A Study on Query Modification Methods for Web Search Using Taxonomy and Classification Learning

Doctoral Program in Engineering

University of Tsukuba

2003, March

Said Mirza Pahlevi

Dedication

In memory of Drs. Said Usman S.P.

Acknowledgments

First and foremost, I would like to thank my supervisor, Professor Hiroyuki Kitagawa, whose guidance and encouragement made this dissertation possible. He kept me steered in the right direction, through his support, care and good judgment. His insight and ideas formed the foundation of this dissertation as much as mine did, and his guidance helped me to get over the various hurdles during my graduate years.

Second, I would like to thank Assistant Professor Yoshiharu Ishikawa for his constructive and instructive advice. I am also thankful to the dissertation committe members Professor Nobuo Ohbo, Professor Jiro Tanaka, Professor Seiichi Nishihara and Associate Professor Mikio Yamamoto for their thoughtful comments.

I would like to thank my father and mother, Rukun Hidayat and Hafidhah, for their love and pray. Special thanks to my wife, Indrasari, for her

love and support. It is quite tough to be a wife of a graduate student while living in a foreign country, but she went through the last 3 years without many complaints.

Finally, I would like to thank other members of Kitagawa Data Engineering (KDE) laboratory for providing a stimulating environment during my laboratory life. I am also grateful to the past members Dr. Atsuyuki Morishima and Dr. Shinagawa Norihide for their support and help.

Abstract

A pressing need exists to improve effectiveness of the web search process and result. The need results from the extremely large volume of information available on the web and the large variety of Internet users.

Current web search services are the main starting points to search for information needed from the huge document collection. Their limitations, however, are apparent. Search engines that employ a keyword-based search are good at returning a long list of relevant documents, but the list is also cluttered with many irrelevant candidates. Even if the engines rank the matched documents according to a specific ranking algorithm, they are still not especially helpful in locating the needed information. This problem results mainly from the short queries usually given by users, where it is hard to express the search intent using short queries and the keyword search approach used by the search engines. On the other hand, taxonomy-based search facilities such as web directories generally have better search result

precision than search engines. The precision is better because they use a context-based search in addition to the typical keyword-based search in processing user queries. Nevertheless, the need to manually classify information in their directories limits their web coverage.

This dissertation proposes novel query modification methods for web search to improve effectiveness of the web search process and result. The proposed methods use *taxonomy* provided by a taxonomy-based search facility and a *classification learning algorithm*. The taxonomy is used as a guidance to *determine* the user query intent while the learning algorithm is used as a way to *derive* the query intent in the form of a Boolean condition. The Boolean condition is used to modify the given user query.

More precisely, a user first designates an initial query and a context category in the taxonomy. The system then probes the taxonomy and samples data sets in the taxonomy using the given information. A classifier is then constructed using the sampled data and a learning algorithm, and a query modifier is extracted from the classifier. Finally, the initial query is modified using the modifier sent to search engines. The proposed methods are dynamic so that they can derive an appropriate query modifier fit for the given initial query and context category. Beyond that, they allow the user to trade off search result effectiveness with response time by adjusting parameters to control the amount of sampled data.

We have developed two new classification learning algorithms to adapt the proposed methods to different user requirements on the search result effectiveness and properties of target search engines. The algorithms are aware of the precision/recall measures and constraints on query processing capabilities of the search engines.

Experiments involving real web search services show that the proposed methods can significantly improve the web search process and result effectiveness. The results comply with user requirements under constraints of the target search engines.

Beyond the above contributions, we have developed a prototype system to demonstrate the practicability of the proposed methods. We also discuss implementation issues in the real web environment.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The exponential growth of information on the web continues to escalate demand for improving the web search process and result effectiveness. Although today's search engines provide a certain amount of help in web search, the search results are often cluttered with a huge amount of irrelevant information. To solve the clutter problem, much research has been devoted to web search techniques. Research includes, among others, the analysis of web links [Kle99][BP98][CK97][CDI98], application of text data mining techniques [G+01][OKI+01][CS96], employment of text data clustering techniques [ZEMK97][HKP95] and intelligent web search agents [BH99][AFJM95].

Taxonomy is a useful resource to categorize and classify information. It is a hierarchical arrangement of topics and subtopics that clearly shows the

relationships between them. Some search services provide a taxonomy that pre-classifies the collection of information compiled in their data set, and uses it to improve search result effectiveness. We refer to these search services as *taxonomy-based search facilities*. Typical examples are web directories such as Open Directory Project/ODP [Net] and Yahoo [Yah]. They accept a context category in the taxonomy designating the scope of topics in addition to commonly accepted keywords. The search then focuses on the specified category. The topic-focused query result significantly contributes to improved search result effectiveness. Use of the maintained taxonomy is, however, restricted to searching information stored in their own data set and not applicable to other web search contexts. Also, because pre-classification of the information is usually done manually by web editors, coverage of the search facilities is limited.

In addition to these taxonomy-based search facilities, some approaches have used taxonomies/categorical information to improve the web search process and result effectiveness as surveyed in Chapter 2. They do, however, suffer from the following serious limitations and defects:

1. They are *not scalable* or *robust* in the large and dynamic web environment because they have to create and maintain their own new categorical information for each searched objective.

2. They are *static*. This means their classification schemes are fixed inde-

pendently from given user queries, and do not incorporate user query intents.

3. They are *not flexible* to user requirements on search result effectiveness since they are inherently unable to provide users the ability to adjust search result effectiveness.

This dissertation proposes a *novel query modification framework* that uses existing taxonomy maintained by a taxonomy-based search facility and classification learning to improve the web search process and result effectiveness of search engines. The user first designates an initial query and a context category in the taxonomy, as if information searchable in the target web search engines had been pre-classified according to the taxonomy. To probe the taxonomy, the initial query is issued to the taxonomy-based search facility, and samples of the query results are chosen to construct a classifier. A query modifier is then derived from the classifier to focus the initial query on the selected context category. Finally, the modified query is forwarded to the target web search engines and the query result is returned to the user.

Basic Query Modification Method

The basic query modification method employs existing classification learning algorithms to construct the classifier. Any learning algorithms can be used as

the learner on the condition that a Boolean condition can be derived from the learned classifiers. This method is easy to implement because many existing learning algorithms can be used with it.

Extensive experiments have revealed that the basic method can dynamically derive an appropriate query modifier fit for the given initial query and context category. Beyond that, we have also found that the basic method allows the user to trade off search result effectiveness with response time by adjusting parameters to control the amount of data sampled from the taxonomy-based search facility in the taxonomy probing. If the amount is limited, time spent on taxonomy probing and classifier construction is reduced, but it may lead to degraded search result effectiveness. We show that we can compromise the potentially conflicting factors but still achieve reasonable search result effectiveness and good response time.

Despite the easy implementation and benefits listed above, the basic query modification method suffers from the following problems.

1. It cannot guarantee that the modified queries can always be processed by the target web search engines. This limitation results because web search engines usually have different acceptable query formats with different query sizes and units, and the existing learning algorithms are not aware of the query restrictions.

2. It cannot provide users the control needed to adjust search result effectiveness. This limitation results because, again, the existing learning algorithms are not aware of the search result effectiveness.

Enhanced Query Modification Methods

To adapt the proposed method to different user requirements on search result effectiveness and properties of target web search engines, we have developed two new classification learning algorithms: *constrained decision tree* (CDT) and *constrained classification rule* (CCR). The algorithms are aware of the precision/recall measures and constraints on query processing capabilities of the target web search engines.

Precision and recall are important measures of search result effectiveness. Although the search result should ideally present high precision and recall, we know that high precision and recall are usually in conflict. High precision is generally preferred in the context of web search. It is best, however, if the user can be given some way to express individual requirements on search result effectiveness. The enhanced query modification methods that use the two algorithms to construct a classifier allow the user to express this as a weight value α in the *general effectiveness* (G) *measure* [D$^+$02]. $\alpha = 1$ ($\alpha = 0$) means that only recall (precision) is concerned. Values in between stand for

their relative weight. Our algorithms try to find a query modifier that will lead to the modified query with the maximum *G-measure* value under the given α. The algorithms also take into account the query size and format restriction imposed by the target web search engines to control the size and format of the resulting query modifier. Thus the derived query modifier can fit both the user requirements and constraints of the target web search engines.

Extensive experiments have shown that the enhanced query modification methods share the same characteristics as the basic method. Beyond that, they outperform the basic method in terms of search result effectiveness. We have also found that the enhanced query modification methods allow the user to control search result effectiveness by adjusting the α value while guaranteeing that the modified queries can be processed by the target web search engines.

Prototype System Development

We have developed a prototype system implementing the proposed methods to demonstrate the practicability of the query modification framework. The system is implemented in a *loosely integrated configuration* that uses live taxonomy data from some major web directories. As a result, it not only

provides up-to-date taxonomy data to users but also provides a rich selection of taxonomy data that will be used to formulate their queries. In the current implementation, a user may select taxonomy data and a target web search engine from major web directories and search engines. We also discuss another implementation method called *tightly integrated configuration*, and specify its benefits and limitations.

Main Contributions

The following summarizes the main contributions presented in this dissertation.

1. We propose a novel query modification framework that combines a *dynamic taxonomy probing* and *classifier construction* to improve the web search process and result effectiveness.

2. We perform extensive experiments and show that the proposed framework provides a reasonable range of trade-offs between search result effectiveness and response time.

3. We propose two new classification learning algorithms that are aware of the search result effectiveness and constraints on query processing capabilities of the target web search engines.

4. We show that the proposed framework with the two algorithms allows a user to control search result effectiveness by adjusting effectiveness parameter α while guaranteeing that the modified queries can be processed by the target web search engines.

5. We developed a prototype system implementing the proposed framework and discuss implementation issues.

Organization

The remainder of this dissertation is organized as follows: Chapter 2 presents the background and surveys related work. Chapter 3 discusses the problem formulation and basic ideas to solve the problem. Chapter 4 presents the proposed query modification framework. Chapter 5 describes the basic query modification method derived from the framework and evaluates its performance. The chapter also discusses enhancements needed to put the proposed framework into practice. Chapters 6 and 7 describe CDT and CCR enhanced query modification methods and evaluate their performance. These chapters also describe the proposed classification learning algorithms. Chapter 8 compares the three query modification methods and discusses their usage to obtain optimal performance. Chapter 9 presents a prototype system that implements the proposed framework and discusses implementation issues. The final section lists our conclusions and discusses future work.

Chapter 2

Background and Related Work

2.1 Overview

An estimated 2 billion pages are publicly available on the web today. Search engines and taxonomy-based search facilities are essential starting points for users to seek information needed from the huge web page collection. This chapter describes the two search services, and surveys related work devoted to improved search services. This chapter also describes two popular classification learning algorithms: *decision tree* and *classification rule learning*. These algorithms are frequently used in the web search technologies.

2.2 Web Search Services

2.2.1 Search Engines

Search engines can be divided into two types: *general search engines* and *specialized search engines*. General search engines collect and index material related to any subjects found at websites. They utilize 'crawlers' or 'spiders' that scour the Internet for that purpose. They are usually useful when we are looking for an overview of a subject or general, light information. Two well-known examples of general search engines are Google [Goo] and Altavista [Alt].

In contrast, specialized search engines maintain material that relates to particular fields or subjects. For that reason, these search engines are usually useful in finding detailed or highly specific information about a subject area. The maintained information includes *indexable* and *unindexable* information. The indexable information is on web pages. This information is collected using *topical crawlers* that can 'recognize' web pages on a specific set of topics. Examples of specialized search engines that maintain indexable information are Scirus [Sci], which indexes scientific information, Cora [Res], which indexes research papers on all computer science subjects and SiteSeer [Ins], which indexes Postscript and PDF research articles.

Unindexable information, on the other hand, (usually denoted as *hidden web* or *invisible web* or *deep web*) is on legacy databases that existed before the development of HTML, and usually covers a specific subject area. These databases are accessed from the web through special interfaces (e.g., CGI) capable of translating the stored data into the HTML format on demand. HTML pages are generated on an ad hoc basis, so they cannot be reached by general search engines' crawlers. Examples of specialized search engines providing information from the hidden web are Medline [Pub][Gat], which has 12 million references to journal articles in life sciences with a concentration on biomedicine and FactFinder [Bur], which provides U.S. census data. A complete list containing tens of thousands of the specialized search engines can be found at [Com][Inv][Lib].

The primary method search engines use to search text is keyword-based search. With this search method, a query typically consists of a few keywords. On receiving the keywords, search engines search for the occurrences of these keywords in their databases. Most search engines have separate *advanced search forms* where the user can be more specific by forming complex Boolean searches. Beyond that, some search engines parse HTML tags, allowing the user to look for things specifically as links, or as a title or URL without considering the text on the page.

2.2.2 Taxonomy-based Search Facilities

As described in Chapter 1, taxonomy-based search facilities are search services that pre-classify the stored information into a taxonomy and provide users a category-based search. These search facilities are useful when we only have general topics or we are not sure how to narrow our search from a general topic. 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 result, they seem good starting points for novice users or for those who lack background and domain knowledge to find useful information on the web.

Table 2.1 shows the major web directories. Yahoo! [Yah] is the best known web directory, which also provides news search service and search results from Google. ODP is the largest human-edited web directory with 3.8 million web sites classified into 460,000 hierarchically organized categories. Since ODP makes all of its data publicly and freely available through periodic RDF dumps, there are over 250 sites using the data, including major search engines Google and Lycos [Lyc]. With its ubiquity of content, ODP has replaced Yahoo in importance to web marketers. Looksmart [Loo] is a relatively small web directory about one-fourth the size of ODP, but its data is also used by several major search engines including Altavista and MSN [Cor].

Table 2.1: Major web directories.

Web directory	Sites (million)	Category	Editor	Note
Open Directory Project/ODP	3.8	+460,000	+52,500	• Data is publicly available • There are over 250 sites currently using ODP's data including Google
Yahoo!	1.8	N.A.	100	•Data is not publicly available
LookSmart	1	70,000	200	• Data is not publicly available, but currently used by several search engines including Altavista and MSN

Another type of taxonomy-based search facility is the newsgroup search service provided by Google. This search service indexes over 700 million messages classified into more than ten thousand hierarchically organized groups.

Beyond the keyword-based search approach, taxonomy-based search facilities use categories in the taxonomy as search contexts to process user queries. Users may specify a target category in addition to search keywords, and the search facilities do the search only against site entries in the category. This usually improves search precision of the taxonomy-based search facilities over those of the search engines.

2.3 Work for Web Search Improvement

2.3.1 Query Modification

Much research has been done to improve the web search process and result effectiveness. The first and most closely related to our work are those that use query modification based on categorical information. Inquirus 2 [G+01][GLG+99], developed at the NEC research institute, 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 effectiveness. The query is modified using a set of modification terms extracted from the document collection of the category using the expected entropy loss. They have recently extended the work by extracting the modification terms using SVMs [FGLG02]. Keyword-spice [OKI+01] also modifies a user query based on a specific category, but it uses a decision tree learning algorithm to construct the modification terms.

Our work differs from theirs in that we use an existing taxonomy and then dynamically construct classifiers to determine the user query intent. Using the taxonomy, the user can freely shift the broadness of intended topics by merely selecting an appropriate category from the taxonomy. On the other hand, they use flat categories that must be constructed and provided to users. Another drawback is that the query modification for each category is static;

it is fixed for all queries with the same category.

Another related work is to interactively modify the user query based on initial query results. This approach is used by Altavista Prisma [Pri] and is similar to the interactive query expansion (IQE) approach [Eft96] in information retrieval systems. Altavista scans the top initial result pages and automatically extracts the 12 most frequently occurring terms from them. It then presents the extracted terms to the user, who can select one of these terms to refine the search. Typical search queries, however, often return many noise documents, so the above approach is not effective in the real world.

Another similar interactive query modification method is the one used by AlltheWeb [Tra] and MSN [Cor]. With this approach, the system extracts common search terms from the previous users' queries. However, since the term extraction is based on keywords in the queries, it becomes very hard to provide the user good common terms that will fit sufficiently with the search intent. Further, the system will not be able to provide common terms if there are no previous queries related to the current user query.

2.3.2 Based on User Activities and Profiles

Another related work is to automatically infer context information from an everyday productivity application such as a word processor, to guide the web and database search. The Watson project [BH99] [LSBH99][BH00][BHBK00] and IntelliZap project [FGM+01] analyze the web pages/documents that are currently opened by the user to extract important terms from the pages/documents. The extracted terms are used to construct/modify a query sent to web search services. The Remembrance Agent [RM00][RS96][Rho00] is an Emacs plug-in that suggests information relevant to what the user is reading or writing. This system continually looks for documents (such as e-mail archives and notes files) on the web related to documents that users are reading or writing.

WebGlimpse [MSG97][Web] is a search engine that assists user browsing by merging the benefits of both the searching and browsing system. It does this by analyzing a given web archive and then computing the 'neighborhood' of the documents. It allows the search to be further limited to this neighborhood. Similar to WebGlimpse, Syskill & Webert [PMB96][PB97] also assists users in browsing and searching. With the system, a user first provides a topic name (e.g., machine learning, link analysis etc.) along with the URL of an index page. This index page is manually constructed and is simply any page with many links to other pages on the topic. The user then starts to

explore the links starting from the index page and gives the system feedback on the relevancy of each visited page. Based on the feedback, the system then constructs positive and negative examples, which are used to learn the user profile. Finally, in the next browsing activities, the system sugests links for the user to explore from current page. In the earlier version, the system uses Bayesian classifier [DH73] to learn the profile. Recently, it has been extended to use different classification learning algorithms like ID3 [Qui86], nearest neighbour [DH73], perceptrons [GWH60] and multilayer networks [RHW86] trained with error propagation.

The above systems do not infer the search context from taxonomy. Some use vector space model to infer the context or user profiles while others use the existing machine learning techniques. None of their inference processes can be controlled by the user.

Interactive query learning system was proposed in [CS96] to keep resource directories/index pages up-to-date. The user via an augmented web browser specifies positive and negative examples incrementally for the current topic. The system can be instructed to create new rules by using the positive and negative examples that has been collected. The resulting rules are then transformed into a query for web search interfaces to detect any new instances that may be added in the specific resource directories. The main focus of this system is query generation rather than query modification.

2.3.3 Automatic Web Page Classification and Collecting

Many attempts have been made to classify web content automatically into taxonomy [CDAR98][DC00][WZL99]. The main goal is to deal with the exponential growth in the volume of online text databases. These attemps start with a small sample of corpus that is classified by hand to build a hierarchical classifier. At run time, each web page retrieved by crawlers is classified automatically by the classifier into an appropriate category. [CDI98] goes further by using hyperlink information of web pages in addition to their textual information. For a static and relatively small taxonomy, the above approaches could be very useful. However, it seems very difficult to build and maintain a hierarchical classifier corresponding to a dynamically changing taxonomy with tens of thousands categories, such as the one used by the existing taxonomy-based search facilities.

Another similar approach is to use topical crawlers or spiders to collect web pages related to a particular topic. These crawlers select and index only web pages of interest and avoid the hyperlinks that lead to any off-topic areas. Focused crawler [CvdBD99b][CvdBD99a] is a hypertext resource discovery system that seeks, acquires, indexes, and maintains web pages on a specific set of topics. This system is guided by a classifier that learns to recognize relevance from examples embedded in a topic taxonomy, and a dis-

tiller that identifies topical vantage points on the web. Arachnid [Men97] is a distributed algorithm for information discovery based on a distributed, adaptive population of intelligent agents making local decisions. This algorithm advocates the use of competitive, reproducing and mutating agents for finding information on the web. Other work that falls into this category is CI Spider [CZC01] and Context Graph [DCL+00].

BINGO! [SSTW02] is a new type of focused crawling that starts from user bookmarks as the initial training data and crawling seed, but enlarges the training data as it crawls along and classifies newly visited documents. To this end, it indentifies, among the crawled and positively classified documents of a topic, characteristic "archetypes" and uses them for periodically retraining the classifier; this way the crawler is dynamically adapted based on the most significant documents seen so far.

The above approaches deal with improving web directory coverage and the document collection quality of specialized search engines. They do not work to improve the quality of user queries.

2.3.4 Search Result Clustering and Filtering

Another approach to improve the web search process is to cluster or filter the search result. [ZEMK97][ZE98] cluster short snippets returned by search

engines. They show that clusters based on the snippets are as good as those created using the full text of web documents. To provide reasonable small clustering time, they propose an incremental, linear time algorithm called *Suffix Tree Clustering* (STC), which creates clusters based on phrases shared between documents. In their implementation system, called Grouper [ZE99], a user can simply select an interesting cluster/phrase and then zoom in on it.

Scatter/Gather interface [HKP95] uses text clustering as a way to group documents according to overall similarities in their content. Scatter/Gather is so named because it allows the user to scatter documents into clusters, or groups, then gather a subset of these groups and re-scatter them to form new groups. Each cluster in Scatter/Gather is represented by a list of topical terms — a list of words that attempt to give the user the gist of what the documents in the cluster are about. The user can also look at the titles of documents in each group. The documents in the cluster can have other representations as well, such as summaries, or TileBars [Hea95].

Automatic classification of web documents into pre-specified categories was studied in [CGRU96]. They start by building a classifier for a set of categories using a 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 of interest. The system retrieves documents that match the query,

and then filters them by comparing their pre-computed categories against those chosen by the user.

NorthernLight [Div] is a search engine that presents its search result with a hierarchical Custom Search Folders. These folders are created dynamically from the search results and can sort the results by one of four types: subject, type, source and language. Teoma [Ask] also presents its search result in the same way, but it clusters/organizes the result into naturally occurring communities related to the subject of each search query.

Information filtering agents, such as Amalthaea [Mou96][MM98], WebMate [CS98] and NewsWeeder [Lan95], work to provide a user only those sites/pages that are interesting and relevant. It is done by removing irrelevant and unwanted information from a flow of information to the user. Typically, these agents try to find pertinent information based on a user profile of interests and needs.

The above systems only cluster/filter the search result or an information flow. They are neither consider query modification nor use existing taxonomy.

2.4 Classification Learning Algorithms

In this section we describe two typical supervised learning algorithms: *decision tree* and *classification rule*. These algorithms are not only popular in machine learning and other disciplines for solving classification and related problems, but are also frequently used in the web search domain [OKI+01][IGS01][Coh96][CS96]. This is due to the easiness to transform the resulting classifier/hypothesis[1] into a Boolean condition that can be processed by many web search services.

The learning algorithms take a set of instances whose class labels are known (denoted as *training set*) and derive a hypothesis that best fits the training set. For the decision tree learning algorithm the hypothesis takes the form of a flow-chart like tree structure; the classification rule learning algorithm takes the form of If-Then rules. The hypothesis will be used to classify unseen/unlabeled instances.

2.4.1 Decision Tree

Among the many decision tree learning algorithms proposed, ID3 and its successor C4.5 are well known state of the art algorithms [Qui86][Qui93].

[1] The terms 'classifier' and 'hypothesis' are used interchangeably in this dissertation.

Figure 2.1 shows the ID3 algorithm. Given a training set ($TSet$) and their instance attribute set ($attSet$), ID3 builds a decision tree in top-down fashion. After creating a *Root* node (line 1), to expand the tree, the algorithm selects the best attribute based on a statistical measure (line 10), called *information gain* [Qui86]. The information gain measure evaluates how well a given attribute separates examples in the training set according to their classes. For each value of the best attribute, ID3 creates a tree branch (line 13) and divides training examples according to the attribute value (line 15). Using the subset of the training examples as a base, it then decides whether to stop expanding the tree by creating a leaf node below the branch (line 17) or to continue expanding the tree by creating a new subtree below the branch (line 20). The subtree is created by recursively calling function ID3 with the training examples and unused attributes. ID3 will also stop expanding the tree if the gain is zero (lines 3 and 5) or there are no unused attributes in $attSet$ (line 7).

Classification of an instance begins from the root node with testing of the attribute specified at this node. Based on the instance attribute value, we move down the tree branch to the subsequent node. This process is then repeated for the subtree rooted at the node. When we reach a leaf node, the instance is classified according to the class label of the node. For example, Figure 2.2 shows an example of a decision tree that classifies an instance into two classes $\{I, R\}$ by testing attributes $\{a, b, c\}$, where each

```
ID3 (TSet, attSet, C)
1 :  Create a Root node for the tree;
2 :  If all examples in TSet are from class C
3 :    Return a tree with a single node labeled C;
4 :  If all examples in TSet are not from C
5 :    Return a tree with a single node labeled not C;
6 :  If attSet is empty
7 :    Return a tree with a single node labeled by the
8 :    majority class of TSet;
9 :  Otherwise
10:    A ← Find the best attribute from attSet;
11:    Decision attribute for Root ← A;
12:    For each possible value, i, of A
13:      Add a new tree branch below Root corresponding
14:      to test A = i;
15:      TSet_i ← Subset of examples satisfying test A = i;
16:      If TSet_i is empty
17:        Below this branch create a leaf node labeled by
18:        the majority class of TSet;
19:      Else
20:        Below this branch add a subtree ID3(TSet_i, attSet\{A}, C);
21:  Return Root;
```

Figure 2.1: ID3 learning algorithm.

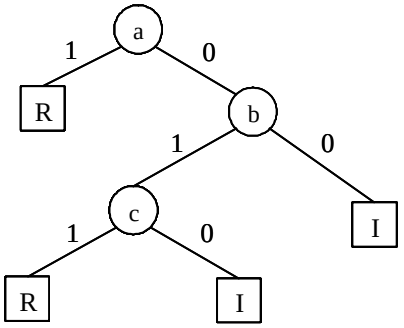

Figure 2.2: An example of a decision tree.

attribute has values $\{0, 1\}$. According to the procedure above, an instance $(a : 0, b : 1, c : 1)$ is classified as R.

