データベースシステム 128－65
（２００２．７．１９）

Web 検索におけるタキソノミーに基づく適応型問合せ拡張

サイド ミルザ パレビ[†]　北川 博之[††]
[†]筑波大学工学研究科
[††]筑波大学電子・情報工学系

本論文は Web 上にある他の情報源の検索精度を上げるためにタキソノミーベース検索エンジンから提供される情報の使用を考える。これを実現するために、ユーザからの問合せをタキソノミーベース検索エンジンに送り、その問合せ結果から作成された決定木で問合せ拡張を行い、拡張された問合せを他の情報源に送る。問合せ拡張はユーザの問合せとタキソノミーから選択されたコンテクストカテゴリに依存し、適応的である。また，利用者が望む問合せ結果の質や対象情報源の検索インタフェースの制約を考慮した決定木構築アルゴリズムを提案する。最後に、提案手法とアルゴリズムの有効性と効率性を示すために評価実験を行う。

An Adaptive Taxonomy-based Query Modification Method for the Web Retrieval

Said Mirza Pahlevi[†] and Hiroyuki Kitagawa[††]
[†]Doctoral Program in Engineering, University of Tsukuba
[††]Institute of Information Sciences and Electronics, University of Tsukuba

We consider using the information provided by the existing taxonomy-based search engines for facilitating searches in other information resources in the web. For doing this, we send a user query to a taxonomy-based search engine, modify the query using a decision tree built from the retrieval results and send the modified query to other search interfaces available in the web space. The query modification is adaptive in that it depends on the user query and a context selected from the taxonomy by the user. Furthermore, to give flexibility to users to control the retrieval performance and to deal with the variety of the search interface constraints, we propose a decision tree construction algorithm adapted for the web retrieval tasks. We conduct some experiments showing the usefulness and effectiveness of the proposed method and the algorithm.

1 Introduction

The exponential growth of the Internet has led to a great deal of interest in providing users better search precision of web retrieval. One of the effective ways to improve the search precision is to use taxonomy. A typical example of web retrieval systems using this approach is taxonomy-based search engines such as Yahoo! (www.yahoo.com) and Open Directory Project/ODP (dmoz.com). The key of precision improvement in the systems is that they classify web pages (manually) into a hierarchically organized taxonomy and process a query based on categories in the taxonomy. However, the searches can only be done in their local

databases and their web coverage is very limited due to the manual classification.

Ideally, the searches done by the taxonomy-based search engines can also be applied in other information resources so that we can get similar retrieval performance from the resources. Furthermore, since different users usually have different retrieval performance requirements (in terms of precision and recall), it is better to give them control over the retrieval results based on their need.

In this paper, we consider to use the information provided by the taxonomy-based search engines (i.e. the high precision of retrieval results and the document collections indexed by the taxonomy) for facilitating searches in other information resources. In other words, we want to make the keywords and context-based search that successfully used in the taxonomy-based search engines possible for the other information resources in the web, but without constructing a new taxonomy to deal with it. For doing this, we send a query given by a user to a taxonomy-based search engine, modify the query using a decision tree built from the retrieval results and send the modified query to other search interfaces available in the web space. The query modification is adaptive in that it depends on the user query and context category selected from the taxonomy. Furthermore, in order to give flexibility to users to control the retrieval performance and to deal with the variety of the search interface constraints, we propose a decision tree construction algorithm adapted for the web retrieval tasks. We adapt the tree by controlling its size and its performance in terms of the weighted F1 measure [1]. This point is different with our previous work [2] where we create a classifier without considering the information resource constraints and the ability to control retrieval result performance.

The remainder of this paper is organized as follows: Section 2 introduces related work. Section 3 describes the proposed method and decision tree construction algorithm. Section 4 presents the evaluation results of the proposed method and algorithm. In the final section, we conclude with a summary.

2 Related Work

Several literature has described category specific web search methods. Inquirus 2 [3] is a domain specific metasearch engine developed at NEC research Institute. This engine takes a query with context information in the form of a category of information desired and modifies the user query based on the context information to improve the precision of the query. The query is modified by using a set of modification terms or phrases constructed by calculating expected entropy loss for each feature term extracted from document collection of the category. Keyword-spice [4] also modifies a user query based on a specific category, but it uses a decision tree to construct the modification terms. Both methods do not utilize existing taxonomy, rather they require the system administrators to construct the (flat) context categories prior to running time. In addition, the modification terms for each category is static, that is, it is fixed for all queries.

