Adapting to the User's Internet Search Strategy

Jean-David Ruvini

e-lab Bouygues SA
1avenue Eugène Freyssinet, 78061 St Quentin en Yvelines, France
jdruvini@bouygues.com
http://e-lab.bouygues.com

Abstract. World Wide Web search engines typically return thousands of results to the users. To avoid users browsing through the whole list of results, search engines use ranking algorithms to order the list according to predefined criteria. In this paper, we present Toogle, a front-end to the Google search engine for both desktop browsers and mobile phones. For a given search query, Toogle first ranks results using Google's algorithm and, as the user browses through the result list, uses machine learning techniques to infer a model of her search goal and to adapt accordingly the order in which the results are presented. We describe preliminary experimental results that show the effectiveness of Toogle.

1 Introduction

Today World Wide Web search engines like Google, Altavista or AllTheWeb among others search documents for user specified keywords and return a list of document snippets (called results) where the keywords were found. With the vast amount of online information sources available on the web, this list typically contains thousands of results.

To deal with this amount of information, users have two choices : to browse through the list of results which is often a tedious task or to repeatedly select new keywords and query the search engine until they estimate they have selected the right keywords and... it's worth browsing the list! This problem is particularly critical on small devices offering web browsing (like mobile phones) which generally have limited display and input capabilities, making browsing a very difficult task.

To limit the browsing process and to help users quickly locate relevant documents, search engines use ranking algorithms to order the list according to certain criteria, presenting relevant documents in the top of the list and less relevant documents below. Criteria can include the number of keywords matched, proximity and frequency of keywords, document length, number of links made to a document, number of times a document is accessed from a result list and other factors. Alternatively, [8] proposed to learn the ranking function from clickthrough data (the logfile of results the users clicked on) of a group of users.

These ranking algorithms have two major limitations. First, they are global to all users and do not adapt to a specific user and to a specific search session or query.

Second, they are exclusive in that they cannot be combined with - or, more precisely - benefit from other ranking algorithms.

In this paper, we present ongoing research efforts to overcome these limitations that leaded to the implementation of Toogle, an intelligent front-end to the Google search engine. The Toogle hypothesis is that a typical browsing session over a list of results where the user clicks on some results and ignore some others, conveys information about his search goal. Toogle first ranks search results using Google's algorithm and, as the user steps through the result list, uses machine learning techniques to infer a model of the user's search goal from clickthrough data and to adapt accordingly the order in which results are presented. Toogle is available for both desktop computer and wireless device web browsers but has been mainly designed to improve usability of small devices.

This paper is structured as follows: First, we present Toogle's interface. Second we explain how it works. Third, we report preliminary experimental results for wireless device web browsers. Fourth, we discuss related research. The paper concludes with a summary of contributions and plans for future research.

2 Toogle: an Intelligent Front-end to Google

As we mentioned above, Toogle, as Google, is accessible from desktop internet browsers and mobile phone offering web browsing, namely i-mode mobile phone. First introduced in Japan in February 1999 by NTT DoCoMo, i-mode is one of the world's most successful services (currently more than 41 millions subscribers in Japan) offering wireless World Wide Web (WWW) access and e-mail from mobile phones. It was launched in France by Bouygues Telecom by November 2002.

For both their desktop and i-mode versions there is almost no difference, from the user perspective, between Toogle and Google. Toogle is accessed from a web browser and its homepage is a clone of Google's homepage. To query Toogle, the user just has to type in some keywords and to press the "Google Search" button exactly as he uses to do with Google. Similarly, Toogle's results are displayed in the same format as Google's results, grouped into pages that can be accessed individually from links located at the bottom of the current result page. Figure 1(a)[1] shows the homepage of Toogle i-mode. Figure 1(b) and 1(c) show the result page of Toogle i-mode for the query "korean restaurant new york". Figure 1(b) presents the first lines of that result page; on Figure 1(c) the user has scrolled down this page to review the first result.

