DEWS2003 2-B-02

Building a Taxonomy-based Integrated Web Search Service

Said MIRZA PAHLEVI† and Hiroyuki KITAGAWA††

† Doctoral Program in Engineering, University of Tsukuba Tsukuba, Ibaraki, 305–8573 Japan
†† Institute of Information Sciences and Electronics, University of Tsukuba Tsukuba, Ibaraki, 305–8573 Japan

Abstract Taxonomy-based search services such as web directories are good starting points for users to search information needed from the web. In this paper we propose a *taxonomy-based context conveyance method* employing the search services. The method enables ones to build an integrated web search service where the user's current context on taxonomy of a taxonomy-based search service can be easily conveyed to the searches conducted with other web search services. The basic idea is to learn the context from the search result returned by the taxonomy-based search service and use it to modify the user query before forwarding the query to the target web search services. We conduct experiments using real web search services to show the effectiveness of our method.

Key words Taxonomy, Integrated web search service, Query modification, Rule learning

1. Introduction

Taxonomy is a hierarchical arrangement of topics. Using taxonomy in the web search has been proved to be useful to improve the search precision. Typical examples of the taxonomy-based search services are web directories. These search services are useful when we only have general topics or we are not sure how to narrow our search from a general topic. In addition, they can also help users understand how topics within a specific area are related and may suggest useful terms in conducting a search.

As a motivating example, imagine that a user wants to buy *salsa sauce* and its related products. He starts to find the information from the ODP [1], a web directory. He goes to directory "/Shopping/Food/Condiments/" and finds dozens of relevant website entries there. After browsing the directory, suppose now he wants to find another type of information such as web pages, images or news that is also related to the salsa sauce products but by using other web search services that are not based on the taxonomy. He then puts keyword "salsa" in the search field of the search service and gets many unrelevant matches such as web pages describing salsa dance and music. He then tries to refine his search by adding keyword "sauce" but still gets many unrelevant matches, for example pages related to salsa recipe and cooking. He can inspect the entries in the "/Shopping/Food/Condiments/" directory to refine his query. But it is a rather hard and time-consuming task. The idea here is that we can automatically convey the current context on the directory/taxonomy to the search conducted with the other web search services so that the user can easily get many relevant matches from them.

The conveyance of taxonomy context becomes more important in the current situation where many search sites have started to integrate different search services including the taxonomy-based search services to enrich and improve their search functionalities. For example, Google [2] and Altavista [3] provide images, news and other types of search services besides the ordinary web and taxonomy-based search services. Lycos [4] also provides news search service but with an additional shopping search service.

In this paper, we propose a *taxonomy-based context conveyance method* for integration of multiple web search services. A user starts a search from a taxonomy-based search service such as a web directory by specifying search keywords and a category. The system then probes the taxonomy-based search service using the given information and learns the user's current context in the form of a classification rule from the probe result. Finally, a query modifier is extracted from the rule, which in turn is used to modify the user query sent to the target web search services. To adapt the proposed method to different user requirements on search result effectiveness and properties of the target web search services, we have developed a *new fast classification rule learning algorithm*. The algorithm is aware of the precision/recall measures and query processing constraints of the target web search services.

In the previous work [5], we proposed a method to integrate web search services but did not consider the user requirements on search result effectiveness and search service properties. In subsequent work [6], we proposed a method called TAX-PQ to extend the first work by taking into ac-

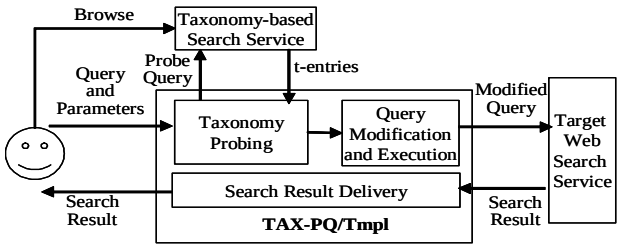

Figure 1 The architecture of the TAX-PQ/Tmpl system.

count the two points in the query modifier construction process. TAX-PQ, however, has a limited target web search services. It only guarantees that the modified queries can be processed by search services accepting ordinary Boolean queries; search services, such as Google, that accept only template-based Boolean queries (i.e., simpler queries without nested Boolean expression) are excluded. The limitation originates from the use of a decision tree learning algorithm to learn the user's current context, where the condition derived from the resulting decision tree usually is in the form of the nested Boolean expression.

This paper proposes a novel classification rule learning algorithm and extends TAX-PQ so that search services accepting only the template-based Boolean queries can also be the target. Beyond that, this paper reveals that the proposed method is also effective with different taxonomy data and discusses the implementation issues in the real world situation. For clarity, we call the extended method TAX-PQ/Tmpl.