Another similar method is WebSifter II [5]. In the system, first a user creates personalized search taxonomies expressing his/her query intent via the proposed Weighted Semantic-Taxonomy Tree. The node/category labels in the tree are then further refined by consulting a web taxonomy agent such as Wordnet to eliminate the term ambiguity problem. Finally, the concepts represented in the tree are transformed into

Boolean queries processed by existing search engines. Although the system uses taxonomy, it does not employ classifiers or decision trees. In addition, the system needs a new taxonomy for each query intent.

Automatic classification of web documents into pre-specified categories was studied in [6], with the objective of increasing the precision of web search. They start by building a classifier for a set of categories using pre-classified training set of pages. In the query formulation step, the user specifies not only the query terms, but also one or more categories in which he/she is interested. The system retrieves documents matching the query, and then filters them by comparing their pre-computed categories against those chosen by the user. This method only classifies the query results and do not modify the user query.

3 Proposed Method and Decision Tree Construction Algorithm

3.1 Proposed Method

As mentioned before, the main goal is to make the keywords and context-based search used in the taxonomy-based search engines possible for the other information resources in the web. One way to do this is to learn/extract useful information from the engines based on a given user query and a selected context category. The extracted information can then be used to enrich the user query so that the query result quality can be improved. However, the matter becomes complicate because the variety of the size constraint imposed by the search interfaces. Many of search interfaces typically support Boolean query, but they have different allowable maximum query sizes. Therefore, the enriched user query should be in an acceptable Boolean form and within the allowable maximum query sizes of the search interfaces.

In this paper, we assume that a taxonomy-based search engine allows searches based on all categories existing in its taxonomy and provides additional information about the category of each matched document. (Most of the major taxonomy-based search engines support this.)

There are three steps involving in the proposed method: *query formulation step*, *taxonomy probing step* and *query modification step*. In the query formulation step, first the user navigates the taxonomy provided by the taxonomy-based search engine. After the user has found a category related to the topic sought, he/she then creates an initial Boolean condition (denoted as *probing condition*) that will be sent to the engine. In this paper, we call the category selected by the user as a *context category* and a pair of probing condition and context category as a *query*. The user may choose the context category after browsing some documents under the category.

In the next taxonomy probing step, the system sends the probing condition to the taxonomy-based search engine and separate the retrieval results into relevant and irrelevant documents based on the context category as follows.

- If category associated with a document is the context category (or a subcategory under the context category)[†], the document is considered to be relevant to the user query. This conforms to the method used by the taxonomy-based search engines to catch the user intent

[†] In the remaining part, we refer to the subcategory of the context category just as "context category".

- Otherwise it is considered to be irrelevant to the query.

In the final step (i.e. query modification step), the system constructs a decision tree for two categories: *relevant* and *irrelevant categories*. (The detail of the decision tree construction algorithm is given in the next section.) The relevant category is a category for the relevant document set while the irrelevant category is for the irrelevant document set. The decision tree is constructed to produce a Boolean condition (denoted as *condition modifier*) that is used to modify the probing condition by ANDing it with the condition modifier. The modified probing condition is then sent to the search interfaces and the results are presented to the user.

3.2 Decision Tree Construction Algorithm

```
DTWITHWF1 (trainingSet){
1:  Divide trainingSet into grow set (gSet) and evaluation set (vSet);
2:  Create a node called root;
3:  Loop{
4:    eNode ← Find an expandable leaf node that has the biggest error rate;
5:    if ((eNode = null) or (eNode.attSet = null)), then break the loop;
6:    A ← the attribute in eNode.attSet that best classifies eNode.gSet;
7:    if (A = null), then break the loop;
8:    For each attribute value v∈{0, 1} of A, add a new tree branch labeled A=v below eNode,
9:    where the branch points to the following leaf node lNode;
10:       lNode.gSet = examples in eNode.gSet with A = v;
11:       lNode.attSet = eNode.attSet – {A};
12:       lNode.label = majority class of examples in lNode.gSet;
13:   WF1 ← calculate the weighted F1 value of current relevant subtree using vSet;
14: }
15: Return a relevant subtree with the maximum WF1 value;
}
```

