

# Query Refinement for Domain-Specific Web Search

Satoshi Oyama



# Abstract

The expansion of the Internet and the number of its users has raised many new problems in information retrieval. The most common way to find information in the web is using web search engines. However, gathering information from the web is a difficult task for a novice user even if he uses the search engines. The user must have experience and skill to find the relevant pages from the large number of documents returned, which often cover a wide variety of topics.

One solution to the problem is to build a domain-specific search system that returns only web documents relevant to a certain domain. In this thesis, we adopt query refinement approach when we build domain specific search systems. This approach is based on the idea that the main difficulty in searching the web is result from the fact that users cannot formulate appropriate queries representing their information needs. We try to support users by changing their queries to appropriate forms.

In this thesis, we present following issues on domain-specific web search.

1. A simple and domain-independent method for building domain-specific web search engines.

Previous domain-specific web search engines are based on human knowledge about the domain in question. Accordingly, they are hard to build and cannot be applied to other domains. We contrive a new method that improves search performance by adding the domain-specific keywords, called *keyword spices*, to the user's input query;

the modified query is then forwarded to a general-purpose search engine. This method enables us to build high quality domain-specific search engines in various domains with a common and simple architecture. We describe a machine learning algorithm, which is a type of decision-tree learning algorithm that can extract keyword splices as a disjunctive normal form of keywords from web documents. We conduct experiments in the cooking domain with a commercial search engine and the results show the high values of precision and recall.

2. A method to generate query refinement rules for finding web pages that mostly satisfy their requests from the collection of domain pages.

Usually the user cannot obtain the most relevant page after the first search. The user has to examine the search results and modify the query. The system that can support these query refinement processes of a variety of users, including mobile users, is required. Therefore, we propose that the system presents an initial search result to the user and gradually navigates him/her to better information. We describe following two navigation methods using association rules of the keywords, automatic modification of the query to satisfy search conditions and proposing a new search condition to be considered. The key to the success of the navigation is the quality of rules. Thus we have developed two pruning algorithms of association rules. We remove rules that contain keywords derived from the formats of web pages based on strongly connected components of the keyword graph. We also apply  $\chi^2$  test for the significance of keyword co-occurrence in the rules for eliminating redundant rules.

3. An architecture for building information agents based on multiple domain-specific web search systems.

Around the world, social information infrastructures based on the Internet that support people's access to urban information have been developed. For example, in Kyoto, we have developed Digital City Kyoto Prototype. To make full use of dynamical and various in-

formation in digital cities, interactive agent interfaces are requested. However, it is difficult to develop general-purpose agents that can treat various information needs of users. Therefore, we design an architecture that combines multiple domain-specific web agents. In this architecture, back-end agents, which extract and organize relevant information from the Internet, are built on the basis of domain-specific web search engines. Front-end agents, which have natural language and animated character interfaces, cooperate with back-end agents for meeting various information needs.



# Acknowledgments

I would like to express my sincere gratitude to my thesis supervisor, Professor Toru Ishida at Kyoto University, for his continuous guidance, valuable advice, and helpful discussions. Without his warm encouragement, I would not have been able to accomplish this thesis.

I gratefully acknowledge valuable comments of other members of my thesis committee, Professor Yahiko Kambayashi and Professor Hiroshi Okuno at Kyoto University. I am deeply indebted to my adviser, Professor Toyooki Nishida at the University of Tokyo for his kind advice and useful discussions.

I wish to thank Associate Professor Yasuhiko Kitamura at Osaka City University. He is the leader of the agent project in Laboratories of Image Information Science and Technology, where the initial idea of the keyword spice method was born. I am also thankful to Teruhiro Yamada and the other project members for their kind support.

I am grateful to Kaoru Hiramatsu at NTT for his cooperation in Digital City Kyoto Experiment Forum.

Lecturer Hirofumi Yamaki and Assistant Professor Hideyuki Nakanishi at Kyoto University always gave me kind advice and helpful comments. I would also thank my colleagues at Professor Ishida's laboratory. The research of domain specific search engines is collaborative work with Takashi Kokubo and Takayuki Yoshizumi.

Finally, I wish to thank all members of Professor Ishida's laboratory for their help and fruitful discussion.

This research was supported in part by followings:

- The Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research(A), 11358004, 1999.
- The Ministry of Education, Science, Sports and Culture, Grant-in-Aid for University and Society Collaboration, 11792025, 1999.
- Laboratories of Image Information Science and Technology
- JSPS Research Fellowships for Young Scientists

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Background . . . . .	1
1.2	Approach . . . . .	3
1.3	Outline of the Thesis . . . . .	5
<b>2</b>	<b>Web Search</b>	<b>9</b>
2.1	General-Purpose Web Search Engines . . . . .	9
2.1.1	Crawlers . . . . .	10
2.1.2	Query language . . . . .	11
2.1.3	Ranking function . . . . .	13
2.2	Meta Search Engines . . . . .	14
2.3	Evaluation Methods for Search Engines . . . . .	15
2.3.1	Precision and Recall . . . . .	16
2.3.2	Harmonic Mean of Precision and Recall . . . . .	17
2.4	Difficulty in General-Purpose Web Search . . . . .	17
<b>3</b>	<b>Keyword Spice Method for Building Domain-Specific Web Search Engines</b>	<b>21</b>
3.1	Introduction . . . . .	21
3.2	Keyword Spice Model . . . . .	24
3.3	Extracting Keyword Spices . . . . .	27
3.3.1	Collecting Sample Documents . . . . .	27
3.3.2	Decision Tree Learning Algorithms . . . . .	28
3.3.3	Rule Post-Pruning for Simplifying Keyword Spices . . . . .	34