Many extensions have been made to the algorithm [Qui93][Mit97]. These include tree pruning strategies to avoid overfitting the training examples, incorporating continuous-values attributes, using other attribute selection measures such as *Gain Ratio* and handling training examples with missing attributes.

2.4.2 Classification Rule

Most classification rule learning algorithms use a *separate-and-conquer strategy*. The term separate-and-conquer comes from a strategy for learning: learn a rule that covers a part of the given training examples, remove the covered examples from the training set (the *separate* part) and recursively learn another rule that covers some of the remaining examples (the *conquer* part) until no examples remain.

Typical rule learning algorithms induce a set of rules in the form of "$H \mapsto C$", where H is a conjunction of attribute tests and C is the target class name. Figure 2.3 shows the typical classification rule learning algorithm. It starts from an empty rule set (line 1) and successively adds rules to it (line 10) until all examples from a given target class C are covered.

The learning of a single rule starts with a rule whose head is a *true* condition (line 3). As long as it still covers examples from classes other than C, the current rule is specialized by adding attribute tests to its head (lines 5–9). To find an 'optimal' rule that only covers instances from target class C, the algorithm selects an attribute test that maximizes the purity of the rule, i.e., a test that maximizes the percentage of examples from C covered by the rule (lines 7 and 8). When a rule has been created, it is added to the rule set (line 10) and all covered examples are removed from $TSet$ (line 11). Another rule is then learned from the remaining examples. The procedure is repeated until no examples from C remain. Thus it is to ensure that the learned rules together cover all examples from C (*completeness*), but no examples from other classes (*consistency*).

The induction of complete and consistent rules can lead to overfit the training examples if the data are noisy. To prevent overfit, some improvements have been made to the rule learning algorithm by applying *rule pre-pruning* techniques that add additional stop criterion [CN89][TC96], *rule-post pruning* techniques that simplify the rules in a post-processing phase [MMHL86][BMMZ92][PH90] and a combination of both techniques [FW94][Coh95].

```
TypicalRule (TSet, attSet, C)
1 :  ruleSet ← ∅;
2 :  While TSet contains examples from class C
3 :    Create an initial rule R : true ↦ C;
4 :    Do until R is perfect
5 :      For each A ∈ attSet, and each value i
6 :        R' ← add test A = i to the head of R;
7 :        If Purity(R') > Purity(R)
8 :          bestRule ← R';
9 :      R ← bestRule;
10:    ruleSet ← ruleSet ∪ {R};
11:    TSet ← TSet\Cover(R, TSet);
12:  Return ruleSet;
```

Figure 2.3: Typical classification rule learning algorithm.

Chapter 3

Problem Definition

3.1 Overview

This chapter starts by describing several problems faced by contemporary web search services: search engines and taxonomy-based search facilities. Problems in search engines originate from the use of keyword-based search while those in the taxonomy-based search facilities originate from the manual classification of web sites into the taxonomy. The chapter then describes the goal of this dissertation and the challenges. Finally, it describes the basic idea to achieve the goal and points out criteria that must be met.

3.2 Contemporary Web Search Problems

As mentioned in the previous chapter, search engines use crawlers to gather web pages from the Internet. This process is usually done automatically with only a bit of human intervention; the result is that web coverage of the search engines grows quite large. The crawled information is stored/indexed as a group of keywords associated with the page and no topic organization of the pages is made. This makes the search engines typically support keyword-based search, which returns pages containing the given search keywords.

Keyword-based search can potentially have many limitations. There could be *ambiguity* in the case of synonymy (different words have the same meaning). It could also be *vague* in the case of polysemy (a single word has different meanings). Another problem is that users usually express their query intent with only a few keywords, which renders the search engines unable to completely understand what the users want. The three factors together seriously hinder search engine performance, and a strategy that ranks the search results according to a specific ranking algorithm does not help a great deal.

Taxonomy-based search facilities such as web directories take another approach to store their information. They utilize taxonomy to index the stored information. Pages/site entries with similar topics are (logically)

stored/grouped in the same category on the taxonomy in addition to being indexed according to the terms they contain. This approach has the following advantages that make the search precision of the taxonomy-based search facilities usually better than that of the search engines. First, they can use *category-based search* in which the user can narrow the search scope by specifying a category on the taxonomy. Second, they give the user another option (an option to specify a category of interest) to express the query intent in addition to specifying search keywords.

Usually, web editors manually classify pages/site entries into categories in the taxonomy. Information providers submit their site information to the search facilities and the web editors then classify the sites into an appropriate category. Manual classification results in web coverage of the taxonomy-based search facilities being small compared to search engines. In addition, use of the maintained taxonomy is restricted to searching information stored in an own data set and not applicable to other web search contexts.

3.3 Problem Definition and Basic Idea

Our goal is to use document collection and taxonomy in the existing taxonomy-based search facilities to facilitate searches in other web search services (i.e., search engines). We achieve this by letting the user formu-

late queries via a taxonomy-based search facility and letting search engines process the query. We make it possible for the user to improve queries by formulating them via the taxonomy-based search facility (search keywords associated with context information).

More precisely, first we learn/extract the user query intent from the taxonomy-based search facility using the given search keywords and context information. The extracted query intent is then used to enrich/modify the user query (i.e., search keywords). The motivation behind this is that initial search keywords usually carry less information about the user query intent. So by enriching them with the extracted query intent, search result effectiveness from the search engines can be greatly increased. The challenge is to propose a query modification method that will bridge the gap between the two different search services. Several requirements should be met here.

1. The method should *dynamically* modify the user query so that the modified query can precisely meet the user query intent.

2. The method should be aware of query processing constraints imposed by the target search engines. That is, it must modify the user query so that the resulting modified query can always be processed by the target search engines.

3. The method should be flexible to user requirements on search result

effectiveness. This is needed because different users have differing requirements on how their search result should be delivered to them.

Note that none of the existing modification methods surveyed in Chapter 2 satisfy the above requirements.

Chapter 4

Proposed Query Modification Framework

4.1 Overview

This chapter discusses the proposed query modification framework [PK02b][PK02a]. Figure 4.1 shows the search procedure in the framework. It consists of three steps: the *taxonomy browsing and query formulation step*, the *taxonomy probing step* and the *query modification and execution step*. In the taxonomy browsing and query formulation step, the user browses taxonomy to formulate a query condition and select a context category. The user also specifies additional parameters and a target web search engine. In the taxonomy probing step, data in the taxonomy-based search facility are

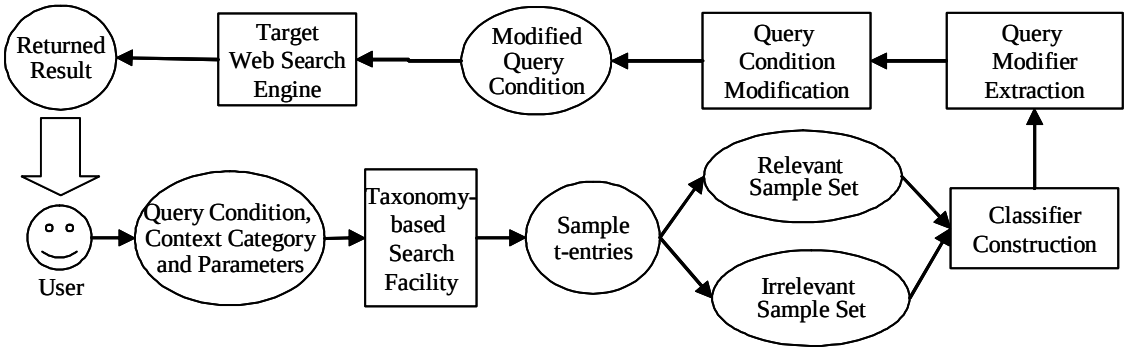

Figure 4.1: Procedures in the query modification framework.

sampled. In the query modification and execution step, a classifier is constructed using the sampled data and the given query condition is modified. The modified condition is then sent to the target web search engine and the result is returned to the user.

We assume that the taxonomy-based search facility satisfies the following conditions. Most of the major web directories meet 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*[1].

2. Accepting a Boolean query 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

[1]In typical taxonomy-based search facilities, a t-entry is a combination of URL and short description of a web site.

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. Note that the number of matched t-entries is given at the beginning of the returned result.

For simplicity, we assume that the modified query is sent to one target web search engine. Extending it to multiple targets is trivial. We also assume that the target web search engine accepts Boolean queries.

4.2 Taxonomy Browsing and Query Formulation Step

In the taxonomy browsing and query formulation step, a user interactively browses the taxonomy and selects an appropriate category (denoted as *context category*) that matches the query intent. The user then defines a query condition Q to be sent to the system. 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). The user can also specify search parameters to control the modification and search process.

4.3 Taxonomy Probing Step

On receiving a query (Q, G) from the user, the system uses the query to sample the matched t-entries. 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 procedure. 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 facility. 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 taxonomy-based search facility. 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*. They are fetched so that they represent even samples from all the top-level categories g_i. (If g_i itself is the context category, it is excluded.)

When the context category is a subcategory of g_i, the search result may contain both relevant and irrelevant t-entries. The t-entries are separated by inspecting associated category information.

Note that p is the minimum bound for the number of relevant and irrelevant samples and that q is the number of additionally fetched samples. We provide p to ensure that the system gets a reasonable number of relevant and irrelevant samples to construct a classifier, even if the distribution of relevant/irrelevant t-entries is significantly skewed. Note that, when $N_1 < p + q \cdot N_1/(N_1 + N_2)$, the number of relevant samples is considered too small to modify the query. When $N_2 < p + q \cdot N_2/(N_1 + N_2)$ but $N_1 \geq p + q \cdot N_1/(N_1 + N_2)$, the query condition is left unchanged and sent directly to the target web search engine[2].

4.4 Query Modification and Execution Step

In the query modification and execution step, the system first creates a classifier using the relevant and irrelevant samples fetched in the previous step. The classifier is used to distinguish the *relevant class* from the *irrelevant class*. The relevant class is a class for the relevant samples while the irrelevant class is for the irrelevant samples.

[2]The system need not modify a query condition if it is already focused.

The system then extracts a Boolean condition M called *query modifier* from the classifier, which is used to modify the initial condition Q. Q is modified by AND-ing it with M (i.e., $Q \wedge M$). Finally, the modified query (condition)[3] is sent to the target web search engine and the returned result is presented to the user.

An Example

As a concrete example, suppose that the query is ("ATM $\wedge$ company", "/Computers/Networks/"), where "ATM $\wedge$ company" is a query condition and "/Computers/Networks/" is a context category. After the taxonomy probing step, the system gets t-entries, most of which contain information about the asynchronous transfer mode as relevant samples; most of the others contain unrelated information (e.g., automated teller machines) as irrelevant samples. In the query modification and execution step, the system creates a classifier and extracts a query modifier to distinguish the relevant class from the irrelevant class. Let the query modifier be "networks $\wedge$ (switch $\vee$ asynchronous)". Thus, the modified query is "ATM $\wedge$ company $\wedge$ (networks $\wedge$ (switch $\vee$ asynchronous))".

By sending the query condition with the query modifier (i.e., modify the

[3]The terms *modified query condition* and *modified query* are used interchangeably in this dissertation.

query condition with the query modifier) to the target web search engine, we get topic-focused information relevant to the query condition and the context category. Note that the query modifier is created dynamically based on the given query condition and context category. This dynamic characteristic is very important because it can significantly increase the search result effectiveness (as experimentally validated in the subsequent chapters). The meaning of a query depends strongly on its query condition and context category. So two queries with the same query condition but different context categories usually have different meanings, e.g., ("ATM $\wedge$ company", "/Computers/Networks/") and ("ATM $\wedge$ company", "/Business/Banking_Services/"). The same holds for queries with the same context category but different query conditions, e.g., ("ATM $\wedge$ company", "/Computers/Networks/") and ("Windows", "/Computers/Networks/").

Chapter 5

Basic Query Modification Method

5.1 Overview

This chapter describes a basic query modification method [PK02c] based on the proposed query modification framework. The classifier in the basic query modification method is built using the existing learning algorithms. Any learning algorithm can be used as the learner as long as a Boolean condition can be derived from the learned classifiers. Examples of algorithms are RIPPER [Coh95], CN2 [CN89] and C4.5Rule [Qui93].

The chapter also evaluates retrieval performance of the basic method

to show its effectiveness. We use RIPPER in the performance evaluation because it is a fast and accurate rule learning algorithm. Finally, the chapter discusses limitations of the basic method and suggests enhancements needed to put the proposed framework into practice.

5.2 RIPPER Rule Learning Algorithm

RIPPER is a variant of the typical classification rule learning algorithm described in Section 2.4.2. It too is a separate-and-conquer algorithm that constructs one rule at a time and removes all examples covered by a new rule as soon as the rule is constructed.

The main difference between RIPPER and the typical rule learning algorithm is that RIPPER splits the training set into two parts, where the first part is used to construct a rule as in the typical algorithm and the second part is used to prune the rule. Another difference is that RIPPER adds a new stop condition based on a Minimum Description Length (MDL) [Qui95] and compresses the final rule set before returning it.

Figure 5.1 shows the main procedure in RIPPER. It starts by splitting the uncovered examples ($TSet$) into $GrowData$ and $PruneData$ (line 3). It then creates a rule using the $GrowData$ (line 4), and then simplifies or

```
IREP* (TSet, C)
1 :  ruleSet ← ∅;
2 :  While TSet contains examples from class C
3 :    Split TSet into GrowData and PruneData;
4 :    R ← GrowRule(GrowData);
5 :    R ← PruneRule(R, PruneData);
6 :    ruleSet ← ruleSet ∪ {R};
7 :    TSet = TSet\Cover(R, TSet);
8 :    If DL(ruleSet) > DL(ruleSet_opt) + d;
9 :      Return Compress(ruleSet, TSet);
10:  Return Compress(ruleSet, TSet);
```

Figure 5.1: RIPPER rule learning algorithm.

prunes the created rule by using the $PruneData$ (line 5). The rule creation process is similar to the typical algorithm. That is, a rule is "grown" by repeatedly adding attribute tests to an initial rule with an empty head. The rule pruning process considers the deletion of any final sequence of attribute tests from the head of the rule, then chooses the deletion that maximizes the function $f(R') = \frac{p-n}{p+n}$, where p (n) is the number of examples from class C (classes other than C) in the pruning set covered by the new rule R'. After pruning the rule, RIPPER adds the pruned rule to the rule set (line 6) and removes the examples covered by it (line 7).

RIPPER stops creating rules when the description length of the current rule set is more than d bits larger than the smallest description length obtained so far (lines 8 and 9) or when there are no more examples from class C in $TSet$ (the While loop has completed). It then compresses the rule set by

examining each rule in turn, starting with the last rule added, and deleting any rules that increase the description length.

Using RIPPER in the Proposed Framework

The classifier constructed by RIPPER is used to classify t-entries into relevant and irrelevant classes. A t-entry is regarded as a tuple and each attribute represents the presence or absence of a word in the t-entry as a binary feature. We use the set of the relevant and irrelevant samples obtained in the taxonomy probing step as the training set ($TSet$) given to RIPPER. We label relevant sample t-entries as *relevant* and irrelevant ones as *irrelevant*.

A query modifier M used to modify an initial query condition Q is extracted from the learned classifier as follows. Let $Rule = \{r_1, \cdots, r_n\}$ be the rule set for the relevant class, where $r_i = H_i \mapsto Relevant$. Then, M is $H_1 \vee \cdots \vee H_n$.

5.3 Experimental Evaluation

5.3.1 Overview

We evaluate the basic query modification method with experiments. In particular, we make the following comparisons and measurements.

- We measure the *query modification time* of the basic method with respect to different numbers of t-entries fetched in the taxonomy probing step (different (p, q) parameter values). We compare them with the modification time of the case in which we fetch most of the matched t-entries. For clarity, let us call the former *p_q partial probing* and the latter *full probing*. This experiment is explained in Section 5.3.3.

- Similar to the first point, we also measure *search result effectiveness* of the basic method with various probing types. Search result effectiveness is given by the *generalized effectiveness measure* (*G-measure*) [D$^+$02] shown below.

$$G - measure = \frac{1}{\alpha \cdot (1/recall) + (1 - \alpha) \cdot (1/precision)} \qquad (5.1)$$

  where we can trade-off *precision* and *recall* by adjusting α ($0 \leq \alpha \leq 1$). We also show the relationship between the sample number and the modification time. This experiment is explained in Section 5.3.4.

- To show effectiveness of the basic method's *dynamic behavior*, we compare its search result effectiveness with that of a static query modification method. In the static method, advanced preparation of a classifier is assumed for each context category using a data set associated with the taxonomy. In the method, the same classifier is used for a given context category, regardless of the query conditions given by the user. This experiment is explained in Section 5.3.5.

- We measure the *document level precision* of queries modified by the basic method using Altavista [Alt], a real web search engine. This experiment is explained in Section 5.3.6.

5.3.2 Evaluation Method

To calculate the *G-measure* 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 facility in this experiment. Since each t-entry in the result returned by the taxonomy-based search facility 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 classifier construction and performance evaluation.

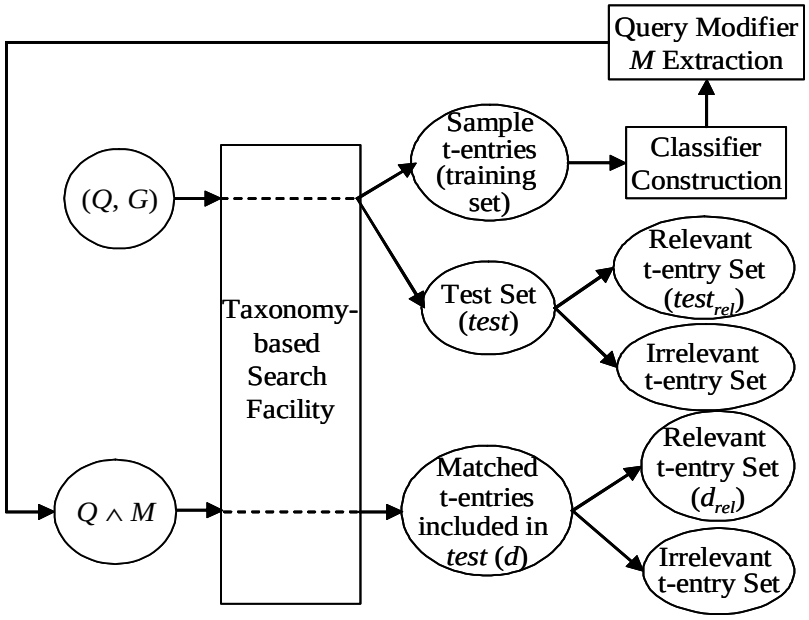

Figure 5.2: Evaluation method (G-measure calculation).

Figure 5.2 shows the flow of the experiment. First, a query (Q, G) is defined and sent to the taxonomy-based search facility. 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 [2]. 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[3]. The training set is used to construct a classifier, and a query modifier M is extracted from it to modify the initial query condition Q.

[2]In the experiment, we send $(Q, null)$ to the taxonomy-based search facility, get all the matched t-entries, and divide them into the training and test sets randomly using the ratio 2:1.

[3]In the experiment, we construct the test set by randomly taking t-entries fetched in the full probing that are not included in the training set of the p_q partial probing.

The *modified query* $Q \wedge M$ is then sent "back" to the taxonomy-based search facility 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 ($prec_{init}$) are given below. (Note that *recall* of Q is always 1).

$$precision = \frac{|d_{rel}|}{|d|} \qquad recall = \frac{|d_{rel}|}{|test_{rel}|} \qquad prec_{init} = \frac{|test_{rel}|}{|test|}$$

We also calculate the query modification time. It is the time to sample the matched t-entries from the taxonomy-based search facility and to construct a classifier.

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 [Net] as the taxonomy-based search facility. We define 50 queries, which are divided into two types: 25 queries with *broad context categories* and 25 queries with *narrow context categories*. We say a context category is broad if it is a direct subcategory of a top category of the taxonomy; we define it as

Table 5.1: Some queries and their meanings.

Query 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 trees (farming)
Nepal	/Recreation/ Travel/	Travel information of Nepal (including travel business)	/Recreation/Travel/ Travelogues/	Personal travelogues of Nepal
oil ∧ product	/Shopping/ Health/	Business in oil products for health	/Shopping/Health/ Beauty/	Business in oil products for beauty
first ∧ 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 ∧ product	/Business/ Industries/	Fabrication of oil finished products (e.g., petroleum and cooking oil)	/Business/Industries/ Energy/	Fabrication of oil finished products related to energy (e.g., oil and gas)

narrow if it is a subcategory of a broad context category. Table 5.1 lists 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. As mentioned before, the meaning of each query depends on the query condition and the context category. This being the case, the meanings of queries with narrow context categories differ from those with broad context categories. More precisely, the meanings of queries with narrow context categories are more specific than those with broad context categories. We divide the queries in this way because we also want to evaluate dynamic behavior of the proposed method. Terms in the query conditions are taken from the popular terms used in Google [Zei] and Yahoo! [Buz].

Since ODP provides 20 t-entries on each result page, for partial probing, we set $p = 20$ and $q = \{0, 80, 160, 240, 320\}$. The value of α is set to $\{0,$

0.25, 0.5}.

5.3.3 Evaluation of Query Modification Time

Figures 5.3 and 5.4 show the query modification time for various probing types averaged over 25 queries with broad and narrow context categories. The figures show that the query modification time of partial probing is much smaller than that of full probing. This occurs because the partial probing get only a fixed number of matched t-entries, which is usually much smaller than the number of all matches. We see also that the learning time is much smaller than the probing time. Beyond that, no significant difference exists between the modification time for queries with the broad and narrow context categories because each probing type always fetches the same number of t-entries in the two context categories.

5.3.4 Evaluation of Search Result Effectiveness

Figures 5.5 and 5.6 show the *G-measure* ratio averaged over 25 queries with the broad and narrow context categories. The *G-measure* ratio is the relative *G-measure* value taking *G-measure* of the initial (not modified) query condition Q as the base. It is obtained by dividing the *G-measure* value of the modified query $(Q \wedge M)$ by that of the initial query condition (Q), then

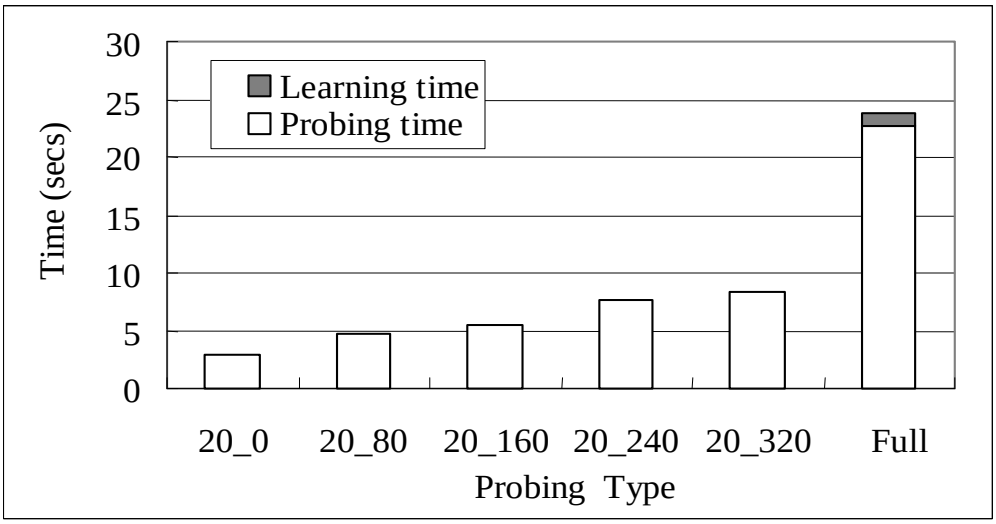

Figure 5.3: Query modification time for various probing types and queries with broad context categories.

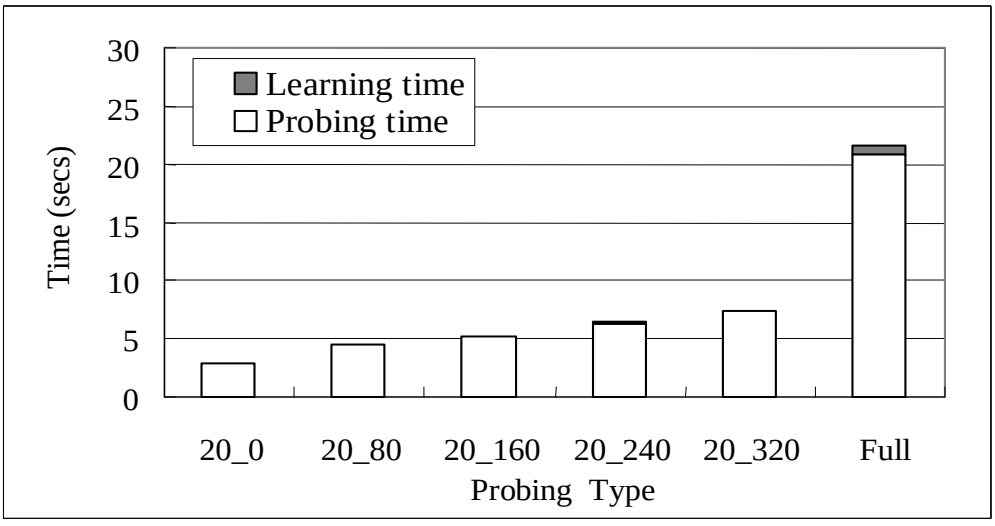

Figure 5.4: Query modification time for various probing types and queries with narrow context categories.

taking the average over 25 queries.