Fig. 1. Decision tree construction algorithm with *WF1* measure

The decision tree construction algorithm is summarized in Figure 1. Training set consists of documents labeled *relevant* and *irrelevant*. A *relevant subtree* is a subtree of a decision tree where its leaf nodes are the ones labeled relevant, while the size of the tree is the number of branches existing in the tree. The algorithm builds a decision tree using *gSet* and selects the best relevant subtree from the ones that have been built that has a maximum weighted F1 (*WF1*) value with respect to *vSet*.

Line 2 initializes a decision tree by creating a root node and initializes the properties of the node by setting *root.gSet=gSet*, *root.attSet=initial_attribute_set* and *root.label*=majority class. Line 4 selects a leaf node at which current decision tree will be expanded further. The leaf node should be expandable with respect to a given maximum relevant subtree size (*maxSize*) and has the biggest error rate value that is greater than zero. We say a leaf node is expandable, if we expand the current tree at the node then the relevant subtree from the resulting expanded tree will not exceed *maxSize*. The error rate of a leaf node is defined as the proportion of examples from a minority class in the node's *gSet*.

Line 6 gets the best attribute from the attribute set of the selected node that best classify the node's grow set. The best attribute is the one with the highest information gain that is similar to the one used in ID3 algorithm [7]. Lines 8 and 9 expand the tree by creating a branch pointing to a new leaf node for each value of

the best attribute. The properties of the new leaf node are then set at lines 10 through 12.

Line 13 calculates the *WF1* value of current relevant subtree using the following equation: $WF1 = (precision \cdot recall) \div (\alpha \cdot recall + (1 - \alpha) \cdot precision)$. The calculation is done by first converting the subtree into a Boolean condition and then calculating its precision and recall against *vSet*. Finally, line 15 returns the best relevant subtree with the maximum *WF1* value that will be converted into a modifier condition having the same size as the subtree.

