Transactional Query Identification in Web Search

In-Ho Kang

Computing LAB,
Samsung Advanced Institute of Technology
inho97.kang@samsung.com

Abstract. User queries on the Web can be classified into three types according to user's intention: informational query, navigational query and transactional query. In this paper, a query type classification method and Service Link information for transactional queries are proposed. Web mediated activity is usually implemented by hyperlinks. Hyperlinks can be good indicators in classifying queries and retrieving good answer pages for transactional queries. A hyperlink related to an anchor text has an anticipated action with a linked object. Possible actions are reading, visiting and downloading a linked object. We can assign a possible action to each anchor text. These tagged anchor texts can be used as training data for query type classification. We can collect a large-scale and dynamic train query set automatically. To see the accuracy of the proposing classification method, various experiments were conducted. From experiments, I could achieve 91% of possible improvement for transactional queries with our classification method.

1 Introduction

The Web is rich with various sources of information. Therefore, the need behind a user's query is often not informational. Classic IR that focuses on content information cannot satisfy various needs [3]. Fusion IR studies have repeatedly shown that combining multiple types of evidence such as link information and url information can improve retrieval performance [10]. However, it is not easy to make a general purpose method that shows good performance for all kinds of need. There are studies that try classifying users' needs and solving each category with a different method [5]. [2] showed that users' queries can be classified into three categories according to the intention of a user.

- Informational Query
- Navigational Query
- Transactional Query

Users are interested in finding as much information as possible with informational queries. For example, "*What is a prime factor?*" or '*prime factor*' is an informational query. Its goal is finding the meaning of '*prime factor*'. Contrary to Informational queries, users are interested in navigating a certain site

with navigational queries. For example, "*Where is the site of Johns Hopkins Medical Institutions?*" or '*Johns Hopkins Medical Institutions*' is a navigational query. The goal of this query is finding the entry page of '*Johns Hopkins Medical Institutions*'. Users are interested in finding a document that offers the service described in a transactional query. For example, "*Where can I find Beatles' lyrics?*" or '*beatles lyrics*' is a transactional query. Users do not desire to learn about lyrics of Beatles' songs, but simply a desire to view the lyrics themselves [9].

Previous studies for query analysis employed handcrafted rules and used a natural language query to infer the expected type of an answer [6]. These rules were built by considering the first word of a query as well as large patterns of words identified in a query. For example, query "*Where was Babe Ruth born?*" might be characterized as requiring a reply of type *LOCATION*. While previous studies focused on informational queries, nowadays we should consider not only informational queries but also navigational queries and transactional queries for Web search. Previous methods need large-scale training data to extract rules and patterns for analyzing query [6]. However, there is a case in Web search when we cannot analyze a query with extracted rules and patterns. For example, two queries, "*Where is Papua New Guinea?*" and "*Where is Google?*" have the same sentence patterns. However, '*Google*' has special meaning on the Web. User wants to visit '*Google*' site on the Web. The intention of the latter query is not informational but navigational. To make matters worse, the properties of words are changed. For example, Korean query '*EOLJJANG*'[1] cannot be analyzed without the usage information on the Web. '*EOLJJANG*' was formerly an informational query. As an informational query, a possible answer document may contain a definition such as "*EOLJJANG means a good looking person.*" Nowadays, '*EOLJJANG*' is a transactional query. What a user wants is finding and downloading pictures of '*EOLJJANG*'s. We may use a dictionary to find the property of '*EOLJJANG*', like previous query analysis methods. However it is not feasible to contain all new words and maintain their changes of properties.

[5] automatically classified queries into informational queries and navigational queries. They also proposed dynamic combination of content information, link information and URL information according to a query type. In this paper, I extend their classification and combining method to include transactional queries. Each hyperlink related to an anchor text has an anticipated action with a linked object. Possible actions include reading, visiting and downloading a linked object. These tagged anchor texts can be used training data for a query classification module. We can collect a large-scale and dynamic training data automatically.

The structure of this paper is as follows. In section 2, previous classification methods are briefly explained. In section 3, a query type classification method for classifying queries into informational, navigational and transactional

[1] the Romanization of Korean word

is presented. In section 4, Service Link information is presented for transactional queries. Various experiments are conducted to show the performance of the proposing classification and combining method, in section 5. Conclusion is followed in section 6.