3.3.4	Comparison with Other Pruning Methods . . . . .	42
3.3.5	Keyword Spice Extraction Algorithm . . . . .	49
3.4	Experiments . . . . .	51
3.4.1	Experimental Settings . . . . .	51
3.4.2	Precision . . . . .	51
3.4.3	Estimated Recall . . . . .	53
3.5	Application: A Cooking Recipe Search Engine . . . . .	55
3.6	Summary . . . . .	57
<b>4</b>	<b>Applying Association Rules to Information Navigation</b>	<b>59</b>
4.1	Introduction . . . . .	59
4.2	Information Navigation using Association Rules . . . . .	61
4.2.1	Association Rules . . . . .	61
4.2.2	Using Association Rules in Interactive Web Search . . . . .	62
4.2.3	Search Aspects in Domain-Specific Web Search . . . . .	64
4.2.4	Association Rules to Satisfy Search Aspects . . . . .	65
4.2.5	Association Rules to Propose New Search Aspects . . . . .	67
4.3	Pruning Association Rules . . . . .	67
4.3.1	Keyword Clustering for Pruning Association Rules . . . . .	68
4.3.2	Statistical Tests for Pruning Association Rules . . . . .	68
4.4	Experiments . . . . .	70
4.4.1	Experimental Settings . . . . .	70
4.4.2	Pruning Association Rules . . . . .	72
4.4.3	Effectiveness of Association Rules in Information Navigation . . . . .	73
4.5	Related Work . . . . .	76
4.6	Summary . . . . .	77
<b>5</b>	<b>Cooperative Information Agents for Supporting Daily Life</b>	<b>79</b>
5.1	Introduction . . . . .	79
5.2	Cooperative Information Agents on the Internet . . . . .	80
5.2.1	Architectures for Information Agents on the Internet . . . . .	80
5.2.2	Multi-agent system: Venus & Mars . . . . .	82

5.3	Digital Cities . . . . .	83
5.4	Cooperative Information Agents for Digital Cities . . . . .	87
5.4.1	Architecture for Cooperative Information Agents . . .	87
5.4.2	Front-end Agents . . . . .	89
5.4.3	Back-end Agents . . . . .	91
5.5	Summary . . . . .	93
<b>6</b>	<b>Conclusion</b>	<b>95</b>
6.1	Contributions . . . . .	95
6.2	Future Directions . . . . .	98
	<b>Publications</b>	<b>109</b>



# List of Figures

2.1	Architecture of general-purpose search engines . . . . .	11
2.2	Architecture of meta-search engines . . . . .	15
2.3	Precision and recall . . . . .	16
3.1	The filtering model of building domain-specific web search engines . . . . .	23
3.2	The keyword spice model of building domain-specific web search engines . . . . .	25
3.3	Sampling with input keywords to increase the ratio of positive examples . . . . .	26
3.4	A simple decision tree . . . . .	29
3.5	A decision tree of the training set . . . . .	33
3.6	The decision tree pruned by statistical testing . . . . .	45
3.7	Effect of reduced-error pruning . . . . .	48
3.8	The decision tree pruned by reduced-error pruning . . . . .	48
3.9	Precision of queries forwarded to Goo . . . . .	52
3.10	Precision of the query “pork AND salt”forwarded to Goo . . . . .	54
3.11	A cooking recipe search engine . . . . .	56
4.1	Keyword Associations . . . . .	63
4.2	Applying association rules to a query . . . . .	65
4.3	Pruning association rules by keyword clustering . . . . .	69
4.4	Distribution of keyword appearance . . . . .	71
5.1	Venus & Mars (provided by Teruhiro Yamada) . . . . .	83

5.2	The Three Layers Model for Digital Cities . . . . .	86
5.3	Architecture for Cooperative Information Agents for Digital Cities . . . . .	87

# List of Tables

2.1	Statistics of the number of terms per query submitted to Altavista in September 1998[Silverstein <i>et al.</i> , 1998]	18
2.2	Statistics of the number of operators per query submitted to Altavista in September 1998[Silverstein <i>et al.</i> , 1998]	18
3.1	Collected web documents in the cooking domain	28
3.2	The number of relevant and irrelevant documents	31
3.3	The initial decision tree	32
3.4	Pruning results	42
3.5	2×2 contingency table	43
3.6	The number of nodes after statistical test pruning	46
3.7	Pruning results by reduced error pruning	49
3.8	Average precision over the index of Goo	53
3.9	Estimated recall of the queries with keyword spices over the index of goo	53
4.1	Keyword co-occurrence with no significance	70
4.2	Search aspects	73
4.3	Number of extracted rules for each aspect	74
4.4	Rate of effective rules	76



# Chapter 1

## Introduction

### 1.1 Background

The number of people using the Internet in the world was estimated at more than 500 million at the end of August 2001. In Japan, the number of Internet users is more than 40% of the population and still increasing rapidly. The popularization of the Internet does not only mean the increase of its users but also change the usage of it. In the beginning, the Internet was mainly used for research or business but now people have started to use the Internet for their daily life. Among various internet services the Web is one of the most popular, and most of internet users access it frequently. We can obtain information on various topics of daily life such as shopping, eating, health-care, education etc. The number of web pages is also rapidly increasing year by year and it amounts to more than 800 million pages.

The expansion of the number of web users and the amount of information stored in the web has raised many new problems in information retrieval. The most common way to find information in the web is using a web search engine. Web search engines have common appearances with conventional information retrieval systems. The user forms query words to express his information needs and the system returns documents that are estimated to be relevant. Information retrieval has a long history and many researches for improving search systems have been carried out. However,

information retrieval in the web has new problems that have not been dealt with preceding information retrieval researches.

First, there is a difficulty in treating unorganized and enormous information sources. Conventional information retrieval systems usually treat more homogeneous information such as news articles or research papers. In contrast, the Web is composed of pages on various topics, which are written by various people. Therefore, each page in the web is different from each other in its format and quality.

Second, the users of the web are different people from that of conventional information retrieval systems. These systems are intended for specialized people such as researchers or librarians. On the other hand, web search engines are used by various people that include those who are not accustomed to information systems. In other words, the most users of the web are “casual users.”

These users have different requirements for the search system. The professional users usually think comprehensiveness is important even if it requires many costs. For example in search of patent information, one oversight will be fatal. When searching information for daily life the comprehensiveness is not usually required but finding sufficient information in reasonable time is more important. For example, no one spends several hours to find a recipe for today’s dinner.

Thus, we cannot assume a user’s precise knowledge about a search target or skills to utilize complicated search functions. With this condition we have to help them to find the pages that satisfy their needs from vast amount of heterogeneous web pages without taking much time.