All probing types can significantly increase search result effectiveness of the modified queries. The increase in the *G-measure* ratio of queries with narrow context categories is greater than that of queries with broad context categories. In addition, the *G-measure* ratio increases with an increase in sample size. This occurs because, as the sample size increases, so does the accuracy of the classifier.

Figures 5.7 and 5.8 plot the query modification time versus the *G-measure* ratio for various probing types and queries with broad and narrow context categories. 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. 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

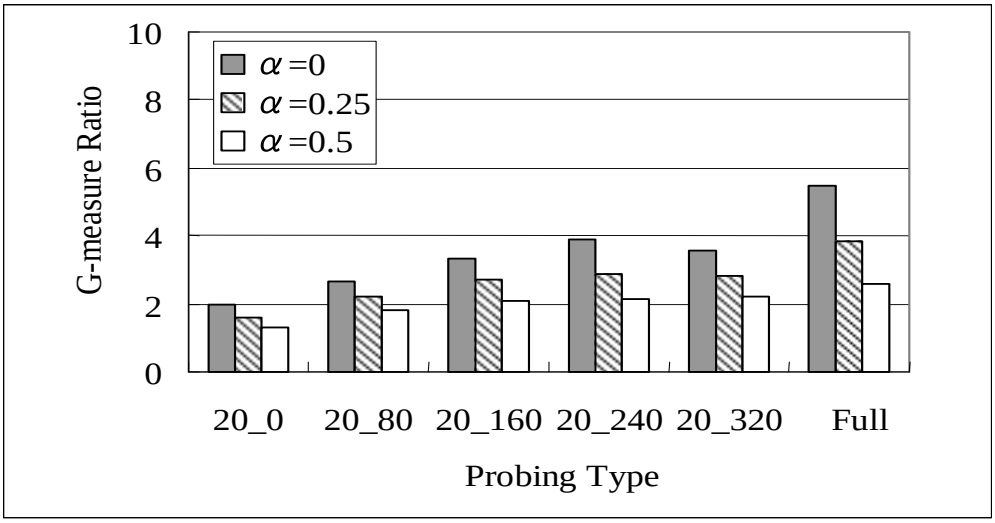

Figure 5.5: G-measure ratio for various probing types and queries with broad context categories.

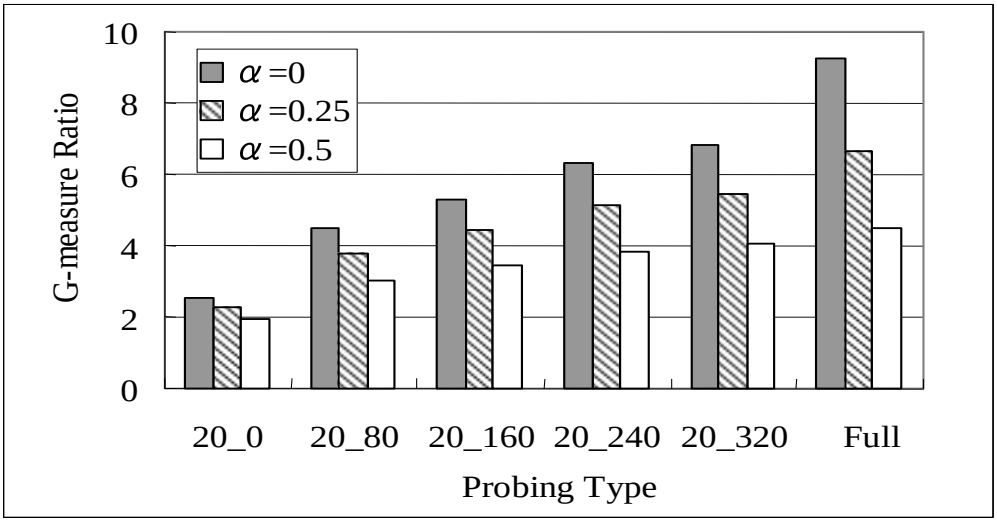

Figure 5.6: G-measure ratio for various probing types and queries with narrow context categories

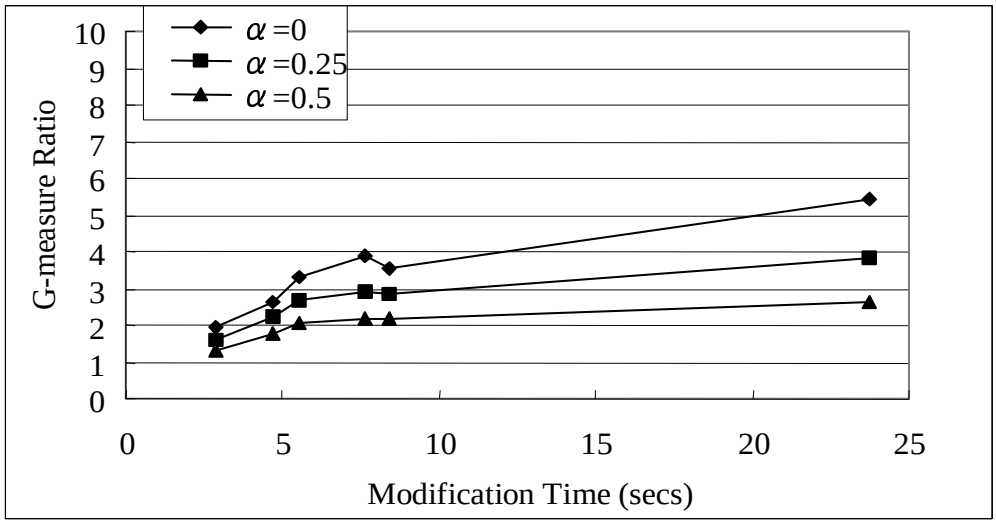

Figure 5.7: Query modification time and G-measure ratio for various probing types and queries with broad context categories.

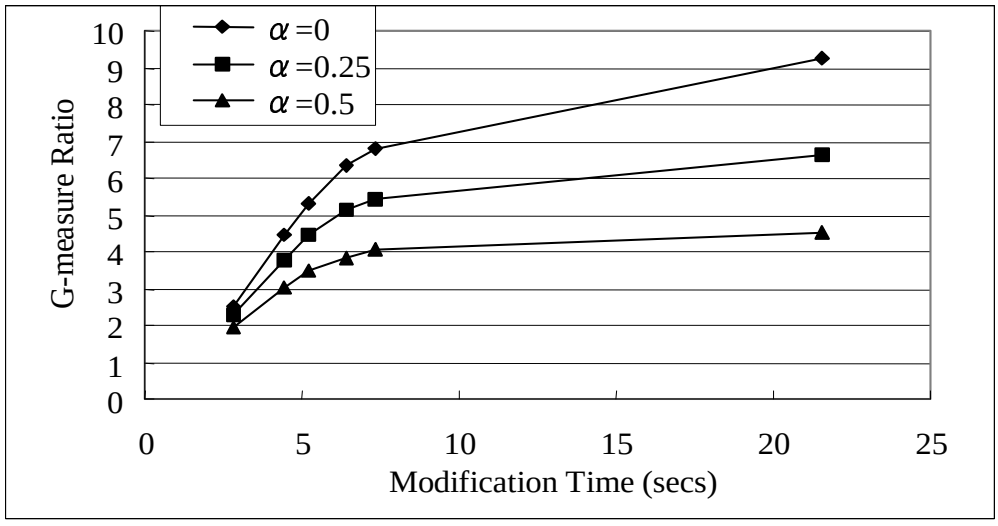

Figure 5.8: Query modification time and G-measure ratio for various probing types and queries with narrow context categories.

the parameters.

5.3.5 Comparison with a Static Method

In this experiment we compare the basic query modification method with a *static method*. The static method modifies a query with a pre-computed fixed query modifier. That is, different queries with the same context category are modified by the same query modifier belonging to the context category. In this sense, this method is similar to [G+01][OKI+01] discussed in Section 2.3.1.

In the static method, the classifier for each category of a taxonomy is created before run time by treating t-entries in the category as relevant and those in the other categories as irrelevant. Note that the relevant/irrelevant t-entries in this context do not necessarily meet the query condition Q. In this experiment, the classifiers are built using the RIPPER algorithm described in Section 5.2. For simplicity, we construct classifiers only of the selected context categories. We collect relevant samples of a category by randomly taking 30 percent of t-entries from the category, but limiting the number to 6000 t-entries. We collect irrelevant samples from other categories so that, for broad context categories, the number of irrelevant samples is made three times larger than the number of its respective relevant samples. For narrow context categories, the number is made six time larger. This rel-

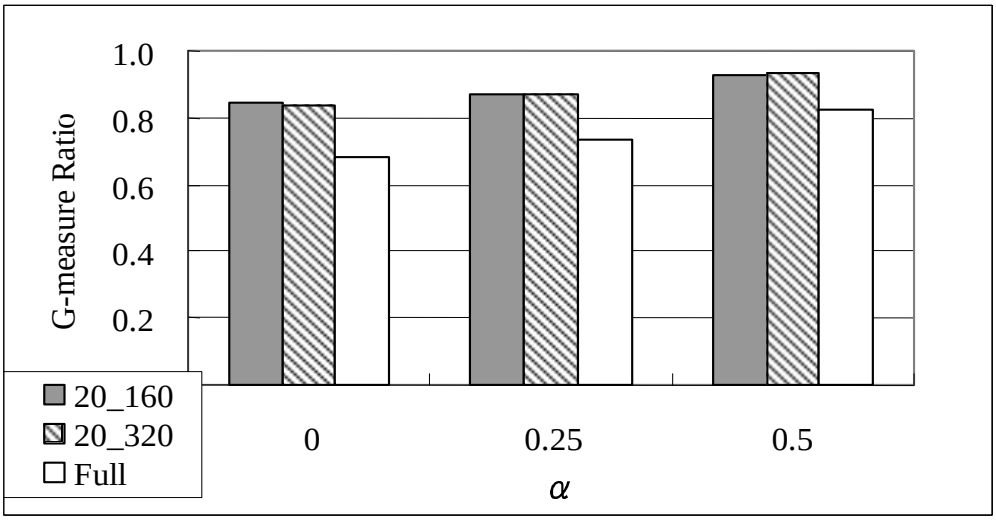

Figure 5.9: Comparison with a static method for queries with broad context categories.

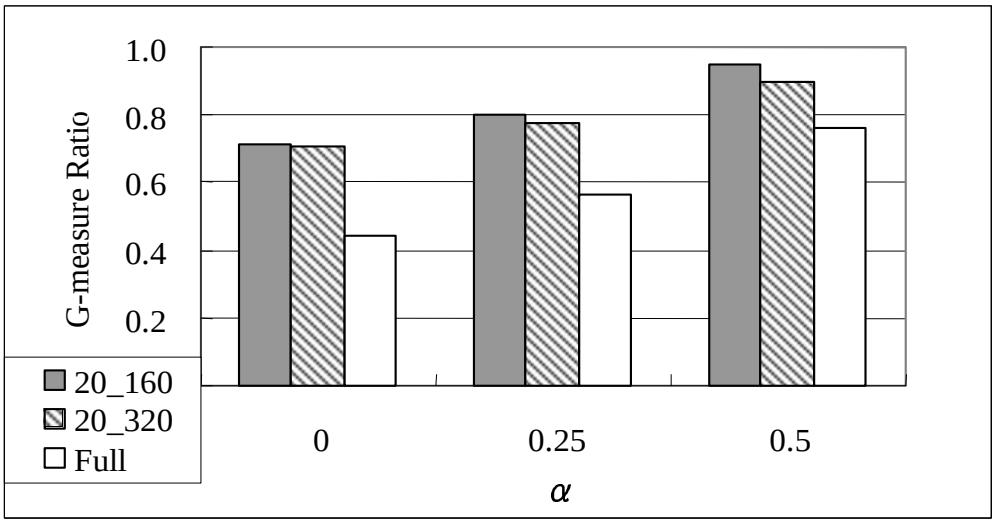

Figure 5.10: Comparison with a static method for queries with narrow context categories.

evant/irrelevant sample size ratio at the two context category types is the same with the relevant/irrelevant t-entry ratio at the contexts averaged over 25 queries.

Figures 5.9 and 5.10 show the *G-measure* ratio averaged over 25 queries with the broad and narrow context categories. The *G-measure* ratio here is the relative *G-measure* value, taking that of the modified queries using the basic method as the base. The results show that the basic method always outperforms the static method (i.e., the ratio is always less than one). The static method performs poorly because the fixed classifiers of the context categories are forced to cover many topics that may exist in the categories. As a result, the query modifiers derived from the classifiers cannot "fit" the queries well. In contrast, classifiers derived by the basic method need to cover specific topics implied by the given queries.

5.3.6 Evaluation with Altavista Search Engine

In this experiment we evaluate the performance of the basic query modification method with a real web search engine. We use Altavista [Alt] as the target web search engine and *document level precision* [DM96][CR96][GW96] as a performance measurement. Document level precision is computed after a given number of documents/matches in the ranked query result have been fetched. We calculate the precision until the cutoff 30 is reached. Relevance

is judged manually by directly checking whether cited pages conform to the meaning of the combined keywords given and the context category.

In this experiment, we again use ODP [Net] as the taxonomy-based search facility and take the 50 queries from the previous experiment. We set α to 0, and compare the basic method using the 20_320 and full probing with the initial query condition and static method.

Figures 5.11 and 5.12 show the precision averaged over 25 queries with the broad and narrow context categories. The document level precision of the basic method with partial and full probing is clearly better than that of the initial query condition and static method. Beyond that, no significant difference exists between the precision of queries with narrow and broad context categories. Again, performance of the static method is inferior because of its "stationary nature". Many less relevant terms to the query are included in the query modifier. This seriously impacts the document ranks returned by the search engine.

5.4 Limitations and Discussion

Many search engines are available on the web. Most accept Boolean queries, but they have differing acceptable query formats. For example, Altavista

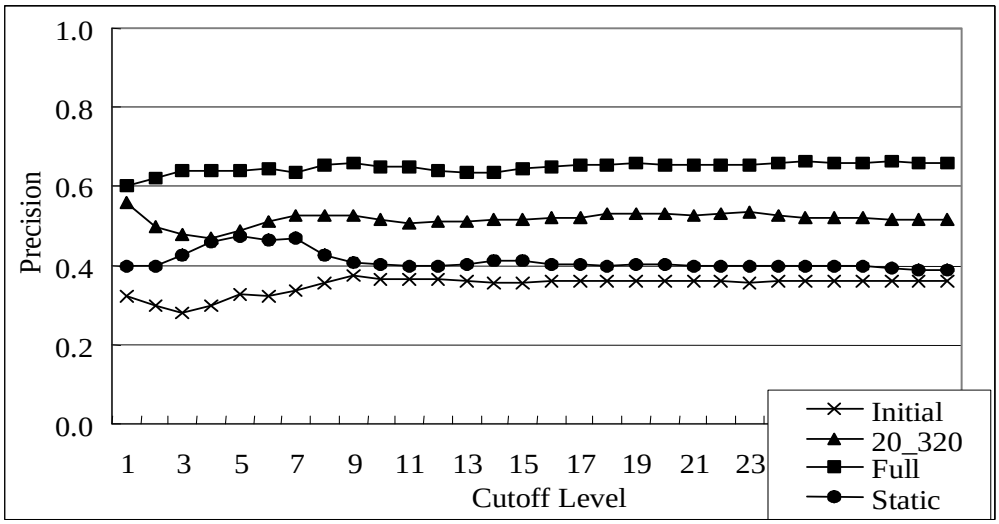

Figure 5.11: Precision of Altavista for queries with broad context categories.

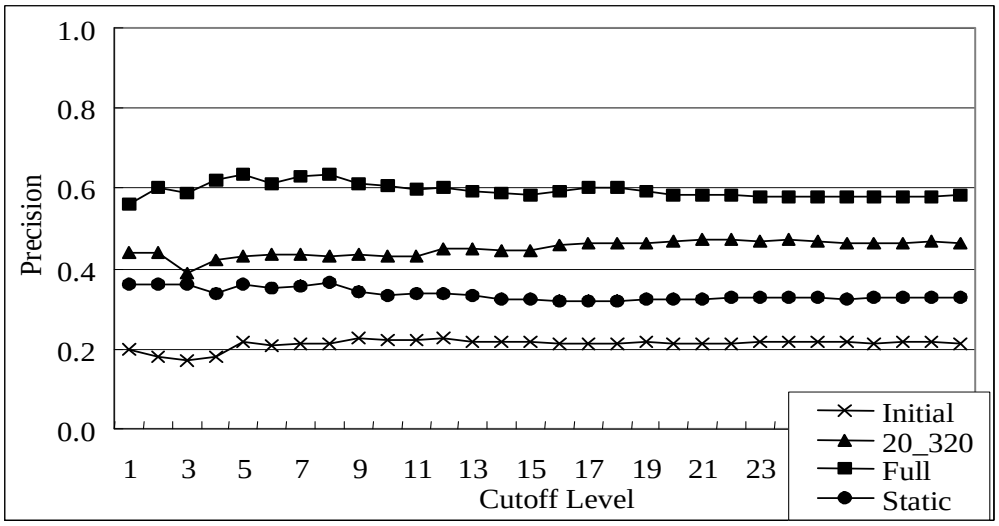

Figure 5.12: Precision of Altavista for queries with narrow context categories.

[Alt] and MSN [Cor] accept Boolean queries having nested expressions with parentheses (called *ordinary Boolean query format*). Usually, a query is directly formulated and put into a search field provided by the search engines. Another type of query format is the one supported by Google [Goo] and AlltheWeb [Tra]. In their advanced search facilities, a query is formulated indirectly using a *query template/form* consisting of fields associated with Boolean operators. For example, the advanced search of AlltheWeb provides a template consisting of fields labeled by "must include" corresponding to the AND operator, "must not include" corresponding to the NOT operator and "should include" corresponding to the OR operator. Hence, the expressive power of the Boolean query expression supported by the template-based query format is more limited than that supported the ordinary Boolean query format. Some search engines such as Google and AlltheWeb support only query expressions of this type, and cannot accept queries in the ordinary Boolean query format. For clarity, we call search engines accepting queries in the ordinary Boolean query format *ordinary Boolean search engines*; Search engines that *only* accept queries in the template-based query format is called *template-based search engines*.

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

The above description reveals that creating a classifier without considering the *format* and *size* of the extracted query modifier, such as done by the basic method, will not guarantee that the target web search engine can always process the modified query.

Another limitation of the basic method is that users cannot control search result effectiveness of the modified query sent to the target web search engine. This feature is important in the web environment because different users usually have different requirements on how they want their search result delivered. Both limitations originate from the use of the existing classification learning algorithm, which is not aware of search result effectiveness and query processing constraints of the target web search engine.

The key to the solution is obvious. We must control the learning process so that we can "shape" the query modifier derived from the learned classifier. To achieve control, we have developed two new classification learning algorithms: *Constrained Decision Tree* (CDT) learning algorithm and *Constrained Classification Rule* (CCR) learning algorithm. The algorithms are aware of the precision/recall measures and query processing constraints imposed by the target web search engine.

Using the two algorithms, we derive two query modification methods from the proposed framework: First, a query modification method that uses the

CDT learning algorithm to construct the classifier, called *CDT enhanced query modification method*. Second, we derive a query modification method that uses the CCR learning algorithm to construct the classifier, called *CCR enhanced query modification method*. The first is for the ordinary Boolean search engines; the second is for the template-based search engines.

Chapter 6

CDT Enhanced Query Modification Method

6.1 Overview

This chapter describes the CDT enhanced query modification method [PK03]. This method constructs a classifier using a proposed decision tree learning algorithm called *Constrained Decision Tree* (CDT). This algorithm differs significantly from the existing algorithms in two ways: First, it builds a decision tree by explicitly constraining its size. Second, it chooses the most promising decision tree based on the estimated search result effectiveness of the modified query. Specifically, it picks a decision tree that will lead to the maximum *G-measure* given by Equation 5.1, where a trade-off between

precision and *recall* can be made by adjusting α ($0 \leq \alpha \leq 1$).

These two points are crucial in the context of web search. First, target web search engines have different constraints on the acceptable query sizes as explained at the end of the previous chapter. Second, the decision tree here is used to modify the initial query condition Q. Therefore, the most promising decision tree should definitely be chosen on the basis of how well it can improve search result effectiveness of the modified query. Beyond that, the use of *G-measure* as the search result effectiveness measure gives the user an additional advantage. It gives the user an option in the process. When the user chooses a small (large) α value, more (less) emphasis is placed on the improvement of *precision*.

We evaluate performance of the CDT enhanced query modification method in the same way as with the previous experiments. We also evaluate effectiveness of the proposed algorithm by comparing it with an alternative algorithm, which is similar to the ID3 algorithm.

6.2 CDT Learning Algorithm

As with the basic method, t-entries sampled in the taxonomy-probing step are used as the training examples ($TSet$), which are split into two disjointed

subsets: *grow set* ($GSet$) and *validation set* ($VSet$). $GSet$ is used to build a decision tree, while $VSet$ is used to estimate the *G-measure* of the modified query derived. The *relevant subtree* (RST) of a decision tree is defined as a tree that contains all the nodes and branches on paths from the root to leaves labeled relevant. The number of branches in RST is referred to as its *size*. For the decision tree in Figure 2.2, its RST consists of the two leaves labeled R (relevant), three internal nodes, and the branches connecting them. The size is 4.

The algorithm is outlined in Figure 6.1. After splitting $TSet$ into $GSet$ and $VSet$ (line 1), the algorithm initializes the decision tree by creating a root node and determines the attribute set (line 2). Terms included in $GSet$ are extracted and go through stopword elimination. The information gain of each term is then calculated. Terms with the k largest information gain values are actually used for the (initial) attribute set for the decision tree construction. Others have reported that this preliminary feature filtering can significantly decrease decision tree construction time and boost classification accuracy [LR94][AD91].

The size of RST directly affects the size of the modified query condition. To guarantee that the size of RST meets the given constraint, the algorithm chooses only an expandable leaf node for tree expansion. Given the maximum relevant subtree size ($maxSize$), a leaf node n is considered *expandable* if the

```
CDT(TSet, maxSize, α)
1 :  Split TSet into GSet and VSet;
2 :  Create root node (root) and determine root.attSet;
3 :  root.GSet ← GSet;
4 :  Loop {
5 :    eNode ←expandable leaf node with the largest error rate;
6 :    If eNode or eNode.attSet is null, then exit the loop;
7 :    A ← the best attribute from eNode.attSet;
8 :    If A is null, then exit the loop;
9 :    For each possible value, i, of A
10:      Add a new tree branch below eNode corresponding
11:      to test A = i;
12:      Add a new leaf node lNode below the branch where
13:        lNode.GSet ← instances in eNode.GSet with A = i;
14:        lNode.attSet ← eNode.attSet\{A};
15:        Label lNode with the majority class in lNode.GSet;
16:    If G-measure(current RST) > G-measure(RST_opt)
17:      RST_opt ← current RST;
18:  }
19:  Return RST_opt;
```

Figure 6.1: CDT learning algorithm.

size of RST of the tree obtained by expanding the current decision tree at n is less than or equal to $maxSize$. Among the expandable leaves, the algorithm chooses the one with the largest positive *error rate*. Let S_R (S_I) be the set of relevant (irrelevant) samples associated with a leaf node in the current decision tree. Then, the error rate is given by $min(|S_R|, |S_I|)/(|S_R| + |S_I|)$. The rationale here is that leaf nodes with large error rates most likely harm purity of the classification; hence the algorithm gives higher priority to "repairing" the tree at those nodes[1]. This operation is done in line 5.

[1]We have experimented with *error count* ($min(|S_R|, |S_I|)$), but did not see improved

The algorithm then selects the best attribute from the attribute set of the selected expandable node ($eNode.AttSet$) based on the information gain measure (line 7) and expands the tree at the selected leaf node using the selected attribute (lines 9 to 15). This procedure is similar to the ID3 algorithm described in Section 2.4.1.

After the expansion process, the algorithm estimates *G-measure* of the query modified using a query modifier extracted from the new tree. It uses $VSet$, which does not overlap $GSet$. This is done to obtain a less biased estimation. First, it converts the RST of the tree into a query modifier (explained latter) and calculates *precision* and *recall* of the modifier for $VSet$. *G-measure* is then calculated using the given α. If the *G-measure* is bigger than that of an optimal RST obtained so far (RST_{opt}), then the RST is put into RST_{opt} (lines 16 and 17). At the end, the algorithm returns the optimal RST (RST_{opt}) (line 19). It meets, of course, the size constraint given by $maxSize$ (imposed by line 5).

The algorithm stops expanding the tree if there is no expandable leaf node with a positive error rate (line 6), no unused attribute in the selected expandable node (line 6), or no attribute in the selected expandable node with a positive information gain (line 8).

performance.

Query Modifier Extraction

The RST returned by the algorithm is converted into a query modifier that will be used to modify the initial query condition Q. The conversion can be done easily by traversing the RST. For example, the query modifier obtained from the RST of the decision tree in Figure 2.2 is "$a \vee (\neg a \wedge b \wedge c)$". Note that the number of literals in the query modifier equals the size of the RST. This being the case, we can guarantee that the number of literals in the modified query condition is always within $maxSize$ plus the number of literals in Q. Even if the size of acceptable queries is constrained by other restraints, such as the length of characters, the proposed algorithm can handle the constraint with only a minor change.

6.3 Experimental Evaluation

6.3.1 Overview

We evaluate the CDT enhanced query modification method with experiments. The evaluation method is similar to that of the previous experiment with different points given below.