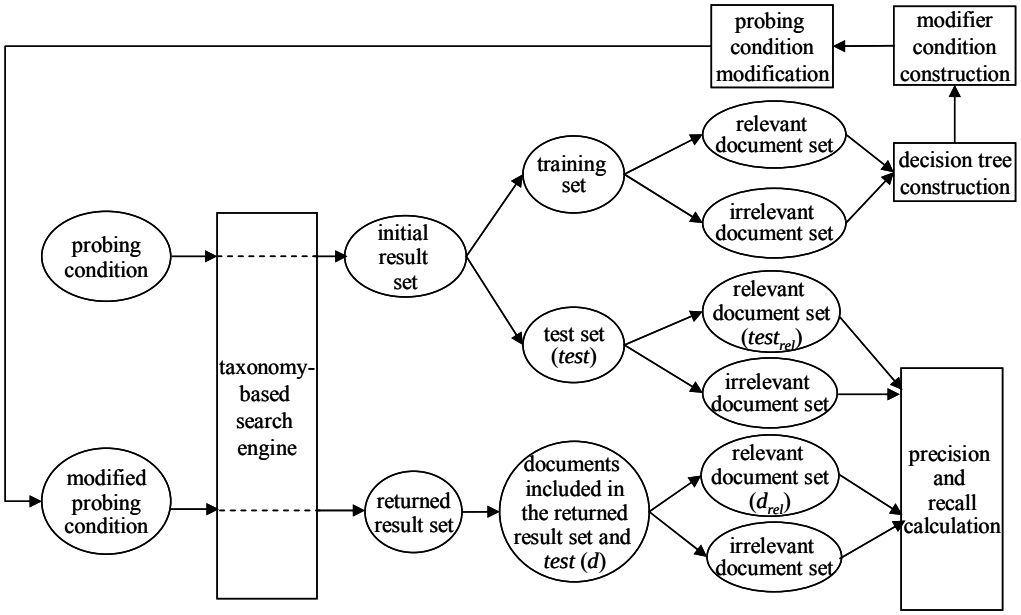

Fig. 2. Evaluation method

4 Performance Evaluation

4.1 Evaluation Method

The main purpose of this experiment is to see the effectiveness of selecting the best relevant subtree based on *WF1* value in the decision tree construction process. We do this by comparing the proposed method with a simple information gain (IG)-based method. The simple IG-based method is a method that builds a decision tree without considering *WF1* value of the decision tree. That is, the algorithm builds a decision tree until one of the stop conditions is satisfied and use the relevant subtree extracted from the final decision tree to construct a condition modifier.

In this experiment, we calculate and compare the performance of the two methods with respect to their set-based *WF1* values (*WF1* of an unordered set of retrieved documents). In order to calculate the set-based *WF1* value of a given query we need to know the "true answer" of the query. To make relevance judgment easy, we simulate the search interface (where the modified probing condition is sent) with a taxonomy-based search engine. This can be done by having the search carry out against documents in all categories of the taxonomy-based search engine. The "true" answer of a query from the simulated search interface is a subset of documents that matches the probing condition and that are classified into the context category. (Note that, since the documents come from a taxonomy-based search engine, they are associated with their categories.)

Figure 2 shows the flow of the experiment. First, we construct a query by defining a probing condition

and selecting a context category from the taxonomy. After the probing condition is submitted to the taxonomy-based search engine, we get an initial result set. The result set is then divided into training set and test set (*test*), which in turn are divided into relevant and irrelevant document sets based on the selected context category. The relevant and irrelevant documents in the training set are used to construct decision trees of the two methods, which in turn are used to modify the probing condition.

Table 1. Queries and their meanings

Initial condition	Broad context category	Meaning	Narrow context category	Meaning
christmas	/Business/Industries/	Industries of Christmas related products	/Business/Industries/Agriculture_and_Forestry/	Industries of Christmas tree (farming)
nepal	/Recreation/Travel/	Travel information of Nepal (including travel business)	/Recreation/Travel/Travelogues/	Personal travelogues of Nepal
oil AND product	/Shopping/Health/	Business in oil products for health and beauty	/Shopping/Health/Beauty/	Business in oil products for beauty only
ginger	/Home/Cooking/	Food and drink recipes using ginger	/Home/Cooking/Beverages/	Drink/beverage recipes only using ginger
first AND aid	/Health/Public_Health_and_Safety/	First aid topics related to public health and safety	/Health/Public_Health_and_Safety/Emergency_Services/	First aid topics related to emergency services only (e.g. rescue squads)
oil AND product	/Business/Industries/	Fabrication of oil finished products (e.g. petroleum and food)	/Business/Industries/Energy/	Fabrication of oil finished products related to energy (e.g. oil and gas)

The resulting modified probing conditions from the two methods are then sent to the simulated search interface (in this case the taxonomy-based search engine itself) and the precision and recall of the returned results are calculated based on *test* as follows. Let d be a set of documents that are included both in the result set of the modified condition and in *test*. Let d_{rel} be a set of relevant documents in d. Similarly, let $test_{rel}$ be the set of relevant documents in *test*. In this experiment, $test_{rel}$ is the "true" answer of the query because it is a relevant document set that is not involved in constructing decision trees of the two methods. Then, the precision and recall of the modified probing condition are calculated as follows: $precision = |d_{rel}| \div |d|$ and $recall = |d_{rel}| \div |test_{rel}|$. (Note that precision of the initial condition is $|test_{rel}| \div |test|$ and recall is always 1.)

We conduct the evaluation process with 3-fold cross validation and present the average of the three times evaluation results. We use ODP as the taxonomy-based search engine and refer each site in the returned results as a document. There are two query types used in the experiment: queries with broad context categories and with narrow context categories. The number of queries of the first type is 50 with 49 probing conditions and 33 broad context categories, while the second type is also 50 with 49 probing conditions and 47 narrow context categories. We say a context category as a broad one, if it is a direct subcategory of the main category of the taxonomy and as a narrow one if it is a subcategory of the broad context category. Table 1 shows some of the queries and their meanings. As shown in the table, the meanings of a probing condition at broad and narrow contexts are similar, but it is more specific at the narrow context. Furthermore, the meaning becomes completely different at other different context categories. For example, the meaning of

probing condition “oil AND product” at context categories “/Shopping/Health/” and “/Business/Industries/”.