In addition, conventional information retrieval systems are aimed at the users in front of desktop terminals, but the web is accessed from everywhere. Actually, in Japan, the people who access the Internet via mobile phones are rapidly increasing these few years, and they take up the greater part of all internet users. The more useful for daily life the information on the web become, the more they want to use the information anytime and anywhere. However, the existing user interfaces of web search engines are designed for large screens and are not suitable for people using mobile ter-

minals.

These days the idea of universal design, the design of products or services to be usable by all people, is considered more important. Designing information retrieval system on the web based on this idea is required because the information on the web will become indispensable for daily life in near future. Especially the consideration for people who have difficulty in using direct manipulation interface like the aged or the blind is important.

With this background various software agents that support interactively users to search information on the web have been developed. However, the most of these agents are designed for guiding users in a certain web site and the architecture is specialized for each site. Thus, the general architecture that enables us to develop software agents for various information sources on the web is required.

## **1.2 Approach**

The objective of this research is to explore various methods for resolving difficulties in web search addressed in the previous section. We propose methods for developing web search system that can be easily used by ordinary users in their daily life.

To accomplish the object mentioned above, we apply domain-specific approach, in which we made special-purpose search systems for each domain. By targeting the specific purpose, we can reduce the difficulty caused by heterogeneity of the web and ambiguity of user's needs. Of course, many domain-specific web search system have been developed so far. However, existing methods for building domain-specific search systems require specialized facilities or human expertise knowledge and it is hard to apply same method for developing the system in other domains. Therefore, in this thesis we pursue domain-independent methods that enable us to develop domain-specific search system for various domains.

In this thesis, we will explore new methods for developing domain-specific web search systems at low cost. In these methods we needs no

large facilities such as local databases or web spiders that largely consume network bandwidth. Because many domain-specific search systems will be built for each domain, the requirement for saving resources is indispensable to the success of this approach.

We also aim at a method for building domain-specific web search systems without human expertise. For this purpose we try to extract the knowledge for web search in a certain domain automatically using machine learning and data-mining methods from domain documents collected and classified by hand. Preparing training examples is a time-consuming task, but still much easier than finding effective search knowledge by hand for most developers of the system.

In this thesis, we adopt query refinement approach when we build domain specific search systems. The system supports a user to improve his ambiguous query to more appropriate one. This approach is based on the idea that the main difficulty in searching the web is result from the fact that users cannot formulate appropriate queries representing their information needs.

Domain-specific web search system has two major research issues as follows:

- How to gather web pages relevant to a certain domain from the web that contain pages of various topics.
- How to find web pages that mostly satisfy user's request among collected domain pages.

To solve the former problem, we try to construct domain-specific search engines. By using a domain-specific search engine, the user can only obtain web pages of a certain domain and needs not to be troubled by many irrelevant pages. In our method, the query submitted by a user is refined automatically by the system so as to obtain only relevant pages to the domain in question.

For the latter problem, we pursue a method in which the system gradually navigates a user to better information while refining the query. To

refine the query, the system uses information obtained from the interaction with the user. This method realizes quite different interaction patterns from that of existing search engines and it is especially suitable for people using mobile terminals. We try to extract query refinement rules used in this method automatically from the web documents of a certain domain.

We also address needs and usefulness of information agents that support users to obtain information for their everyday life from the web. We adopt the domain-specific approach to realize these agents as well. First, we build domain-specific agents for various domains and then we make them cooperate each other. We propose a cooperative architecture for information agents that support various aspects of life.

### **1.3 Outline of the Thesis**

This thesis consists of six chapters, including this chapter as the introduction.

In Chapter 2, the general framework of web information retrieval is presented. This research is motivated by the problem in using existing general-purpose search engines. Thus, we describe functions and architecture of general-purpose search engines. This research is based on the idea of meta search engines. Therefore, we also explain the architecture of meta search engines. As a preliminary to succeeding chapters, we also introduce the criteria for evaluation search systems, like precision or recall, and explain their meanings and way to calculate them. We refer to preceding surveys of web search engine users and discuss why general-purpose search engines are difficult to use for many users. We argue that the main difficulty of web search is come from the fact that users cannot construct appropriate queries express their requirements.

In chapter 3 we argue that building domain-specific search engines is an effective solution to the above problem. We describe the two popular existing methods for developing domain-specific search engines, building domain-specific induces and using information filtering techniques. We

show that these methods requires massive human and network resources. Therefore, we propose a new method that enables us to build domain-specific search engines without troubled with these problems. Our keyword spice method improves search performance by adding domain-specific keywords, called keyword spices, to the user's input query; the modified query is then forwarded to a general-purpose search engine. We describe a machine learning algorithm, which is a type of decision-tree learning algorithm that can extract keyword spices. The point of this algorithm is to find small size of keyword spices while keeping the performance of them. Thus, we describe the pruning method of our algorithm in detail. To demonstrate the value of the proposed approach, we conduct experiments in the domain of cooking. We confirm the effectiveness of our method using the common criteria for evaluating information retrieval systems.

Chapter 4 describes a method for supporting the user to find relevant information from the collection of web documents in a certain domain. We argue that interaction patterns of existing search engines cannot answer the various web users, especially those who access the web with mobile terminals. To solve the problem we describe a new interface that people can refine their queries while interacting with the system. We propose to use association rules of keywords in navigation, which is discovered from the web documents of the domain. We show that applying association rule while considering domain-specific search aspects enable effective navigation. The key to the success of the navigation is the quality of rules. Thus, we describe algorithms for selecting appropriate association rules. One is deleting keywords irrelevant to the domain using keyword clustering and the other is removing redundant rules using statistical significance tests.

Chapter 5 explores the architecture of information agents on the web, especially, agents that support people's daily life. We first introduce various activities that construct information infrastructure for supporting people and discuss the roles of software agents on them. We argue that a multi-agent architecture that combine many domain-specific web agents is effective to build a system that can handle various information request arise in daily life. We introduce a prototype of multi-agent information system based on this

domain-specific approach.

Chapter 6 concludes the thesis summarizing the result obtained through this research. We also address the prospect of the future research considering the evolution of web technologies in near future.