There are few differences between desktop and i-mode versions of Toogle (respectively Google). The main noticeable one is the number of results displayed on each result page. Whereas desktop browser version displays 10 results per page, i-mode version displays only 5 results per page. In the following we use "page" to refer with no distinction to a list of 5 or 10 results.

The main reason why we have chosen to emulate Google rather than any other search engine is that Google can be accessed programmatically using a web service

[1] The i-mode emulator shown in Figure 1 is WAPAG (see http://www.wapag.com/).

programming toolkit (released by the Google corporation). This toolkit makes possible a simple proxy architecture for Toogle. When the user initiates a search query, the query is first sent to Toogle and then forwarded to Google. The results returned by Google are first processed by Toogle which then display them in the user's browser. Similarly, when the user clicks a proposed result to examine the corresponding document, the document request is processed by Toogle. This architecture allows Toogle to record the list of results presented to the user and which results she actually examined. It uses this information in the adaptation process described in the next section.

(a) (b) (c)

Fig. 1. The homepage (a) and a result page (b) and (c) of the i-mode versions of Google and Toogle.

3 Adapting to the User's Search Strategy

Toogle is based on the observation that there is not an exact mapping between search queries and search goals. Since result lists typically present a lot of different information, users may type the same query for accessing different documents and, as a consequence, exhibit different browsing behaviors for that query. In other words, the browsing behavior of the user for a given query depends on his search goal (or interest).Toogle elaborates on this observation to infer, for a each search query, a model of

the user interest and to adapt the list of results accordingly. More precisely, it first presents to the user the list of results proposed by Google with no modification. As the user browses through the list, Toogle tries to infer a model of his search goal from his browsing actions. It then uses this model to reorder the list of results the user has not yet considered in order to present most relevant results first. In the following we explain how Toogle builds a model of the user's interest and how and when it uses this model to reorder the list.

3.1 Building a model of the user's interest

Toogle's goal is to build, for a given user U, a search query Q and its corresponding list of results L, the following target function:

$$ResultInterest_{U,Q,L} : Result \rightarrow \{0,1\}$$

Given a result, the value of $ResultInterest_{U,Q,L}$ is interpreted as a measure of the interest the user has in it. A value of 1 indicates that he has a strong interest in it and is likely to click it to examine the corresponding document, whereas a value of 0 indicates that he is not likely to examine the document.

Toogle uses a machine learning approach to build the $ResultInterest_{U,Q,L}$ function. More precisely, because results are textual data, it employs a text classifier to learn it. This approach requires to identify positive examples of results the user is likely to click and negative examples i.e. results he is not likely to click. Remember that Toogle (as Google) groups results into pages. For a given search query, Toogle considers, within all the result pages the user has visited, results he has clicked as positive examples and the results he has not clicked as negative examples. Suppose for example that, using an i-mode web browser (5 results per page), the user has clicked the results ranked in position 3, 5 (first page) and 8 (second page) in the list. Toogle considers results 3, 5 and 8 as positive examples and results 1, 2, 4, 6 and 7 as negative examples. The reason why Toogle uses the document snippets (results) rather than the documents themselves is straightforward: whereas a click on a result provides an explicit indication of the interest the user has in it, it is difficult to draw any conclusion about the content of a document from the fact that the user read it. Also, working with results is possible even if some documents are not available ("page not found" status).

Once it has identified the examples, Toogle represents them in a text classifier exploitable form, namely the bag-of-words representation. It this representation, a result is encoded as a feature vector, with each feature indicating the presence or absence of a word. Table 1 shows an example of a Google result for the search query "french food". A result is made of five main parts : the title of the document referenced by the result, a snippet of the document where the keywords composing the search query were found, a summary of the document, its category and its url. Some of these parts may be absent from the result and it is often the case for the summary and the category. Toogle makes no difference between the five parts of the results in the encoding.