The remaining of the paper is organized as follows. Section 2 presents the architecture of the TAX-PQ/Tmpl system. Section 3 explains the proposed rule learning algorithm. Section 4 presents the experimental evaluation. Section 5 discusses the implementation issues. Section 6 surveys related work. Section 7 concludes the paper and mentions future work.

2. Taxonomy-based Context Conveyance

TAX-PQ/Tmpl utilizes a taxonomy-based search service satisfying the following conditions. (Most of the major web directories satisfy these conditions.) (1) It maintains a taxonomy and information is pre-classified according to the hierarchy in the taxonomy. We call searchable units of information in the taxonomy *t-entries*. (2) Accepting a Boolean query/condition and a category in the taxonomy, it executes search of t-entries located under the specified category. In addition, with just a Boolean query, it executes search of t-entries from the entire taxonomy hierarchy. We assume the Boolean query can be a conjunction of terms. (3) Information on the number of matched t-entries and the category of each matched t-entry is provided in the returned result.

Many web search services are available on the web. Most accept Boolean queries, but they have differing acceptable query formats. For example, Altavista [3] accepts Boolean queries having nested expressions with parentheses. Another type of query format is the one supported by Google [2] and Lycos [4]. In their advanced search facilities, a query is formulated indirectly using a *query template* consisting of fields associated with Boolean operators. For example, the advanced search of Google provides a template consisting of fields labeled by "with all of the words" corresponding to the AND operator, "without the words" corresponding to the NOT operator and "with one of the words" corresponding to the OR operator. Hence, the expressive power of the Boolean query expression supported by the query template is very limited compared to that of the ordinary Boolean query expression. Some search services such as Google and Lycos only support query expression of this type, and cannot accept ordinary Boolean queries.

More precisely, the form of the *template-based Boolean query* can be expressed as follows:

$$C = (w_{1,1} \wedge \ldots \wedge w_{1,i}) \wedge (\neg w_{2,1} \wedge \ldots \wedge \neg w_{2,j}) \wedge (w_{3,1} \vee \ldots \vee w_{3,k}) \qquad (1)$$

where $w_{m,n}$ corresponds to a search term, $i, j \geqq 0$ and $k = 0$ or $k \geqq 2$. We denote $w_{m,n}$ and its negation $\neg w_{m,n}$ as *positive* and *negative literals*. Also we denote the first, second and third subexpression of Eq. 1 as $Conj_p$, $Conj_n$ and $Disj_p$.

Another query constraint imposed by the web search services is query size and its units. For example, Google accepts Boolean queries with maximum 10 keywords while MSN [7] with maximum 150 characters.

As a consequence, in order to guarantee that the modified queries can always be processed by commonly used web search services, we must modify the user query such that its form conforms to the format supported by target web search services and its size is within the maximum acceptable size. Since the template-based Boolean query can also be processed by search services accepting ordinary Boolean queries, in this paper we focus on the Boolean query format shown by Eq. 1. Some search services support only Boolean queries with no $Disj_p$. The algorithm proposed here can cope with this case by imposing an additional constraint $k = 0$.

Fig. 1 shows the architecture of the TAX-PQ/Tmpl system. There are three steps involved in the system: *taxonomy-browsing*, *taxonomy probing* and *query modification and execution*.

2.1 Taxonomy Browsing Step

In this step the user interactively browses the taxonomy and selects an appropriate category called a *context category*. He then defines a query condition Q to be sent to TAX-PQ/Tmpl. Note that Q is a conjunction of terms. The

pair of a query condition Q and a context category G forms a *query* (Q, G).

2.2 Taxonomy Probing Step

In this step, the system uses query (Q, G) given by the user to fetch the matched t-entries from the taxonomy-based search service. It does not obtain all the matched t-entries. Since all the matched t-entries are usually arranged in multiple result pages, fetching all of them lead to many HTTP requests and takes a lot of time, deteriorating the response time. The following is the sampling-based probing procedure [6]. We call the matched t-entries (not) under the specified context category including its subcategories *relevant t-entries* (*irrelevant t-entries*).

(1) Get the number of all the relevant t-entries N_1 and the irrelevant t-entries N_2 by sending queries (Q, G) and $(Q, null)$ to the taxonomy-based search service. Note that $(Q, null)$ means that no context category is specified.

(2) Fetch $p + q \cdot N_1/(N_1 + N_2)$ relevant t-entries, called *relevant samples*, from the context category. This step may need multiple HTTP requests to be issued to the search service. The purpose is to get t-entries contained in the search result for (Q, G) in addition to those returned in step 1.