# Chapter 2

## Web Search

### 2.1 General-Purpose Web Search Engines

The most frequently used method to obtain relevant information from the vast web is using a search engine. About 85% of users use search engines to find web pages [gvu, 1998]. Today web search engines are indispensable parts of the World Wide Web system.

There are two different forms of search engines. The first is search engines based on web directories, represented by Yahoo!\*, in which manually collected web pages are classified into categories according to their subjects. The users can find information they want by following the topic trees. They can also find web pages that contain submitted keywords in their titles or summaries among web pages in the whole directory or under specified categories. The advantage of web directories is users can find web pages of relatively high quality because the experienced staffs have selected web pages to register. The ratio of relevant pages in returned results by keyword search is also high because the scope of search is limited to the titles or human-maintained summaries. The disadvantage is people sometimes cannot find any results because the amount of pages registered is rather small compared with the whole web. The user fails to obtain the web page when

---

\* <http://www.yahoo.com>

he uses keywords that are not included in the title or the summary even if the page actually contains the keyword in its body.

The other is crawler-based search engines such as AltaVista<sup>†</sup> and Google<sup>‡</sup>. This kind of search engines are constructed by making indices to the web documents gathered by software called crawlers. Keywords are extracted from collected documents and user can find web pages by submitting keywords to the search engine. The advantage of crawler-based search engines is the vast amount of web pages collected. Thus, the user can find web pages that meet their needs with high possibility. The disadvantage is the user often obtains too many results that are not relevant to the intended topic because all web pages that contain submitted keywords are returned by the system.

As mentioned above these two types of search engines have both advantages and disadvantages. Thus, search engines combining the merits of these two types are emerged these days. For example, Yahoo! returns the result of crawler based search engine Google if there is no web page that contains user-specified keywords in the directory.

Figure 2.1 is a typical architecture of crawler-based general-purpose search engines.

### 2.1.1 Crawlers

Crawlers are software for automatically collecting pages from the web. Crawlers analyze HTML and extract hyperlinks to other pages, then collect pages recursively by following the hyperlinks. Crawlers are used not only for constructing search engines but also for collecting web pages for personal use. Efficient crawling is a key to the quality and quantity of the search engine's index[Cho *et al.*, 1998]. The larger the index of the search engine is, the greater the possibility of finding web pages on specified topic becomes. The problem in using web crawlers is they consume a lot of network bandwidth. The traffic caused by web crawlers occupies a large por-

---

<sup>†</sup><http://www.altavista.digital.com/>

<sup>‡</sup><http://www.google.com/>

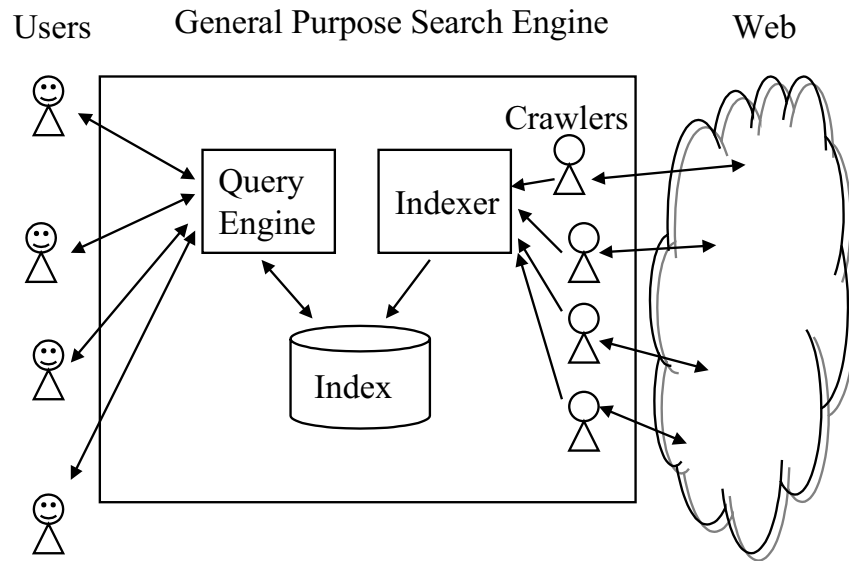


Figure 2.1: Architecture of general-purpose search engines

tion of all web traffic. Thus, the developers of search engines have to carefully implement their crawlers so as not to burden the network facilities.

### 2.1.2 Query language

Query languages determine how users can specify their information need. Most search engines provide the function of “keyword search”, which returns documents that contain users’ input keyword. To specify user’s request more precisely, the user usually have to combine several keywords. How the user can combine keywords is different for each search engines, We will describe following several representative query languages.

## Boolean Query

The most popular form of combining keywords is to use Boolean operators. The operators we can use in a Boolean Query are:

- **AND** This corresponds to the conjunction in Boolean logic. The query ( $k_1$  AND  $k_2$ ) selects all documents that contain both keywords.
- **OR** This corresponds to the disjunction. The query ( $k_1$  OR  $k_2$ ) selects all documents that contain at least one of the two keywords.
- **NOT** This corresponds to the negation. However, the straightforward implementation of this operator causes a problem. The query (NOT  $k_1$ ) selects all documents that do not contain the keyword, but the result is usually the vast majority of Web pages. In commercial search engines, the use of NOT as a monomial operator is not permitted, and this is used the form of ( $k_1$  AND NOT  $k_2$ ).

These operators can be applied to the results of other operators recursively.

The syntax of these operators is different for each search engines. For example, some search engines use “AND” for conjunction but others use “&” for it.

Most of search engines presumes “AND” or “OR” operators when a user inputs several keywords. Many users are not familiar with the syntax of Boolean queries. For these users search engine provide checkboxes annotated “Include all the words” or “Include any of the words” to specify Boolean operators. Some search engines have more elaborate functions for helping users to formulate Boolean queries. For example, in Fresheye<sup>§</sup>, a user inputs keywords by asking several question like “What to search?”, “Any other names to describe it?”, or “Something you want to exclude?”, and then the system automatically formulate a Boolean query.

The main advantage of Boolean queries is its simplicity and the user can grasp the effect of operators intuitively. The disadvantage of Boolean

---

<sup>§</sup><http://www.fresheye.co.jp/>

queries is that it sometimes retrieve too many or too small documents because it is based on exact matching.