Finally, if Toogle could identify at least one positive example and one negative example, it invokes the text classifier to build the model of the user's interest. Of course, result pages the user has not visited are not taken into account in the learning process (no example is extracted from them).

Since Toogle only uses document snippets, the problem has a small dimensionality (the examples contain few words) and Toogle can learn very quickly with most of the algorithms classically used for text classification. As a consequence, it can build a model or revised a learned model, whenever it identifies a new positive example that is to say whenever the user clicks a result. In its current implementation, Toogle's learning algorithm is the Support Vector Machine [16] which has been shown to achieve excellent performances on textual data [4].

The learned $ResultInterest_{U,Q,L}$ function can be seen as a model of the user's interest relatively to the search query Q and the corresponding result list L. It is a short term model since it is valid only in the lifetime of the query Q. In the next section we explain how this model is used.

Table 1. A Google result for the search query "french food"

French Food and Cook : **French** Dinner **French Food** and Cook : the authentic **French** cuisine site. Best typical **French** recipes and advice on **French** cooking. **... French Food** and Cook : Home Page. **...** Description: Complete information on how to organize and cook a dinner, including recipes. Category: Home > Cooking > World Cuisines > European > French www.ffcook.com/Cadres/Dinner.htm - 21k - Cached - Similar pages

3.2 Reordering the result list

To avoid disturbing the user by modifying the ranking of results on a page she has visited, Toogle reorders only pages she has not visited yet. The reordering occurs whenever she requests a new result page and involves three steps.

First, Toogle queries Google for more results and loads them in memory. However, it does not load a single result page but several pages. The number of pages loaded determines the number of results Toogle is able to take into account in the adaptation process. Since the results are obtained from Google, the loaded list is ordered according to the Google's ranking criteria, from the most relevant to the least relevant result.

Second, it uses the text classifier and the learned user's model to classify the loaded results. It then reorders the result list according to the predicted label : every result classified as a positive example is considered as a potentially interesting one and is moved to the top of the list (high ranks). Toogle locally preserves Google's ranking in that it does not perturb the relative ordering of positive examples and the relative ordering of negatives examples : if results r is ranked higher than r' in Google's ordering and r and r' have the same predicted label then r is ranked higher than r' in Toogle's ordering.

Finally, Toogle displays the 5 (i-mode) or the 10 (desktop browser) most relevant results to the user, presenting the most relevant at the top of the result page and the least relevant at the bottom. Suppose for example that, using an i-mode web browser (5 results per page), the user has requested the result page number 3 (results 11 to 15).

Suppose also that Toogle has predicted that results 12, 17 and 23 (according to Google's ordering) may be of interest to the user. Toogle will display a result page

presenting (from top to bottom) results 12, 17, 23, 11 and 13. Results 11 and 13 are negative examples but since they are the negative examples with the highest rank in the Google's ordering they are presented before any other negative example.

Of course, when the user submits a new search query to Toogle, it forgets everything it has previously learned and start learning a new model specific to the new query and the new search goal.

The next section presents preliminary experimental results.

Table 2. Toogle's predictive accuracy and browsing gain for eight queries

		Q1	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**	**Q7**	**Q8**	Mean
Pos. ex.		5	4	7	5	5	11	3	4	5.50
Neg. ex.		10	16	16	10	14	7	9	8	11.25
Predictive accuracy	i-mode									
	NBC	22	93	77	50	50	58	71	14	54.48
	SVM	100	80	100	80	72	54	72	100	82.25
	Desktop									
	NBC	100	89	61	60	44	71	100	50	71.92
	SVM	40	100	89	100	78	75	50	100	79.00
Browsing gain	i-mode									
	Page	1	0	3	1	2	1	0	1	1.125
	Ratio	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.25

4 Experiment

The goal of the experiment presented here is to show that Toogle facilitates browsing through a list of search engine results by reducing user's efforts and information overload. More precisely, it is to answer the two following questions :

1. Can text classification algorithms successfully learn the user's interest from click-through data?
2. Does Toogle reduce the number of result pages users have to browse?