2 Related Work

2.1 Query Taxonomy

[2] and [9] randomly selected queries from a commercial search engine's query logs. They manually classified queries and showed the percentages of each query type. Although their hierarchies are not the same, top-level hierarchies are the same. From their results, over 40% of queries were non-informational. They all insisted the importance of processing transactional queries. [9] showed that navigational queries appear to be much less prevalent than generally believed. This implies that search engines should consider transactional queries to cover more queries.

Table 1. Percentage of Query Type

Query Type	Informational	Navigational	Transactional
Broder (user survey)	39%	24.5%	36%
Broder (log analysis)	48%	20%	30%
Rose (test set1)	60.9%	14.7%	24.3%
Rose (test set2)	61.3%	11.7%	27%
Rose (test set3)	61.5%	13.5%	25%

2.2 Automatic Query Type Classification

An entry page of a site usually does not have many words. It is not an explanatory document for some topic or concept, but a brief explanation of a site. We can assume that site entry pages have the different usage of words. If we find distinctive features for site entry pages, then we can classify a query type using keywords of a query. [5] proposed a method for classifying queries into informational queries and navigational queries. They divided a document collection into two sets whether the URL type of a document is '*root*' type or not. The URL of a '*root*' type document ends with a domain name like 'http://trec.nist.gov'. A document collection that includes only 'root' type documents represents navigational queries and other documents represent informational queries. If a given query's some measures in two collections show large difference, then we can assume the type of a query. Four classifiers were used to determine the type of a query. Four classifiers were as follows.

– Distribution of terms in a query

– Mutual Information of each term in a query
– Usage rate of query term as anchor texts
– POS information

[5] also showed the effects of each information and retrieval algorithm in Web search according to a type of a user's query. For navigational queries, combining link information like PageRank and URL information like the depth of directory improved the retrieval performance of a search engine [7]. However, for informational queries, it degraded the retrieval performance. In addition, retrieval algorithms such as TFIDF and OKAPI also show different effect in Web search. In this paper, I extend this classification in order to include transactional queries. In addition, I propose useful information for transactional queries.

3 Classifying Query Types

3.1 Preparation for Classification Model

Some expression can be used to tell the type of a query. For example, '*winamp download*' is a transactional query and '*the site of SONY*' is a navigational query. We can assume the types of two queries with cue expressions (e.g. '*download*' and '*the site of*').

Query Q is defined as the sequence of words. Punctuation symbols are removed from the query. For example, "*What is a two electrode vacuum tube?*" is expressed as follows.

$$Q = (what,\ is,\ a,\ two,\ electorde,\ vacuum,\ tube) \tag{1}$$

[5] used only keywords of input queries by removing stop words. However, I use all words with the position in a query. I assume that the sequence and the position of word are important in classifying transactional queries. We can say [5] is a keyword-based classifier and my method is an expression-based classifier.

3.2 Extracting Cue Expressions

To extract cue expressions, anchor text and the title of a document are used. The definition or the explanation of a linked object can be extracted from a title and an anchor text. For example, a ticket reservation page has a title related to ticket reservation and a download button has an anchor text that explains a linked object. We can classify linked objects according to possible user's activities with them. Possible activities include reading, visiting and downloading. If a linked object is a binary file that is not readable, then its possible activity is downloading. However, in some cases, there is a linked object whose activity is ambiguous. For example, if a linked object is an html file, then assuming the action of the hyperlink is not easy. So, we clustered hyperlinks according to the

Table 2. Hyperlink Type

Type	Description	Example
Site	URL ends with a domain name, a directory name, and 'index.html'	http://cs.kaist.ac.kr/index.html
Subsite	URL ends with a directory name, with an arbitrary depth	http://trec.nist.gov/pubs/
Music	URL ends with a music file (mp3, wav, midi, etc.)	http://real.demandstremas.com/ragmen/mp3/corinth14.mp3
Picture	URL ends with a picture file (jpg, bmp, gif, etc.)	http://lylpan.com.ne.kr/pds/9.jpg
Text	URL ends with a text file (doc, ps, pdf, etc.)	http://www.loc.gov/legislation/dmca.pdf
Application	URL ends with an executable file (exe, zip, gz, etc.)	http://download.nullsoft.com/winamp/client/winamp3_0-full.exe
Service	URL ends with a cgi program (asp, pl, php, cgi, etc.)	http://www.google.com/search?q=IR
Html	URL ends with an html file	http://www.kaist.ac.kr/naat/interdept.html
File	other hyperlinked documents	http://ods.fnal.gov/ods/root-eval/out.log