### **Other operators**

Some search engines provide operators that are more sophisticated. For example, in Lycos the query ( $k_1$  NEAR  $k_2$ ) retrieves documents in which the two keywords appear within 25 words and the query ( $k_1$  BEFORE  $k_2$ ) operator retrieves documents in which the two keywords appear in this order. Users may also specify where the keywords appear (in title, URL, etc.)

### **Natural Languages**

Users can submit queries in natural languages, like “Suggest a good Chinese restaurant in Kyoto” to some search engines. The system make morpheme analysis for a submitted query and extract keywords, then translate them into a Boolean query using presumed operator “AND” or “OR”. Using Natural Language in queries is not popular in web search so far. It cannot show definite advantage to keyword-based search because of its limited power of natural language processing and inability to handle a series of queries in a dialogue.

### **2.1.3 Ranking function**

A ranking function determine the order in which matched documents are displayed to users. This heavily influences the effectiveness of search because in many cases the number of search results is much larger than the user can read at a glance. In the classical Boolean retrieval model, all documents that satisfy a Boolean query are treated equally, but in practical most search engines that adopt Boolean query give weight to the results. Even if the input query is ( $k_1$  OR  $k_2$ ), most search engines return documents that contain both of the keywords in higher ranks.

There are several heuristics to rank documents, for example, a document that contains keywords in its title may be more relevant to the topic. Google

has made public the basic idea of its PageRank algorithm[Brin and Page, 1998] that use link analysis of the web documents. However, details of ranking function are usually company secrets and are not made public. The information used to rank documents is not limited to that can be derived from the set of documents, but external information such as past access to the pages is also used.

## 2.2 Meta Search Engines

The web is growing so rapidly that no single search engine can catch up with it. Now the web contains about 800 million pages and no engine indexes more than about 16% of the web[Lawrence and Gilg, 1999]. The overlap among search engines is very small[Bharat and Broder, 1998] and this indicates that combining the results of search engines improves coverage of search results.

A meta search engine enables us to search several search engines in parallel. The meta search engine automatically modifies queries before forwarding because each engine differs in the syntax of acceptable queries. It eliminates duplication of the URLs from results returned by search engines and displays them to the user according to its ranking policies. The general architecture of meta search engines is presented in Figure 2.2.

MetaCrawler[Selberg and Etzioni, 1997]<sup>¶</sup> is the most popular meta search engine and integrates general-purpose search engines such as Lycos, Infoseek, WebCrawler, Excite, AltaVista, Yahoo and so on.

Search engines are different from each other in their strong topics. SavvySearch is a meta search engine that learns to identify which search engines are most appropriate for each query[Howe and Dreilinger, 1997]. It maintains associations between search engines and query terms based on past search results. Not only just forwarding an input query to search engines, some meta search engines can modify input query for improving search results. Inquirus[Lawrence and Giles, 1998] is a meta search engine

---

<sup>¶</sup><http://www.metacrawler.com/>

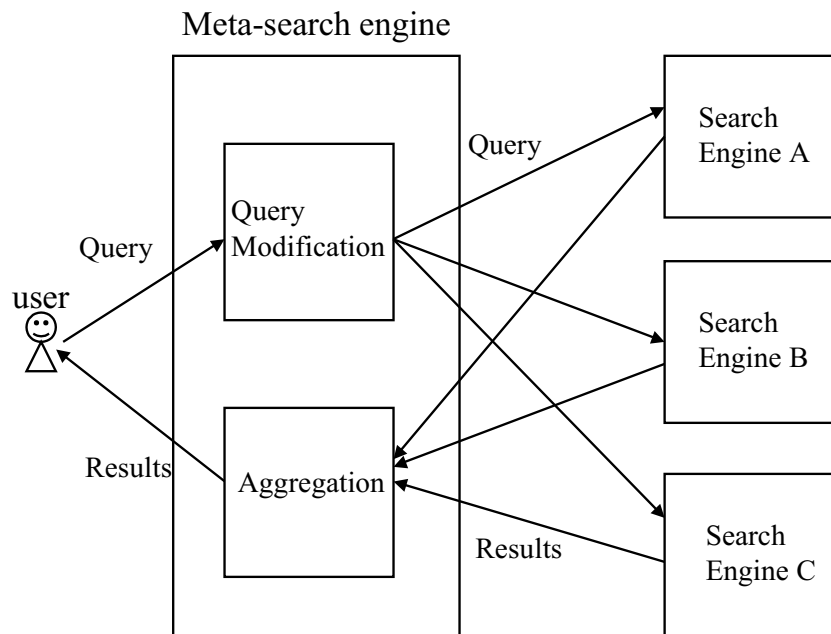


Figure 2.2: Architecture of meta-search engines

that can transform queries so that they achieve high precision. It transforms a query in an interrogative sentence into a specific expressive form for the answer. We adopt the idea of meta-search when we build domain-specific search engines described in Chapter 3.

## 2.3 Evaluation Methods for Search Engines

The methods for evaluating traditional information retrieval systems are also used for evaluating search engines. However, there are several characteristic aspects in the evaluation of search engines.

