) with regard to the query term \( q \) as the correlation between the vectors \( \vec{k}_i \) and \( \vec{q} \). This correlation can be quantified, for instance, by the cosine of the angle between these two vectors. That is,

\[
\text{sim}(k_i, q) = \frac{\vec{k}_i \cdot \vec{q}}{||\vec{k}_i|| \times ||\vec{q}||} = \frac{\sum_{j=1}^{N} w_{i,j} \times w_{q,j}}{\sqrt{\sum_{j=1}^{N} w_{i,j}^2} \times \sqrt{\sum_{j=1}^{N} w_{q,j}^2}}
\]

(1)

where \( ||\vec{k}_i|| \) and \( ||\vec{q}|| \) are the norms of the term and query vectors. The factor \( ||\vec{q}|| \) does not affect the ranking because it is the same for all terms. The factor \( ||\vec{k}_i|| \) provides a normalization in the space of the documents. Since \( w_{i,j} \geq 0 \) and \( w_{q,j} \geq 0 \), \( \text{sim}(k_i, q) \) varies from 0 to 1. Thus, instead of attempting to predict whether a term is relevant or not, the EVM ranks the terms according to their degree of similarity to the query.

If the user query contains multiple terms, denoting as a set \( Q \), we compute the sum of \( \text{sim}(k_i, q) \) for all \( q_j \in Q \) by Equation 2 before ranking.

\[
\text{sim}(k_i, Q) = \sum_{q_j \in Q} \text{sim}(k_i, q_j)
\]

(2)

\[
tf_{ij} = \frac{\text{freq}_{ij}}{\max \{\text{freq}_{ik,j}\}}
\]

(3)

\[
idf_k = \log \frac{N}{n_i}
\]

(4)

\[
w_{i,j} = \text{tf}_{ij} \times idf_k
\]

(5)
where all the notations used in these equations are also summarised in Table 1. The maximum in Equation 3 is computed over all terms which are mentioned in the text of the document \( d_j \). It is in common that the term-weight \( w_{i,j} \) can present the importance of term \( k_i \) in the document \( d_j \), which is also the main reason why we reuse \( w_{i,j} \) in our re-ranking model.

### Re-ranking model

Our re-ranking model is implemented in three separated processing phases, corresponding to three operations to related words, namely zooming slider, “+” and “−” check boxes. As in our definition, \( S_{core} \) (initialized to zero) denotes the score value of a page \( d_j \), which will be used in different re-ranking processing phase. A page of higher score is ranked before one of lower score. One phase is when the user operates the zoomable interface of related words to zoom-in or zoom-out. At the same time re-ranking will be driven according to Expression 7. In another phase, if the user chooses a certain related word in the list to emphasize it, re-ranking will be applied by Expression 8. Otherwise, Expression 9 will be used for re-ranking while a certain related word is deemphasized from the search results.

\[
\text{Score}_j = \text{Score}_j + \sum_{k \in d_j} \text{sim}(k_i, q)
\]

(7)

\[
\text{Score}_j = \text{Score}_j + w_{i,j} \times \text{sim}(k_i, q) \quad \text{iff} \; k_i \in d_j
\]

(8)

\[
\text{Score}_j = \text{Score}_j - w_{i,j} \times \text{sim}(k_i, q) \quad \text{iff} \; k_i \in d_j
\]

(9)