We conducted a preliminary experiment by submitting to two users eight search tasks using Google on an i-mode emulator. To avoid query formulation bias, the eight search queries (and the corresponding search goals) were given to the users. One of the users was a researcher of our laboratory while the other was a female computer novice with no prior knowledge about i-mode. For the eight queries, the two users exhibited the same browsing behavior (they clicked the same results).

Using a proxy architecture similar to the Toogle's architecture, we recorded users' browsing actions. Using these records we then simulated the users to feed Toogle with their browsing actions and analyzed Toogle's behavior. The results of this experiment are presented in Table 2.

Column labels Q1 to Q8 refer to the eight queries used in the experiment. The table comprises three parts. The first part, composed of the first two rows, lists the number of positive (first row) and negative (second row) examples of each query. For a given query, the positive examples are all the results the user clicked during the browsing session and the negative examples all the results (up to the last positive example) she did not click.

The second part gives the predictive accuracy of Toogle for both i-mode and desktop browsers for two learning methods: the Support Vector Machine (SVM) and the Naïve Bayes Classifier (NBC) [6]. We used the SVM-Light [7] implementation for the SVM, and the Bow toolkit [14] for NBC.

To evaluate predictive accuracy, we split the example set in two sets : a training set and a test set. For i-mode, the training set was constituted of the five first examples (according to Google's ranking), and of the ten first examples for desktop browser. The training set was used to learn a user's model and the test set to evaluate the performance of this model. The predictive accuracy is the percent of examples of the test set which label ("positive" or "negative") was correctly predicted. This Table shows that Support Vector Machines achieve good performances (around 80%) for both i-mode and desktop browsers. This is an interesting finding (which requires further investigation) because it suggests that this method is able to learn a good user's model even when few training data are available which is often the case in the area of User Modeling and Adaptive Interfaces. This is particularly true for the i-mode version where the training set contained only five examples. Note that, although SVM are known to be theoretically longer to train, we observed no significant difference between NBC and SVM training time. As we mentioned earlier this is due to the fact that the problem has a small dimensionality (few examples, few features).

The third part proposes an evaluation of the browsing gain for the end user of Toogle, in term of "page gain" and of gain "ratio" on the basis of the SVM algorithm. The page gain is defined as the difference between Google's ordering and Toogle's ordering in the number of result pages the user has to visit to be able to click all the interesting results (i.e. the positive examples). For instance, suppose that, on i-mode, the last result the user has clicked is ranked in position 20 by Google and in position 7 by Toogle. Since an i-mode result page presents 5 results, result 20 appears on Google's page 4 whereas result 7 appears on Toogle's page 2. In that case, the page gain is 2.

The page gain must be considered with respect to the maximum possible gain. The gain ratio measures how far from this optimal Toogle reordering is. Let G_T be Toogle's page gain and G_{Opt} be the maximum possible page gain for a given query. The gain ratio is defined as:

$$Ratio = 1 - \frac{G_T}{G_{Opt}}$$

Concerning i-mode, results presented in Table 2 are very encouraging. They show that there is a real gain in using Toogle (page gain is optimal) for more most of the queries (6 out of 8).

Since Toogle has been mainly designed to improve usability of i-mode WWW search engines, the queries used in the experiment were strongly biased in disfavor of desktop browsers. More precisely, since the search tasks submitted to the users required them to examine less than 21 results for most of the queries (the only exception

is query Q3), they had to visit only 2 result pages. Because in that case the page gain can only be 0, results are not presented for desktop browsers in Table 2. However, because Toogle exhibits a high prediction accuracy, we believe that further experiments will show that its effectiveness on i-mode generalizes to desktop browsers.