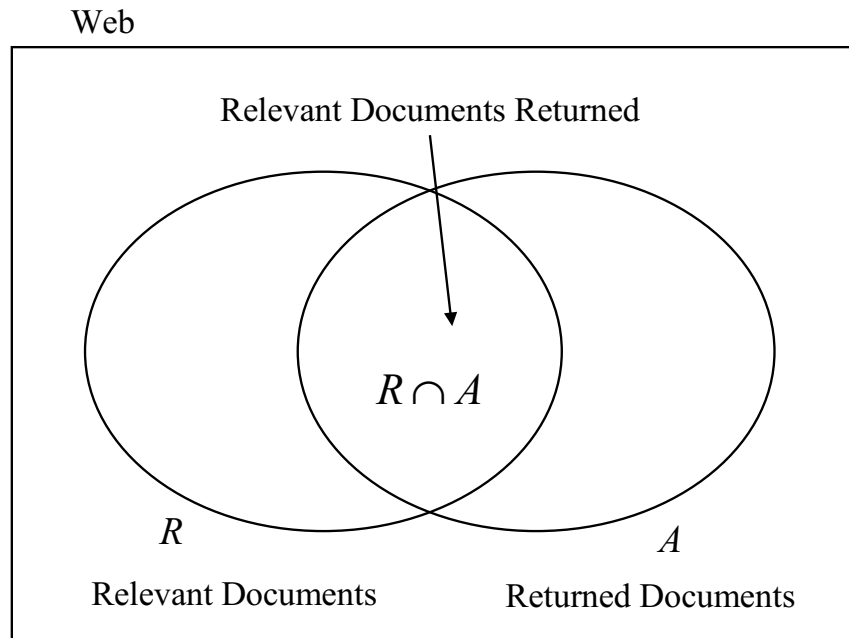


Figure 2.3: Precision and recall

### 2.3.1 Precision and Recall

Precision and recall are the most commonly used measures for evaluating search results. Let  $R$  be the set of all relevant documents ( $R$  is usually unknown) and  $A$  be the set of all returned documents by the query (Figure 2.3).

Precision and recall are defined as follows:

- **Precision** The ratio of number of relevant documents to the number of returned documents i.e.,

$$Precision = \frac{|R \cap A|}{|A|}$$

- **Recall** The ratio of the number of relevant documents returned to the

number of relevant documents in existence i.e.,

$$Recall = \frac{|R \cap A|}{|R|}$$

The user is not presented with all matched documents when he uses a search engine. Instead, a search engine presents matched documents sorted by its ranking function. Therefore, the precision of “Top N” ranked documents is more appropriate for evaluating search engine’s results.

When we evaluate web search engines, the recall of a query is much harder to calculate than the precision because no one knows how many relevant documents exist in the web.

### **2.3.2 Harmonic Mean of Precision and Recall**

Precision and recall are not independent each other. Usually a query with high precision results in low recall and a query with high recall yields low precision. In some situation, we want to use some single measure that combines precision and recall.

The harmonic mean of precision and recall is defined as follows[Shaw Jr. *et al.*, 1997]:

$$F = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}}$$

$F$  takes values between 0 and 1. It is 1 when both precision and recall are 1, and 0 when no relevant document is retrieved. The harmonic mean weights low values more heavily than high values. High values of  $F$  occur only when both precision and recall are high. Thus seeking a query with a high  $F$  value results in obtaining a well-balanced query between precision and recall.

## **2.4 Difficulty in General-Purpose Web Search**

Traditional information retrieval systems are designed for experts in the domain, for example, librarians or scientific researchers. They usually have

detailed domain knowledge and eager to learn the usage of their information retrieval systems.

In contrast, in many cases, a casual user does not know exactly what he/she really wants and so cannot give enough information about the goal to the search engine. It is said that users submit shorter queries to web search engines than those submitted to traditional information retrieval systems. There are several surveys on usage of general-purpose search engines [Jansen *et al.*, 1998; Silverstein *et al.*, 1998] that confirm this. Analysis of the query log of Altavista in September 1998 is precisely reported in [Silverstein *et al.*, 1998].

Table 2.1: Statistics of the number of terms per query submitted to Altavista in September 1998 [Silverstein *et al.*, 1998]

0 terms in query	20.6%
1 term in query	25.8%
2 terms in query	26.0%
3 terms in query	15.0%
more than 3 terms in query	12.6%
average terms in query	1.74

Table 2.2: Statistics of the number of operators per query submitted to Altavista in September 1998 [Silverstein *et al.*, 1998]

0 operators in query	79.6%
1 operators in query	9.7%
2 operators in query	6.0%
3 operators in query	2.6%
more than 3 operators in query	2.1%
average operators in query	0.41

Figure 2.1 shows the statistics of the number of terms per query. More

than the half of the all queries contain only 1 or 2 terms. The average number of terms in a query is 2.35.

Using Boolean operators is much more difficult for ordinary users. Figure 2.2 presents the statistics of usage of operators in query. Queries without any operators account for about 80 % of all queries.

Short queries usually cause poor search results because of ambiguity of words. The web contains documents of various topics and ambiguous queries match with many documents that are not relevant to user's requests.

In addition, most web users are not accustomed to information retrieval systems and hesitate to read "How to Use" or "Search Tips" provided by search engines. This indicates that if search engines become more sophisticated, people will not be able to utilize advanced search options. Thus, we have to develop a simple user interface for searching the web without assuming user's effort to understand complicated functions of search engines.



# Chapter 3

## Keyword Spice Method for Building Domain-Specific Web Search Engines

### 3.1 Introduction

One solution to the difficulty in web search is to build a domain-specific search engine[McCallum *et al.*, 1999]; an engine that returns only those web pages relevant to the topic in question.

This chapter proposes a new method for building domain-specific search engines automatically that it is based on applying machine learning technologies to determine keyword occurrence in web documents.

Several research papers have described domain-specific web search services. A straightforward method to build a domain-specific web search engine is to make indices to domain documents by running web-crawling spiders that collect only relevant pages. Cora\*[McCallum *et al.*, 1999] is a domain-specific search engine for computer science research papers. It web-crawling spiders effectively explore the web by using reinforcement learning techniques. SPIRAL[Cohen, 1998] or WebKB[Craven *et al.*, 1998] also use crawlers. These systems offer sophisticated search functions be-

---

\*<http://cora.whizbang.com/>

cause they establish their own local databases and can apply various machine learning or knowledge representation techniques to the data. Unfortunately, domains such as personal homepages or cooking pages which are dispersed across many web sites, are not well handled by spiders since the time and network bandwidth consumed are excessive. Accordingly, such types of systems are suitable only for those domains that have few web sites.

Reusing the large indices of general-purpose search engines to build domain-specific ones is a clever idea[Etzioni, 1996]. For example, Ahoy!<sup>†</sup> [Shakes *et al.*, 1997] is a search engine specialized for finding personal homepages. It forwards the user's query to general-purpose search engines and sifts out irrelevant documents from the returned ones to increase precision by domain-specific filters. We call this the *filtering model* for building domain-specific search engines (Figure 3.1). Ahoy! has a learning mechanism to assess the patterns of relevant URLs from previous successful searches, but overall accuracy basically depends on human knowledge.

