Search Intent Estimation from User’s Eye Movements for Supporting Information Seeking

Kazutoshi Umemoto¹, Takehiro Yamamoto¹,², Satoshi Nakamura¹,², Katsumi Tanaka¹
¹Kyoto University, Yoshida-Honmachi, Sakyo, Kyoto, Japan
²JST CREST
{umemoto, tyamamot, nakamura, tanaka}@dl.kuis.kyoto-u.ac.jp

ABSTRACT
In this paper, we propose a two-stage system using user’s eye movements to accommodate the increasing demands to obtain information from the Web in an efficient way. In the first stage the system estimates a user’s search intent as a set of weighted terms extracted based on the user’s eye movements while browsing Web pages. Then in the second stage, the system shows relevant information to the user by using the estimated intent for re-ranking search results, suggesting intent-based queries, and emphasizing relevant parts of Web pages. The system aims to help users to efficiently obtain what they need by repeating these steps throughout the information seeking process. We proposed four types of search intent estimation methods (MLT, nMLT, DLT and nDLT) considering the relationship among intents, term frequencies and eye movements. As a result of an experiment designed for evaluating the accuracy of each method with a prototype system, we confirmed that the nMLT method works best. In addition, by analyzing the extracted intent terms for eight subjects in the experiment, we found that the system could estimate the unique search intent of each user even if they performed the same search tasks.

Categories and Subject Descriptors
H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval—Relevance feedback

General Terms
Design, Human Factors

Keywords
Eye tracking, search intent, query suggestion, re-ranking

1. INTRODUCTION
While the recent growth of the Web has enabled users to obtain various kinds of information, it also makes their search costs higher because they have to search for the information they need from a large amount of Web pages. Because of this, there is an increasing demand for more efficient ways to find relevant information from the Web. When one searches for information from the Web, one’s actions generally consist of the following two processes: a search process, which is the process of searching for pages that seem relevant from a search engine results page (SERP), and a browsing process, which is the process of finding relevant information in a Web page that the user has selected.

Regarding the search process, Spink et al. [23] reported that about 50 percent of search tasks include only a single query, and in the remaining half cases users perform several query reformulations during the task. In addition, more than 25 percent of the queries are modifications of a previous one, as stated in [11]. These data suggest that it is not easy for Web users to obtain relevant search results by submitting a single query. Considering these aspects, existing search engines introduce a diversification technique into SERPs for ambiguous queries [18]. This technique partly supports users whose information needs are unclear; however, it is disadvantageous for users who have fixed information needs since the search results do not seem to be ranked in order of their needs. Therefore, they find it difficult to obtain relevant search results. There are also problems in the browsing process. For example, even if users select good results from a SERP, it takes a long time for them to find information that meets their needs from the corresponding pages especially in those that are lengthy or have a complex design.

As described in [10], relevant information may well differ with users, time, and circumstances, even when different users submit exactly the same queries. Accordingly, it is important for future search services to be able to estimate each user’s search intent ac-
curately in all situations in order to provide relevant information for each user.

In this research, we focus on where users look when browsing Web pages. Figure 1 shows a heat map visualization of the eye movements of one of the authors when searching for information after issuing the query “Capri Island Italy”. From the highlighted areas of this figure, we can see that the user probably wants information about tours or hotels in Capri. In line with this thinking, we assume that users’ search intents influence their eye movements during information seeking. On the basis of this assumption, we propose a method to estimate users’ search intents in real time from ongoing information seeking by using their eye movements. Using the intent estimated by this method, we also propose some methods for supporting the users’ search and browsing processes in information seeking.

At present, it is difficult to obtain a high-performance eye tracker (which is a device for measuring eye positions and eye movements) at a low price. In recent years, however, webcams have become popular, as is clear from the fact that about one in four Internet users has a webcam, and these devices are improving in quality year by year. These cameras are just beginning to be used for detecting users’ eye movements. In the near future, it is expected that most PCs will be equipped with a high-performance camera and that software that uses eye movement data will become common. In fact, Tobii and Lenovo demonstrated a prototype eye-controlled laptop in March 2011.

The remainder of this paper is organized as follows. In Section 2 we discuss some studies related to ours. Section 3 describes our method for estimating users’ search intents using their eye movements. Section 4 presents methods that support their information seeking by using the estimated intents. Specifically, we propose a way to improve the search process by re-ranking search results and suggesting intent-based queries, as well as a way to improve the browsing process by emphasizing relevant parts of a Web page. We describe details of an implemented system in Section 5. Section 6 describes the results of user experiments and discusses their implications. Finally, Section 7 concludes the paper with a mention of future work to be done.