### System architecture

Figure 2 gives the overview of the system architecture, and processes in each phase. According to the processing flows in Figure 2, we can track three processing engines — Ajax engine, search engine and analysis engine. Firstly, the searching request comes to the Ajax engine, which posts the request to the search engine running in the background on the Web server, and then the search engine fetches the top 100 pages (documents) from Yahoo!JAPAN by its WebSearch API13. It also immediately transfers these pages to analysis engine for continuing to do text analyzing and related words computation. After that, it outputs the top 20 related words with their similarity value as response data to Ajax engine.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>( k_i )</td>
<td>( i )th term(word) in candidate set</td>
</tr>
<tr>
<td>( d_j )</td>
<td>( j )th document in returned search result</td>
</tr>
<tr>
<td>( q )</td>
<td>a query term(word)</td>
</tr>
<tr>
<td>( Q )</td>
<td>( Q = [q_1, q_2, \ldots, q_n] ), the term set in the query</td>
</tr>
<tr>
<td>( w_{i,j} ), ( w_{q,j} )</td>
<td>weight associated with a pair ((k_i, d_j), (q, d_j))</td>
</tr>
<tr>
<td>( \overrightarrow{q}, \overrightarrow{k_i} )</td>
<td>vector of ( q, k_i )</td>
</tr>
<tr>
<td>(</td>
<td>\overrightarrow{q}</td>
</tr>
<tr>
<td>( N )</td>
<td>total number of documents in returned search result</td>
</tr>
<tr>
<td>( \text{sim}(k_i, q) )</td>
<td>degree of similarity between ( k_i ) and ( q )</td>
</tr>
<tr>
<td>( f_{freq_{i,j}} )</td>
<td>raw frequency of term ( k_i ) in document ( d_j )</td>
</tr>
<tr>
<td>( tf_{i,j} )</td>
<td>normalized frequency of term ( k_i ) in document ( d_j )</td>
</tr>
<tr>
<td>( idf_{i} )</td>
<td>inverse document frequency for term ( k_i )</td>
</tr>
<tr>
<td>( n_i, n_q )</td>
<td>document frequency for term ( k_i, q )</td>
</tr>
<tr>
<td>( \text{Score}_{j} )</td>
<td>Score of ( d_j ) for re-ranking</td>
</tr>
</tbody>
</table>

Table 1

Definitions

The Ajax engine responds to users while dealing with interactive interface operations, such as page changing action, related words zooming and re-ranking list representation, and so on. Indeed, the analysis engine is an important component of our system architecture, and it depends on the EVM computation model and re-ranking model we proposed.

System implementation

Related words analysis

In the related words analysis processing phase, we defined wordAnalysis as analysis procedure in Procedure 1. It collects the title and summary text of documents Dtop100 with the query Q as input data, and outputs the top 20 most related words for the next processing phase, attaching with their degree of similarity $sim(k_i, Q)$ to the query Q.

From line 1 to line 7 in Procedure 1, the text strings in title and summary of documents are preprocessed by doing tokenization, stopwords elimination, stemming and splitting into a meaning word set orderly. In our implementation, we used a big stop list\(^{14}\) containing 900 stopwords, which is provided by Dr. Drott’s information retrieval resource. The stemming processing is cited from Porter Stemming Algorithms (Rijsbergen, Robertson, & Porter, 1980).

For offering the top 20 most related words to next phase, we apply EVM mentioned before to compute degree of similarity $sim(k_i, Q)$ to the query Q for all terms $k_i$ in candidate set Setwords. It is confirmable from line 8 to line 14 in Procedure 1.

Re-ranking methods

According to the re-ranking model mentioned in Section 3, we proposed three computation expressions for different re-ranking purposes. When the user slides the zooming bar in the zoomable interface, related words are displayed according to the zooming level, and the page list is re-ranked according to the Score values calculated using Expression 7. The re-ranking computation by Expression 8 will be applied when user checks the “+” check-box to emphasize a certain word in the target documents. Meanwhile, the pages containing the emphasized words will be given a higher rank. Based on the idea that the concerned word $k_i$ must be treated as a very important factor, we multiply $w_{ij}$ by its $sim(k_i, Q)$ value to enhance the ranking order of document $d_j$. Similarly, if user clicks the “−” check-box to eliminate the influence of a word, we will use Expression 9 to lower the ranking order of documents containing this word.

---

\(^{14}\) http://drott.cis.drexel.edu/retrieval.html
As shown in Figure 2, as soon as the analysis engine finishes its processing via the EVM computation model to compute the degree of similarity for each word, the main task to improve user experience comes to the Ajax engine. It is the key point of this paper, so we give the implementation of our interactive interface, consisting of the following phases.