(3) Fetch $p + q \cdot N_2/(N_1 + N_2)$ irrelevant t-entries, called *irrelevant samples*, from categories other than G. It is done by sending queries (Q, g_i) to the search service, where g_i is a top-level category. When G is a subcategory of g_i, the search result may contain both relevant and irrelevant t-entries. The t-entries are (automatically) separated by inspecting associated category information.

Note that p and q are controlled by the user as search parameters.

2.3 Query Modification and Execution Step

In this step, first the system creates a classification rule using the relevant and irrelevant t-entries fetched/sampled in the previous step. The classification rule is used to distinguish the *relevant class* from the *irrelevant class*. The relevant class is a class for the relevant t-entries while the irrelevant class is for the irrelevant t-entries. The classification rule takes the form of "$H \mapsto relevant$", where H is a Boolean condition conforming to the Eq. 1. Details of the classification rule construction will be given in the next section. The system then extracts H from the rule and makes it a *query modifier*, which in turn is used to modify the query condition Q. Q is modified by AND-ing it with H. The modified query $Q \wedge H$ is then sent to the target web search service.

3. Rule Learning Algorithm

There are many rule learning algorithms that have been proposed so far [8] [9]. Three points differentiate the algorithm proposed here from them. First, it constructs a rule for relevant class "$H \mapsto Relevant$" with H conforming to Eq. 1. Second, it induces a rule by explicitly constraining its size. Third, it selects the most promising rule based on the estimated *generalized effectiveness measure* (*G-measure*) of the modified query. Specifically, it selects a rule which would lead to the maximum *G-measure* given below [10]:

$$G\text{-}measure = \frac{1}{\alpha \cdot (1/recall) + (1-\alpha) \cdot (1/precision)} \quad (2)$$

where we can trade-off *precision* and *recall* by adjusting α $(0 \leqq \alpha \leqq 1)$.

We use the set of the relevant and irrelevant t-entries fetched in the taxonomy probing step as the training set ($TSet$). A t-entry is regarded as a tuple and each attribute represents the presence or absence of a term in the t-entry as a binary feature. We label relevant t-entries as *relevant* and irrelevant ones as *irrelevant*.

The algorithm is summarized in Fig. 2. $TSet$ is randomly split into two disjointed subsets: $GSet$ and $VSet$ with ratio 1:1. $GSet$ is used to construct candidate rules, while $VSet$ is used to select the best rule. The algorithm induces rule "$H \mapsto relevant$," which covers as many relevant t-entries and as few irrelevant t-entries in $GSet$. The rule also has the best estimated *G-measure* calculated with $VSet$ under a given α. H is created by first greedily constructing $Conj$ $(=Conj_p \wedge Conj_n)$, and then $Disj_p$. $Conj$ is constructed by greedily AND-ing positive/negative literals, while $Disj_p$ by greedily OR-ing positive literals. The size of condition x, denoted as $size(x)$, is the number of literals included in it. Thus, it is equal to the number of terms in the condition[1].

After splitting $TSet$ into $GSet$ and $VSet$, the algorithm extracts literal set $litSet$ from $GSet$ (line 2). This is done by first extracting terms from $GSet$, constructing positive and negative literals for each extracted term and then selecting the best literals. The best literals are selected by calculating the *weighted information gain* (WIG) of each literal when it is used to expand an empty/true condition and picking literals with the k largest WIG values. The WIG [11] is given below.

$$WIG(c_{i+1}, c_i) = rel_{i+1} \cdot \left(log_2 \frac{rel_{i+1}}{rel_{i+1} + irrel_{i+1}} - log_2 \frac{rel_i}{rel_i + irrel_i} \right) \quad (3)$$

where rel_i ($irrel_i$) is the number of relevant (irrelevant) t-entries in $GSet$ covered by condition c_i. c_i and c_{i+1} are a condition before and after the expansion. Note that a t-entry is said to be covered by a condition if it satisfies the condition. This measure rewards condition c_{i+1}, which increases the density of relevant t-entries that are covered without greatly

(1) The algorithm can be easily adapted to other size units.

```
CreateRule(TSet, maxSize, α)
1:  Split TSet into GSet and VSet;
2:  Determine literal set litSet;
3:  Conj ← CrConj(GSet, VSet, litSet, maxSize, α);
4:  maxSize ← maxSize − size(Conj);
5:  Disj_p ← true;
6:  Remove negative literals from litSet;
7:  If |litSet| and maxSize > 1
8:    Disj_p ←CrDisj(GSet, VSet, litSet, Conj, maxSize, α);
9:  H ← Conj ∧ Disj_p;
10: Return rule "H ↦ relevant";

CrConj(GSet, VSet, litSet, maxSize, α)
11: c_0 ← true;
12: c_best ← true;
13: Loop (i = 0 to maxSize − 1) {
14:   l_i ←BestLiteral(GSet, litSet, c_i);
15:   If no best literal, exit the loop;
16:   c_{i+1} ← c_i ∧ l_i;
17:   If G-measure(c_{i+1}) > G-measure(c_best)
18:     c_best ← c_{i+1};
19:   litSet ← litSet \ {l_i};
20:   if |litSet| = 0, exit the loop;
21: }
22: return c_best;
```

Figure 2 Proposed rule learning algorithm

reducing the total number of covered relevant t-entries relative to c_i.