extension of a linked object instead of using three categories: informational, navigational and transactional. If an expression occurs more frequently in a specific link type, then we can assume that it is a cue expression for a certain link type. For example, '*download file*' and '*reserve ticket*' can be key evidence that the type of a given query is a transactional query.

To cluster titles and anchor texts, I classified hyperlinks according to a linked object (Table 2). In the Table 2 *Site* and *Subsite* are usually for navigational queries. For example, anchor texts '*Fan Club*' and '*Homepage*' can be extracted as *Site* and *Subsite* types.

Both titles and anchor texts are used to extract cue expressions for each tagged hyperlink. The first two words and the last two words are used to extract cue expressions. There are 5 templates for extracting candidates of cue expression.

Table 3. Template for Extracting Cue Expression

Type	Extracted Expression
ALL	Q
F_1	w_1
F_2	$w_1\ w_2$
L_1	w_l
L_2	$w_{l-1}\ w_l$

In the Table 3, l implies the number of words in query Q. For example, anchor text '*winamp full version download*' has 5 cue expression candidates (Table 4).

Table 4. Example Cue Expression

Type	Expression
ALL	*winamp full version download*
F_1	*winamp*
F_2	*winamp full*
F_3	*download*
F_4	*version downoad*

The WT10g collection is used to collect the frequency of each candidate expression [1]. The frequency of each candidate is normalized with the total number of expression in each link type.

$$freq(link\ type) = \sum_i freq(exp_i \cap link\ type) \tag{2}$$

$$score(exp_i \cap link\ type) = \frac{freq(exp_i \cap link\ type)}{freq(link\ type)} \tag{3}$$

To use cue expressions, we use all input queries without removing stop words. With a given query, we can calculate a link score for each link type by adding score of each candidate expression. For example, the link score of a site type is calculated as follows.

$$\begin{aligned} LinkScore_{site} = {} & score(ALL \cap site) + score(F_1 \cap site) + \\ & score(F_2 \cap site) + score(L_1 \cap site) + score(L_2 \cap site) \end{aligned} \tag{4}$$

To use $LinkScore$, we need flags that indicate the type of a linked object and whether an index term is from an anchor text or not.

3.3 Combining Classifiers

To detect a transactional query, I extend the method of [5]. I used TiMBL [4], a Memory-Based Learning software package, to combine multiple classifiers. A Memroy-Based Learning is a classification-based supervised learning approach. It constructs a classifier for a task by storing a set of examples. Each example associates a feature vector (the problem description) with one of a finite number of classes (the solution). Given a new feature vector, the classifier extrapolates its class from those of the most similar feature vectors in memory. The metric defining similarity can be automatically adapted to the task at hand.

A vector for classification consists of following types of information.

- $IN(Q)$: whether a given query is an informational query or a navigational query?
- $isFileName$?: whether a given query is a file name or not? (e.g. '*stand by me.mp3*')
- w_1: the first word of a query
- w_l: the last word of a query
- $LinkScoreVec$: $LinkScore_{site}$, $LinkScore_{subsite}$, $LinkScore_{music}$, $LinkScore_{picture}$, $LinkScore_{text}$, $LinkScore_{application}$, $LinkScore_{service}$, $LinkScore_{html}$, $LinkScore_{file}$

where '$IN(Q)$' is the result of the Kang's method. The result of '$IN(Q)$' is Informational or Navigational. w_1 and w_l provide a special action verb such as '*download*' and an interrogative such as '*where*'. A query can be the name of the file that a user wants to achieve. '$isFileName$?' indicates whether a given query is a file name or not. Simple regular expression is used to decide the value of $isFileName$. $LinkScoreVec$ is a vector that consists of all kinds of link score.