- Page changing action
- Re-ranking list representation
- Related words zooming
- Interactivity correlation

We use Ajax engine to deal with the computation for reranking and operations of interactive interface. We also fetch the top 100 search results by the way of Yahoo!JAPANWeb-Search API\(^\text{15}\). Our system implementation consists of approximately 3500 lines of commented PHP and JavaScript codes. In the following subsections, details in each phase of implementation will be given.

**Page changing action.**

In order to make the page changing action more smooth, and to reduce the waiting cost for getting the next ten page results like the traditional search engines, we implemented a slider interface to control the moves among pages conveniently. As shown in Figure 3, the slider bar is divided into ten pieces like a ruler, with a sliding ball. When users slide the ball in the interval of each piece, every part (which includes ten page) of the result will be displayed in the ranking list. Certainly, the sliding ball can be dragged or clicked to move as well. At the same time, there are two triangle buttons at the both sides of the slide bar, the user can also click them or the tips word “Prev” and “Next” to make the ball moving step by step. When the mouse is over on the sliding ball, the position of current ten pages will be hinted by a tip layer. For instance, in Figure 3, the current position is at the fourth part.

\(^{15}\text{http://developer.yahoo.co.jp/search/}\)
Re-ranking list representation.

The re-ranking list representation consists of the comparison between original and new ranks, the emphasis in different highlight colors for different words, and the elimination of influence for a certain word. For example, in Figure 4, there are two numbers in front of the title of each document, of which the first one is the new rank for the document, and the other one in the couple of parentheses is the original rank.

On the other hand, three words of “basketball”, “player” and “draft” are emphasized in different highlight colors and bold font. In this case, the rank of the documents containing these emphasizing target words has been enhanced, such as the original rank 33 up to rank 1 in Figure 4. If the user eliminated the influence of a certain word, the ranking order of the documents will be lowered and the word will not be emphasized in any color either.

![Figure 4. Re-ranking list representation](image)

Related words zooming.

We implemented a zooming interface like Google Maps to control the representation for the related word list as shown in Figure 5. The left side is a related word list which can be zoomed dynamically according to the degree of similarity held for each word. The slider at the right side is divided into eleven zooming levels from zero to ten. Accordingly, the length of the related word list will be adjusted dynamically when it is zoomed into a certain level. It is a little different from Google Maps that our zooming interface can only be driven on one-dimensional space in this work. Beside this feature, we can see two columns of check boxes at the left side with each word. The “+” column is prepared for emphasizing the target word in the search result, oppositely the “−” column will drive eliminating the influence of the target word to the ranking results. The user can immediately search the Web again by clicking the hyperlink “Search again” at the bottom of the related word list after selecting several related words. Due to the efficiency and convenience of our zooming idea to adjust the related words, we will try to extend the display on two or more dimensional space in the future.

Interactivity correlation.

The main idea of interactive interface is based on the interactivity correlation as shown in Figure 6. When the user operates the zooming interface, the related word list will be adjusted dynamically. Meanwhile, we considered that changing related word list affects the ranking list, because the more related word has more influence to the ranking result for the user. Thus, it is better that the search results should be reranked interactively at the same time. Similarly, in allusion to a certain related word, the emphasizing or deemphasizing operation should be considered that it affects the ranking result much more. Consequently, re-ranking will be also processed interactively in this case by using the re-ranking model mentioned before.

![Figure 5. Related words zooming](image)
Figure 6. Interactivity correlation

As a running example, we give a snapshot of our interactive interface as shown in Figure 7. It also shows a sample of search result of keyword “NBA” which appears in the head of the page. In the area of page changing slider bar, the information message denotes the summary of search result, and the total number of pages hit, running time, displaying page number and so on. The main area on the left part is a screen holding 10 ranking elements per page, and the leftmost column is the order numbers after re-ranking, the numbers in the second column are the original ranking order. The key zoomable interface and related word list are embedding in the right pane. Lastly, at the bottom of related word list, The “Search again” button enables the user to do refinement search. When clicked, the system searches the Web again with the checked words in the “+” column added to the original query. The checked words are also being emphasized in different highlight colors.