Another interesting fact that does not appear in Table 2 is that Toogle is very conservative. There are two observations that support this claim. First, Toogle incorrectly ranked a result lower than it was in Google's ordering only once. It occurs for query Q6 on desktop browser: result ranked 16 by Google was ranked 17 by Toogle. Second, there is a particular case where Toogle's prediction errors do not penalize the user: when it predicts the same label for all the examples of the test set (all positive or all negative). In that case Toogle's reordering is identical to Google's ordering. This occurs for queries Q2 and Q7 where gain ratio is equal to 100%. In other words, when Toogle is unsure about the user's interest, it does not reorder the results.

The results of the preliminary experiment presented in this section show that Toogle is clearly effective on i-mode browsers where it decreases the users' information overload. The next section details related systems.

5. Related work

Besides Toogle illustrates the application of adaptive hypermedia techniques [3] to search engines, works related to Toogle can be grouped in four categories: browsing assistants, small device usability, search engine optimization and relevance feedback.

Browsing assistants like Letizia [10], WebWatcher [9], SurfLen [5] or PageGather [15] are similar to Toogle in that they recommend yet-unseen documents. As opposed to WebWatcher, SurfLen and PageGather who incorporate knowledge about other users, Letizia (like Toogle) estimates the visitor interest based solely on its actions (links he followed, documents he bookmarked, etc.). Whereas these agents act as guides and explicitly recommend documents to the user, Toogle is invisible and unobtrusive and, as a consequence, has been integrated into a small device with limited display capabilities.

Concerning small device usability, [1] proposed an algorithm for automatically suggesting shortcut links in real time to users of wireless PDAs or cell phones. The algorithm finds shortcuts by using a model of web visitor behavior learned on server access logs. They have shown that using a mixture of Markov models, their algorithm can save wireless visitors more than 40% of the possible link savings. This value can be put in parallel with Toogle's gain ratio of 25% which means that Toogle saves users near 75% of the possible information overload saving.

Several techniques have been proposed to optimize search engines. [12] built (from search engine access logs) Bayesian networks to model the dynamics of users' search activities. They have shown that these models can be used successfully to infer the probability of the user's next action, the time delay before taking the action and the user's informational goal. [2] used clickthrough data for identifying clusters of similar queries and similar URLs. More recently, [13] proposed to use common sense reasoning to translate any search query into an effective query. Closer to Toogle, [8] pro-

posed to use clickthrough data to optimize the order in which results are presented to the user. However, the proposed approach differs from our approach in that its goal is not to adapt the ordering to a single user within the context of a specific search query, but to learn a ranking algorithm from the browsing habits of a group of users, independently of their queries. Another advantage of our approach over Joachims's approach is that Toogle elaborates on an existing ordering and can benefit from any search engine ranking algorithm (including Joachims's algorithm) whereas Joachims's learned algorithm is used in replacement of any other ranking algorithm.

The closest work to Toogle [17] comes from relevance feedback research. Relevance feedback in document retrieval systems is an iterative process wherein the set of retrieved documents is updated based on the user's feedback. The update phase typically consists in augmenting the user's query with terms extracted from documents marked as relevant by the user. In [17], the authors investigate implicit relevance feedback where relevance is inferred from the user's behavior. As Toogle, their system first displays an ordered list of documents summaries and re-orders the list as the user interacts with it. As opposed to Toogle it bases its implicit feedback model around summary viewing time and uses term extraction techniques to infer user's search goal. Also, the authors do not address on search engine optimization and browsing gain, particularly in the context of small devices.

6. Conclusion

In this paper, we presented Toogle, a front-end to the Google search engine, that reduces search engine users information overload by adapting to the user's inferred search goal the order in which search results are presented. Toogle first ranks results using Google's algorithm and, as the user steps through the result list, uses machine learning techniques to build a model of his search goal and reorders the list of results accordingly. A preliminary experiment shows that Toogle performs well in practice on wireless phones offering WWW browsing, reducing substantially the number of result pages the user has to visit. Toogle is unobtrusive and, since it elaborates over an existing ordering and locally preserve this ordering, can benefit from any ranking algorithm.