Conjunct $Conj$ is created by calling function CrConj (line 3). It is created by repeatedly AND-ing literals starting from an empty/true condition. At each iteration i, condition c_i is AND-ed with literal $l_i \in litSet$ producing a more restricted condition c_{i+1} (line 16). l_i is the one that yields the largest WIG for c_{i+1} relative to c_i. The literal selection is done by function BestLiteral at line 14. After c_{i+1} is produced, its *G-measure* for the given α is estimated with $VSet$ (line 17). If the *G-measure* value is greater than the maximum *G-measure* obtained so far, then c_{i+1} is put into c_{best} (line 18). The loop stops if there is no literal with positive gain (line 15), no unused literal in $litSet$ (line 20), or the size of c_{i+1} is already equal to a given $maxSize$ (i.e., the loop has completed). Finally, the best conjunct c_{best} is returned at line 22.

Next, if $Disj_p$ can be created (i.e., line 7 is satisfied), CrDisj is invoked (line 8) to construct $Disj_p$. Similar to $Conj$, $Disj_p$ is created by repeatedly OR-ing positive literals. At each iteration i, condition c_i is OR-ed with (positive) literal $l_i \in litSet$ producing a longer condition c_{i+1}. l_i is selected such that it yields the largest weighted information gain for c_{i+1} relative to $Conj$. This is done to ensure that we always create a disjunct that can further improve the weighted information gain of the previously created $Conj$. The remaining procedure is similar to CrConj, where CrDisj returns the best disjunct obtained so far based on its *G-measure*. Finally, at the end of the algorithm (lines 9 and 10), $Conj$ is AND-ed with $Disj_p$ yielding H, and rule "$H \mapsto relevant$" is returned.

4. Experimental Evaluation

4.1 Query Modification Time and Search Result Effectiveness

In this section we reveal the relationship between the *query modification time* and *search result effectiveness* (*G-measure*) for different number of fetched t-entries. Query modification time is the time to sample the matched t-entries from the taxonomy-based search service and to construct a rule. For clarity, let us call taxonomy probing that fetches t-entries based on parameter p and q as *p_q partial probing* and the one that fetches all the matched t-entries as *full probing*.

To calculate the *G-measure* (Eq. 2) of a modified query we need to know the "true answer" of the query. To ease relevance judgment, instead of sending the modified query to the target web search engine, we send it "back" to the taxonomy-based search service in this experiment. Since each t-entry in the result returned by the taxonomy-based search service is associated with the category information, we can easily identify the "true answer" of the query. Of course, to get an unbiased evaluation, we should not use the same data for rule construction and performance evaluation.

Fig. 3 shows the flow of the experiment. First, a query (Q, G) is defined and sent to the taxonomy-based search service. For full probing, 2/3 of the matched t-entries are fetched and used as the training set; the remaining 1/3 is used as the test set. For p_q partial probing, $(2p+q)$ matched t-entries are sampled and used for the training set. The test set is formed to include the same number of t-entries as the full probing case. The training set is used to construct a rule, which in turn is used to modify Q.

The *modified query* $Q \wedge H$ is then sent "back" to the taxonomy-based search service and the *precision* and *recall* of the returned result are calculated based on the test set ($test$) as follows: Let d be a set of t-entries that are included both in the returned result set and $test$, and d_{rel} be a set of relevant t-entries in d. Similarly, let $test_{rel}$ be a set of relevant t-entries in $test$. In this experiment, $test_{rel}$ is the "true answer" of the query because it is a relevant t-entry set that is not involved in constructing the classifier. The *precision* and *recall* of the modified query and the *precision* of the initial query condition Q ($prec_i$) are given below. (Note that *recall* of Q is always 1).

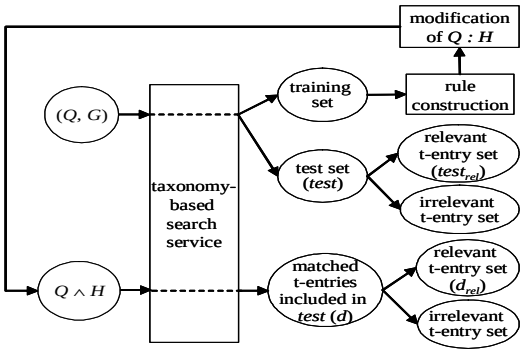

Figure 3 G-measure calculation.