- In measuring the modification time, the learning time is the time

needed to construct a decision tree.

- In comparing the CDT enhanced query modification method with the static method, the classifiers in the static method are built by using the CDT learning algorithm.

- The document level precision of the CDT enhanced query modification method is evaluated using MSN search engine [Cor]. This search engine has a restricter constraint on the maximum acceptable query size compared to Altavista used in the previous experiment.

In addition to the above experiments, we also evaluate effectiveness of the proposed CDT learning algorithm. We do this by comparing the CDT enhanced query modification method with a method that uses an alternative learning algorithm similar to ID3 to construct the decision tree. This experiment is explained in Section 7.3.4.

In the experiments, the cardinality of the initial attribute set for the CDT learning algorithm is set equal to the maximum query size ($maxSize$). Similar to the previous experiments, the value of $maxSize$ in calculating *G-measure* is set to 10.

6.3.2 Evaluation of Query Modification Time

Figures 6.2 and 6.3 show the query modification time for various probing types averaged over 25 queries with broad and narrow context categories. The results are similar to those of the basic method. That is, the query modification time of the partial probing is much smaller than that of the full probing, and there is no significant different between the modification time at the broad and narrow context categories.

6.3.3 Evaluation of Search Result Effectiveness

Figures 6.4 and 6.5 show the *G-measure* ratio of queries with broad and narrow context categories and, Figures 6.6 and 6.7 plot the query modification time versus the *G-measure* ratio of queries with broad and narrow context categories. The results are also similar to those of the basic method, but they have a greater *G-measure* ratio than the basic method. As a result, the rise of the curves in Figures 6.6 and 6.7 is smaller than that of the basic method.

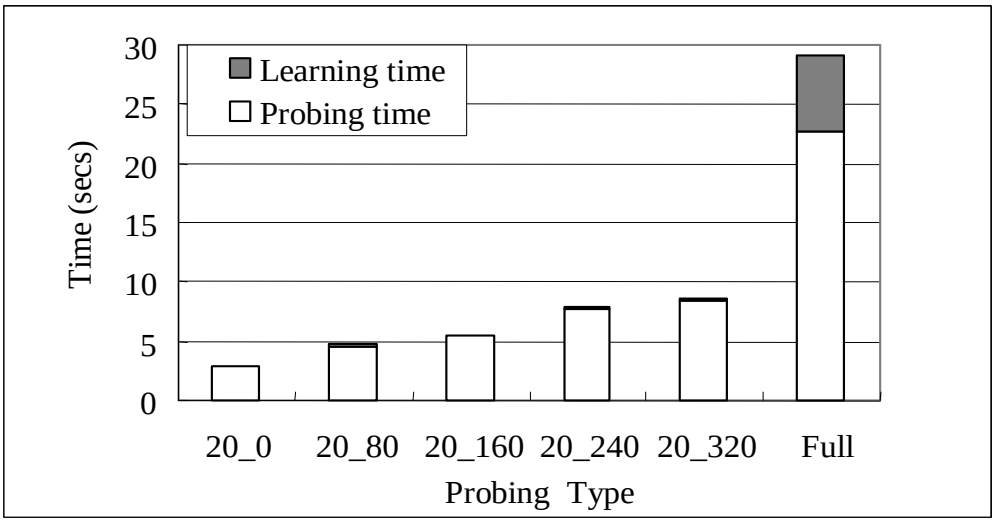

Figure 6.2: Query modification time for various probing types and queries with broad context categories.

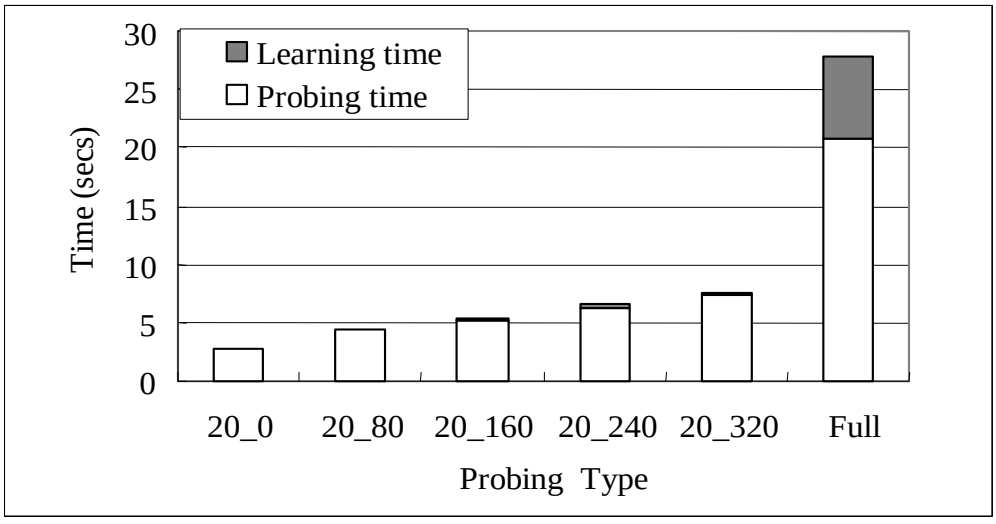

Figure 6.3: Query modification time for various probing types and queries with narrow context categories.

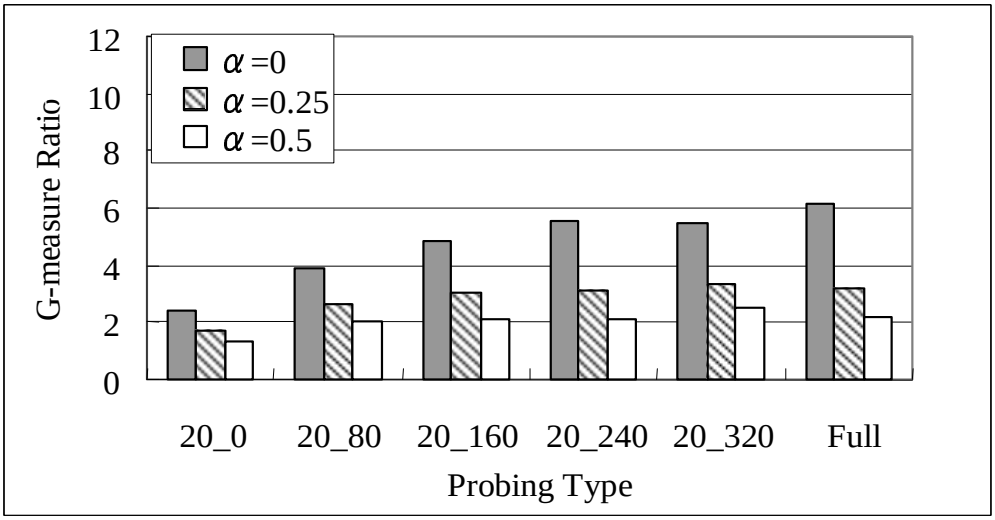

Figure 6.4: G-measure ratio for various probing types and queries with broad context categories.

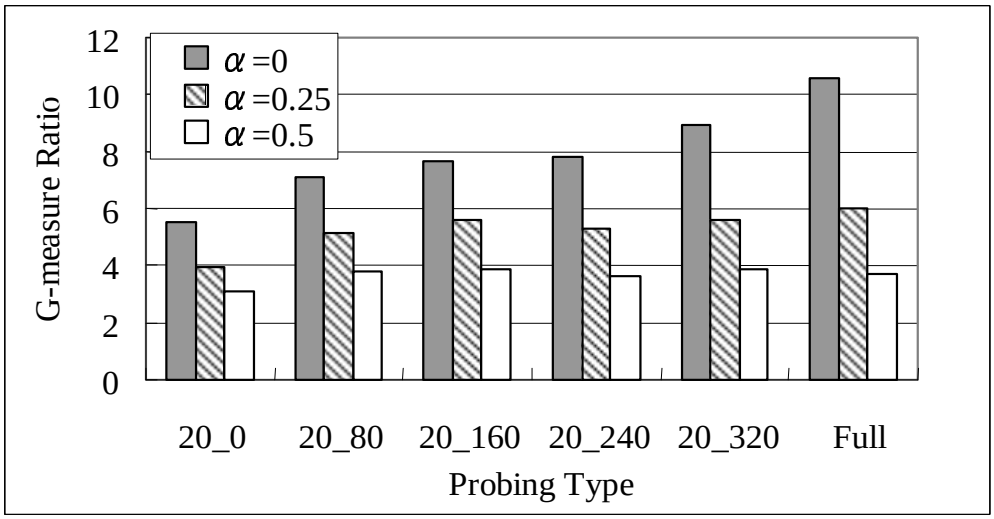

Figure 6.5: G-measure ratio for various probing types and queries with narrow context categories.

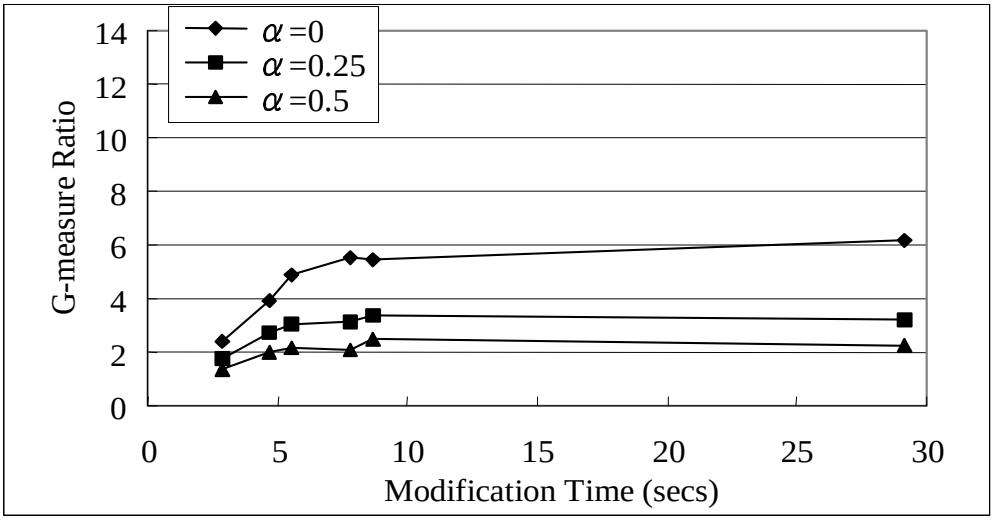

Figure 6.6: Query Modification time and G-measure ratio for various probing types and queries with broad context categories.

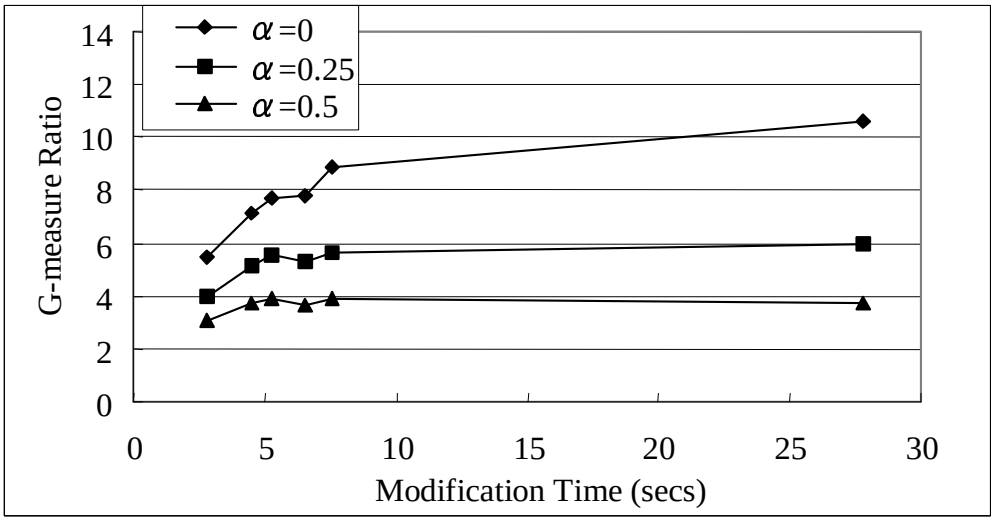

Figure 6.7: Query Modification time and G-measure ratio for various probing types and queries with narrow context categories.

6.3.4 Comparison with a Static Method

The static method here is similar to that of the previous experiment. The difference is that the classifier/decision tree of each category is constructed using the CDT learning algorithm. Figures 6.8 and 6.9 show the *G-measure* ratio averaged over 25 queries with the broad and narrow context categories. Similar to the results of the basic method, it is clear that the CDT enhanced method always outperforms the static method.

6.3.5 Evaluation with MSN Search Engine

In this experiment we evaluate the CDT enhanced query modification method using MSN as the target search engine. MSN has a shorter maximum acceptable query length (150 characters) than Altavista (700 characters). For simplicity, we assume that MSN accepts queries with a 12-keyword length, so we set $maxSize$ to 12. We use ODP [Net] as the taxonomy-based search facility and the 50 queries from the previous experiment. We set α to 0, and compare the basic method using the 20_320 and full probing with the initial query condition and static method.

Figures 6.10 and 6.11 show the result for queries with broad and narrow context categories. Document level precision of the CDT enhanced method

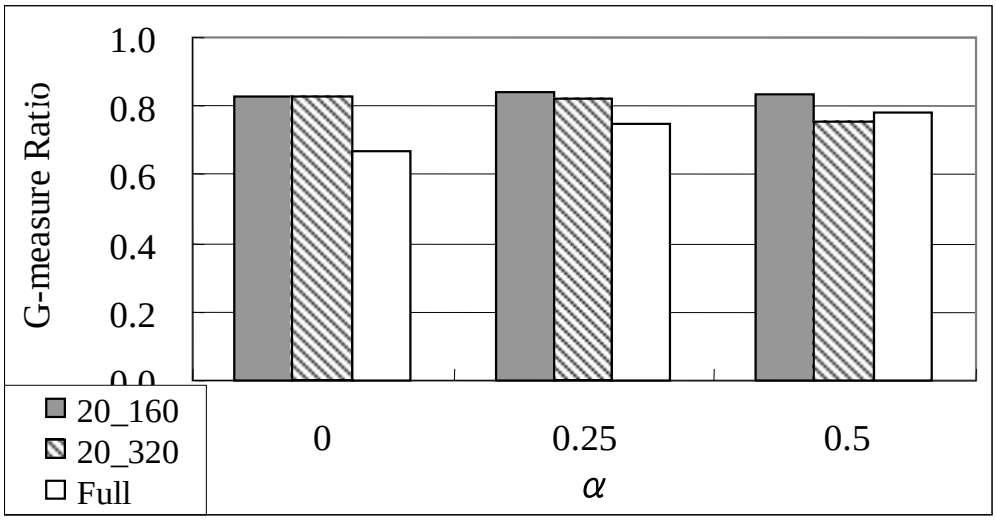

Figure 6.8: Comparison with a static method for queries with broad context categories.

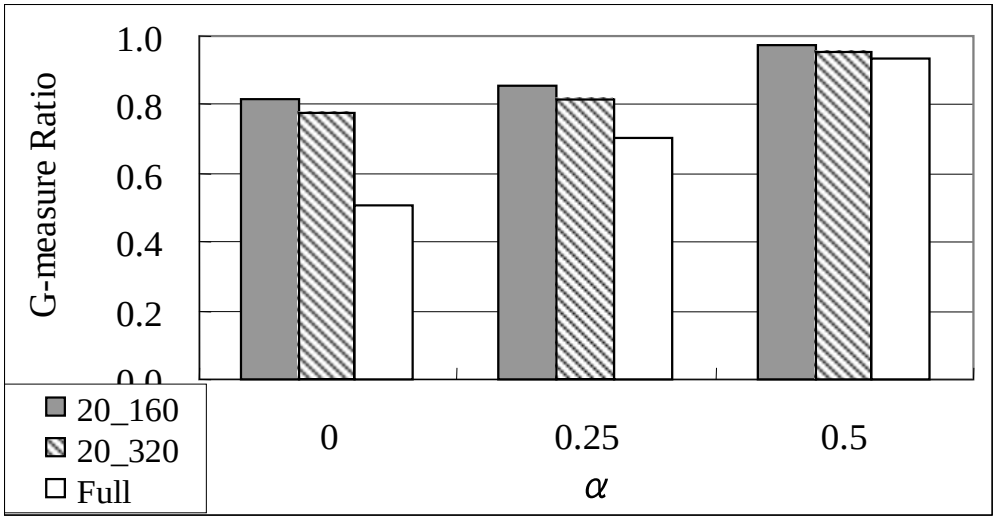

Figure 6.9:
Comparison with a static method for queries with narrow context categories.

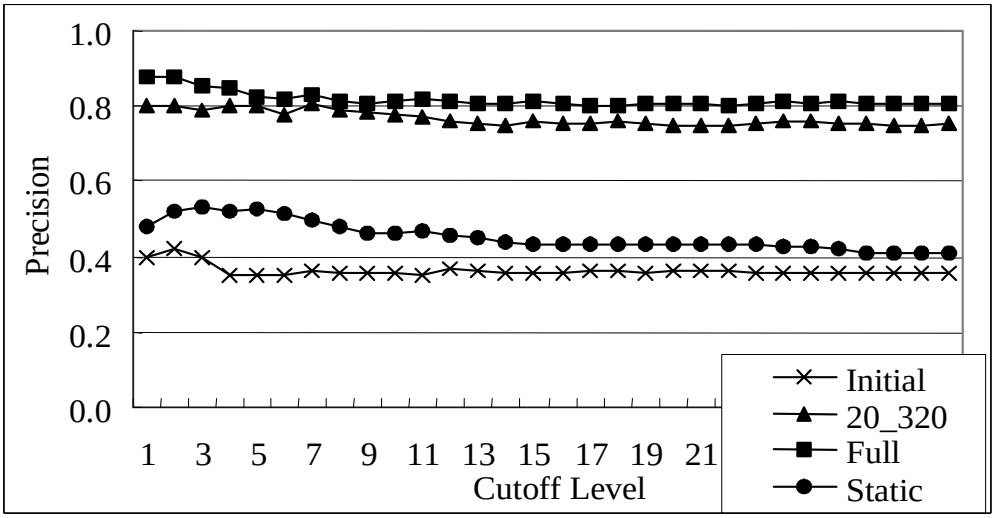

Figure 6.10: Precision of MSN for queries with broad context categories.

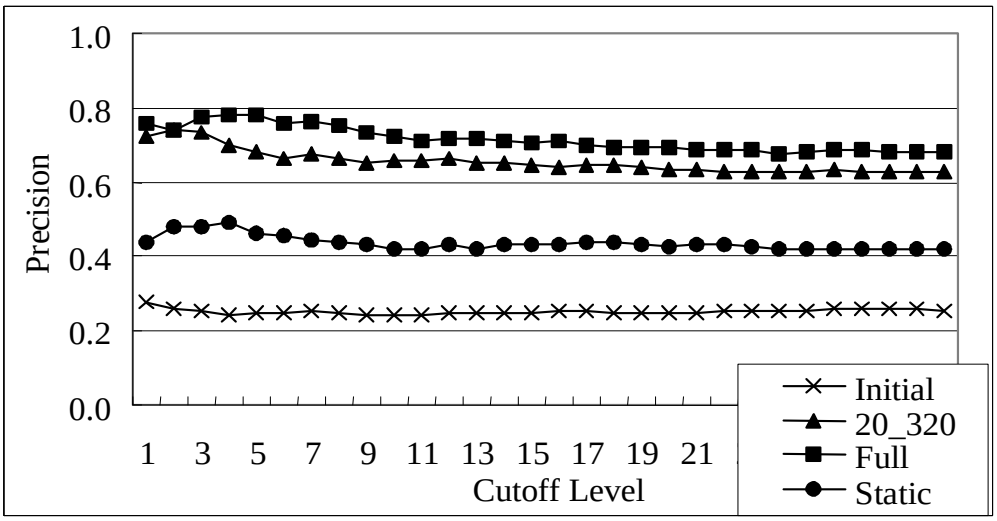

Figure 6.11: Precision of MSN for queries with narrow context categories.

using the 20_320 partial and full probing is much better than that of the initial query condition and static method. Further, precision of the 20_320 partial and full probing is comparable. And there is no significant difference between the precision of queries with broad and narrow context categories. This tells us that the CDT enhanced method is effective in the real web environment.

6.3.6 Use of G-measure in CDT Learning Algorithm

The main purpose of this experiment is to see the effectiveness of selecting the best relevant subtree (*RST*) based on the *G-measure* value. We do this by comparing the *G-measure* of queries modified using the CDT enhanced query modification method and those modified using a method employing an alternative algorithm (denoted as *CDT alternative method*). The alternative algorithm constructs a decision tree until one of the stop conditions is satisfied and extracts the *RST* from the final decision tree regardless of its *G-measure* value[2].

We calculate the *G-measure* value of the modified queries in much the same way as previous experiments (Section 5.3). First, we take sample t-entries with the 20_320 and full probing techniques. For each sample set,

[2]The algorithm is without lines 16 and 17, and line 19 returns the *RST* of the final decision tree (Figure 6.1).

we build a decision tree and collect two $RSTs$: the one with a maximum *G-measure* value and the one extracted from the final decision tree. After extracting query modifiers from the $RSTs$, we modify query condition Q with those modifiers and calculate their *G-measures* based on the test set. Note that since the alternative algorithm does not take into account the *G-measure* value of the decision tree, the *precision* and *recall* of the modified queries from it do not depend on α (i.e., the relevant subtree is the same for all α).

Figures 6.12 and 6.13 show the results for queries with broad context categories; Figures 6.14 and 6.15 show the results for queries with narrow context categories. The *G-measure* ratio here is the relative *G-measure* value taking that of the CDT enhanced method as the base. That means it is obtained by dividing the *G-measure* value of the CDT alternative method by that of the CDT enhanced method. 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 CDT enhanced method and the CDT alternative method is statistically significant.

The *G-measure* of queries modified by the CDT enhanced method always outperforms that of queries modified by the CDT alternative method for all

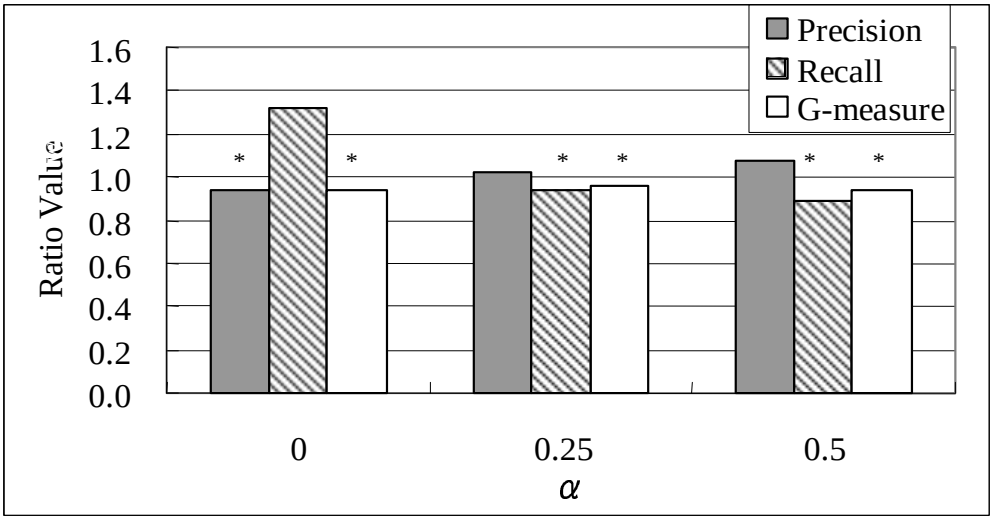

Figure 6.12: Comparison with CDT alternative method for the full probing and queries with broad context categories.

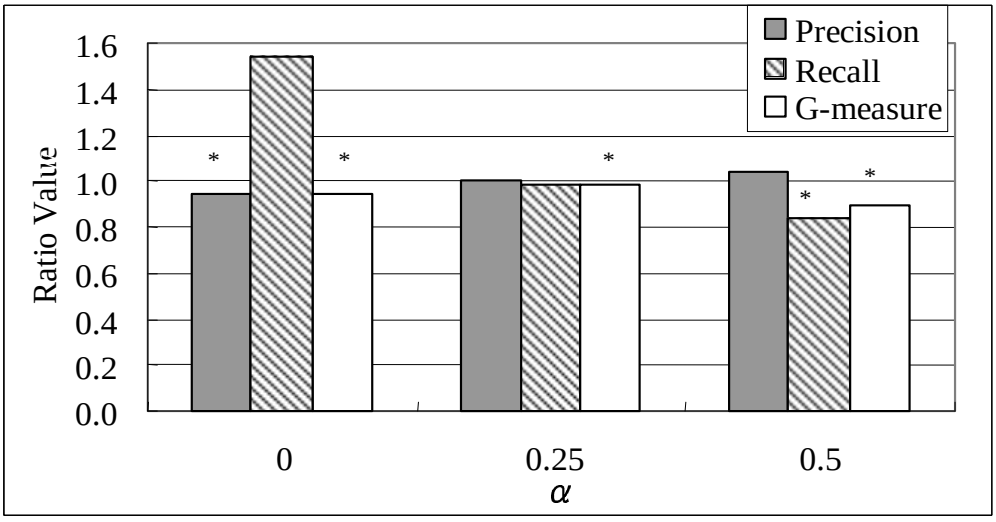

Figure 6.13: Comparison with CDT alternative method for the 20_320 probing and queries with broad context categories.