2. RELATED WORK

2.1 Existing Support for Information Seeking

Query suggestion [2, 7] is well known as a means of conventional support for information seeking. In this method, candidate queries for better search results are shown at the top of a SERP when a user makes a relatively short and general query. Query suggestions are generally given on the basis of query log analysis [2] or click-through data analysis [7]. This process can create candidate queries that are popular for most users. While this method does not work well when users’ information needs are unusual, our method can suggest queries even in this case by estimating users’ specific search intents during information seeking from their eye movements in real time.

Relevance feedback [19, 21] is another conventional means of providing support for information seeking. In this method, search results are personalized for each user on the basis of relevance information input from the user. Implicit relevance feedback [15], which is one type of relevance feedback, performs personalization that is not based on a user’s explicit input (e.g., relevance ratings for search results by the user) but on the user’s action such as link clicking [13] and mouse scrolling [5]. This technique is almost cost-free for users and its effectiveness is shown in [1]. The method we propose in this paper can be regarded as a type of implicit relevance feedback because it estimates a user’s search intent implicitly from data about where the user looked while browsing Web pages. Our method has an advantage over these methods in that it can work well even if users are satisfied with no need to click on any search result (a case known as good abandonment in [16]) or when a user works on a task in another window (e.g., writing an e-mail or creating a document) while displaying some parts of a Web page with a Web browser.

2.2 Applying Eye Tracking to IR

Recently, a wide variety of studies on IR have focused on tracking users’ eye movements, and the use of high-performance cameras or eye-trackers has made applying this technique much easier than before. For example, users’ eye movement data is used for investigating their strategies in the search process [9], for understanding their visual attention to Web pages through their browsing patterns [3], and for summarizing documents according to where each user’s attention is directed when reading [26].

Xu et al. [25] proposed a method of applying implicit relevance feedback to search results for documents, images, and videos on the Web by using eye tracking data. They showed the effectiveness of using eye tracking data in the search process from the results of experiments. While their method only supports the search process in information seeking, our method proposed in this paper supports both the search and browsing processes. In addition, our method differs from theirs in that ours considers the order in which a user browses Web pages and the term frequencies of terms that the user looks at.

Buscher et al. [4] proposed a method to extract terms for query expansion using gaze-based attention to sub-documents in Web pages. This method is similar to ours in that both calculate candidate queries on the basis of users’ eye movements. However, their method needs background data about where sub-documents begin and end for all Web pages to be put to practical use. In contrast, the method we propose does not require such data in advance. Moreover, our method has the potential for estimating users’ search intents at smaller granularity levels by considering their attention not to sub-documents but to terms in Web pages.

As described in [20], there are some ways in which a mouse is used in addition to just clicking hyperlinks in Web pages (e.g., using it as a reading aid or to mark an interesting result). Detecting these mouse movement patterns may make it possible to partly estimate users’ search intents from their information seeking. However, as discussed in [12], the cursor positions are not very close to the positions at which users look during the browsing process. Therefore, we think it is difficult to estimate users’ real search intents using only mouse movement data, and instead we chose to use their eye movement data in the work described in this paper.

3. ESTIMATING SEARCH INTENT

3.1 Search Intent Model

When starting information seeking, a user generally has an information need (the perceived need for information that leads to someone using an information retrieval system in the first place as defined in [22]). In this paper, we assume that the user is interested in parts of Web pages that concretely express this information need,
and we estimate the user’s search intent on the basis of this assumption.

We represent a user’s search intent as a set of pairs of a term and its weight. This idea is based on the vector space model [17], which is widely known in the IR research field and in which the weight of each term expresses the extent to which the term is related to the user’s intent. For example, suppose user uA wants to buy a digital camera that is popular and inexpensive, and user uB wants to buy one that is popular and has high-performance. Also, suppose both of them seek information by making the query “digital camera popular”. In this case, the two users’ search intents can be represented as follows: \{(camera, 0.5), (popular, 0.4), (price, 0.3), (discount, 0.3), \ldots\} for uA and \{(camera, 0.5), (popular, 0.4), (wide-angle-lens, 0.3), (face-recognition, 0.3), \ldots\} for uB.