Query condition	Context category	Meaning
Nepal	/Recreation/ Travel/	Travel information of Nepal (including travel businesses)
oil ∧ product	/Shopping/ Health/	Business in oil products for health
oil ∧ product	/Business/ Industries/	Fabrication of oil finished products

Figure 4 Queries and their meanings.

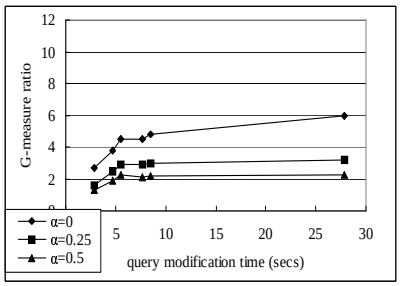

Figure 5 Query modification time and *G-measure* ratio.

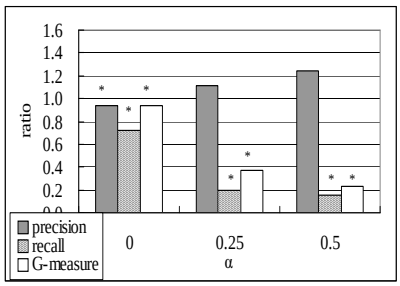

Figure 6 Comparison with an alternative method (full).

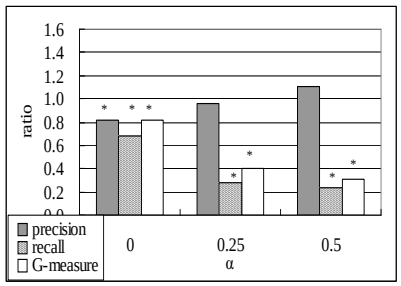

Figure 7 Comparison with an alternative method (20_320).

$$precision = \frac{|d_{rel}|}{|d|} \quad recall = \frac{|d_{rel}|}{|test_{rel}|} \quad prec_i = \frac{|test_{rel}|}{|test|}$$

We conduct the evaluation process with 3-fold cross validation and present the average of the three evaluation results. In the experiment we use ODP [1] as the taxonomy-based search service. We take the popular terms used in Google [12] and Yahoo [13] and define 25 queries. Fig. 4 shows some queries and their meanings used in the experiment. We derive the meaning of each query from the category description of its context category provided by ODP. Note that the meanings of the second and third queries are different even though their query condition is the same.

Since ODP provides 20 t-entries in each result page, for the partial probing, we set $p = 20$ and $q = \{0, 80, 160, 240, 320\}$. The cardinality of the literal set ($|listSet|$) is set equal to the maximum query size $maxSize$. The value of $maxSize$ is set to 10 and α is set to {0, 0.25, 0.5}.

Fig. 5 plots the query modification time versus the *G-measure* ratio for various probing types. The points in each line represent the 20_0, 20_80, 20_160, 20_240, 20_320 and full probing cases from left to right. The *G-measure* ratio is the relative *G-measure* value taking *G-measure* of the initial query condition Q as the base. It is obtained by dividing the *G-measure* value of the modified query ($Q \wedge H$) by that of Q, then taking the average over 25 queries.

The figure shows that the query modification time of partial probing is much smaller than that of full probing. This occurs because the partial probing gets only a fixed number of matched t-entries, which is usually much smaller than the number of all matches. Longer modification time usually results in a higher search result effectiveness. However, the difference between partial and full probing search result effectiveness is not significant compared to the differences in their modification times. This indicates that choosing appropriate parameters (p, q) yields almost comparable search result effectiveness to full probing, but with much less modification time. Beyond that, the increase in search result effectiveness lessens as α value increases. This tells us that for α of 0.25 or more, roughly, full probing search result effectiveness can be obtained with a relatively small sample size (e.g., $p = 20$, $q = 160$). Since the modification time and search result effectiveness depend on sample size, the user may control the trade-off between them by adjusting the parameters.

4.2 Use of G-measure in Rule Construction

The main purpose of this experiment is to see the effectiveness of selecting the best rule condition based on *G-measure*. We do this by comparing *G-measure* of queries modified using TAX-PQ/Tmpl and those modified using a method employing an alternative algorithm (denoted as an *alternative method*). The alternative algorithm is the one that constructs $Conj$ and $Disj_p$ without considering *G-measure* value of the condition. That is, the condition is repeatedly expanded until one of the stop conditions is satisfied and the finally constructed condition is returned.