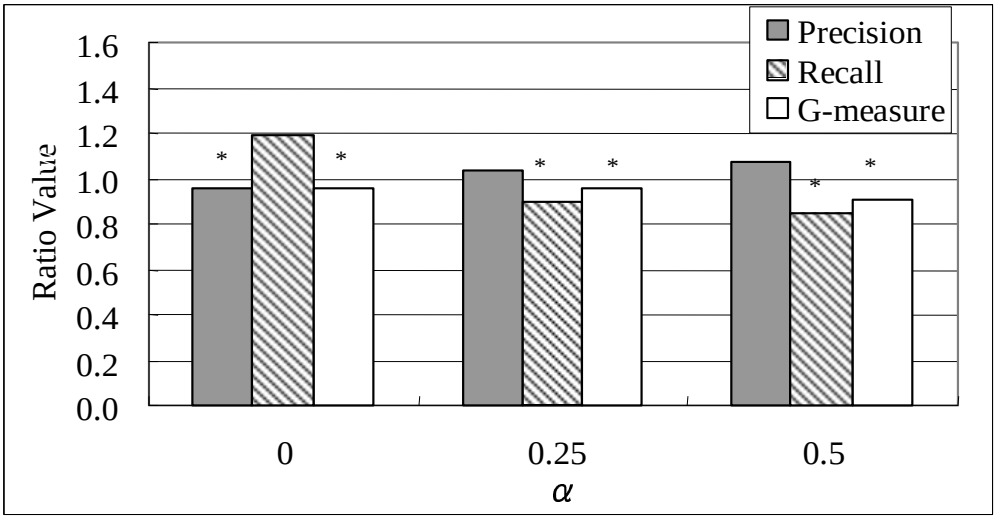

Figure 6.14: Comparison with CDT alternative method for the full probing and queries with narrow context categories.

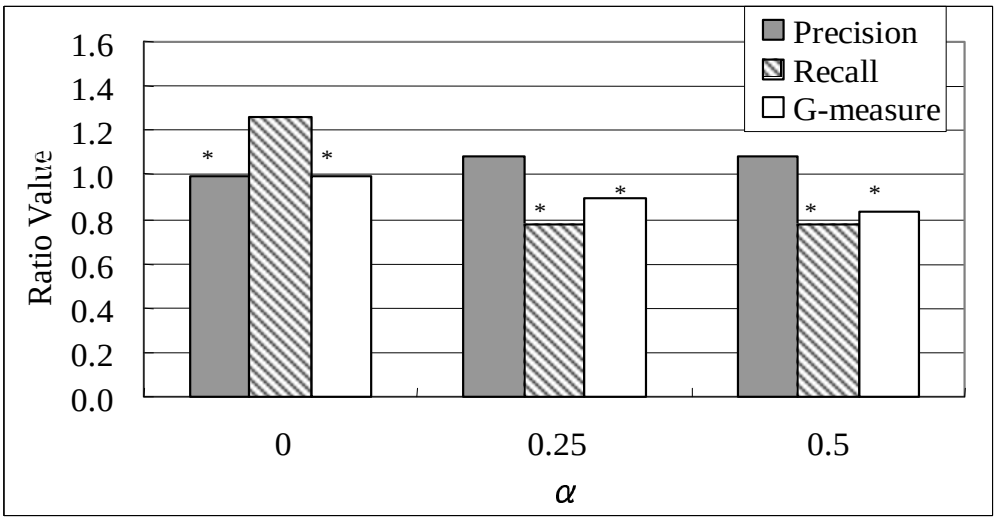

Figure 6.15: Comparison with CDT alternative method for the 20_320 probing and queries with narrow context categories.

α. This achievement is also verified by results of the t-test. This performance advantage results because the CDT learning algorithm used in the CDT enhanced method always selects the best RST having the maximum *G-measure* value.

For $\alpha = 0$, precision of the CDT enhanced method 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 Equation 5.1 and indicates that the α is reflected in the search result of modified queries from the CDT enhanced method. (Note that *precision* and *recall* of the CDT alternative method do not change with the α value.)

Chapter 7

CCR Enhanced Query Modification Method

7.1 Overview

This chapter describes the second proposed enhanced query modification method called *CCR enhanced query modification*. This method is also based on the proposed framework but it uses a proposed *Constrained Classification Rule* (CCR) learning algorithm to construct the classifier. This method is used to modify queries sent to the template-based search engines.

More precisely, queries in the template-based query format can be ex-

pressed as follows:

$$C = (w_{1,1} \wedge \ldots \wedge w_{1,i}) \wedge (\neg w_{2,1} \wedge \ldots \wedge \neg w_{2,j}) \wedge (w_{3,1} \vee \ldots \vee w_{3,k}) \quad (7.1)$$

where $w_{m,n}$ corresponds to a search term/keyword, $i, j \geq 0$ and $k = 0$ or $k \geq 2$. We denote $w_{m,n}$ and its negation $\neg w_{m,n}$ as *positive* and *negative literals*. For easy reference, we denote the first, second and third subexpression of Equation 7.1 as $Conj_p$, $Conj_n$ and $Disj_p$. Note that $set(Conj_p)$, $set(Conj_n)$ and $set(Disj_p)$ are mutually disjointed, where $set(x)$ is a set of terms/keywords in expression x. Some search engines support only Boolean queries with no $Disj_p$. The CCR learning algorithm can cope with these search engines by imposing an additional constraint of $k = 0$.

Three points differentiate the CCR learning algorithm from the existing rule learning algorithms. First, the CCR learning algorithm constructs a rule "$H \mapsto Relevant$" with H conforming to Equation 7.1. Second, it induces a rule by explicitly constraining its size. Third, it selects the most promising rule based on the estimated *G-measure* of the modified query. The second and third points are similar to the approach of the CDT learning algorithm.

We evaluate performance of the CCR enhanced query modification method and the proposed CCR learning algorithm with experiments. The evaluation method and points are the same as those of the experiments car-

ried out for the CDT enhanced method explained in the previous chapter.

7.2 CCR Learning Algorithm

The CCR learning algorithm is summarized in Figure 7.1. Similar to the CDT learning algorithm, $TSet$ is also split into two disjointed subsets: $GSet$ and $VSet$. $GSet$ 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 conjunct $Conj$ ($=Conj_p \wedge Conj_n$), and then disjunct $Disj_p$. $Conj$ 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

[1]The algorithm can be easily adapted to other size units (such as the number of characters) with minor changes.

```
CCR(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}, VSet) > G-measure(c_best, VSet)
18:        c_best ← c_{i+1};
19:      litSet ← litSet \ {l_i};
20:      if |litSet| = 0, exit the loop;
21:    }
22:    return c_best;
```

Figure 7.1: CCR learning algorithm

the empty/true condition and pick literals with the k largest WIG values.

The WIG [Qui90][F$\ddot{9}$9] is given below.

$$\begin{aligned} & WIG(c_{i+1}, c_i) \\ &= rel_{i+1} \cdot \left(log_2 \frac{rel_{i+1}}{rel_{i+1} + irrel_{i+1}} - log_2 \frac{rel_i}{rel_i + irrel_i} \right) \end{aligned} \tag{7.2}$$

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 reducing the total number of covered relevant t-entries relative to c_i. This measure fits well to our goal of constructing a rule for the relevant class.

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 one obtained so far, then c_{i+1} is put into c_{best} (line 18). The For 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.

Query Modifier Extraction

The query modifier used to modify an initial query condition is extracted from the returned rule. This is done by extracting the rule header H, making it the query modifier.

7.3 Experimental Evaluation

7.3.1 Overview

Here we will evaluate the CCR enhanced query modification method with experiments. The evaluation method and points are similar to those of the CDT enhanced method shown in the previous chapter with the minor differences given below.

- In measuring the modification time, the learning time is the time needed to construct a classification rule using the CCR learning algorithm.
- In comparing the CCR enhanced method with the static method, the classifiers in the static method are built by using the CCR learning algorithm.
- Document level precision of the CCR enhanced method is evaluated using Google [Goo] search engine, which has a restricter constraint on the acceptable query form.
- In evaluating effectiveness of the CCR learning algorithm, the alternative learning algorithm is the one that does not take into account *G-measure* value of every candidate rule constructed.

7.3.2 Evaluation of Query Modification Time

Figures 7.2 and 7.3 show the query modification time averaged over 25 queries with broad and narrow context categories. The results are similar to those of the CDT enhanced method. Query modification time of the partial probing is much shorter than that of the full probing, and there is no significant difference between the modification time of queries with the broad and narrow context categories.

7.3.3 Evaluation of Search Result Effectiveness

Figures 7.4 and 7.5 show the *G-measure* ratio of queries with broad and narrow context categories, and Figures 7.6 and 7.7 plot the query modification time versus the *G-measure* ratio of queries with broad and narrow categories. Again, the results are similar to those of the CDT enhanced method. The important point is that the method lets the users control the trade-off between search result effectiveness and modification time by adjusting the sampling parameters (p, q).

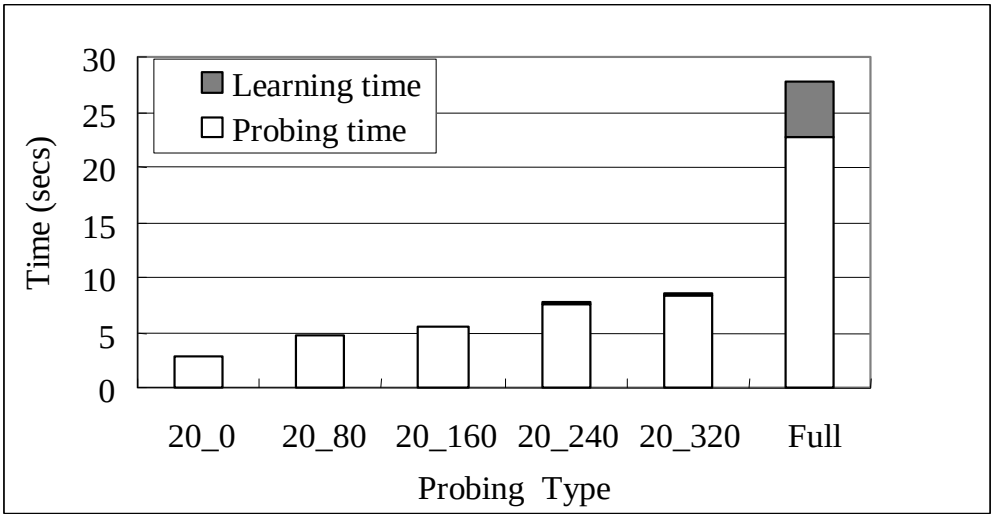

Figure 7.2: Query modification time for various probing types and queries with broad context category.

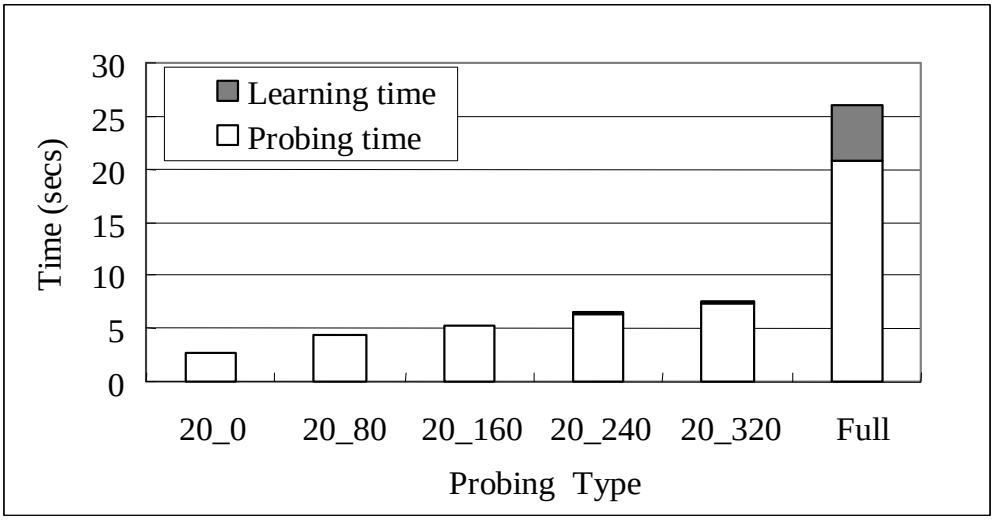

Figure 7.3: Query modification time for various probing types and queries with narrow context category.

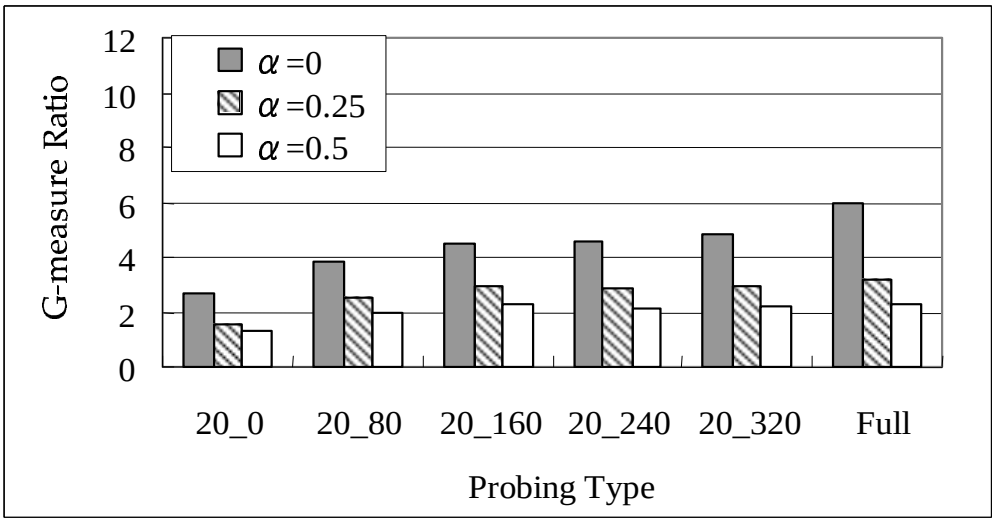

Figure 7.4: G-measure ratio for various probing types and queries with broad context categories.

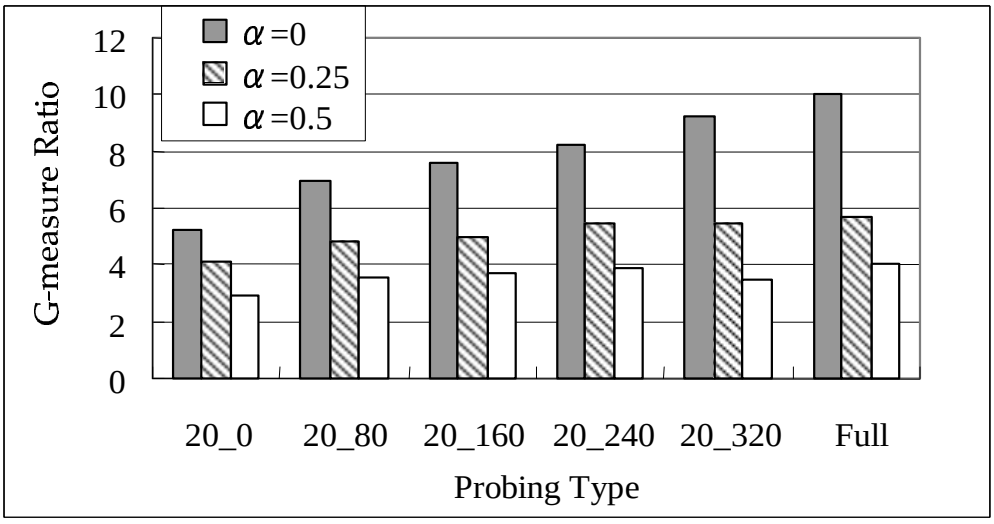

Figure 7.5: G-measure ratio for various probing types and queries with narrow context categories.

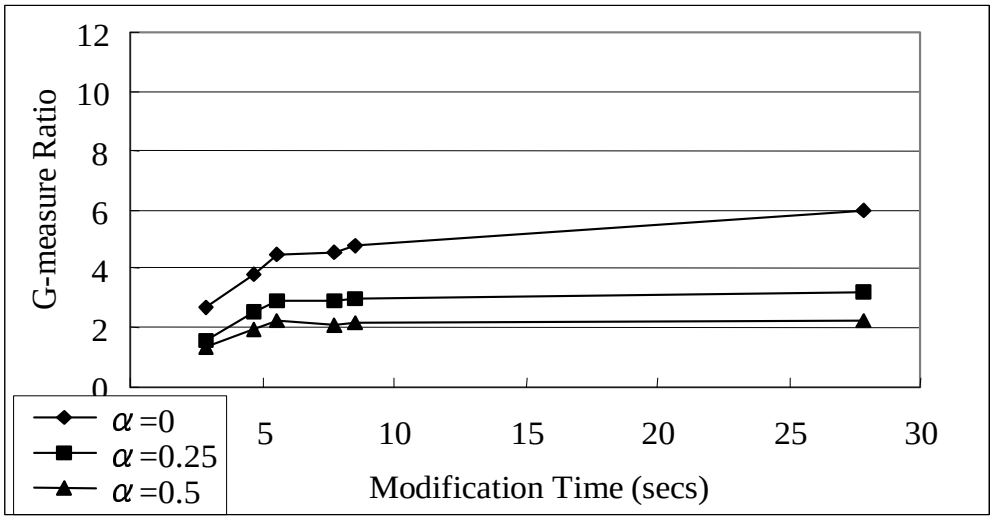

Figure 7.6: Query Modification time and G-measure ratio for various probing types and queries with broad context categories.

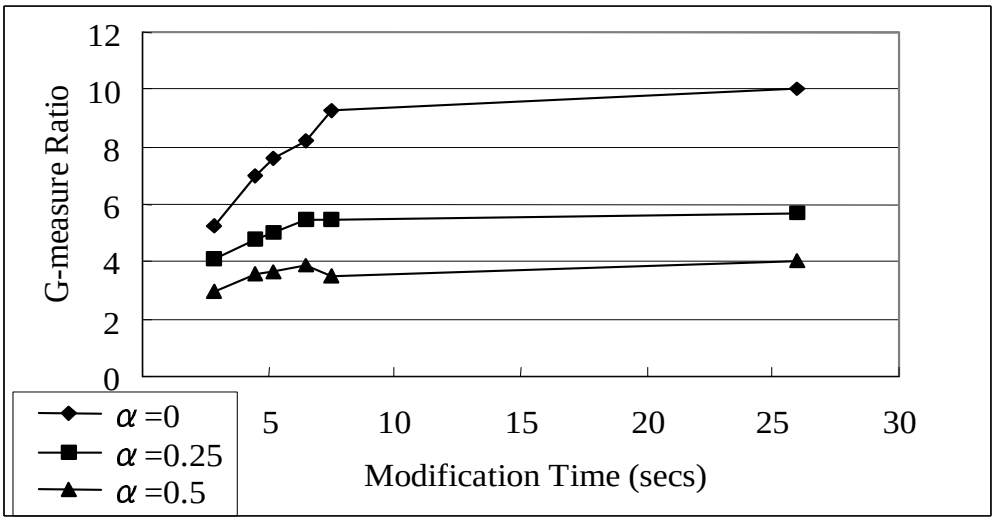

Figure 7.7: Query Modification time and G-measure ratio for various probing types and queries with narrow context categories.

7.3.4 Comparison with a Static Method

Figures 7.8 and 7.9 show the comparison with the static method. Similar to the results of the CDT enhanced method, the CCR enhanced method always outperforms the static one. This shows the advantage of the dynamic behavior of the enhanced modification method.

7.3.5 Evaluation with Google Search Engine

In this experiment we evaluate the CCR enhanced query modification method using Google as the target search engine. Google only accepts queries in the form shown by Equation 7.1, so it cannot process queries in the ordinary Boolean format. We set $maxSize$ to 10 which is the maximum query size supported by Google, and α to 0. We use ODP [Net] as the taxonomy-based search facility and the 50 queries from the previous experiment. We compare the CCR enhanced method using the 20_320 and full probing with the initial query condition and static method.

Figures 7.10 and 7.11 show the result for queries with broad and narrow context categories. From the figures, document level precision of the CCR enhanced method using the 20_320 partial and full probing is much better than that of the initial query condition and static method. Further, the

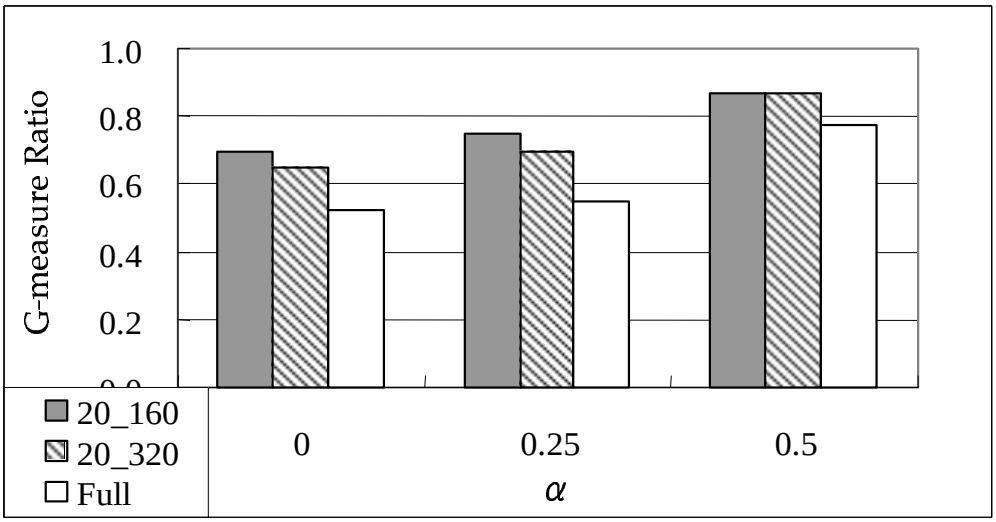

Figure 7.8: Comparison with a static method for queries with broad context categories.

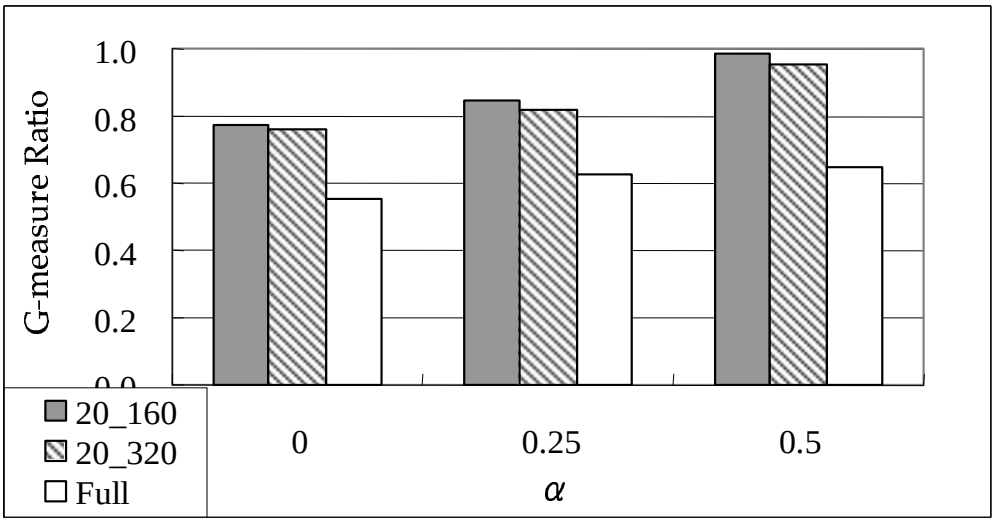

Figure 7.9: Comparison with a static method for queries with narrow context categories.

precision when using 20_320 partial and full probing is comparable. And there is no significant difference between the precision of queries with narrow and broad context categories. This tells us that the CCR enhanced query modification method is also effective in the real web environment.

Using Newsgroups as the Taxonomy-based Search Facility

In order to see behavior of the modification framework for different taxonomy data, we use the Newsgroups search service provided by Google as the taxonomy-based search facility. We define 15 new queries and compare the performance of the CCR enhanced method using the 20_320 and 20_160 partial probing with the initial query condition. We cannot do the full probing with the Newsgroups search service because the number of matched t-entries is too large and Google's TOS does not allow automated queries[2]. The other parameter settings are same as in the previous experiment.

Figure 7.12 shows the experiment result. As before, the CCR enhanced method using the partial probing can greatly increase the *precision* of the initial query condition, especially with the 20_320 partial probing. However, the precision increase is smaller than with the ODP case because the quality

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

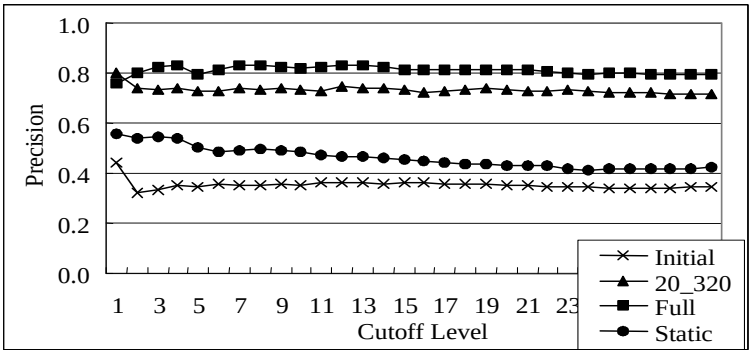

Figure 7.10: Precision of Google for queries with broad context categories.