Figure 7. Snapshot of interface

Evaluations

In this section we describe the evaluation we performed for the ZmSearch system. Section Precision of related words discusses the precision of related words in our experimentations, while section User experience describes the user experience. Section Online run-time performance evaluates the run time performance of our system.

Precision of related words

Experimental setup.

To compare the precision of related words provided in different approaches, we performed a user study. We chose 11 hot keywords as our search targets, which are gathered from Google Trends\(^{16}\) and Yahoo! Buzz\(^ {17}\). These 11 hot keywords are about three different categories, 4 keywords about programming language, 3 keywords about software technology and the rest 4 keywords about sports. Details are presented in Table 2.

\(^{16}\) http://www.google.com/trends

\(^{17}\) http://buzz.yahoo.com/
We applied these keywords to search the Web and gathered top 10 related words returned by three different systems, which are related search words service API\(^{18}\) (we call it Yahoo! API in the paper) provided by Yahoo!JAPAN, Rerank.jp and our ZmSearch. We then put the answers in ascendant order with omission to make sure each word unique. We organized a questionary sample as Table 3, where each related word list is separated according to 11 different search keywords. Combining 11 tables, we organized a complete questionary sheet in the end, which contains about 300 related words.

\(^{18}\)http://developer.yahoo.co.jp/search/

Table 2
Search keywords

We delivered our questionary sheets to 21 volunteers for rating. Our volunteers included 1 professor and 20 students, who are 2 females and 19 males. They are all computer science professionals (mostly our colleagues at our Database Research Laboratory). To make the comparison fair, we did not present the names of the three systems on the questionary sheet. Instead, we only informed the volunteers that all the related words came from three different search systems. The volunteers were instructed as follows:  

\textit{We want to measure how much the listed words are related to the keywords, which were gathered from three different search systems and combined together in ascending order. Please give your judgement according to the memo in our questionary tables.}

User study results.

From the answers we received, we evaluated average precision at top 10 related words for three different systems, similar to the evaluation method that Jeffrey Dean proposed in (Dean & R.Henzinger, 1999). Our statistic analysis is dealt as the following procedure.

\[^*\] She is a famous Japanese figure skater.

Table 2
Search keywords

<table>
<thead>
<tr>
<th>Search keywords</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>ajax, java, ruby, python</td>
<td>4</td>
</tr>
<tr>
<td>database, data mining, open source</td>
<td>3</td>
</tr>
<tr>
<td>NBA, ESPN, NHK, Miki Ando*</td>
<td>4</td>
</tr>
</tbody>
</table>

* She is a famous Japanese figure skater.

### Table 2
Search keywords

We applied these keywords to search the Web and gathered top 10 related words returned by three different systems, which are related search words service API\(^{18}\) (we call it Yahoo! API in the paper) provided by Yahoo!JAPAN, Rerank.jp and our ZmSearch. We then put the answers in ascendant order with omission to make sure each word unique. We organized a questionary sample as Table 3, where each related word list is separated according to 11 different search keywords. Combining 11 tables, we organized a complete questionary sheet in the end, which contains about 300 related words.

\[^{18}\]http://developer.yahoo.co.jp/search/

Table 3
Questionary Sample

We delivered our questionary sheets to 21 volunteers for rating. Our volunteers included 1 professor and 20 students, who are 2 females and 19 males. They are all computer science professionals (mostly our colleagues at our Database Research Laboratory). To make the comparison fair, we did not present the names of the three systems on the questionary sheet. Instead, we only informed the volunteers that all the related words came from three different search systems. The volunteers were instructed as follows:  

\textit{We want to measure how much the listed words are related to the keywords, which were gathered from three different search systems and combined together in ascending order. Please give your judgement according to the memo in our questionary tables.}

User study results.