The evaluation method and queries used in this experi-

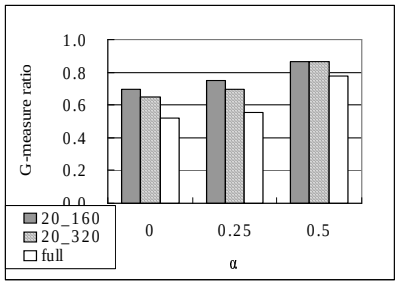

Figure 8 Comparison with a static method.

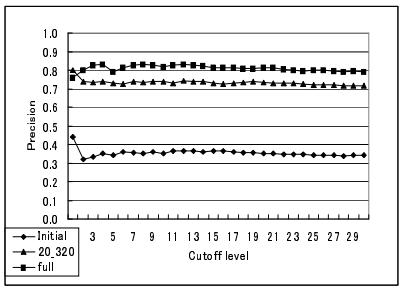

Figure 9 Document level precision with Google (ODP).

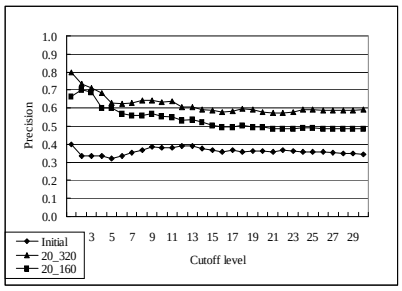

Figure 10 Document level precision with Google (Newsgroups).

ment are same as the previous one. Figs.6 and 7 show the experiment result. The *G-measure* ratio here is the relative *G-measure* value taking that of the TAX-PQ/Tmpl as the base. Thus, it is obtained by dividing the *G-measure* value of the alternative method by that of the TAX-PQ/Tmpl. The *precision* and *recall* ratio are obtained in the same way. To judge significance of the difference between the two methods, we also calculate its test statistics values from a *paired one-sided t-test* at a 5% level. An asterisk above the bar in the figures indicates that the *G-measure* difference between the TAX-PQ/Tmpl and the alternative method is statistically significant. As can be seen, the *G-measure* of TAX-PQ/Tmpl always outperforms that of the alternative method for all α. This achievement is also verified by results of the t-test. This performance advantage results because the proposed rule learning algorithm used in TAX-PQ/Tmpl always selects the best rule condition having the maximum *G-measure* value.

For $\alpha = 0$, precision of TAX-PQ/Tmpl outperforms that of the alternative method. As α increases, the precision ratio also increases but with a decrease in recall ratio (i.e., the precision difference becomes less significant but the recall difference is the reverse). This conforms to Eq. 1, and indicates that the α is reflected in the search result of modified queries from TAX-PQ/Tmpl. (Note that the *precision* and *recall* of the alternative method do not change with the α value.)

4.3 Comparison with a Static Method

In this experiment we compare TAX-PQ/Tmpl with a *static method*. The static method is a method that modifies a query with a pre-computed fixed classification rule. That is, different queries with the same context category are modified by the same rule associated with the category.

In the static method, a classification rule is created for each category of a taxonomy prior to query processing. It is done by treating t-entries in the category as relevant ones and those in the other categories as irrelevant ones. Note that the relevant/irrelevant t-entries do not necessarily meet the query condition Q. The rules are created by the proposed rule learning algorithm described in Section 3. We sample relevant t-entries of a category by taking randomly 30 percent of t-entries from the category, but limiting the maximum number to 6000 t-entries. For irrelevant t-entries, we sample them from the other categories, such that the number of irrelevant t-entries is three times larger than that of the relevant t-entries.

The evaluation method and queries used in this experiment are same as the previous one. Fig. 8 shows the *G-measure* ratio averaged over 25 queries. The *G-measure* ratio here is the relative *G-measure* value taking that of TAX-PQ/Tmpl as the base. From the results, for all cases, TAX-PQ/Tmpl always outperforms the static one. The results of t-test also conforms to this fact. The reason why the performance of the static method is poor is that the fixed rules associated with the context categories are forced to cover many topics that may exist in the categories. As a result, the query modifier derived from the rules cannot "fit" well to the queries. In contrast, rules derived in TAX-PQ/Tmpl need to cover specific topics implied by given queries.

4.4 Evaluation with Google

In this experiment, we evaluate the performance of TAX-PQ/Tmpl with a real web search service. We use Google [2] as the target web search service and the *document level precision* as a performance measure. Document level precision is precision computed after a given number of documents/matches in the ranked query result have been fetched. We calculate the precision until cutoff 30. Relevance judgment is done manually by directly checking whether cited pages conform to the combination of the given keywords and context category.