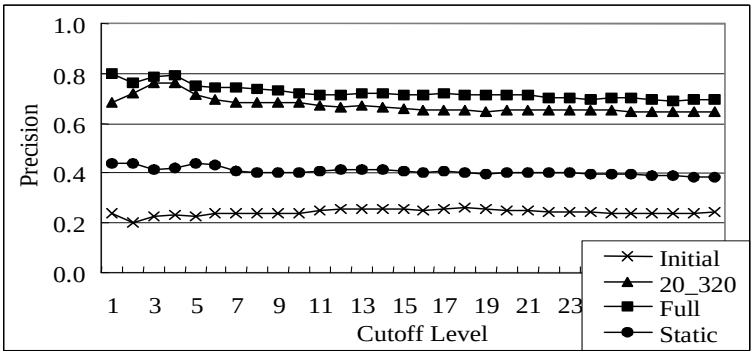

Figure 7.11: Precision of Google for queries with narrow context categories.

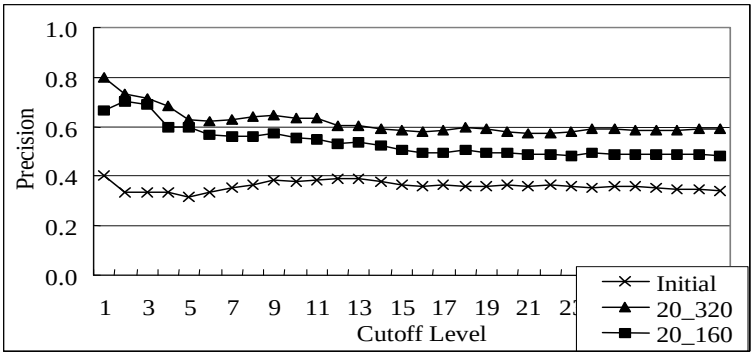

Figure 7.12: Precision of Google with the Newsgroups search service as the taxonomy-based search facility.

of document collection in Newsgroups is lower than that of ODP[3].

7.3.6 Use of G-measure in CCR Learning Algorithm

Similar to the experiments carried for the CDT learning algorithm shown in the previous chapter, the main purpose of this experiment is to see the effectiveness of selecting the best rule based on *G-measure*. We do this by comparing the *G-measure* of queries modified using the CCR enhanced query modification method and those modified using a method employing an alternative algorithm (denoted as *CCR alternative method*). The CCR alternative algorithm is the one that constructs $Conj$ and $Disj_p$ without taking into account the *G-measure* value of the condition. The condition is repeatedly expanded until one of the stop conditions is satisfied and the finally constructed condition is returned[4].

Calculation of the *G-measure* value of the modified queries is similar to that of experiments using the CDT enhanced method. Figures 7.13 and 7.14 show the results for queries with broad context categories, while Figures 7.15 and 7.16 show the results for queries with narrow context categories. Again, the results are similar to those of the CDT learning algorithm shown in the

[3]The quality is low because most of the context categories selected from the Newsgroup hierarchy are unmoderated groups (i.e., groups without moderators).

[4]For CrConj in Figure 7.1, it is without lines 17 and 18, and line 22 returns the finally created condition.

previous chapter. The important point is that the search result effectiveness of modified queries from the CCR enhanced method can be controlled by adjusting α.

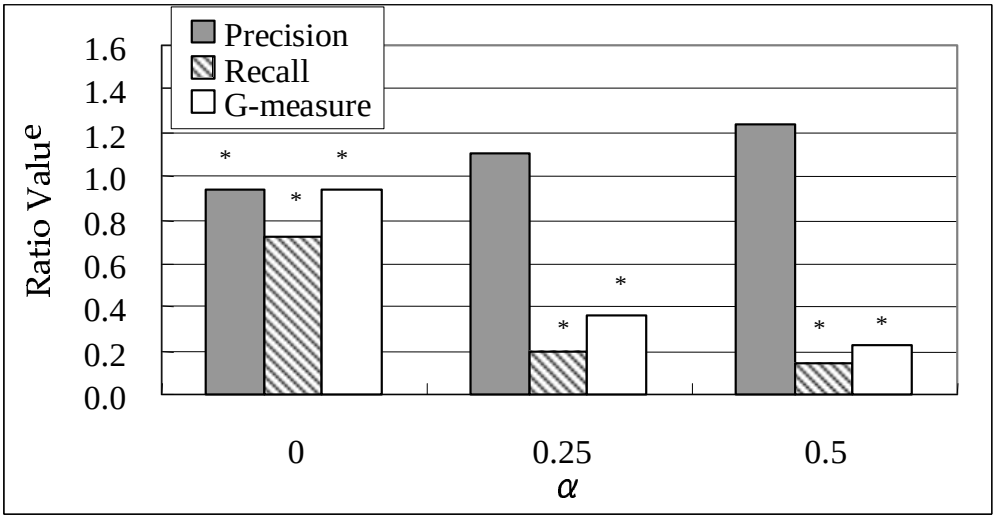

Figure 7.13: Comparison with CCR alternative method for the full probing and queries with broad context categories.

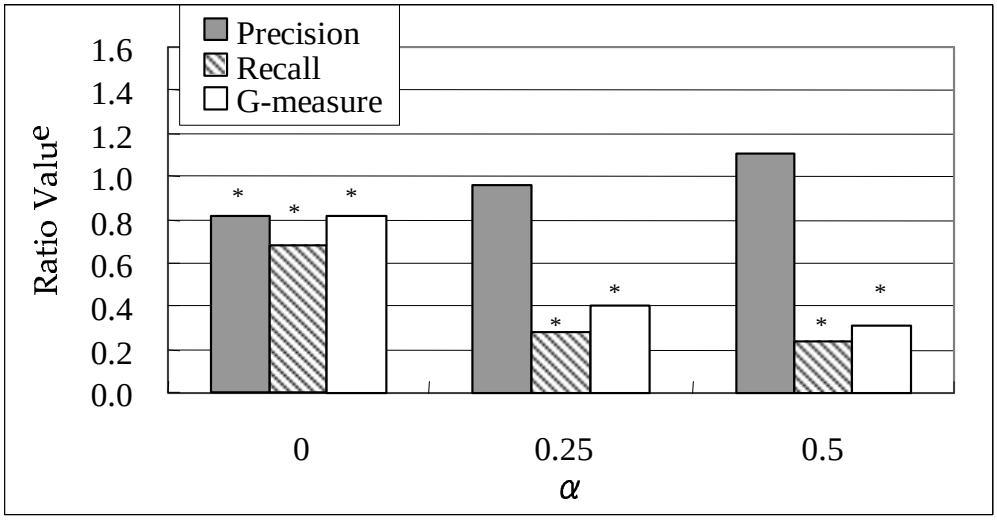

Figure 7.14: Comparison with CCR alternative method for the 20_320 probing and queries with broad context categories.

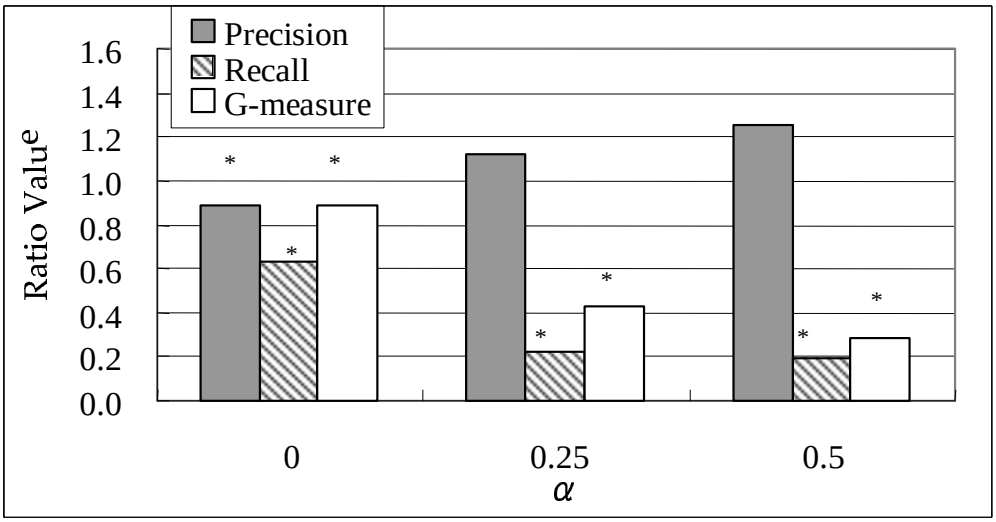

Figure 7.15: Comparison with CCR alternative method for the full probing and queries with narrow context categories.

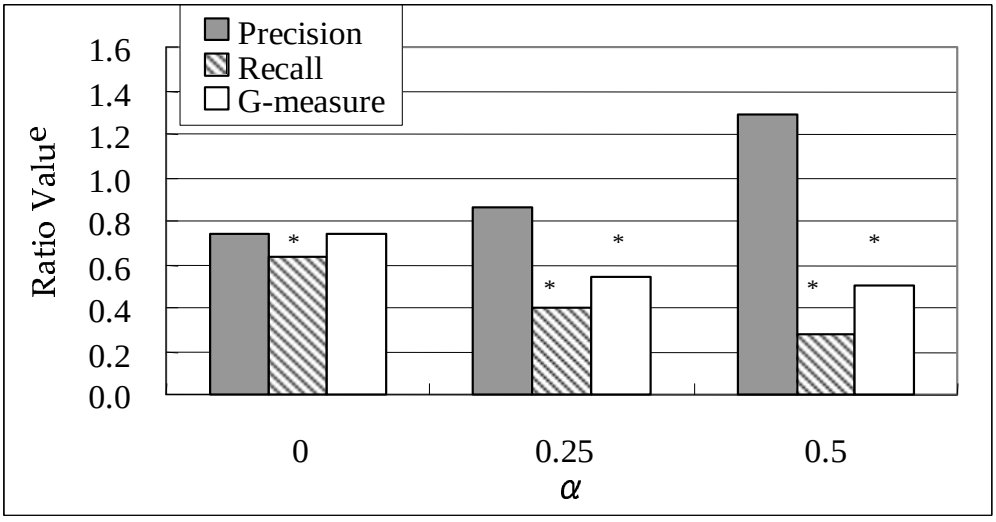

Figure 7.16: Comparison with CCR alternative method for the 20_320 probing and queries with narrow context categories.

Chapter 8

Comparison of the Proposed Query Modification Methods

8.1 Overview

In this chapter we compare the three proposed query modification methods: the basic method, the CDT enhanced method and the CCR enhanced method. The comparison is done as follows.

- Compare search result effectiveness (*G-measure*) of the three query modification methods for various probing types and queries with broad and narrow context categories.
- Compare document level precision (until cutoff 30) of the three query

modification methods using the Altavista [Alt] search engine.

The experiment results reveal the usage of each enhanced method to get maximum retrieval performance.

8.2 Comparison of Search Result Effectiveness

In this section we compare search result effectiveness of the three query modification methods. As in the previous experiment, the cardinality of the initial attribute set for CDT learning algorithm and literal set ($|litSet|$) for CCR learning algorithm are set equal to $maxSize$; the value of $maxSize$ is set to 10.

Figures 8.1 to 8.3 show the *G-measure* ratio of the three query modification methods for various α values and queries with broad context categories. The *G-measure* ratio here is the relative *G-measure* value taking *G-measure* of the initial (not modified) query condition Q as the base. In general the enhanced methods (i.e., CDT and CCR enhanced methods) outperform the basic method. This is also true for the result of queries with narrow context categories, which is shown in Figures 8.4 to 8.6. The reason is that the proposed algorithms used by the enhanced methods always select the best

RST/rule having the maximum *G-measure* value.

The basic method (which uses RIPPER to construct a classifier) is better than the two enhanced methods when α is relatively high (e.g., 0.5) and the sample size is large (e.g., in the full probing case). This occurs because RIPPER tends to construct many rules as the sample size increases, resulting in a relatively high *recall*. The high *recall* takes the advantage as α increases.

Beyond that, the *G-measure* ratio of queries modified by the CDT enhanced method generally outperforms those modified by the CCR enhanced method. This occurs because the hypothesis search space of the CDT learning algorithm (which is given by the ordinary Boolean query format) is larger than that of the CCR learning algorithm (which is given by Equation 7.1). Thus the CDT learning algorithm tends to more precisely find a query modifier with the best *G-measure* value.

8.3 Comparison Using Altavista Search Engine

This section compares retrieval performance of the three query modification methods using a real web search engine. The basic method uses RIPPER to construct a classifier, where RIPPER does not limit the size of the clas-

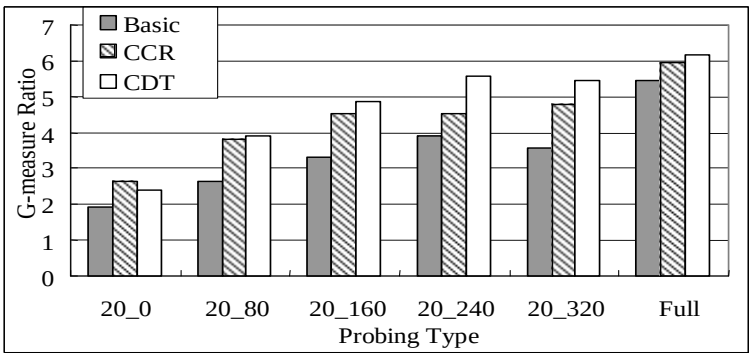

Figure 8.1: Search result effectiveness for $\alpha = 0$ and queries with broad context categories.

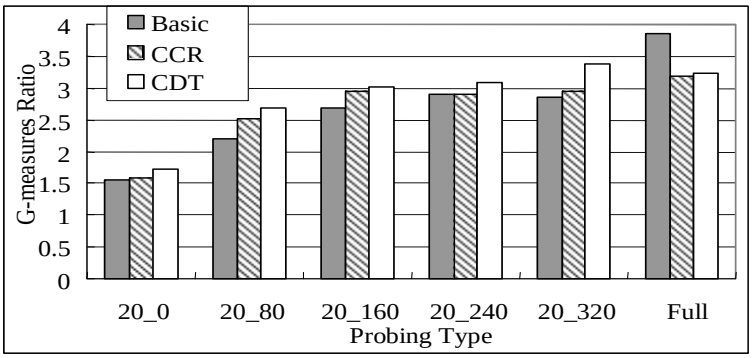

Figure 8.2: Search result effectiveness for $\alpha = 0.25$ and queries with broad context categories.

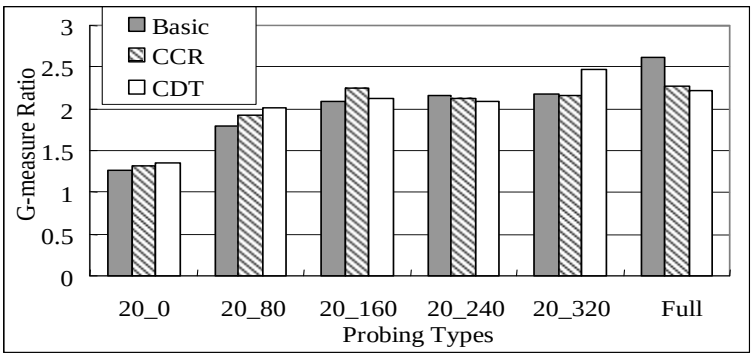

Figure 8.3: Search result effectiveness for $\alpha = 0.5$ and queries with broad context categories.

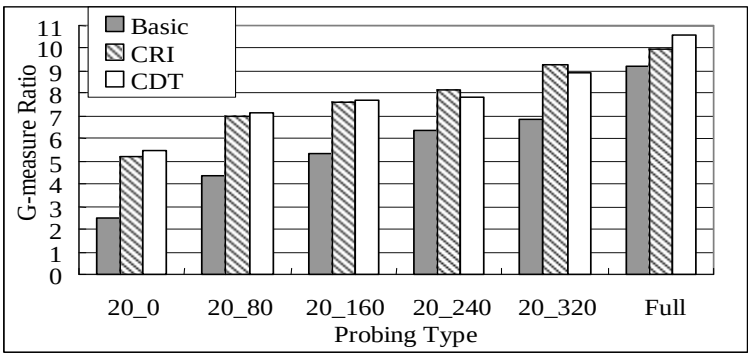

Figure 8.4: Search result effectiveness for $\alpha = 0$ and queries with narrow context categories.

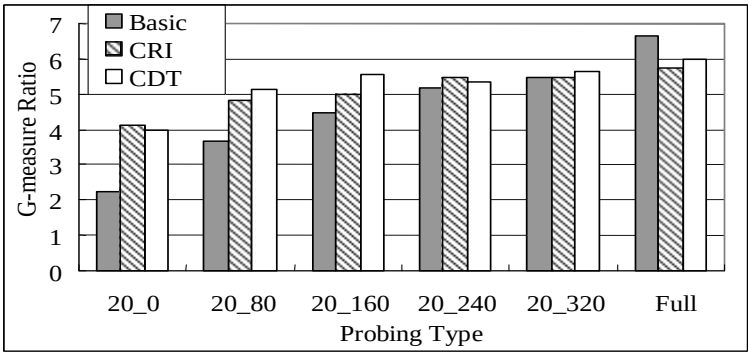

Figure 8.5: Search result effectiveness for $\alpha = 0.25$ and queries with narrow context categories.

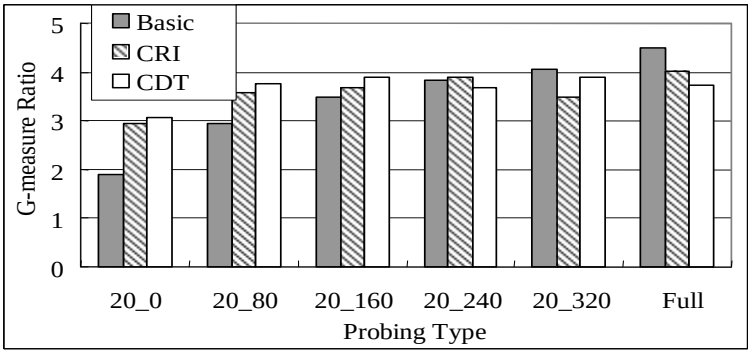

Figure 8.6: Search result effectiveness for $\alpha = 0.5$ and queries with narrow context categories.

sifier constructed. To compare performance of the three query modification methods fairly, we set the value of $maxSize$ for CDT and CCR learning algorithms to a large value, and use the Altavista [Alt] search engine, which accepts very long queries (700 characters), as the target search engine. We use 50 queries from the previous experiments and set α for CDT and CCR enhanced methods to 0.

Figures 8.7 and 8.8 show the comparison of the document level precision of the three query modification methods for the 20_320 and full probing cases and queries with broad context categories. The results for queries with narrow context categories are shown in Figures 8.9 and 8.10. We see that document level precision of queries modified by the two enhanced methods outperforms that modified by the basic method. For the same reason stated before, the CDT enhanced method outperforms the CCR enhanced method. Another interesting point is that precision of the 20_320 and full probing is comparable. This conforms to the experiment results given in previous chapters.

8.4 Result Summary

One goal is to guarantee that the modified queries are always *processable* by the target search engines and that search result effectiveness of the mod-

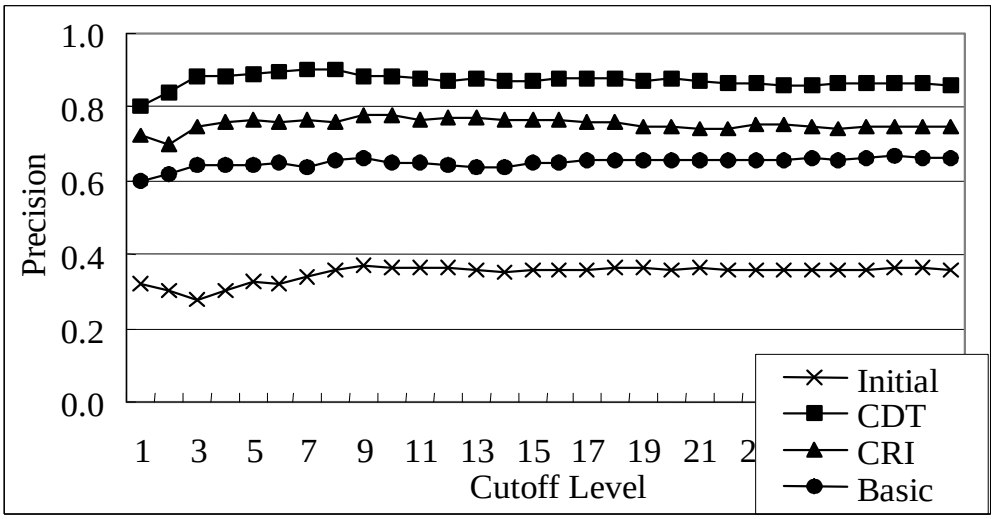

Figure 8.7: Precision of Altavista for the full probing and queries with broad context categories.

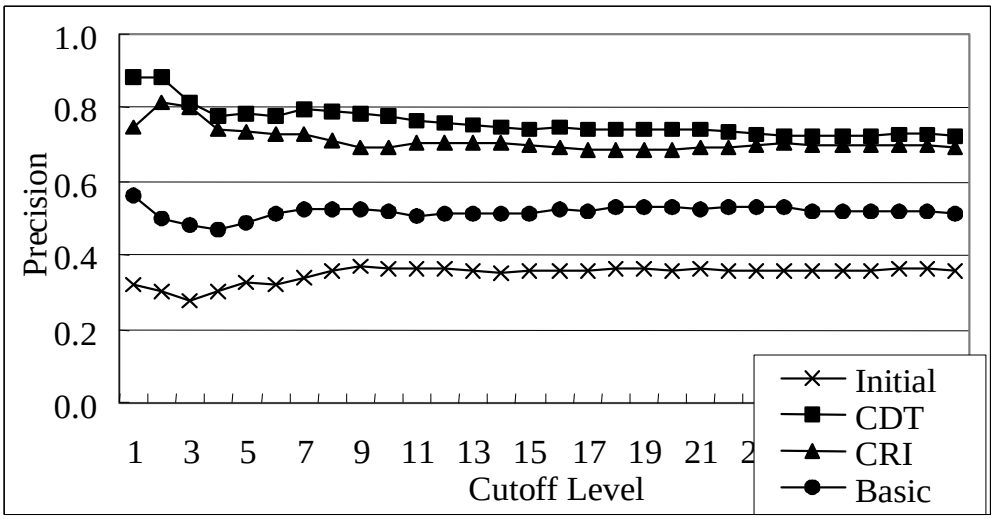

Figure 8.8: Precision of Altavista for the 20_320 probing and queries with broad context categories.

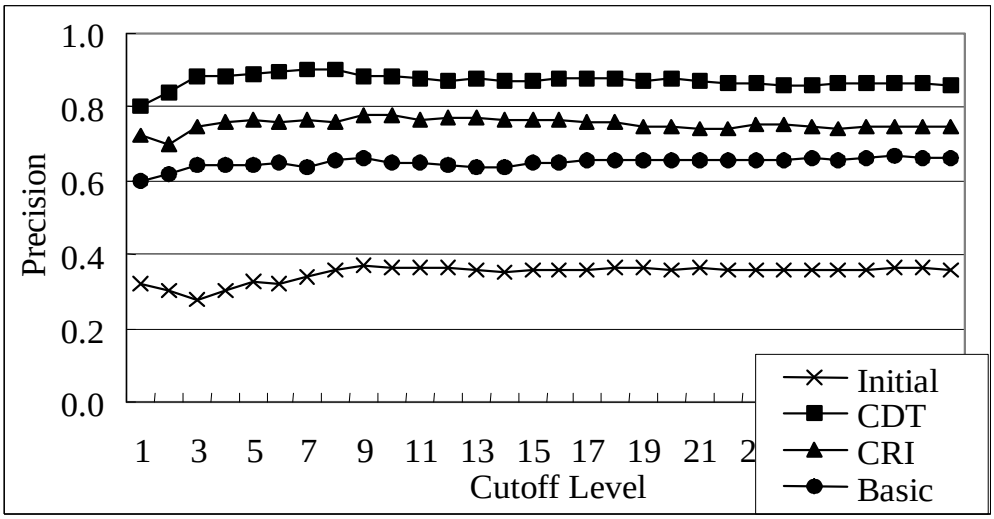

Figure 8.9: Precision of Altavista for the full probing and queries with narrow context categories.

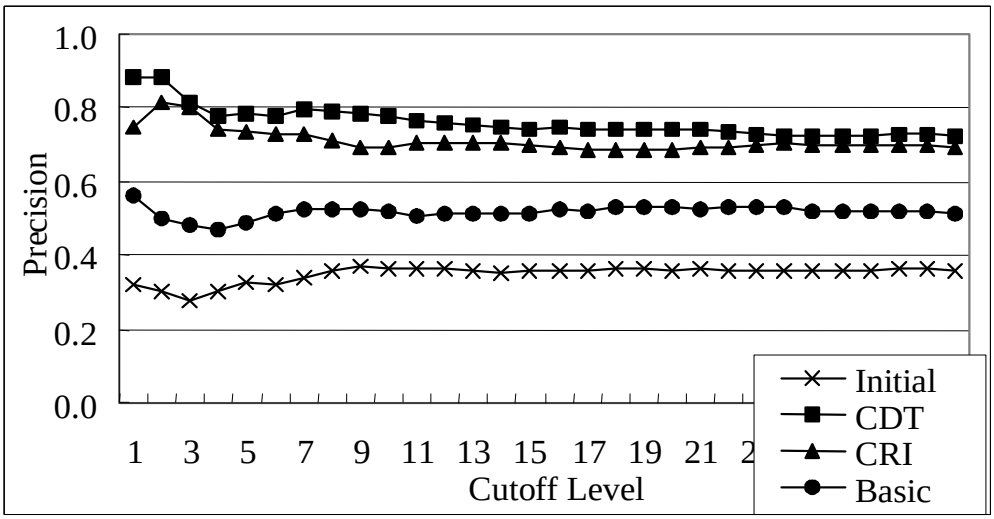

Figure 8.10: Precision of Altavista for the 20_320 probing and queries with narrow context categories.

ified queries is *controllable* by the user. For this purpose, we propose the CDT enhanced method, which modifies queries so that they are ordinary Boolean query format; and the CCR enhanced method, which yields (modified) queries in the template-based query format.