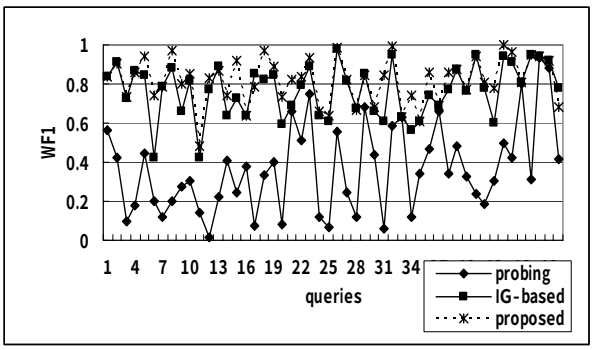

Fig. 3. *WF1* at broad context (α=1, *maxSize*=10)

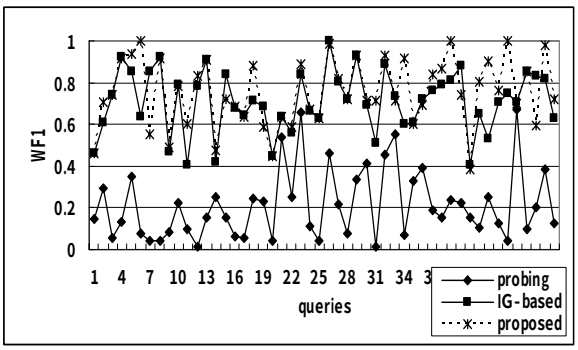

Fig. 4. *WF1* at narrow context (α=1, *maxSize*=10)