In the experiment, we use ODP [1] as the taxonomy-based search service again and take the 25 queries from the previous experiment. The comparison is done for 20_320 partial and full probing. The $|litSet|$ is set equal to $maxSize$. The value of $maxSize$ is set to $10 - initSize$, where 10 is the maximum query size of Google and $initSize$ is the size of the initial query condition. In addition, α is set to 0.

Fig. 9 shows the result. As can be seen, the document level precision with the 20_320 partial and full probing is

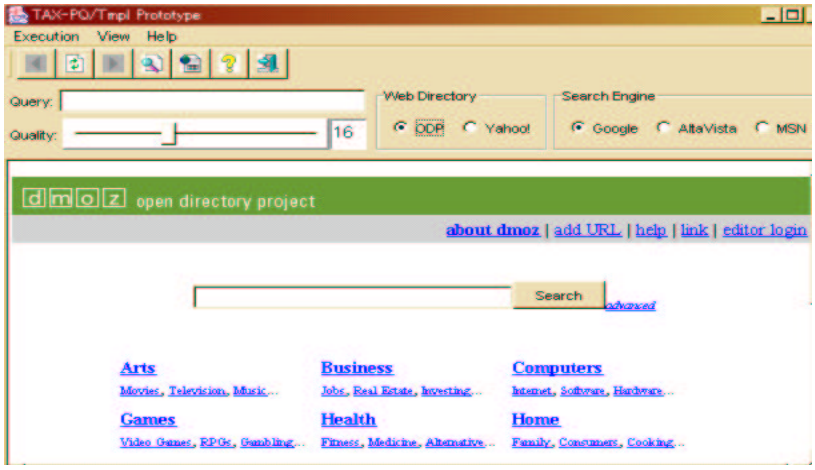

Figure 11 A screenshot of the prototype system.

much better than that of the initial query condition. Beyond that, the precision of the 20_320 partial and full probing is comparable.

To see the behavior of TAX-PQ/Tmpl for different taxonomy data, we use the Newsgroups search service provided by Google as the taxonomy-based search service. We define new 15 queries and compare the performance of the 20_320 and 20_160 partial probing. We cannot carry out the full probing with the Newsgroups search service since the number of matched t-entries is too large and Google's TOS does not allow us to send automated queries(2). The other parameter setting is same as the previous experiment.

Fig. 10 shows the result of the Newsgroups. As before, TAX-PQ/Tmpl can greatly increase the precision of the initial query condition, especially with the 20_320 partial probing. However, the precision increase is smaller compared to the ODP case because the quality of document collection in Newsgroups is lower compared to that of ODP(3).

5. Implementation Issues

As shown in Fig. 1, TAX-PQ/Tmpl stands among the user, a taxonomy-based search service and a target web search service. The taxonomy-based and target web search services can locate on the same platform or they can locate on geographically separated search service platforms. In either cases, TAX-PQ/Tmpl can locate on one of the search service platforms or can independently locate on a different platform. For easy reference, we call the case where the taxonomy-based search service and TAX-PQ/Tmpl locate on the same platform a *tightly integrated configuration*. The case on different platforms is called a *loosely integrated configuration*.

In the tightly integrated configuration, taxonomy probing time is not a concern. We need not worry about it because the network delay time needed to transfer the probing queries and results can be ignored. In addition, this configuration can take advantage of the full probing technique, which can boost search result effectiveness to the maximum. The shortcoming of the tightly integrated configuration is that it is costly to manage and maintain the taxonomy data locally. Beyond that, the data is often out of date. This occurs because it does not use live taxonomy data and has to update the copy periodically.

On the other hand, in the loosely integrated configuration, taxonomy probing time is a matter of concern. It matters because the network delay times needed to transfer the probing queries and results are relatively longer. This configuration is the main focus of this paper. As the experiments clarify, the partial probing technique makes it possible to shorten the delay time while maintaining high search result effectiveness. The main advantage of this configuration is that the association of a taxonomy-based search service to the system can easily be made by merely adding the correspondence wrapper to the system. Also, since this configuration uses live taxonomy data from the associated taxonomy-based search services, the data is always up-to-date, eliminating the need to periodically refresh data.

Fig. 11 shows a screenshot of the prototype system. In the current implementation, the system provides taxonomy data from two major web directories: ODP [1] and Yahoo [14], and allows the user to select one of the three major web search services, Google [2], AltaVista [3] and MSN [7], as the target. It also allows the user to adjust parameters to control the search and modification process.

6. Related Work

There are several work that also convey the context in-

(2) For the partial probing, we send queries using an ordinary web browser, fetch some returned pages and process them offline.