From the answers we received, we evaluated average precision at top 10 related words for three different systems, similar to the evaluation method that Jeffrey Dean proposed in (Dean & R.Henzinger, 1999). Our statistic analysis is dealt as the following procedure.
Firstly, we converted the three kinds of symbol (Ο, Δ, ×) in the returned questionary sheets into weights (1, 0.5, 0), so that all the judgements are computable. Ο-1 means that the volunteer considered the word as related to search keyword, oppositely ×-0 denotes not related. Δ has 50% probability that the word is related to search keyword, we assigned the weight 0.5 for computation.

Secondly, for each search keyword, we computed three average precision $p_s$ by Equation 10 independently, where s denotes the label number of three system compared here. Yahoo! API, Rerank.jp and ZmSearch are labeled respectively as 1, 2, and 3.

$$p_s = \frac{\sum_{j=1}^{21} \sum_{i=1}^{10} \text{weight}_j(w_i)}{21}, \quad s = 1, 2, 3$$

Thirdly, we achieved numerical results as Table 4 after finishing the computation, as well as plotted a clearly comparable view as Figure 8.

<table>
<thead>
<tr>
<th>Keywords</th>
<th>Yahoo!API</th>
<th>Rerank.jp</th>
<th>ZmSearch</th>
</tr>
</thead>
<tbody>
<tr>
<td>ajax</td>
<td>0.38</td>
<td>0.37</td>
<td>0.79</td>
</tr>
<tr>
<td>java</td>
<td>0.69</td>
<td>0.46</td>
<td>0.52</td>
</tr>
<tr>
<td>ruby</td>
<td>0.20</td>
<td>0.47</td>
<td>0.57</td>
</tr>
<tr>
<td>python</td>
<td>0.50</td>
<td>0.54</td>
<td>0.61</td>
</tr>
<tr>
<td>database</td>
<td>0.37</td>
<td>0.37</td>
<td>0.66</td>
</tr>
<tr>
<td>data mining</td>
<td>0.00</td>
<td>0.57</td>
<td>0.68</td>
</tr>
<tr>
<td>open source</td>
<td>0.00</td>
<td>0.47</td>
<td>0.75</td>
</tr>
<tr>
<td>NBA</td>
<td>0.42</td>
<td>0.65</td>
<td>0.81</td>
</tr>
<tr>
<td>ESPN</td>
<td>0.79</td>
<td>0.60</td>
<td>0.69</td>
</tr>
<tr>
<td>NHK</td>
<td>0.79</td>
<td>0.33</td>
<td>0.74</td>
</tr>
<tr>
<td>Miki Ando</td>
<td>0.00</td>
<td>0.67</td>
<td>0.79</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>0.38</strong></td>
<td><strong>0.50</strong></td>
<td><strong>0.69</strong></td>
</tr>
</tbody>
</table>

Table 4

Statistic result

According to the statistical result in Table 4, our Zm-Search system achieved an average precision of 0.69 at top 10 related words, which is 38% better than Rerank.jp system, moreover 81.6% better than Yahoo!API.

Figure 8 shows a visualized result by comparable curves as well. We can observe that our ZmSearch system performed absolutely better than Rerank.jp system for suggesting the top 10 related words to search keywords. However, there are some outliers when comparing with Yahoo!API seeking information about “java”, “ESPN” or “NHK”. Because Yahoo!API does not support multi-word search for related words, its precision leads to zero while searching “data mining”, “open source” and “Miki Ando”.

User experience

Experimental user trial.

To compare the user experience with different search engines, we performed a experimental user trial. Firstly, in order to evaluate user experience, we setup seven evaluation criteria and organized a feedback questionary sheet. All the seven evaluation criteria are in the following list.