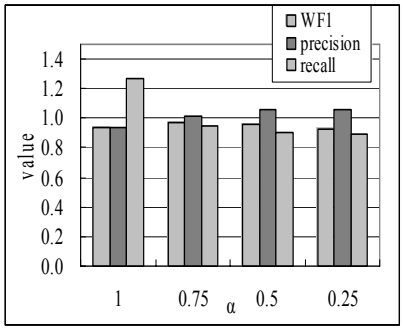

Fig. 5. Broad context & *maxSize*=10

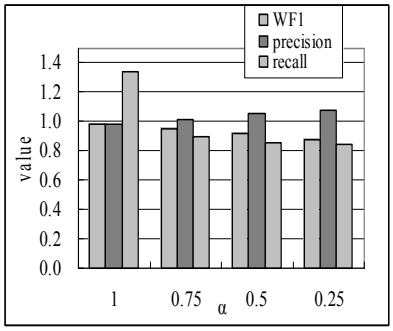

Fig. 6. Broad context & *maxSize*=20

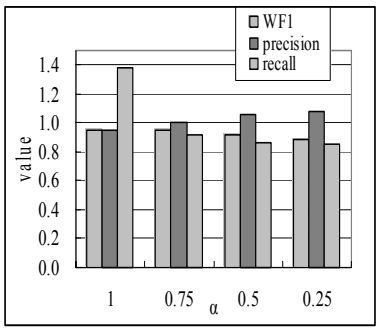

Fig. 7. Broad context & *maxSize*=30

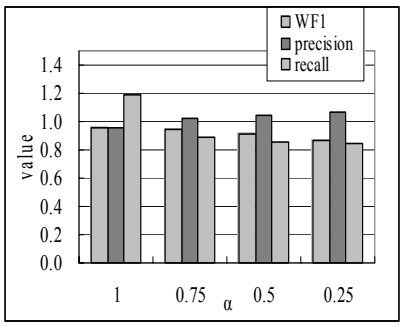

Fig. 8. Narrow context & *maxSize*=10

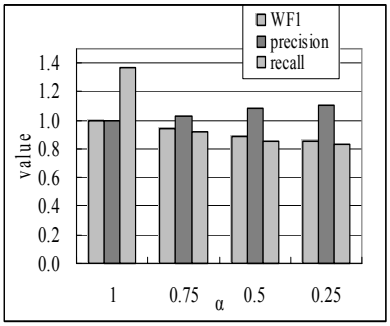

Fig. 9. Narrow context & *maxSize*=20

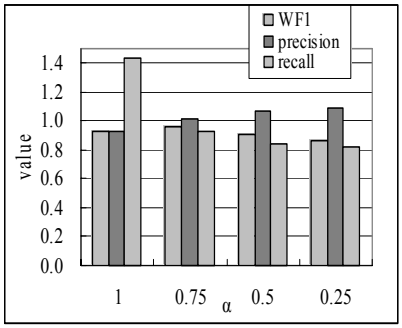

Fig. 10. Narrow context & *maxSize*=30

4.2 Evaluation Result

In the experiment, *maxSize* is set to 10, 20 and 30 and α of *WF1* is set to 0.25, 0.5, 0.75 and 1.0. Figures 5 and 4 show *WF1* comparison among the probing condition, IG-based and the proposed method for α=1 and *maxSize*=10 (for *maxSize*=20 and 30 the results are similar). As can be seen, the two methods can significantly increase the *WF1* value of the probing condition.

Figures 5 through 10 show the average of the ratio between *WF1*, precision and recall of simple IG-based method and those of the proposed method. Note that, the precision and recall of IG-based method do not change with the change of α because the method does not consider the WF1 value of the decision tree. To see the significant difference between the proposed and simple IG-based methods, we also calculate test

statistics using a paired one-sided *t*-test at 5% level. Due to space constraint we cannot show the result here, but it indicates that for the ratio value less than 1 (shown in the figures) the difference between the proposed and simple IG-based method is significant (i.e. the proposed method outperforms the IG-based one).

From the figures, it is clear that the proposed method outperforms simple IG-based method with respect to *WF1* value for all α, *maxSize* and context category types. The reason is obvious. It is because the algorithm always selects the best relevant subtree having a maximum *WF1* value in the decision tree construction time to modify a probing condition. Furthermore, for α=1, the precision of the proposed method always outperforms that of the simple IG-based method. As α decreases, the precision also decreases but with the increase of recall. This conforms to *WF1* formula telling that to set α to a larger (respectively smaller) value if the precision (respectively recall) is the main concern. This indicates that the proposed method can modify the probing condition based on the importance of precision and recall.

4 Conclusions

We have proposed an adaptive query modification method using the information provided by the existing taxonomy-based search engines that makes the keywords and context-based search possible in the web space. We also have proposed a decision tree construction algorithm adapted for the web retrieval tasks taking into account Boolean condition size supported by existing search interfaces and the performance of the tree in term of *WF1* measure. We have shown by experiments that the proposed method can significantly increase the performance of the probing query and outperforms the query modification using a traditional decision tree construction algorithm. Finally, we have also shown that the algorithm can control the retrieval result performance based on a given α value.

References

[1] C. J. VanRijsbergen: Information Retrieval. London: Butterworths, 1979.

[2] S. Mirza and H. Kitagawa: Taxonomy-based Adaptive Web Search Method, *Proc. 3rd IEEE International Conference on Information Technology: Coding and Computing*, pp. 320-325, 2002.

[3] Eric J. Glover et al: Improving Category Specific Web Search by Learning Query Modifications, *Symposium on Applications and the Internet (SAINT),* pp.23-31, 2001.

[4] S. Oyama, T. Kokubo, and T. Ishida: Keyword Spices: A new Method for Building Domain-Specific Web Search engines, *International Joint Conference on Artificial Intelligence (IJCAI-01)*, pp.1457-1463, 2001.

[5] L Kerschberg, W. Kim and A. Scime: A Semantic Taxonomy-Based Personalizable Meta-Search Agent, *Proceedings of the second International Conference on Web Information Systems Engineering (WISE)*, pp.53-62, 2001.

[6] C. Chekuri and M. H. Goldwasser: Web Search Using Automatic Classification, *Poster at the Sixth International WWW Conference*, 1997.

[7] J. R. Quinlan: Induction of Decision Tree, Machine Learning, 1(1), pp. 81-106, 1986.