For example, with query "*Do beavers live in salt water?*", '*beavers live salt water*' are extracted as keywords by excluding stop words. We assigned a Part-of-Speech Tag to each word. With tagged keywords, '$IN(Q)$' is calculated. Then calculate LinkScores for each link type. The result vector looks like as follows.

Table 5. Example Vectors with "*Do beavers live in salt water?*"

Type	Value
$IN(Q)$	*Informational*
w_1	*do*
w_l	*water*
$isFileName$	*No*
$LinkScore_{site}$	0.00001888148
$LinkScore_{subsite}$	0.00007382660
$LinkScore_{music}$	0.00000028322
$LinkScore_{picture}$	0.00000066085
$LinkScore_{text}$	0.00000000000
$LinkScore_{application}$	0.00000018881
$LinkScore_{service}$	0.00000037763
$LinkScore_{html}$	0.00006051515
$LinkScore_{file}$	0.00003068241

Since $LinkScore_{subsite}$ and $LinkScore_{html}$ are high among $LinkScores$, we can assume that this query does not have special cue expressions for transactional queries.

4 Service Link Information for Transactional Queries

In this section, useful information for a transactional query is proposed. A good result document for a transactional query should provide good service. To provide service, the further action of a user is needed. A user wants to buy some products, do some game, download a music file and a picture file, and so on. For example, query '*winamp download*' has the intention that a user wants to download a '*winamp*' program file by clicking or saving a linked file. Retrieved Web documents should have a download button or a linked file. For a transactional query, designated service is implemented by some mechanisms. These mechanisms contain a CGI program, a linked file (except an html file), and so on. These mechanisms usually implemented by a hyperlink. If a Web document has useful hyperlinks, then we can assume it is a kind of a document that provides some types of transaction. I propose a formula that accounts for the existence of hyperlinks. We call this '*Service Link information*'.

$$Service\ Link\ Information(d) = \frac{\#service\ links}{\#service\ links + \gamma_1 + \gamma_2 \times \frac{link\ count(d)}{avg.\ link\ count}} \quad (5)$$

where, '*service links*' means the union of music, picture, text, application, service, and file links. $\#service\ links$ is normalized with the number of links. *link count* means the number of all hyperlinks in a document and $avg.\ link\ count$ means the average number of all hyperlinks. In this paper, I set the value of γ_1 to 0.5 and γ_2 to 1.5.

$$\begin{aligned} link\ count(d) = {} & \#site\ links + \#subsite\ links + \#music\ links + \\ & \#picture\ links + \#text\ links + \#application\ links + \\ & \#service\ links + \#html\ links + \#file\ links \end{aligned} \quad (6)$$

Service Link information is normalized by the number of $link\ count(d)$. *Service Link information* is added to content information as follows to reorder result documents [5].

$$rel_{new}(d) = \alpha \times rel_{old}(d) + \beta \times Service\ Link\ Information(d) \quad (7)$$

5 Experiments

In this section, we conduct various experiments to see the usefulness of our classification method. In addition, experiments to see the usefulness of Service Link information for transactional queries are also conducted.

5.1 Test Set

Six query sets are used for experiments. For informational queries, queries of the TREC-2000 topic relevance task (topics 451-500) and queries of the TREC-2001 topic relevance task (topics 501-550) are used. We call them $QUERY_{I-TRAIN}$ and $QUERY_{I-TEST}$. For navigational queries, queries for randomly selected 100 homepages[2] and 145 queries of the TREC-2001 homepage finding task are used. We call them $QUERY_{N-TRAIN}$ and $QUERY_{N-TEST}$, respectively. For transactional queries, 100 service queries extracted from a Lycos log file are used for training and testing. I divided 100 queries into $QUERY_{T-TRAIN}$ and $QUERY_{T-TEST}$. Table 6 shows selected examples of transactional queries. The WT10g collection is used for making a classification model. Training queries are used to calculate the value of constants such as α and β in the Eq. 7.

Table 6. Example of Transactional Queries

airline tickets
acdsee.zip
free native American clip art
superbowl tickets
Trident blade 3d driver
download video card driver

5.2 Distinguishing Query Types

To measure the performance of our classification model, precision and recall are calculated with the following equations.

$$Precision = \frac{\#correct\ classification}{\#total\ trials} \tag{8}$$

$$Recall = \frac{\#correct\ classification}{\#queries} \tag{9}$$

For the value of '$IN(Q)$', I used navigational queries as a default result [5]. Every query has at least one query type. Therefore the precision and the recall of classification are the same. Table 7 shows the results of classifying query types.