One solution to the above problem is to make domain filters automatically from sample documents. Automatic text filtering, which classifies documents into relevant and non-relevant ones, has been a major research topic in both information retrieval[Baeza-Yates and Ribeiro-Neto, 1999] and machine learning[Mitchell, 1997].

We can use various machine learning algorithms to find such filters if the training examples, which consist of documents randomly sampled from the web together with their manual classification, are available. Unfortunately, making such training examples is real barrier because the web is very large, and randomly sampling the web will provide only a small likelihood of encountering the domain in question. In fact, most studies on text classification have been applied to e-mail, net news, or web documents at limited sites where the ratio of positive examples is rather high. Thus, previous methods of text classification cannot be directly applied to the problem of building domain-specific web search engines.

---

<sup>†</sup><http://ahoy.cs.washington.edu:6060/>

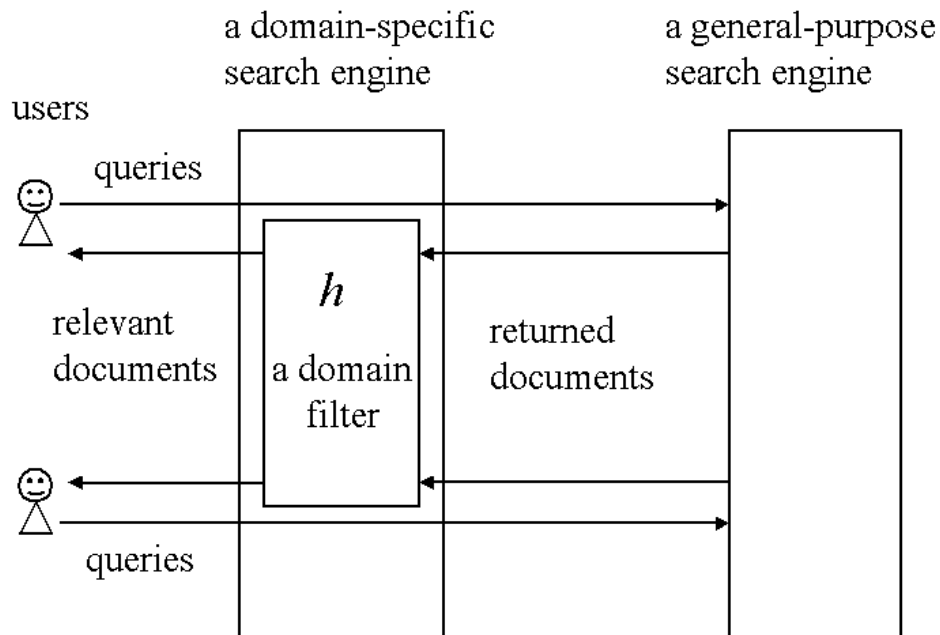


Figure 3.1: The filtering model of building domain-specific web search engines

When one of the authors was using a popular Japanese search engine (Goo<sup>‡</sup>) to find some beef recipes, he input the obvious keyword *gyuniku* (beef), but only 15 of the top 25 returned pages (60%) pertained to recipes. He hit on the idea of adding another keyword *shio* (salt) to the query, at which point all but one of the returned pages(96%) contained recipes. Surprised at this enhancement, he used the same approach for other ingredients such as pork and chicken... the same improvement in search performance was seen. This indicated the possibility of making a domain-specific search engine simply by adding a few keywords to the user's query and forwarding

<sup>‡</sup><http://www.goo.ne.jp/>

the modified query to a general-purpose search engine. Our *keyword spice* method is generalization of this finding.

The keyword-spice method considers only those web pages that contain the user’s input query keyword, not all web pages. This eliminates the problem of finding positive examples and enables us to make domain-specific search engines at low cost.

The remainder of this chapter is organized as follows: Section 3.2 presents the idea of building domain-specific search engines using keyword spices. Section 3.3 describes a machine learning algorithm for discovering keyword spices. Section 3.4 evaluates our method. Section 3.5 shows the application of keyword spices and our conclusions are given in Section 3.6.

## 3.2 Keyword Spice Model

Here we introduce some notations to define the machine learning problem. We let  $\mathcal{D}$  denote the set of all web documents;  $\mathcal{D}_t$  denotes the set of documents relevant to a certain domain. The target function (an ideal domain filter) that correctly classifies any document  $d \in \mathcal{D}$  is given as

$$f(d) = \begin{cases} 1 & \text{if } d \in \mathcal{D}_t \\ 0 & \text{otherwise} \end{cases}$$

We let  $\mathcal{K}$  be the set of all keywords in the domain and let  $\mathcal{H}$  be the hypothesis space that is composed of all Boolean expressions where any keyword  $k \in \mathcal{K}$  is regarded as a Boolean variable. We adopt the Boolean hypothesis space because most commercial search engines can accept queries written in Boolean expressions.

A Boolean expression of keywords can be regarded as a function from  $\mathcal{D}$  to  $\{0, 1\}$  when we assign 1(true) to a keyword (Boolean variable) if the keyword is contained in the document and 0(false) otherwise. In the filtering model, the problem of building a domain filter is equal to finding hypothesis  $h$  that minimizes the error rate