Figure 2 shows the system flowchart when a user seeks information. If the user visits a new Web page by clicking a search result, the system starts to estimate the user’s search intent and extracts candidate terms for the intent set by eliminating stop words from terms that appear in the page. After this step, the system continues to detect terms the user looks at and calculates the extent of the user’s attention to the terms while she browses the current page. If the user jumps from the current page to another page, the system re-executes this processing procedure. If the user goes back to the SERP, the system calculates the user’s intent by aggregating the attention scores of all terms. The system then stops the estimation and waits until the user browses another Web page.

In the rest of this section, we describe how to calculate an attention score for each term and an intent score of a user. We describe how to identify terms looked at by a user in Section 5.

### 3.2 Estimation Methods

In this paper, we define four equations \(\text{TermScore}_{m}(t, p)\) that calculate the attention score for each candidate term \(t\) in a Web page \(p\) as follows:

\[
\text{TermScore}_{m}(t, p) =
\begin{cases}
\text{Look}(t, p) \cdot TF(t, p) & (m=\text{MLT}) \\
\frac{\text{Look}(t, p)}{\max_{s \epsilon \text{Look}(t, p)} \text{TF}(s, p)} \cdot \frac{\text{TF}(t, p)}{\max_{s \epsilon \text{TF}(t, p)}} & (m=\text{mMLT}) \\
\text{Look}(t, p) \cdot \frac{\text{TF}(t, p)}{\max_{s \epsilon \text{TF}(t, p)}} & (m=\text{DLT}) \\
\frac{\text{Look}(t, p)}{\max_{s \epsilon \text{Look}(t, p)} \text{TF}(s, p)} \cdot \frac{\text{TF}(t, p)}{\max_{s \epsilon \text{TF}(t, p)}} & (m=\text{mDLT})
\end{cases}
\]

where \(\text{Look}(t, p)\) is the number of times that a user looked at \(t\) in \(p\) and \(\text{TF}(t, p)\) is the term frequency of \(t\) in \(p\).

These methods assume that terms looked at many times by a user are related to the user’s search intent. The MLT (Multiply Look by TF) method is also based on the idea that the more times a term appears, the more relevance the term has to the user’s intent. The DLT (Divide Look by TF) method, on the other hand, is based on the idea that even if a term has low frequency in a page, this term has a strong relationship with the user’s intent if the user looks at it many times. Methods mMLT and mDLT are normalized versions of MLT and DLT; these methods absorb the difference in the range of values that \(\text{Look}(t, p)\) or \(\text{TF}(t, p)\) takes among Web pages.

Using these scores, the system calculates the intent score, defined as in the equation below, for every candidate term appearing in at least one of the Web pages browsed by the user. It then outputs the top \(n\) terms having the highest intent scores as the estimated search intent of the user. In this paper, we propose two methods of calculating the intent score of a term \(t\) considering the shift types of users’ intents. We express the score as \(\text{Intent}_{d_m}(t)\), when a user browses pages \(p_1, p_2, \ldots, p_k\) in this order. The calculation methods are:

\[
\begin{align*}
\text{Intent}_{d_m}(t) &= \sum_{i=1}^{k} \text{TermScore}_{m}(t, p_i) \\
&= \alpha \cdot \text{TermScore}_{m}(t, p_k) + \sum_{i=1}^{k-1} (1 - \alpha)^{k-i} \cdot \text{TermScore}_{m}(t, p_i)
\end{align*}
\]

where \(0 \leq \alpha \leq 1\) in Equation (2b) is a weight coefficient for terms at which a user looked in the latest page.

A uniform model calculates the score by treating every term in all pages that the user browsed as equally weighted. This model assumes that the user’s intent is almost fixed and uniformly distributed during a search session. In contrast, there also exist search sessions in which users’ search intents come and go (unclear information need may be one factor in these searches). A damping model can deal with these cases. It calculates the score by weighting terms that frequently appear in pages that the user browsed most recently. This means this model regards a newly browsed page as more relevant to the user’s search intent than a previously browsed one. This idea is based on the Ostensive Model in [6], which is a model of the progressive development of information needs.

### 4. SUPPORTING INFORMATION SEEKING

In this section, we describe methods of supporting users’ information seeking from both the search and browsing process standpoint, using the intents estimated through our method given in
Section 3. First, we propose re-ranking and query suggestion methods to support the search process. We then propose a method which emphasizes information relevant to the intents in Web pages to support the browsing process.