ALL in the table means that all classifiers are combined for classification. The last column *QUERY-TEST* means queries from all test sets were used. By combining each classifier, we could increase precision and recall. From the table, '$IN(Q)$' is good for distinguishing informational queries and navigational queries. '*LinkScore*' shows good performance in detecting transactional queries.

[2] available at http://www.ted.cmis.csiro.au/TRECWeb/Qrels/

Table 7. Performance of Query Type Classification

	$QUERY_{I-TEST}$		$QUERY_{N-TEST}$		$QUERY_{T-TEST}$		$QUERY$-$TEST$	
	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
$IN(Q)$	58.0%	58.0%	97.9%	97.9%	0%	0%	69.8%	69.8%
w_1	34.0%	34.0%	97.9%	97.9%	14.0%	14.0%	67.8%	67.8%
w_l	2.0%	2.0%	99.3%	99.3%	34.0%	34.0%	66.1%	66.1%
$isFileName?$	0%	0%	100%	100%	2.0%	2.0%	59.6%	59.6%
$LinkScore$	62.0%	62.0%	73.8%	73.8%	74.0%	74.0%	71.4%	71.4%
ALL	74.0%	74.0%	79.3%	79.3%	78.0%	78.0%	78.0%	78.0%

5.3 Service Link Information for Transactional Queries

For the evaluation, binary judgment was used to measure the retrieval performance of transactional queries. The version of a program and the media type should be matched with a description in a query. The result document that needs further navigation to listen or download an object is not correct. Since we do not know all relevant results, precision at 1, 5, and 10 documents are used.

Table 8 shows a retrieval performance of Google search engine[3] with transactional queries. Top 100 documents were retrieved with Google, and then I combined Service Link information to get new top 10 documents. Since we did not know the similarity score of each result document, estimated score according to a rank was used. 0.9 was calculated from a training phase with training data.

$$rel(d) = \frac{1}{e^{r(d)}} + 0.9 \times Service\ Link\ Info(d) \tag{10}$$

where, $r(d)$ is the rank of document d. I used an estimated score instead of a real similarity score. Thus, the improvement of retrieval performance was low. However, we can conclude that *Service Link Information* is good for transactional queries.

Table 8. Retrieval Performance of Google over Transactional Queries

Model	$QUERY_{T-TRAIN}$			$QUERY_{T-TEST}$		
	P1	P5	P10	P1	P5	P10
Google	0.28	0.25	0.34	0.38	0.31	0.35
Google+Serv.	0.38	0.33	0.39	0.48	0.34	0.38

5.4 Retrieval Performance with Classification

Three ranking algorithms were used according to a query type. OKAPI algorithm was used for informational queries. For navigational queries, PageRank and URL

[3] http://www.google.com

information were combined with OKAPI score. For transactional queries, Service Link Information was combined with OKAPI score [5]. Table 9 shows the retrieval performance with the best and worst possible performance that we could achieve. Average Precision was used for measuring the retrieval performance of informational queries, MRR for navigational queries and precision at 10 documents for transactional queries. '*BEST*' is obtained when we have all correct answers and '*WORST*' is obtained when we have all wrong answers in classifying query types.

Table 9. Retrieval Performance with Classification

	$QUERY_{I-TEST}$	$QUERY_{N-TEST}$	$QUERY_{T-TEST}$
BEST	0.182	0.691	0.14
WORST	0.154	0.278	0.04
OURS	0.170	0.648	0.13

With our proposed query type classifier, 57% of possible improvement in informational queries, 90% in navigational queries and 91% in transactional queries could be achieved.

I also tested dynamic weighting of each type of information. Based on the probability of a query type, the proper weight of each type of information is decided, and then combined. The following equation was used to weight each type of information.

$$rel(d) = OKAPIscore + \lambda_1 \times URLInfo. + \lambda_2 \times PageRank + \lambda_3 \times ServiceInfo. \quad (11)$$

where, the values of λ_1, λ_2, and λ_3 are determined based on the probability of query type. Table 10 shows the retrieval performance of dynamic weighting.