Although the experimental results presented in this paper are encouraging, we need to evaluate Toogle in depth through extensive experiments and user tests. We are currently carrying these experiments. There are several possible directions of research to improve Toogle. One direction is to increase the informative value of the text composing the search results in order to facilitate the learning of the user's search goal model. For example, this can be done by adding a field "keywords" to the result description, containing the most informative words of the document referred by the result in respect with the other documents proposed in the result page. Since result pages typically presents less than a dozen of results, these keywords could be extracted very efficiently using classical term extraction. Another direction is to "take advantage of the user think time" [11] to refine the learned model and increase the predictive per-

formance of Toogle. Several techniques have been proposed in the area of machine learning (ensemble methods, re-sampling) to achieve this goal.

References

1. Anderson C. R, Domingos P. and Weld D. S. Adaptive web navigation for wireless devices. In Proceeding. of the International Conference on Artificial Intelligence, Morgan Kaufmann, (2001).
2. Beeferman D. and Berger A. Agglomerative clustering of a search engine query log. In Proceeding. of the International Conference on Knowledge Discovery and Data Mining, ACM, (2000).
3. Brusilovsky, P. Methods and Techniques of Adaptive Hypermedia. User Modeling and User-Adapted Interaction 6(2-3), (1996), 87-129.
4. Dumais S., Platt J., Heckerman D. and Sahami M. Inductive Learning Algorithms and Representations for Text Classification. Proceedings of the International Conference on Information and Knowledge Management, ACM, (1998), 148-155.
5. Fu X., Budzik, J. and Hammond K.J. Mining navigation history for recommendation. Proceedings of the International Conference on Intelligent User Interfaces, ACM, (2000).
6. Good I.J. The Estimation of Probabilities: An Essay on Modern Bayesian Methods. MIT Press, (1965).
7. Joachims T. SVM-Light Support Vector Machine; (1999). http://svmlight.joachims.org/.
8. Joachims T. Optimizing Search Engine using Clickthrough Data. In Proceedings of the ACM Conference on Knowledge Discovery and Data Mining, ACM, (2002).
9. Joachims T., Freitag D. And Mitchell T. WebWatcher: a Tour Guide for the World Wide Web. In Proceeding of the Fifteenth International Conference on Artificial Intelligence, Morgan Kaufmann, (1997).
10 Lieberman H. Letizia: An Agent That Assists Web Browsing. In Proceeding of the Fifteenth International Conference on Artificial Intelligence, Morgan Kaufmann, Montreal, Canada, (1995).
11. Lieberman, H. Autonomous Interface Agents. In Proceeding of the International Conference on Human Computer Interaction, ACM, (1997).
12. Lau T. And Horvitz E. Patterns of Search: Analyzing and Modeling Web Query Refinement. In Proceedings of the International Conference on User Modeling, ACM, (1998).
13. Liu H., Lieberman H., Selker T. GOOSE: A Goal-Oriented Search Engine With Commonsense. In Proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web Based System, LNCS 2347, p. 253, (2002).
14. McCallum A. K. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering, 1996. http://www.cs.cmu.edu/~mccallum/bow.
15. Perkowitz M. And Etzioni O. Towards adaptive web sites: conceptual framework and case study. Artificial Intelligence Journal, 118(1-2), (2000).
16. Vapnik V. The Nature of Statistical Learning Theory. Springer-Verlag, New-York, (1995).
17. White R. W., I. Ruthven and J. M. Jose. Finding Relevant Documents Using Top Ranking Sentences : An Evaluation of Two Alternative Schemes. In Proceedings of the 25th International Conference on Research and Development in Information Retrieval, ACM, 2002.