4.1 Supporting Search Process

4.1.1 Re-ranking Search Results

The system re-ranks a search result set \( R = \{r_1, \ldots, r_m\} \), in the light of a user’s search intent by using a vector \( v_{\text{intent}} \) composed of \( n \) intent terms and the weights described in the previous section. The flow of re-ranking is as follows:

1. Create a term frequency vector \( tf_{ri} \) from a snippet of each result \( r_i (i = 1, \ldots, m) \) by performing morphological analysis.
2. Calculate the cosine similarity between \( v_{\text{intent}} \) and each \( tf_{ri} \), and then re-rank each search result in descending order of its similarity value.

After the user has browsed some pages, this re-ranking method places into a leading position a search result that includes many terms the user looked at frequently during the browsing. It is expected that this will allow users to easily find pages that are relevant to their search intents from among many search results. If the system re-ranks the search results automatically, there is a possibility that the efficiency of a user’s information seeking will be reduced because the re-ranking was unexpected by the user. Therefore, we design the system to perform the re-ranking method only when a user specifies it explicitly.

4.1.2 Intent-based Query Suggestion

We use the estimated intent of a user for query suggestion as well as re-ranking. In a SERP, the system creates a tag cloud from the intent vector \( v_{\text{intent}} \) and displays it at the top of the SERP. The size of each term in the tag cloud reflects the intent score for the term. Thus, if an estimation of a user’s search intent is performed well, it can be said that the user wants to find much more information related to a term whose display size is large.

A user seeking information can take a panoramic view of her search intent by looking through terms in a created tag cloud. In addition, we design the system to work in the following way so that users can carry out iterative searches easily: when a user clicks any term in the tag cloud, the system issues to a search engine a new query composed of the original query and selected term, or only of that term.

It can be said that the user wants to find much more information for a term when the system judges it as interesting information for the user, the system emphasizes this segment. If not, the system makes it less noticeable.

Here a segment means a Web page element that contains meaningful information or important topics of the page, including news headlines on the top page of a newspaper website, and paragraphs in a certain article appearing on that website. We describe the details of each step in the rest of this section.

4.2.1 Extracting Segment Set

First, we describe a method to extract a set of segments from a Web page. We assume that segments have two features: (i) each segment is located on the same level in terms of the page’s tree structure, and (ii) segments cover a larger area of the page than elements that are not segments. We show below the flow of extracting a segment set \( S \) from a Web page on the basis of this assumption. In the following explanation, \( e_p \) indicates the parent element of all elements in the context candidate of the segment set and \( S_c \) is a set that contains a tuple \((S_i, n, s)\) as an element: \( S_i \) is a candidate for the segment set, \( n \) is the number of elements in the candidate, and \( s \) is the average display size of the elements in the candidate.

1. The system constructs a DOM tree of a Web page and assigns \( e_p \leftarrow e_{\text{body}} \), where \( e_{\text{body}} \) is the BODY element in the tree.
2. The system performs the following operations for the set of child elements of \( e_p \), expressed as \( E_{\text{children}} \):
   (a) The system groups elements in \( E_{\text{children}} \) into \( G_1, \ldots, G_m \) by element name. \( (E_{\text{children}} = \bigcup_{i=1}^n G_i) \)
   (b) For each \( G_i (i = 1, \ldots, m) \), the system re-assigns \( S_c \leftarrow S_c \cup \{(G_i, n_i, s_i)\} \), where \( n_i \) is the number of elements in \( G_i \), and \( s_i \) is the average display size of elements in \( G_i \).
3. For each \( e_{\text{child}} \in E_{\text{candidates}} \), the system goes back to Step 2 after performing re-assignment.
4. After finishing the above operations for all elements in the tree, the system calculates \( \text{SegSetScore}(n, s) \) for each tuple \( t = (G_i, n_i, s_i) \) in \( S_c \) using the equation shown below. Finally, the system outputs \( G_i \) as the segment set \( S_c \), where \( G_i \) is the first element of \( t \) whose score is the highest among \( S_c \).
   \[
   \text{SegSetScore}(n, s) = \log_2 c \cdot \log_2 s
   \]

4.2.2 Interest in Segments

Next we describe a method to estimate whether a user is interested in each segment in a Web page extracted by the method given above. In this paper, we consider that all the segments belong to one of the following two classes: \( P \), which refers to segments that interest the user, and \( N \), which refers to segments that do not interest the user. The system judges to which class each segment in the Web page belongs by using the following algorithm, which is a kind of Naïve Bayes classifier. Here, we define \( C \) as the set \((P, N)\). In this algorithm, the system uses as training data the number of segments that belong to each class, terms appearing in each segment in the class, and the frequency of each term. The flow of this algorithm is as follows:

1. For each term \( t_i \) in a segment \( s \), the system counts its term frequency in \( s \) (expressed as \( \text{Occur}(t_i) \)). In this way, the system calculates \( \text{Data}(s) = \{(t_i, \text{Occur}(t_i))\}_{i=1}^{n} \).
2. For each class \( c \in C \), the system calculates the probability \( \Pr(c|s) \) that a segment \( s \) belongs to \( c \) by the following equation, and judges the class to which \( s \) belongs through Equation (3):

\[
\Pr(c|s) \propto \Pr(c) \cdot \Pr(s|c) = \Pr(c) \prod_{(t_i, n_i) \in Data(s)} \Pr(t_i|c)^{n_i}
\]

\[
SegClass(s) = \arg \max_{c \in C} \Pr(c|s)
\]

where \( \Pr(c) \) is the ratio of the number of segments belonging to \( c \) to the number of all the segments in the training data, and \( \Pr(t_i|c) \) is the ratio of the frequency of \( t_i \) appearing in the segments belonging to \( c \) to the sum of the frequency of each term appearing in the segments belonging to \( c \).

3. After a user has finished browsing a Web page \( p \), the system calculates \( SegScore(s_i, p) \) defined as the following equation for each segment \( s_i \). If this value is higher than the threshold \( \sigma_s \), the system labels \( s_i \) as \( P \), otherwise as \( N \). Then the system adds the labeled \( Data(s_i) \) to the training data.

\[
SegScore(s, p) = \sum_{(t_i, n_i) \in Data(s)} \alpha_i \cdot TermScore_m(t_i, p)
\]

where \( \alpha_i \) is the ratio of the frequency of \( t_i \) appearing in the segments belonging to \( c \) to the sum of the frequency of each term appearing in the segments belonging to \( c \).

4.2.3 Emphasizing Segments

We propose three methods to change the display style of each segment in a Web page according to the estimated class of a user’s interest. The methods are:

- **Size Changing**: A segment belonging to \( P \) is emphasized by making its font size larger. In contrast, the system makes the font size smaller for a segment belonging to \( N \).
- **Color Changing**: If a segment belongs to \( N \), the system makes it less noticeable by changing the color of the text in the segment to one that is similar to the background color of the segment.
- **Folding**: If a segment belongs to \( N \), some parts of it are displayed to a user and other parts are not (e.g., the first few lines are displayed and others are not).

5. IMPLEMENTATION

We implemented a prototype system using C# to evaluate the effectiveness of our proposed methods. In implementing this system, we used the Web search API of Yahoo! JAPAN5 to obtain search results, and MeCab6 for morphological analysis. We also used the eye tracking device Tobii T60 and the software development kit Tobii SDK, which makes it possible to handle users’ eye movement data in real time6. We describe the details of the implemented system below.

5.1 Detecting Target Terms

To estimate a user’s search intent using the proposed method, we need to know which terms the user looks at when browsing Web pages. The system that we implemented detects these terms by embedding an original tag in each candidate term. The specific flow for detecting the term at which a user is looking is as follows:

1. When a user visits a new Web page \( p \), the system constructs a DOM tree from the HTML source code of \( p \), and performs the following preprocessing for each text node in the tree to extract candidate terms:
   
   (a) Perform a morphological analysis of the node’s value.
   (b) For each morpheme, if the stop word list includes the morpheme, the system creates a text node whose value is equal to that of the morpheme and inserts this node before the original node. Otherwise, the system creates a FONT element whose content is equal to that of the morpheme and assigns an identifiability class name to it so that the system can instantly recognize this element’s content as a candidate term. The system then inserts this element before the original node as well.
   (c) Remove the original node from the tree.

2. The system uses Tobii SDK to detect the point on the PC monitor at which the user is looking at a certain time interval \( \delta \), which is tentatively set to 10 ms considering the balance between the system load and the recall of users’ gaze events, and uses the method `GetElementFromPoint()` to work out which element is located on that point. It then performs the following step for the element:

   (a) If the class name of the element is equal to that which the system set during pre-processing, the system assumes that the user looked at the term \( t \) of the element, and increments the value of \( Look(t, p) \).