Table 10. Retrieval Performance with Dynamic Weighting

	$QUERY_{I-TEST}$	$QUERY_{N-TEST}$	$QUERY_{T-TEST}$
BEST	0.182	0.691	0.14
WORST	0.154	0.278	0.04
OURS	0.171	0.664	0.16

Dynamic weighting method showed a slightly better performance than the method that uses special engines for transactional queries. When a query type is ambiguous, proper weighting of both types showed a better performance.

6 Conclusion

We have various forms of resources in the Web and consequently purposes of users' queries are diverse. Classic IR that focuses on content information cannot satisfy the various needs of a user. Search engines need different strategies to meet the purpose of a query. In this paper, we presented a method for classifying queries into informational, navigational and transactional queries. To classify a query type, link type that exploits anchor texts is used. I tagged each anchor text according to a possible action with a linked object. 9 link types were used to tag an anchor text. These tagged anchor texts can be used as large-scale training data. Cue expressions of each type are automatically extracted from tagged anchor texts. After we classified the type of a query, different types of information for a search engine are used. To retrieve better results for a transactional query, we add Service Link information. In addition, I proposed dynamic weighting method that combines each type of information according to the probability of each query category. Dynamic weighting showed a slightly better performance than the method that uses special engines for transactional queries. When a query type is ambiguous, proper weighting of both types showed a better performance.

For a future work, the fine-grained categories have to be considered. In this paper, we use 9 link types for three categories: informational, navigational and transactional. However, we can pinpoint target categories with 9 clusters. Categories for explaining the need of user should be extended. In addition, research on indexing model for our classification method is needed.

References

1. Bailey, P., Craswell, N., Hawking, D. : Engineering a Multi-Purpose test Collection for Web Retrieval Experiments. Information Processing and Management 39(6), pages 853-871, (2003)
2. Broder, A. : A Taxonomy of Web Search. SIGIR Forum, pages 3-10, 36(2), (2002).
3. Croft, W. B. : Combining Approaches to Information Retrieval. In: W. B. Croft (Ed.) Advances in Information Retrieval: Recent Research from the center for intelligent information retrieval, pages 1-36, Kluwer Academic Publishers, (2000)
4. Daelemans, W., Zavrel, J., Sloot, K. van der, Bosch, A. van den. : TiMBL: Tilburg Memory Based Learner, version 3.0, reference guide (Tech. Rep.). ILK Technical Report 00-01, Available from http://ilk.kub.nl/~ilk/papers/ ilk0001.ps.gz, (2000)
5. Kang, I.-H., Kim, G.: Query Type Classification for Web Document Retrieval. In Proceedings of the 26th Annual International ACM SIGIR conference on Research and Development in Information Retrieval. pages 64-71, Toronto, Canada, (2003)
6. Moldovan, D., Harabagiu, S., Pasca, M., Mihalcea, R., Girju, R., Goodrum, R., Rus, V.: The Structure and Performance of an open-domain Question Answering System. In: Proceedings of the Conference of the Association for Computational Linguistics. pages 563-570, Hong Kong, (2000)
7. Page, L., Brin, S., Motwani, R., Winograd, T.: The PageRank citation ranking: Bring order to the Web. (Tech. Rep.), Stanford Digital Library Technologies Project, (1998)

8. Ratnaparkhi, A.: A Maximum Entropy Part-of-Speech Tagger. In: E. Brill & K. Church (Eds.), Proceedings of the Conference on Empirical Methods in Natural Language Processing, pages 133-142, Somerset, New Jersey: Association for Computational Linguistics, Available from http://www.cis.upenn.edu/~adwait/statnlp.html, (1996)
9. Rose, Daniel E. & Levinson, Danny: Understanding User Goals in Web Search. In: Proceedings of the 13th international conference on World Wide Web, pages 13-19, New York, New York, (2004)
10. Westerveld, T., Kraaij, W., and Hiemstra, D. : Retrieving Web Pages using Content, Links, Urls and Anchors, In Proceedings of Text REtrieval Conference (TREC-10), pages 663-672, (2001)