For the template-based search engines, such as Google and AlltheWeb, we clearly must use the CCR enhanced method. We must point out, however, that either of the two enhanced methods can be used for the ordinary Boolean search engines. The experiments shown in previous sections reveal that the CDT enhanced method is highly recommended for these search engines. It is recommended because retrieval performance of the CDT enhanced method outperforms that of the CCR enhanced method, yielding maximum retrieval performance from these search engines.

Chapter 9

Prototype System Development

9.1 Overview

This chapter describes the prototype system development implementing the proposed query modification framework. More specifically, we implement the CDT and CCR enhanced query modification methods to demonstrate practicability of the framework. The chapter starts by describing the architecture of the prototype system; it then explains the main part of the system called the *taxonomy probing and querying* (TAX-PQ) engine. The prototype system does the following:

- It provides taxonomy data from two major web directories (ODP and

Yahoo).

- It allows the user to select one of the three major target web search engines (Google, AltaVista and MSN).
- It allows the user to adjust parameters to control the search and modification process.
- It allows the user to interactively choose a modifier from several candidates to modify his query.

This chapter also presents a sample session of the prototype system and discusses implementation issues in a real world web environment.

9.2 System Architecture

Figure 9.1 shows the architecture of the prototype system. The system consists of two main parts: a *user interface part* and a *taxonomy probing and querying* (*TAX-PQ*) *engine part*. The parts are in the user, taxonomy-based search facilities and target web search engines.

The user interface part gives the user a way to formulate a query. It displays taxonomy of a taxonomy-based search facility and allows the user to browse the category hierarchy of the search facility. It provides some fields

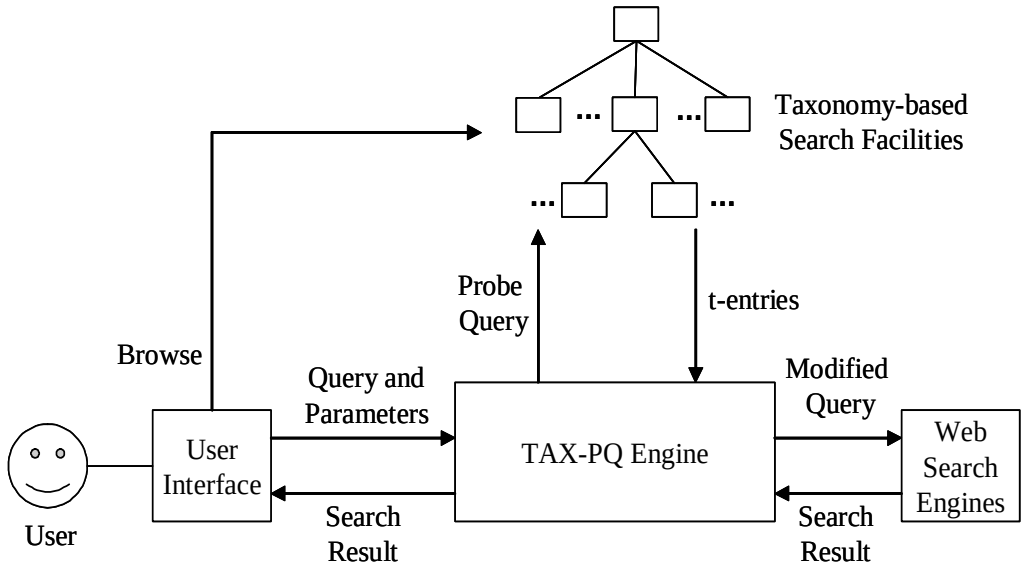

Figure 9.1: Prototype system architecture.

and buttons where the user can put an initial query condition and control the modification process:

- Probing parameters (p, q) that can be used to trade-off between modification time and search result effectiveness.
- Learning parameter α that can be used to control search result effectiveness of the modified queries.
- The selection of taxonomy data used to formulate a query and the selection of a target search engine where the modified query is sent.
- The location of search term occurrences (in title or body) and the use

of stemming.

- Interactive selection of query modifiers.

The TAX-PQ engine part is a core element of the prototype system. It implements the enhanced query modification methods and bridges the gap between the taxonomy-based search facilities and the search engines. Specifically, this part does the following:

- Fetches the sample t-entries from the taxonomy-based search facility to form training examples.
- Creates a classifier using the proposed learning algorithms and extracts a query modifier from the created classifier.
- Modifies the initial query condition, sending it to a target search engine and delivering the search result to the user.

Details of the TAX-PQ engine part are explained below.

9.3 TAX-PQ Engine

Figure 9.2 shows the architecture of the TAX-PQ engine part. It consists of six modules:

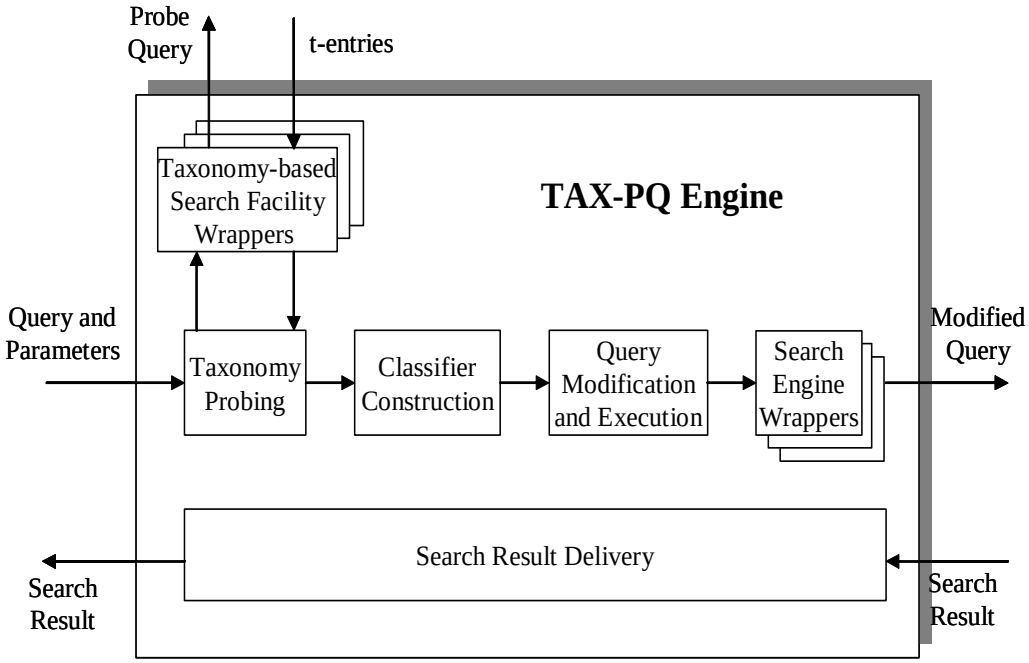

Figure 9.2: TAX-PQ engine.

- Taxonomy probing module
- Taxonomy-based search facility wrappers
- Classifier construction module
- Query modification and execution module
- Search engine wrappers
- Search result delivery module

The taxonomy probing module creates probe queries based on the given user query and parameters and sends them to the wrapper of the selected taxonomy-based search facility. Each probe query requests a page of matched t-entries. Basically, it requests a list of matched t-entries in a specific range located in a specific category. The number of probe queries (or requested pages) depends on the sampling parameters (p, q) given by the user.

The taxonomy-based search facility wrapper forwards each probe query received to a taxonomy-based search facility after transforming it into a query format recognized by the search facility. The wrapper then gets the probe query result, extracts t-entries with their associated category information and passes them to the taxonomy probing module. To shorten the probing and t-entry extraction time, the taxonomy probing module and wrapper are implemented as thread processes.

The classifier construction module takes sample t-entries from the taxonomy probing module and constructs training examples. This is done by transforming each t-entry as a tuple with binary features and labeling it as relevant or irrelevant, depending on its associated category name. The module then constructs a classifier using one of the proposed learning algorithms, where the algorithm is selected on the basis of the target search engine selected by the user.

The query modification and execution module extracts a query modifier from the learned classifier, modifies the initial query condition given by the user using the modifier and passes the modified condition to an appropriate search engine wrapper. On receiving the modified query condition, the wrapper passes it to the target search engine after transforming it into a Boolean query format recognized by the target search engine. The search result delivery module takes the search result from the search engine and passes it to the user.

In the interactive mode, the classifier construction module repeats the classifier construction n times, where $n = 10$ in this prototype. Each time it divides the training set randomly into $GSet$ and $VSet$, it changes the random seed. In this manner, the resulting $GSet$ and $VSet$ in each division are different and might result in different classifiers. The query modification and execution module then ranks the resulting modifiers based on their occurence

number and lets the user choose one of the modifiers to modify his query.

9.4 Session Example

We now give a session example demonstrating the use of the prototype system. Suppose a user tries to find information about *diet products* (the name, price etc.). He opens the system, sets ODP as the taxonomy-based search facility, and Google as the target search engine. He then goes to category "/Shopping/Health/" of the ODP and enters keyword "diet" in the query field. The default value of probing parameter (p, q) is $(20, 100)$ and α is 0. Figure 9.3 shows the screen shot of the graphical user interface.

Suppose the user wants to modify his query interactively. Figure 9.4 shows the screen shot of the query modifier selection. The user may select one of the modifiers to modify his query. Figure 9.5 shows the results from Google when the user selects the first modifier "weight AND supplements". Precision of the top ten matches is 1, in contrast to precision of the initial condition "diet" which is only 0.2. Suppose now he sets the target search engine to AltaVista and lets the system choose the modifier for him. Figure 9.6 shows the query result from AltaVista. The modified query condition is "diet AND ((loss AND book) OR (loss AND supplements))". Precision of the top ten matches is 1, in contrast to precision of the initial condition

which is only 0.4.

9.5 Implementation Issues

As the experiment results in previous chapters show, most of the query modification time is occupied by the probing time (the time to sample t-entries from the taxonomy-based search facility). The probing time strongly depends on the "distance" between the system and the taxonomy-based search facility. In the real world, the system and the taxonomy-based search facility can locate on the same search service platform or on geographically separated search service platforms. For easy reference, we call the case where the system and taxonomy-based search facility are 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 tightly integrated configuration can be used by web sites that provide

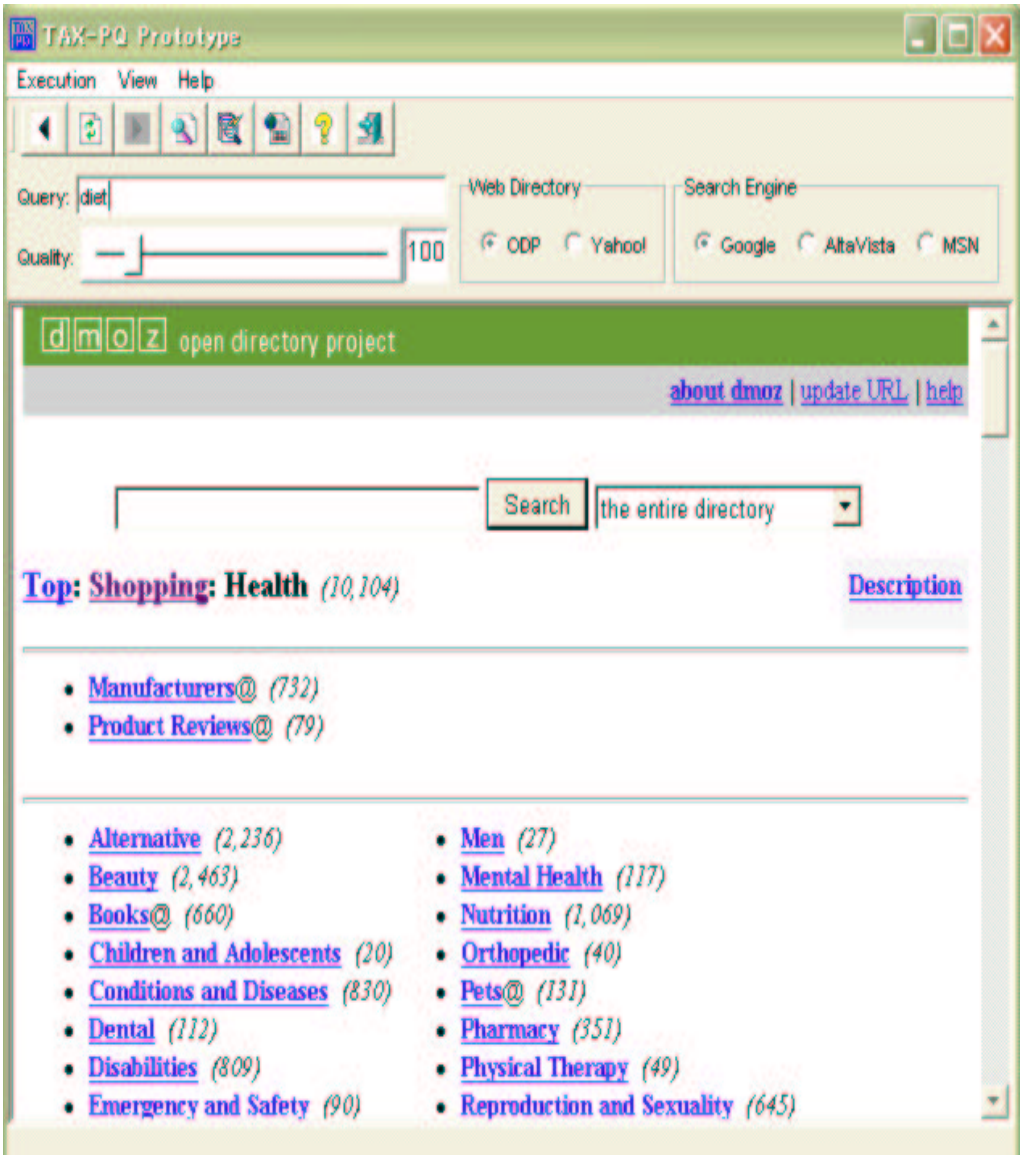

Figure 9.3: Screen shot of the user interface of the prototype system.

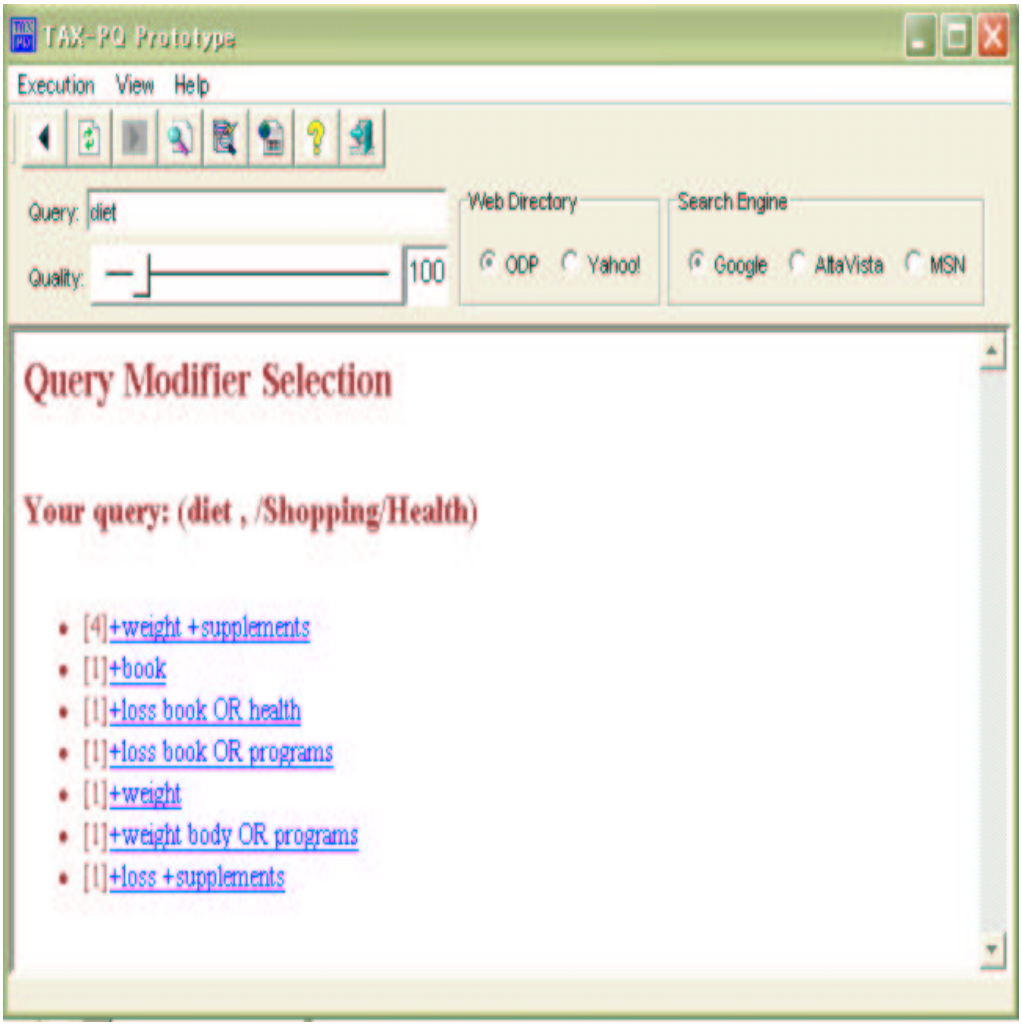

Figure 9.4: Query modifier selection in the interactive mode.

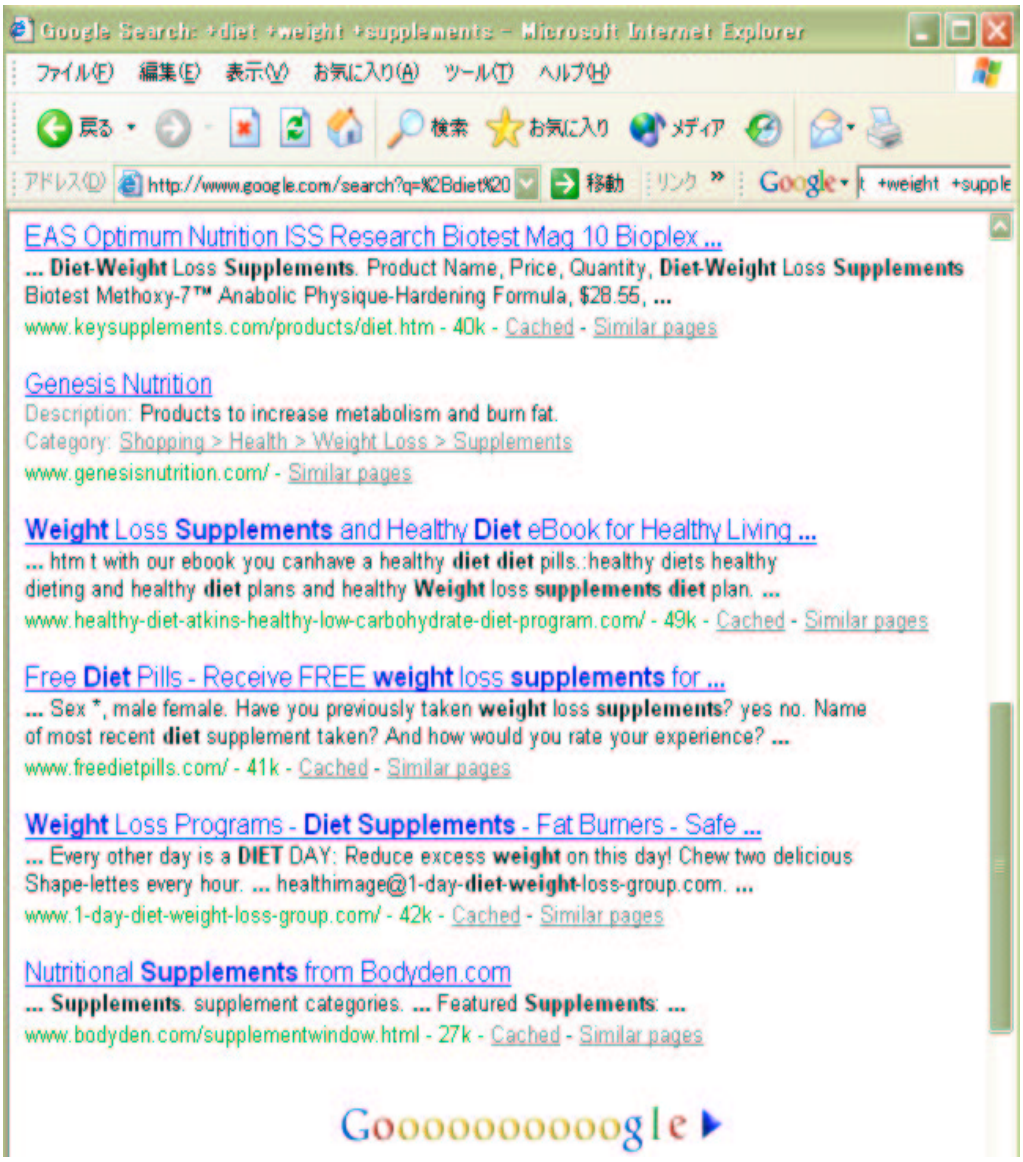

Figure 9.5: Result from Google for query ("diet", "/Shooping/Health/").

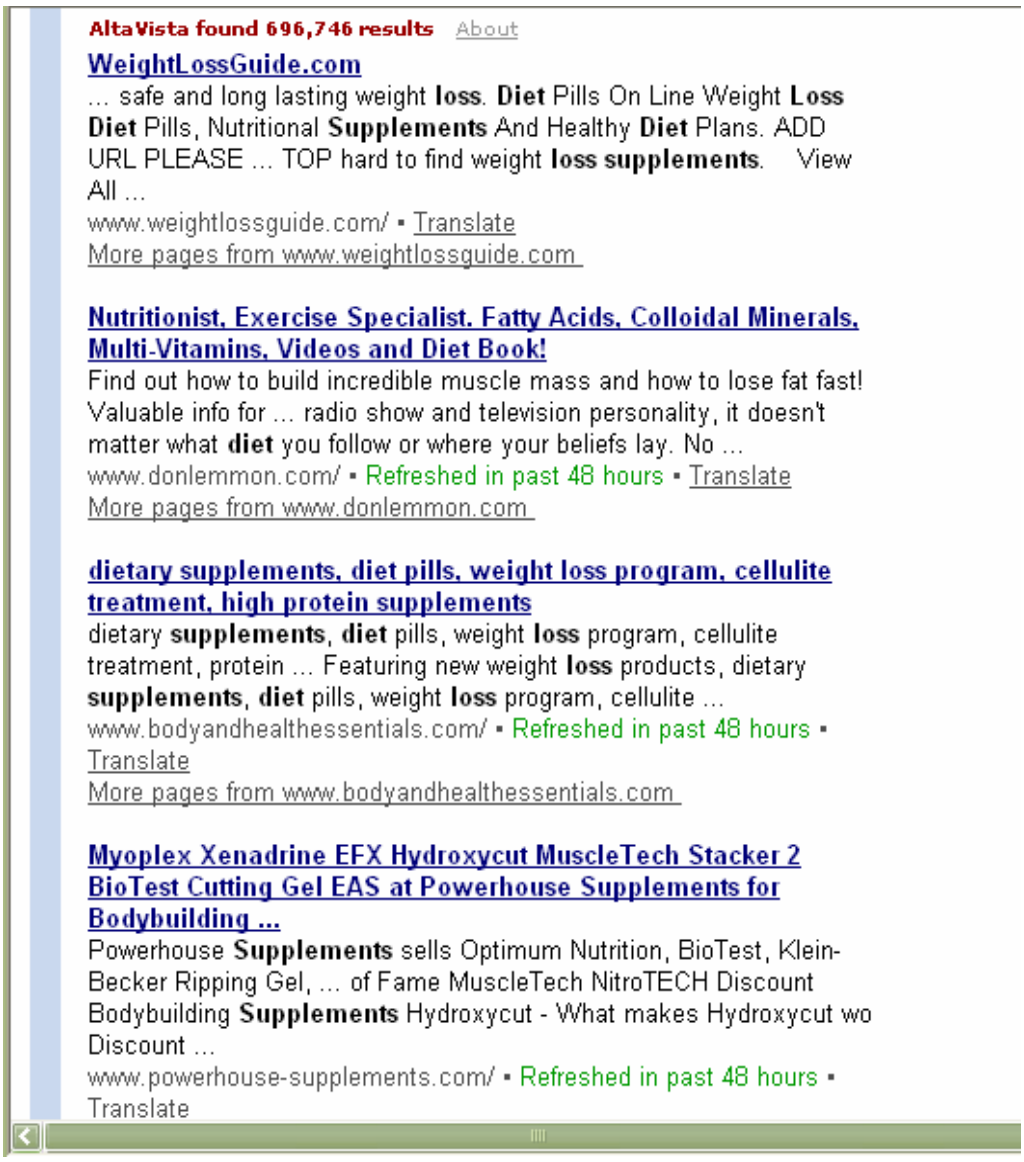

Figure 9.6: Result from AltaVista for query ("diet", "/Shooping/Health/").

a taxonomy-based search facility such as ODP [Net] and Yahoo [Yah], and web sites that provide an integrated web search service, where the taxonomy-based search is included in it. Examples are Google [Goo], Altavista [Alt] and MSN [Cor]. Note that these web sites could also be managed by an individual person since taxonomy data such as that owned by ODP is publicly available.