**Eval. 1:** Which provides the most number of related words

**Eval. 2:** Which suggests the most correlated words

**Eval. 3:** Which has the top re-ranking capability

**Eval. 4:** Which performs the top re-ranking efficiency

**Eval. 5:** Which is the most convenient user interface

**Eval. 6:** Which supports the most quick user interface response

**Eval. 7:** Which presents the best holistic user experience

Secondly, we invited 14 volunteers to participate in our experimental user trial. Our volunteers included 1 professor and 13 students, who are all computer science professionals. Thus certainly they are skilled searching users and they require advanced search support or better user experience. The volunteers were instructed as follows:

We want to evaluate whether our ZmSearch system with interactive interface can give better user experience than other search engines. Please try to use the following four search engines to finish at least three searching tasks. Please pay attention to their user interface, suggested related words, re-ranking function and their holistic experience during your tasks. After that, please fill in the questionary sheet to vote one or more in the four search engines according to each evaluation criterion, and then give us feedback about your trial.
Feedback result.

From the feedback questionnaire sheets we received, we evaluated user experience of our interactive interface according to all the evaluation criteria that we setup above. As soon as finished statistic analysis from user trial feedback, we plotted the comparable result in Figure 9. It predicates that our interactive interface achieved 5/7 of all evaluation criteria better than or the same as the other three search engines (Rerank.jp, Google, and Yahoo!). Especially, our ZmSearch was voted as the absolutely better re-ranking capability and efficiency. Besides, it was considered the same performance with Google in Eval.5 and Eval.6 about user interface conveniency and response speed. Totally, our interactive interface achieved better holistic user experience than Rerank.jp and Yahoo!. On the other hand, we applied the well-known Kendall-tau rank correlation coefficient (Abdi, 2007) to measure the vote agreement between each evaluator and statistic result. About the Eval.3 and Eval.4, their average Kendall-tau values are $\tau = 0.923$, $\tau = 0.667$, which are confident that our evaluators got in highly common agreement.

Online run-time performance

We also evaluated the online run-time performance of our ZmSearch system to confirm its efficiency. The experiment is performed on a Intel-based computer system running Linux.

![Figure 8. Precision at top 10](image)