5.2 Parameters

In our proposed method, there are three parameters. The first parameter \( n \) is the number of terms that the system outputs as an estimated result of a user’s search intent. The second parameter \( \alpha \) is the weight coefficient for terms the user looked at in the most recently browsed Web page, and is used in Equation (2b). The last parameter \( \sigma_s \) is a threshold for classification according to whether or not the user was interested in each segment in a Web page after browsing the page.

If the number of terms expressing a user’s estimated search intent is too small, the recall for real search intent drops to a lower value. In contrast, if the number is too large the precision for real search intent drops instead. Therefore we tentatively set \( n \) to 30 in our implemented system considering the balance between recall and precision. Regarding \( \sigma_s \), we use the median of the scores of all terms, whose scores are calculated according to Equations (1a) – (1d), in the Web page that a user is currently browsing. As for \( \alpha \), we do not set a value for it because its ideal value could vary with different users or situations. Instead, we discuss its ideal value in the experimental results part of the next section.

5.3 Example of Execution

Figures 3 and 4 show execution examples of the system we implemented. The figures are screen shots of the system during the search and browsing processes respectively after a user has browsed a few Web pages. As shown in Figure 3, a tag cloud containing \( n(=30) \) terms that express a user’s estimated search intent is displayed at the top of the SERP. The user can use this tag cloud to re-rank search results or carry out a repeat search. Figure 4 shows how the system changes the display style of each segment in a Web page based on the user’s intent. In this example, the system uses the folding method, one of the three methods described in the previous section, for changing the display styles of segments.
6. EXPERIMENTS AND DISCUSSIONS

We conducted an experiment to evaluate the accuracy of search intents estimated by our proposed methods and to find which of them was the most effective.

6.1 Experimental Design

To confirm the effectiveness of our method to estimate users’ search intents, we prepared two baselines in this experiment: the TF method calculates an attention score for a term using only its term frequency, which means the system estimates search intents using the equation 

\[
\text{TermScore}_{\text{TF}}(t, p) = \text{TF}(t, p);
\]

however, the LOOK method calculates attention score using only the number of times the term was looked at, which corresponds to the equation 

\[
\text{TermScore}_{\text{LOOK}}(t, p) = \text{Look}(t, p).
\]

In the rest of this paper, we express each method in the form of Method_suffix, where Method corresponds to m and suffix to d in Intent \(d_m(t)\) respectively. When the system uses a uniform model for estimation, we express suffix as unif, otherwise (when the system uses a damping model), we express it as the value of weight coefficient \(\alpha\) in Equation (2b). In this experiment, we tested the five values 0.1, 0.3, 0.5, 0.7 and 0.9 as \(\alpha\). Thus, the number of variations for comparing the methods totals 36.

We asked eight volunteers to take part in this experiment, all of whom were male undergraduate or graduate students who were familiar with information seeking on the Web. Because the purpose of the experiment was to evaluate the extent to which the system can estimate the user’s search intent from their eye movements, we configured the system so as not to execute the support methods proposed in Section 4 (re-ranking, query suggestion and highlighting). The experimental procedure was as follows:

1. The system used Tobii T60 to perform calibration to identify the specific characteristics of the user’s eyes.
2. We explained the information seeking tasks with a specified search context to each user, and they worked on each task for 10 minutes. The used tasks are shown in Table 1.
3. Each method estimated up to 15 terms that expressed the user’s search intent from the user’s browsing Web pages and eye movement data during the task.
4. The system merged all terms output using all methods into a list in random order and presented the list to the user.
5. The user rated each term in the list on a three-point scale on the basis of whether the term was related to the user’s search intent or not: zero, one, and two meant irrelevant, somewhat relevant, and relevant, respectively.

6.2 Results

6.2.1 Precision of Estimated Intents

We show the result of nDCG@15 (normalized Discounted Cumulative Gain) for each task and the task total, and MAP@15 (Mean Average Precision) for the task total in the ranking of terms estimated by each method in Table 2. In calculating precision, we treated the terms having a rating value of two as relevant to a user’s search intent and the others as irrelevant. As shown in Table 2, nDCG and MAP obtained the highest values when using the proposed method nMLT_unif. Those of the DLT and nDLT methods were substantially lower than those of TF, LOOK, MLT and nMLT. However, the difference in accuracy between nMLT_unif and baselines was less than we expected.

For the damping model we tested the five values as a weight coefficient. The results show that the best weight depended on the method used. In terms of nDCG the best weight in each method was TF_0.1, LOOK_unif, MLT_0.5, nMLT_unif, DLT_unif, and nDLT_0.7.