The shortcoming of the tightly integrated configuration is that it typically provides only one type of taxonomy data to the user. For example, Google provides taxonomy data from ODP while Altavista and MSN provide taxonomy data from Looksmart [Loo]. The reason is that not all taxonomy data is in the public domain, and it is costly to manage and maintain the taxonomy data locally. Beyond that, for the integrated web search services that usually get a copy of the taxonomy data from their search service partners and for the individually managed web sites that get a copy of public domain taxonomy data, the data is often out of date. This occurs because those services do not use live taxonomy data and have 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 dissertation and is used by the prototype system. As the previous experiments clarify, the partial probing

technique makes it possible to shorten the delay time while maintaining high search result effectiveness. Further, delay time can be further shortened by implementing taxonomy probing as a multi-thread process.

The advantage of the loosely integrated configuration is that the association of a taxonomy-based search facility 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 facilities, the data is always up-to-date, eliminating the need to periodically refresh data. This configuration not only provides up-to-date taxonomy data to users but also offers a rich selection of taxonomy data that can be used to formulate the queries. That is why we use the loosely integrated configuration in the prototype system.

Chapter 10

Conclusions and Future Work

10.1 Conclusions

This dissertation proposes a *novel query modification framework* for web search. This framework uses existing taxonomy maintained by a taxonomy-based search facility and classification learning. It features a dynamic taxonomy probing and classifier construction that can significantly improve search result effectiveness. This framework was described in detail in Chapter 4.

Chapter 5 described a basic query modification method based on the proposed framework that employs existing classification learning algorithms. Extensive experiments reveal the following characteristics of the basic method:

- It can significantly increase search result effectiveness of the modified query condition and outperforms the static method.
- It allows the user to trade-off search result effectiveness with query modification time by adjusting parameters to control the amount of data sampled from the taxonomy-based search facility.

We also discussed limitations of the basic method and presented a solution. The limitations originate from use of the existing classification learning algorithms; the solution is to propose a classification learning algorithm that is aware of search engine constraints and effectiveness of the modified queries. Taking into account the two Boolean query formats supported by many search engines, we proposed two new classification learning algorithms and enhanced query modification methods.

Chapter 6 started by describing the first proposed learning algorithm called the CDT learning algorithm and then the CDT enhanced query modification method that uses the algorithm. The algorithm has the following two features that distinguish it from the existing learning algorithms. First, it builds a decision tree by explicitly constraining its size. Second, it learns a decision tree based on the estimated search result effectiveness of the modified query. Experiments showed that the algorithm enables the user to control search result effectiveness of the modified queries by adjusting the effectiveness parameter α. We also found that the CDT enhanced query modification

method has the same characteristics as the basic method. Beyond that, it can guarantee that the modified queries can always be processed by the ordinary Boolean search engines.

Chapter 7 described the second proposed learning algorithm called the CCR learning algorithm and the CCR query modification method that uses the algorithm. The CCR learning algorithm shares the same design philosophy with the CDT learning algorithm. It builds a rule based on estimated search result effectiveness of the modified query while constraining the rule size. Another characteristic of the algorithm is that it builds a rule with a head that conforms to the template-based query format. As a result, modified queries from the CCR query modification method can always be processed by the template-based search engines. We also did extensive experiments to evaluate the CCR enhanced method and found that it has the same characteristics as the CDT enhanced method.

Chapter 8 revealed usage of the two enhanced query modification methods. We compared search result effectiveness and document level precision of the two methods and found that the CDT enhanced method generally outperforms the CCR enhanced method. Based on the experiment results, we encourage use of the CDT enhanced method for the ordinary Boolean search engines. We recommend the CCR enhanced method for the template-based search engines.

Chapter 9 described a prototype system that implements the two enhanced query modification methods. The system uses the loosely integrated configuration combined with the partial probing technique. As a result, beyond providing a reasonably fast response time, it also features up to date taxonomy data from multiple sources. The prototype system proved that our query modification framework, especially the enhanced modification methods, can be implemented in the real world.

10.2 Future Work

The most important point needing further investigation is the need to decrease the taxonomy probing time. As explained, most of the query modification time is occupied by the time needed to sample t-entries from the taxonomy-based search facility. In most cases, the ratio of irrelevant t-entries is much larger than that of relevant t-entries. This means that most of the taxonomy probing time is occupied by the time needed to fetch irrelevant samples. So an obvious way to decrease the taxonomy probing time is to"ignore" irrelevant samples and fetch only the relevant ones. This can be achieved by doing one of the following:

- Provide the irrelevant samples locally before run time.
- Construct the classifier using one-class learning algorithms [TD99]

[SPST+99][CB00].

In the first approach, the irrelevant samples need not match the initial query condition. They can be taken from the same or different taxonomy data or from the cache of the proxy server before run time. Recall that the purpose of building a classifier used to modify a given query condition is to distinguish relevant samples in the relevant class from irrelevant ones in the irrelevant class. This reflects the learning process of the proposed algorithms. In the CCR learning process, the algorithm always selects literals that can further improve the density of relevant samples covered by the new constructed condition; in the CDT learning process the algorithm considers only the $RSTs$, whose leaves are all labeled relevant. This tells us that in the learning processes the relevant samples take the lead while the irrelevant ones are only a supplement part. This being the case, a method that provides irrelevant samples locally will not significantly harm classifier performance.

In the second approach, we use the algorithms to construct a classifier for the relevant class. Thus we need not fetch irrelevant samples from the taxonomy-based search facility nor provide them locally. This approach requires us to use an existing learning algorithm such as the one-class SVM [SPST+99] or PEBL framework [YHC02]. The challenge is whether we can make the classifier aware of search engine constraints and search result effectiveness of the modified queries.

Another approach to decreasing probing time is to cache t-entries while the user browses and queries the taxonomy-based search facility. This approach comes from the fact that the user can also query the taxonomy-based search facilities in addition to browsing their taxonomy. With this approach, on receiving a query from the user, the system looks into the cache see if the needed t-entries have been retrieved. If the cache contains them all, they can be used to build the classifier. Otherwise, the remaining (unfetched) t-entries are fetched from the search facility. This approach is more efficient if combined with one of the other approaches described above.

Bibliography

[AD91] H. Almuallim and T. G. Dietterich. Learning with many irrelevant features. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, volume 2, pages 547–552, Anaheim, California, 1991. AAAI Press.

[AFJM95] Robert Armstrong, Dayne Freitag, Thorsten Joachims, and Tom Mitchell. Webwatcher: A learning apprentice for the world wide web. In *AAAI Spring Symposium on Information Gathering*, pages 6–12, 1995.

[Alt] Altavista. http://www.altavista.com/.

[Ask] AskJeeves. http://www.teoma.com/.

[BH99] Jay Budzik and Kristian J. Hammond. Watson: Anticipating and contextualizing information needs. In *62nd Annual Meeting of the American Society for Information Science*, Medford, NJ, 1999.

[BH00] Jay Budzik and Kristian J. Hammond. User interactions with everyday applications as context for just-in-time information access. In *Proceedings of the 2000 International Conference on Intelligent User Interfaces*, New Orleans, Louisiana, 2000. ACM Press.

[BHBK00] Jay Budzik, Kristian J. Hammond, Larry Birnbaum, and Marko Krema. Beyond similarity. In *Proceedings of the 2000 Workshop on Artificial Intelligence and Web Search*. AAAI Press, 2000.

[BMMZ92] F. Bergadano, S. Matwin, R. S. Michalski, and J. Zhang. Learning two-tiered descriptions of flexible concepts: the poseidon system. *Machine Learning*, 8(1):5–43, 1992.

[BP98] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[Bur] U.S. Census Bureau. http://factfinder.census.gov/.

[Buz] Yahoo Buzz. http://buzz.yahoo.com.

[CB00] Colin Campbell and Kristin P. Bennett. A linear programming approach to novelty detection. In *Advances in Neural Information Processing Systems NIPS*, pages 395–401, 2000.

[CDAR98] Soumen Chakrabarti, Byron Dom, Rakesh Agrawal, and Prabhakar Raghavan. Scalable feature selection, classification and signature generation for organizing large text databases into hierarchical topic taxonomies. *VLDB Journal: Very Large Data Bases*, 7(3):163–178, 1998.

[CDI98] Soumen Chakrabarti, Byron E. Dom, and Piotr Indyk. Enhanced hypertext categorization using hyperlinks. In *Proceedings of SIGMOD-98, ACM International Conference on Management of Data*, pages 307–318, Seattle, US, 1998.

[CGRU96] C. Chekuri, M. Goldwasser, Prabhakar Raghavan, and E. Upfal. Web search using automatic classification. In *Proceedings of WWW-96, 6th International Conference on the World Wide Web*, San Jose, US, 1996.

[CK97] J. Carriere and R. Kazman. Webquery: Searching and visualizing the web through connectivity. In *Proc. 6th International World Wide Web Conference*, pages 701–711, 1997.

[CN89] Peter Clark and Tim Niblett. The CN2 induction algorithm. *Machine Learning*, 3:261–283, 1989.

[Coh95] W. W. Cohen. Fast effective rule induction. In *International Conference on Machine Learning*, pages 115–123, 1995.

[Coh96] W. W. Cohen. Learning to classify English text with ILP methods. In L. De Raedt, editor, *Advances in Inductive Logic Programming*, pages 124–143. IOS Press, 1996.

[Com] CompletePlanet. http://www.completeplanet.com/.

[Cor] Microsoft Corp. http://www.msn.com/.

[CR96] H. Chu and M. Rosenthal. Search engines for the world wide web: A comparative study and evaluation methodology. In *Proceedings of the ASIS 1996 Annual Conference*, pages 127–135, 1996.

[CS96] W. W. Cohen and Y. Singer. Learning to query the web. In *AAAI Workshop on Internet-Based Information Systems*, 1996.

[CS98] Liren Chen and Katia Sycara. WebMate: A personal agent for browsing and searching. In *Proceedings of the 2nd International Conference on Autonomous Agents (Agents'98)*, pages 132–139, New York, 1998.

[CvdBD99a] Soumen Chakrabarti, Martin van den Berg, and Byron Dom. Distributed hypertext resource discovery through examples. In *The VLDB Journal*, pages 375–386, 1999.

[CvdBD99b] Soumen Chakrabarti, Martin van den Berg, and Byron Dom. Focused crawling: a new approach to topic-specific Web re-

source discovery. *Computer Networks (Amsterdam, Netherlands: 1999)*, 31(11–16):1623–1640, 1999.

[CZC01] Michael Chau, Daniel Zeng, and Hsinchun Chen. Personalized spiders for web search and analysis. In *ACM/IEEE Joint Conference on Digital Libraries*, pages 79–87, 2001.

[D+02] J. Ding et al. Mining medline: Abstracts, sentences, or phrases? In *Pacific Symposium on Biocomputing (PSB 2002)*, pages 326–337, 2002.

[DC00] Susan T. Dumais and Hao Chen. Hierarchical classification of Web content. In *Proceedings of SIGIR-00, 23rd ACM International Conference on Research and Development in Information Retrieval*, pages 256–263, Athens, GR, 2000.

[DCL+00] Michelangelo Diligenti, Frans Coetzee, Steve Lawrence, C. Lee Giles, and Marco Gori. Focused crawling using context graphs. In *26th International Conference on Very Large Databases, VLDB 2000*, pages 527–534, Cairo, Egypt, 10–14 September 2000.

[DH73] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis.* John Wiley and Sons, 1973.

[Div] Divine. http://nlresearch.northernlight.com/.

[DM96] Wei Ding and Gary Marchionini. Comparative study of web search service performance. In *Proceedings of the ASIS 1996 Annual Conference*, pages 136–142, 1996.

[Eft96] E.N. Efthimiadis. Query expansion. In Martha E. Williams, editor, *Annual Review of Information Systems and Technology*, pages 121–187. 1996.

[FÖ99] Johannes Fürnkranz. Separate-and-conquer rule learning. *Artificial Intelligence Review*, 13(1):3–54, January 1999.

[FGLG02] Gary Flake, Eric Glover, Steve Lawrence, and C. Lee Giles. Extracting query modifications from nonlinear SVMs. In *International World Wide Web Conference*, Honolulu, Hawaii, May 7–11 2002.

[FGM+01] Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. Placing search in context: the concept revisited. In *Proceedings of the 10th International World Wide Web Conference*, pages 406–414, 2001.

[FW94] J. Furnkranz and G. Widmer. Incremental reduced error pruning. In *Proc. 11th International Conf. on Machine Learning*, pages 70–77. Morgan Kaufmann, New Brunswick, NJ, 1994.

[G+01] Eric Glover et al. Improving category specific web search by learning query modifications. In *Symposium on Applications and the Internet, SAINT*, San Diego, CA, January 8–12 2001.

[Gat] NLM Gateway. http://gateway.nlm.nih.gov/gw/cmd.

[GLG+99] Eric J. Glover, Steve Lawrence, Michael D. Gordon, William P. Birmingham, and C. Lee Giles. Web search – your way. *Communications of the ACM*, 1999.

[Goo] Google. http://www.google.com/.

[GW96] S. Gauch and G. Wang. Information fusion with profusion. In *Proceedings of WebNet 96*, 1996.

[GWH60] G. G. Widrow and M. Hoff. Adaptive switching circuits. In *Institute of Radio Engineers, Western Electronic Show and Convention, Convention Record, Part 4*, pages 96–104, 1960.

[Hea95] Marti A. Hearst. Tilebars: Visualization of term distribution information in full text information access. In *Proceedings of the Conference on Human Factors in Computing Systems, CHI'95*, 1995.

[HKP95] Marti A. Hearst, David R. Karger, and Jan O. Pedersen. Scatter/Gather as a tool for the navigation of retrieval results. In *Working Notes AAAI Fall Symp. AI Applications in Knowledge Navigation*, 1995.

[IGS01] Panagiotis G. Ipeirotis, Luis Gravano, and Mehran Sahami. Probe, count, and classify: Categorizing hidden web databases. In *SIGMOD Conference*, 2001.

[Ins] NEC Research Institute. http://citeseer.nec.com/.

[Inv] InvisibleWeb. http://www.invisibleweb.com/.

[Kle99] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[Lan95] Ken Lang. NewsWeeder: learning to filter netnews. In *Proceedings of the 12th International Conference on Machine Learning*, pages 331–339. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA, 1995.

[Lib] Librarian. http://lii.org/.

[Loo] Looksmart. http://www.looksmart.com/.

[LR94] David D. Lewis and Marc Ringuette. A comparison of two learning algorithms for text categorization. In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 81–93, Las Vegas, US, 1994.

[LSBH99] David B. Leake, Ryah Scherle, Jay Budzik, and Kristian J. Hammond. Selecting task-relevant sources for just-in-time retrieval. In *AAAI-99 Workshop on Intelligent Information Systems)*, Menlo Park, CA, 1999. AAAI Press.

[Lyc] Lycos. http://dir.lycos.com/.

[Men97] Filippo Menczer. ARACHNID: Adaptive retrieval agents choosing heuristic neighborhoods for information discovery. In *Machine Learning: Proceedings of the 14th International Conference*, pages 227–235, 1997.

[Mit97] Tom Mitchell. *Machine Learning.* McGraw Hill, 1997.

[MM98] Alexandros Moukas and Pattie Maes. Amalthaea: An evolving multi-agent information filtering and discovery system for the WWW. *Autonomous Agents and Multi-Agent Systems*, 1(1):59–88, 1998.

[MMHL86] R.S. Michalski, I. Mozetie, J. Hong, and N. Lavrae. The multipurpose incremental learning system AQ15 and its testing application to three medical domains. In *Proc. 5th AAAI*, pages 1041–1045, Philadelphia, PA, 1986.

[Mou96] Alexandros Moukas. Amalthaea: Information discovery and filtering using a multiagent evolving ecosystem. In *Proceedings of the Conference on Practical Applications of Agents and Multiagent Technology*, 1996.

[MSG97] U. Manber, M. Smith, and B. Gopal. Webglimpse: Combining browsing and searching. In *Proceedings of 1997 Usenix Technical Conference*, 1997.

[Net] Netscape. http://dmoz.org/.

[OKI+01] Satoshi Oyama, Takashi Kokubo, Toru Ishida, Teruhiro Yamada, and Yasuhiko Kitamura. Keyword spices: A new method for building domain-specific web search engines. In *IJCAI*, pages 1457–1466, 2001.

[PB97] Michael J. Pazzani and Daniel Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine Learning*, 27(3):313–331, 1997.

[PH90] Giulia Pagallo and David Haussler. Boolean feature discovery in empirical learning. *Machine Learning*, 5(1):71–99, 1990.

[PK02a] Said Mirza Pahlevi and Hiroyuki Kitagawa. Taxonomy-based adaptive web search method. In *Proc. 3rd IEEE International Conference on Information Technology: Coding and Computing (ITTC 2002)*, pages 320–325, 2002.

[PK02b] Said Mirza Pahlevi and Hiroyuki Kitagawa. A taxonomy-based focused retrieval method for the web space. In *Proc. Pan-Yellow-Sea International Workshop on Information Technologies for Network Era (PYIWIT 2002)*, pages 127–134, 2002.

[PK02c] Said Mirza Pahlevi and Hiroyuki Kitagawa. A web search method integrating taxonomy-based and crawler-based search

engines. *IPSJ Transaction on Databases*, 43(SIG 9):15–27, September 2002.

[PK03] Said Mirza Pahlevi and Hiroyuki Kitagawa. TAX-PQ: Dynamic taxonomy probing and query modification for topic-focused web search. In *8th International Conference on Database Systems for Advanced Applications (DASFAA 2003)*, 2003.

[PMB96] Michael J. Pazzani, Jack Muramatsu, and Daniel Billsus. Syskill webert: Identifying interesting web sites. In *AAAI/IAAI, Vol. 1*, pages 54–61, 1996.

[Pri] Altavista Prisma. http://www.altavista.com/help/search/pp.

[Pub] Pubmed. http://www.ncbi.nlm.nih.gov/entrez/query.fcgi.

[Qui86] J. R. Quinlan. Induction of decision trees. In *Machine Learning*, pages 81–106, 1986.

[Qui90] J. R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5, 1990.

[Qui93] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, California, 1993.

[Qui95] J. R. Quinlan. MDL and categorical theories (continued). In *Machine Learning: Proceedings of the Twelfth International Conference*, 1995.

[Res] Just Research. http://www.cora.justresearch.com/.

[Rho00] Bradley Rhodes. Margin notes: Building a contextually aware associative memory. In *Proceedings of the International Conference on Intelligent User Interfaces (IUI '00)*, 2000.

[RHW86] D.E. Rumelhart, G. E. Hinton, and R.J. Williams. *in Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, volume 1. Cambridge, MA: MIT Press, 1986.

[RM00] Bradley Rhodes and Pattie Maes. Just-in-time information retrieval agents. *IBM Systems Journal special issue on the MIT Media Laboratory*, 39(3):685–704, 2000.

[RS96] Bradley Rhodes and Thad Starner. The remembrance agent: A continuously running automated information retrieval system. In *The Proceedings of The First International Conference on The Practical Application of Intelligent Agents and Multi Agent Technology (PAAM '96)*, pages 487–495, 1996.

[Sci] Elsevier Science. http://www.scirus.com/.

[SPST+99] B. Scholkopf, J. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. Technical Report 99-87, Microsoft Research, 1999., 1999.

[SSTW02] Sergej Sizov, Stefan Siersdorfer, Martin Theobald, and Gerhard Weikum. BINGO!: Bookmark-induced gathering of information. In *The 3rd International Conference on Web Information Systems Engineering (WISE 2002)*, 2002.

[TC96] Hendrik Theron and Ian Cloete. BEXA: A covering algorithm for learning propositional concept descriptions. *Machine Learning*, 24:5–40, 1996.

[TD99] D. Tax and R. Duin. Data domain description by support vectors. In *Proc. ESANN*, pages 25–256, 1999.

[Tra] Fast Search & Transfer. http://www.alltheweb.com/.

[Web] WebGlimpse. http://webglimpse.org/.

[WZL99] Ke Wang, Senquiang Zhou, and Shiang Chen Liew. Building hierarchical classifiers using class proximity. In *Proceedings of VLDB-99, 25th International Conference on Very Large Data Bases*, pages 363–374, Edinburgh, UK, 1999.

[Yah] Yahoo. http://www.yahoo.com/.

[YHC02] Hwanjo Yu, Jiawei Han, and Kevin Chen-Chuan Chang. PEBL: Positive example based learning for web page classification using SVM. In *the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 25–256, 2002.

[ZE98] Oren Zamir and Oren Etzioni. Web document clustering: A feasibility demonstration. In *Research and Development in Information Retrieval*, pages 46–54, 1998.

[ZE99] Oren Zamir and Oren Etzioni. Grouper: a dynamic clustering interface to Web search results. *Computer Networks (Amsterdam, Netherlands: 1999)*, 31(11–16):1361–1374, 1999.

[Zei] Google Zeitgeis. http://www.google.com/press/zeitgeis.html.

[ZEMK97] Oren Zamir, Oren Etzioni, Omid Madani, and Richard M. Karp. Fast and intuitive clustering of web documents. In *Knowledge Discovery and Data Mining*, pages 287–290, 1997.

List of Publications

(1) Publications Related to this Dissertation

Refereed Journal Papers

1. Said Mirza Pahlevi and Hiroyuki Kitagawa, "A Web Search Method Integrating Taxonomy-based and Crawler-based Search Engines", IPSJ Transaction on Databases (TOD 15), Vol.43, No.SIG 9, pp.15–27, September 2002.

Refereed International Conference Papers

1. Said Mirza Pahlevi and Hiroyuki Kitagawa, "TAX-PQ: Dynamic Taxonomy Probing and Query Modification for Topic-focused Web Search", In 8th International Conference on Database Systems for Advanced Applications (DASFAA 2003), Kyoto, Japan, March 2003. (to appear)

2. Said Mirza Pahlevi and Hiroyuki Kitagawa, "Taxonomy-based Adaptive Web Search Method", In Proc. 3rd IEEE International Conference on Information Technology: Coding and Computing (ITCC 2002), Las Vegas, pp. 320–325, April. 2002.

3. Said Mirza Pahlevi and Hiroyuki Kitagawa, "A Taxonomy-based Focused Retrieval Method for the Web Space", In Proc. Pan-Yellow-Sea International Workshop on InformationTechnologies for Network Era (PYIWIT 2002), Saga, Japan, pp. 127–134, March 2002.

National (Japan) Conference Papers and Technical Reports

1. Said Mirza Pahlevi and Hiroyuki Kitagawa, "An Adaptive Taxonomy-based Query Modification Method for the Web Retrieval", Technical Report of IEICE, Vol.2002, No.67, pp.487–494, July 2002.

2. Said Mirza Pahlevi, Hiroyuki Kitagawa and Yoshiharu Ishikawa, "Classifier-based Focused Retrieval for Text Databases", Technical Report of IEICE, Vol. 101, No. 192, DE2001-56, pp. 153–158, July 2001.

3. Said Mirza Pahlevi and Hiroyuki Kitagawa, "Similarity Search of XML Documents Based on Tag Structures and Contents", 61st IPSJ General Conference (3), pp. 21–22 October 2000.

(2) Other Publications

Refereed Journal Papers

1. Akio Koyama, Leonard Barolli, Said Mirza Pahlevi and Shoichi Yokoyama, "An Exact Explicit Indication Scheme for each VC in ATM Networks", Trans. IEE Japan, Vol.119, No.6, pp.714–723, June 1999.

Refereed Book

1. Akio Koyama, Leonard Barolli, Said Mirza Pahlevi and Shoichi Yokoyama, "Performance Evaluation of Self Detective Congestion Control Scheme for ATM Networks", Advanced Information Processing Technology Series, Information Networking in Asia, G&B publisher, Vol.3, pp.25–36, February 2001.

Refereed International Conference Papers

1. Akio Koyama, Leonard Barolli, Said Mirza Pahlevi and Shoichi Yokoyama, "An Adaptive Rate-based Congestion Control Scheme for ATM Networks", In Proc. 12th International Conference on Information Networking (ICOIN-12), Tokyo, Japan, pp.14–19, January 1998.

2. Said Mirza Pahlevi and Kuninobu Tanno, "An Explicit Rate Indication Scheme for Congestion Control of ABR Service in ATM Networks", In Proc. 10th International Conference on Information Networking (ICOIN-10), Kyung-ju, Korea, pp.569–577, January 1996.

3. Akio Koyama, Kuninobu Tanno, Said Mirza Pahlevi and S. Noguchi, "Performance Analysis of a Ring Priority Self-Token High-Speed LAN", In Proc. 10th International Conference on Information Networking (ICOIN-10), Kyung-ju, Korea, pp.504–509, January 1996.

4. Kuninobu Tanno, Akio Koyama, Said Mirza Pahlevi, Toshihiro Taketa and S. Noguchi, "Performance Evaluation of High-Speed Self-Token Ring LAN", in Proc. International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN 1994), Kanazawa, Japan, pp.294–301, December 1994.

National (Japan) Conference Papers and Technical Reports

1. Said Mirza Pahlevi and Kuninobu Tanno, "Self Detective Congestion Control Scheme for ABR Service in ATM networks", Technical Report of IEICE, Vol. 95, No. 266, pp. 43–48, September 1995.

2. Said Mirza Pahlevi, Kuninobu Tanno and Akio Koyama, "Analysis of Fairness on Self-Token Protocol for High Speed Ring LANs", IPSJ Sig

Notes Distributed Processing Systems, Vol. 94, No. 56, pp. 61–66, July 1994.