<table>
<thead>
<tr>
<th>Keywords</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
<th>T6</th>
<th>T7</th>
<th>T8</th>
<th>T9</th>
<th>T10</th>
<th>T11</th>
<th>T12</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>ajax</td>
<td>3.22</td>
<td>1.98</td>
<td>1.92</td>
<td>1.97</td>
<td>1.96</td>
<td>2.00</td>
<td>1.92</td>
<td>2.03</td>
<td>1.89</td>
<td>1.96</td>
<td>1.97</td>
<td>2.01</td>
<td>1.97</td>
</tr>
<tr>
<td>java</td>
<td>4.33</td>
<td>1.99</td>
<td>1.93</td>
<td>1.92</td>
<td>1.87</td>
<td>1.97</td>
<td>2.08</td>
<td>1.98</td>
<td>2.00</td>
<td>1.92</td>
<td>2.07</td>
<td>2.07</td>
<td>1.99</td>
</tr>
<tr>
<td>ruby</td>
<td>3.95</td>
<td>2.05</td>
<td>1.95</td>
<td>2.07</td>
<td>2.08</td>
<td>2.02</td>
<td>2.09</td>
<td>2.06</td>
<td>2.02</td>
<td>2.05</td>
<td>2.11</td>
<td>1.98</td>
<td>2.05</td>
</tr>
<tr>
<td>python</td>
<td>3.44</td>
<td>1.94</td>
<td>1.94</td>
<td>1.94</td>
<td>2.04</td>
<td>2.01</td>
<td>1.93</td>
<td>2.23</td>
<td>1.99</td>
<td>2.05</td>
<td>1.90</td>
<td>1.97</td>
<td>2.00</td>
</tr>
<tr>
<td>database</td>
<td>4.09</td>
<td>1.95</td>
<td>2.08</td>
<td>2.01</td>
<td>2.09</td>
<td>2.00</td>
<td>1.96</td>
<td>1.89</td>
<td>1.95</td>
<td>1.90</td>
<td>2.00</td>
<td>1.94</td>
<td>1.99</td>
</tr>
<tr>
<td>data mining</td>
<td>3.67</td>
<td>1.89</td>
<td>1.94</td>
<td>1.98</td>
<td>1.89</td>
<td>2.09</td>
<td>1.88</td>
<td>1.94</td>
<td>2.11</td>
<td>1.91</td>
<td>1.99</td>
<td>1.90</td>
<td>1.96</td>
</tr>
<tr>
<td>open source</td>
<td>3.17</td>
<td>1.95</td>
<td>2.29</td>
<td>1.89</td>
<td>1.94</td>
<td>1.94</td>
<td>2.00</td>
<td>2.01</td>
<td>2.05</td>
<td>1.89</td>
<td>1.90</td>
<td>1.92</td>
<td>1.99</td>
</tr>
<tr>
<td>NBA</td>
<td>3.51</td>
<td>2.18</td>
<td>1.96</td>
<td>2.09</td>
<td>2.02</td>
<td>1.94</td>
<td>2.03</td>
<td>1.99</td>
<td>2.04</td>
<td>2.10</td>
<td>2.02</td>
<td>2.16</td>
<td>2.06</td>
</tr>
<tr>
<td>ESPN</td>
<td>4.71</td>
<td>2.27</td>
<td>2.01</td>
<td>1.95</td>
<td>1.99</td>
<td>1.98</td>
<td>1.93</td>
<td>2.11</td>
<td>1.99</td>
<td>1.95</td>
<td>1.96</td>
<td>2.01</td>
<td>2.02</td>
</tr>
<tr>
<td>NHK</td>
<td>4.24</td>
<td>2.11</td>
<td>2.06</td>
<td>2.24</td>
<td>2.11</td>
<td>2.17</td>
<td>2.13</td>
<td>2.22</td>
<td>2.17</td>
<td>2.27</td>
<td>2.15</td>
<td>2.15</td>
<td>2.17</td>
</tr>
<tr>
<td>Miki Ando</td>
<td>2.04</td>
<td>2.06</td>
<td>1.99</td>
<td>2.13</td>
<td>3.64</td>
<td>1.98</td>
<td>1.96</td>
<td>2.06</td>
<td>2.00</td>
<td>2.08</td>
<td>2.08</td>
<td>2.12</td>
<td>2.05</td>
</tr>
</tbody>
</table>

**Median**: 2.03
The CPU is Intel(R) Xeon (TM) 2.80 GHz and the amount of main memory is 3.2GB. We chose 11 hot keywords as our search targets, which are gathered from Google Trends\textsuperscript{19} and Yahoo!Buzz\textsuperscript{20}. These 11 hot keywords are about three different categories, 4 keywords about programming languages, 3 keywords about software technologies and the rest 4 keywords about sports. During our online run-time experimental testing, we repeated the same search task 12 times for each keyword, and recorded the running cost of each task. For each search keyword, we calculated its median running time but omitting the maximum and the minimal value in the 12 records. The details of online run-time cost are summarized in Table 5, where we can see that the median run-time performance is about 2.03 seconds. \( T_i \) denotes the \( i \)th search task in Table 5. This fact confirms that our system is feasible in the real world as an online Web search system.

**Conclusions and future work**

In this paper, we proposed a new idea about interactive interface aggregation, and implemented an online Web search system, which provides a more effective, convenient and faster support for Web search. It can suggest users a zoomable related word list, by which users can re-rank the original search result selectively. It supports users to search the Web again immediately after they choose some related words.

The experimental evaluation shows that our ZmSearch system provides users with related words in high precision. The online run-time our ZmSearch system performed a good online runtime cost, which is about 2 seconds on average, and it also reached nice feedback from the volunteers who took part in our experimental trial. As future work, we plan to extend the interactive interface to represent high-dimensional data search results.

\textsuperscript{19}http://www.google.com/trends \\
\textsuperscript{20}http://buzz.yahoo.com/
References