When a user has a fixed search intent, as in the tasks used in this experiment, estimating the search intent using the nMLT method in a uniform model or damping model with a small weight coefficient will work well. On the other hand, when a user’s search intent is not fixed (or unclear) and changes during information seeking, the system can estimate this variable search intent flexibly by...
Table 2: Results of nDCG@15 for each method in each task and MAP@15 for each method in the task total. The five highest values are indicated by boldface for each column. In both tasks, the proposed method nMLT showed a good performance.

<table>
<thead>
<tr>
<th>Method</th>
<th>Task 1</th>
<th>Task 2</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>TF_unif</td>
<td>0.758</td>
<td>0.774</td>
<td>0.753</td>
</tr>
<tr>
<td>TF_0.1</td>
<td>0.800</td>
<td>0.796</td>
<td>0.798</td>
</tr>
<tr>
<td>TF_0.3</td>
<td>0.758</td>
<td>0.774</td>
<td>0.766</td>
</tr>
<tr>
<td>TF_0.5</td>
<td>0.727</td>
<td>0.762</td>
<td>0.745</td>
</tr>
<tr>
<td>TF_0.7</td>
<td>0.712</td>
<td>0.764</td>
<td>0.738</td>
</tr>
<tr>
<td>TF_0.9</td>
<td>0.697</td>
<td>0.752</td>
<td>0.725</td>
</tr>
<tr>
<td>LOOK_unif</td>
<td>0.806</td>
<td>0.763</td>
<td>0.784</td>
</tr>
<tr>
<td>LOOK_0.1</td>
<td>0.807</td>
<td>0.732</td>
<td>0.770</td>
</tr>
<tr>
<td>LOOK_0.3</td>
<td>0.793</td>
<td>0.759</td>
<td>0.776</td>
</tr>
<tr>
<td>LOOK_0.5</td>
<td>0.781</td>
<td>0.784</td>
<td>0.782</td>
</tr>
<tr>
<td>LOOK_0.7</td>
<td>0.769</td>
<td>0.783</td>
<td>0.776</td>
</tr>
<tr>
<td>LOOK_0.9</td>
<td>0.761</td>
<td>0.749</td>
<td>0.755</td>
</tr>
<tr>
<td>nMLT_unif</td>
<td>0.823</td>
<td>0.752</td>
<td>0.788</td>
</tr>
<tr>
<td>nMLT_0.1</td>
<td>0.838</td>
<td>0.758</td>
<td>0.798</td>
</tr>
<tr>
<td>nMLT_0.3</td>
<td>0.818</td>
<td>0.788</td>
<td>0.803</td>
</tr>
<tr>
<td>nMLT_0.5</td>
<td>0.800</td>
<td>0.817</td>
<td>0.809</td>
</tr>
<tr>
<td>nMLT_0.7</td>
<td>0.785</td>
<td>0.813</td>
<td>0.799</td>
</tr>
<tr>
<td>nMLT_0.9</td>
<td>0.770</td>
<td>0.786</td>
<td>0.778</td>
</tr>
</tbody>
</table>

Table 3: Mean unique rate (with standard deviation) for users in each task. The three highest values are indicated by boldface for each column.

<table>
<thead>
<tr>
<th>Method</th>
<th>Task 1</th>
<th>Task 2</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>TF_unif</td>
<td>0.54 (0.24)</td>
<td>0.35 (0.32)</td>
<td>0.44 (0.23)</td>
</tr>
<tr>
<td>TF_0.1</td>
<td>0.57 (0.22)</td>
<td>0.40 (0.24)</td>
<td>0.48 (0.17)</td>
</tr>
<tr>
<td>LOOK_unif</td>
<td>0.75 (0.22)</td>
<td>0.57 (0.26)</td>
<td>0.66 (0.22)</td>
</tr>
<tr>
<td>LOOK_0.5</td>
<td>0.86 (0.12)</td>
<td>0.70 (0.32)</td>
<td>0.78 (0.14)</td>
</tr>
<tr>
<td>nMLT_unif</td>
<td>0.79 (0.08)</td>
<td>0.65 (0.15)</td>
<td>0.72 (0.10)</td>
</tr>
<tr>
<td>nMLT_0.1</td>
<td>0.79 (0.15)</td>
<td>0.68 (0.09)</td>
<td>0.74 (0.10)</td>
</tr>
</tbody>
</table>

method TF, which used only term frequencies.