(3) The quality is low because most of context categories selected from the Newsgroup hierarchy are unmoderated groups.

formation to improve the web search result. Inquirus 2 [15] takes a query with context information in the form of a category of information desired and modifies the query based on the context information to improve the search result quality. The query is modified by using a set of modification terms extracted from the document collection of the category using the expected entropy loss. Recently they have extended the work by extracting the modification terms using SVMs [16]. Keyword-spice [17] also modifies a user query based on a specific category, but it uses a decision tree learning algorithm to construct the modification terms instead. Both methods do not utilize existing taxonomy, rather they require the system administrators to construct (flat) context categories and extract the modification terms from them prior to the running time. In this sense, these methods are similar with the static method described in Section 4. 3.

Another related work is to automatically infer context information from the everyday productivity application such as word processor to guide the web and database searching. The Watson project [18], IntelliZap project [19] and Remembrance Agent [20] analize the web pages/documents that are currently opened by the user to extract important terms from the pages/documents. The extracted terms are then used to construct/modify a query sent to the search services. Again, the above systems do not utilize the taxonomy. Some of them use the term weighting algorithm to infer the context while the others use the clustering algorithm.

7. Conclusions and Future Work

We have demonstrated that the taxonomy-based search services can provide a great help for better information searching on other web search services. For the purpose, we have proposed a taxonomy-based context conveyance (TAX-PQ/Tmpl) method for integration of multiple search services. With the method, ones can build an integrated web search service where the user's current context on taxonomy of a taxonomy-based search service can be easily conveyed to the next search conducted with other web search services.

The experiments indicate that by sampling only a small amount of data from the taxonomy-based search service, we can get a reasonably high retrieval effectiveness performance with fast query modification time. This partial probing technique could be used by the loosely integrated configuration where the TAX-PQ/Tmpl locates apart from the taxonomy-based search service. For the tightly integrated configuration, the full probing technique could be used instead to obtained the full retrieval performance.

TAX-PQ/Tmpl is dynamic in that the classification rule constructed to modify the query is different depending on both the selected context category and the query condition given by the user. This dynamic behavior leads to high retrieval effectiveness performance comparing to the static method. Another contribution is the new classification rule learning algorithm which is conscious of the *G-measure* and query processing constraints imposed by the target web search services. This makes TAX-PQ/Tmpl adaptable to the variety of search effectiveness requirements and query processing constraints.

We are going to further investigate the t-entry fetching procedure to decrease the probing time. In the taxonomy browsing step a user browses categories in the taxonomy to find a related category. Another alternative is also to allow the user to do searches on the taxonomy and cache the search result. On receiving query (Q, G) from the user, the system then looks the cache to check whether the search result has been retrieved. If it is, then the search result can be used to modify the query.

8. Acknowledgement

This research has been supported in part by the Grand-in-Aid for Scientific Research from Japan Agency for the Promotion of Science.

References

[1] Netscape, http://dmoz.org/.
[2] Google, http://www.google.com/.
[3] Altavista, http://www.altavista.com/.
[4] Lycos, http://dir.lycos.com/.
[5] M. P. Said and H. Kitagawa, "A Web Search Method Integrating Taxonomy-based and Crawler-based Search Engines." IPSJ TOD, 43 (SIG 9), pp. 15–27, 2002.
[6] M. P. Said and H. Kitagawa, "TAX-PQ: Dynamic Taxonomy Probing and Query Modification for Topic-Focused Web Search." In proc. DASFAA, 2003 (to appear).
[7] Microsoft, http://www.msn.com/.
[8] P. Clark and T. Niblett, "The CN2 induction algorithm." Machine Learning, 3:261–283, 1989.
[9] W. W. Cohen, "Fast effective rule induction." In Proc. ICML, pp. 115–123, 1995.
[10] J. Ding et al, "Mining medline: Abstracts, sentences, or phrases?" In Pacific Symposium on Biocomputing, 2002.
[11] J. R. Quinlan. "Learning logical definitions from relations." Machine Learning, 5, 1990.
[12] Google Zeitgeist, http://www.google.com/press/zeitgeist.html/
[13] Yahoo Buzz, http://buzz.yahoo.com/.
[14] Yahoo, http://www.yahoo.com/.
[15] E. Glover et al. "Improving category specific web search by learning query modifications." In SAINT, 2001.
[16] G. Flake et al, "Extracting query modifications from nonlinear SVMs." In Int. WWW Conf., 2002.
[17] S. Oyama et al. "Keyword spices: A new method for building domain-specific web search engines." In IJCAI, 2001.
[18] D. B. Leake et al. "Selecting task-relevant sources for just-in-time retrieval." In AAAI-99, 1999.
[19] L. Finkelstein et al. "Placing search in context: the concept revisited." In Int. WWW Conf., 2001.
[20] B. Rhodes and T. Starner. "The remembrance agent: A continuously running automated information retrieval system." In the PAAM, 1996.