We describe the characteristics of each method giving example terms output by the system using the method in Task 2. The system using the baseline method TF output many abstract or generic terms such as “shopping”, “hotel” or “hot spring” to every participant. In contrast, when using the nMLT and LOOK methods, the system tended to output concrete or specific terms that differed among participants. For example, the system output names of historic places to one participant and terms related to whale watching to another.

These results suggest that we can estimate each user’s individual search intent by taking into account the extent to which the user has looked at certain terms, even if their search tasks are the same, as was the case in this experiment.

6.3 Discussions

In this section, we first discuss a problem with our proposed methods in light of the experiment results. By checking the Web pages that each participant had browsed in the experiment and the log files in which her rating data was recorded, we found that our proposed methods had not worked well when participants had often browsed pages that contained Adobe Flash™ content or many images such as maps and smart phones images. In the experiment, participants tended to look more at Flash content or images than at textual parts of those pages. Because our proposed method uses terms a user looked at when browsing Web pages, there is a high possibility that the estimation of the user’s search intent will be inaccurate. This may well lead to experiment results in which there is no great difference in accuracy between our proposed method and baseline methods.

From here, we also discuss the proposed methods of supporting users’ search and browsing processes during information seeking from the viewpoint of efficiency, while taking into account the impression of the prototype’s actual behavior. We use the nMLT_unif method to estimate search intents when testing the prototype because it was shown that this method was the most effective of our proposed methods in the experiment described above.

To support users’ search process in information seeking, we have proposed two methods. One is re-ranking search results and the other is suggesting intent-based queries, both of which use a term set estimated as relevant to a user’s search intent. Generally, a snippet in a search result only partially contains Web page contents corresponding to the result. Therefore, a search result whose snippet does not contain information that meets a user’s search intent is re-ranked lower than the original position even if the information appears on the corresponding Web page. In this case, the user spends more time to reach the page, or sometimes cannot visit there at all, and her information seeking becomes less efficient. On the other hand, users can reflect their intents to the search system interactively and dynamically by selecting intent-based queries. For

http://www.adobe.com/products/flash/
novice users in particular, modifying a search query to obtain better search result is hard and takes time. Therefore, it is expected that this method will help these users to find information efficiently.

To facilitate the users’ browsing process, we have proposed a method that emphasizes the parts of a Web page that meet their search intents. When testing the prototype behavior, this method worked well for some pages but not for others. It worked well for the pages whose structures were composed of multiple lists, such as the top page of a news site or a page that lists information on many recipes. In these pages, users can quickly obtain the information they want because the system emphasizes the segments that contain it. However, this method did not work well for the pages which had complicated structures. For such pages, using the size changing method or the color changing method to emphasize segments sometimes failed, partly because of the preference order of the page’s style sheet even if extracting the segment sets worked well. Though the folding method worked relatively well in most cases, it was disadvantageous when the interest estimation for segments failed, because relevant information was also hidden from the user. Therefore, we need to improve our method of emphasizing segments in the future.

7. CONCLUSIONS AND FUTURE WORK

This paper has described two main contributions to the area of information seeking support. One is the proposal of methods for estimating users’ search intents from their eye movements while browsing Web pages. Here it was found that the nMLT method was the most effective since it estimates the unique search intent of each user even if the search tasks are the same among multiple users. The other is the proposal and implementation of a system that helps users to obtain information efficiently in both the search and browsing processes of information seeking by estimating their search intents.

In this paper, we estimate users’ search intents by using the term frequencies of terms the users look at, and the order in which they browse Web pages. In the estimation process, however, there are many other features that can be used for supplementing our proposed estimation method (e.g., the order of the terms the user looks at and the positions in which these terms appear). Furthermore, we use eye movements only for textual parts of Web pages to estimate users’ search intents; however, there are other Web page components at which users often look, such as images and movies.

In future work, we intend to extend our method by applying the features described above. We will then use the extended method to attempt to determine the extent to which our system can estimate users’ search intents from their eye movements. We also intend to use eye movement patterns to determine whether we can estimate more semantic intents of users, such as comparing and re-finding. In addition, we plan to use estimated search intents to generate intent-based thumbnails of Web pages to support users’ search processes as discussed in [24].

8. ACKNOWLEDGMENTS

This work was supported by Grant-in-Aid for Young Scientists (A) (#23680006) from JSPS.

9. REFERENCES