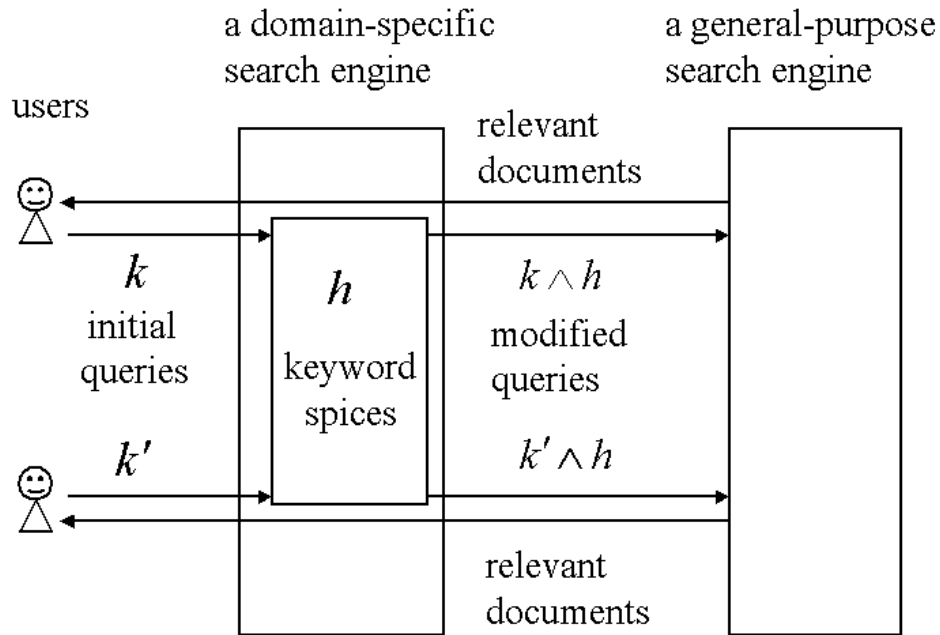


Figure 3.2: The keyword spice model of building domain-specific web search engines

$$\frac{1}{|\mathcal{D}|} \sum_{d \in \mathcal{D}} \delta(h(d), f(d))$$

Note: quantity  $\delta(h(d), f(d))$  is 1 if  $h(d) \neq f(d)$ , 0 otherwise.

The keyword spice model does not filter documents returned by a general-purpose search engine. Instead, it combines the user's input query with a domain-specific Boolean expression (keyword spice), which better classifies the domain documents, and passes the extended query to a general-purpose search engine (Figure 3.2). This model is just the reverse of the filtering model.

Our method is based on the idea that when we build a domain-specific

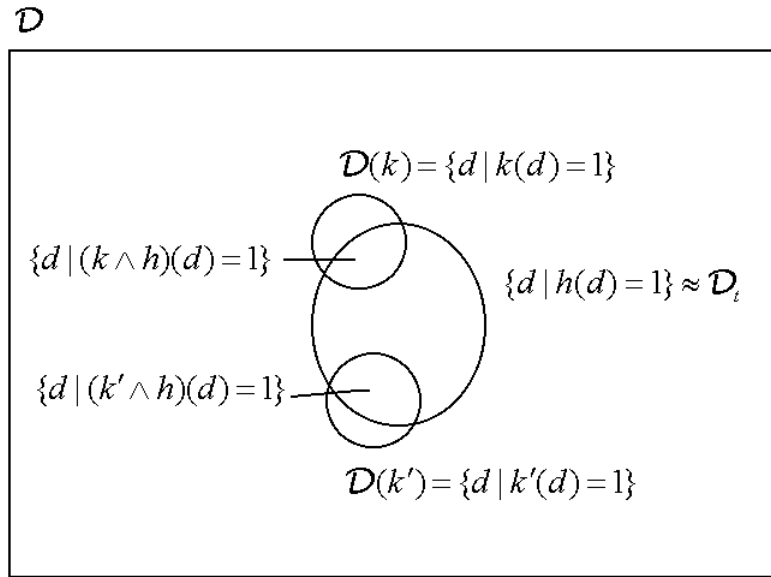


Figure 3.3: Sampling with input keywords to increase the ratio of positive examples

web search engine, we need consider only those web pages that contain the user's input query keywords; not all web pages.

As described in Figure 3.3, the scope of sampling is reduced from set  $\mathcal{D}$ , all web documents, to  $\mathcal{D}(k)$ , the set of web pages that contain input keyword  $k$ ; this increases the ratio of positive examples  $\{d \mid (k \wedge h)(d) = 1\}$ . This idea makes it easier to create training sets and it becomes possible to build a domain filter, which is not possible with random sampling.

By using domain filter  $h$ , we modify the user's input query  $k$  to  $k \wedge h$ , so the returned documents contain  $k$  and are included in the domain. In short,  $h$  is the *keyword spice* for the domain.

## 3.3 Extracting Keyword Spices

### 3.3.1 Collecting Sample Documents

It is rather easy to find good keyword spices for any input keyword  $k$  (for example “beef”), because we input the keyword in a general-purpose search engine and only use web documents returned. The problem is to find the keyword spices that provide enough generalization to handle all future user keywords.

We let  $p(k)$  denote the probability of that a user will input keyword  $k$  to a domain-specific search engine. Then

$$\sum_{k \in \mathcal{K}} p(k) \sum_{d \in \mathcal{D}(k)} \frac{1}{|\mathcal{D}(k)|} \delta((k \wedge h)(d), f(d))$$

is the expectation of the error rate when users try to locate domain documents using this system.

The Boolean expression that minimizes the above expectation value is the most effective keyword spice. It would be best to make training examples using  $p(k)$  but we do not know  $p(k)$  beforehand. Obviously, we have to start with some reasonable value of  $p(k)$ , and modify the value as statistics on input keywords are collected.

In this chapter, we attempt to build a recipe search engine and choose several input keyword candidates in the cooking domain. We assume that all candidates have the same possibility of occurrence and gathered two thousand sample pages of the cooking domain that contained human-entered keywords in Japanese: *gyuniku* (beef), *toriniku* (chicken), *piman* (paprika), *jagaimo* (potato), *kabocha* (pumpkin), *daikon* (radish), *sake* (salmon), *tofu* (tofu), *tomato* (tomato), and *shiromizakana* (whitefish). To find and download web pages containing the above input keywords, we input the keywords in a general-purpose search engine “Goo” which is used in the recipe search engine we tries to build. We ignore dead links or invalid pages and collected the same number of web pages for each keyword. We examined the pages collected and classified them as either relevant or irrelevant by hand (Table 3.1).

Table 3.1: Collected web documents in the cooking domain

Keyword	relevant	irrelevant	total
beef	47	153	200
chicken	88	112	200
paprika	79	121	200
potato	49	151	200
pumpkin	42	158	200
radish	64	136	200
salmon	15	185	200
tofu	45	155	200
tomato	33	167	200
whitefish	103	97	200
Total	565	1435	2000

### 3.3.2 Decision Tree Learning Algorithms

We can apply any decision tree learning algorithm like ID3[Quinlan, 1986b] to discover keyword spices because that correctly classify all examples and it is easy to convert a tree into Boolean expressions, which are accepted by most commercial search engines. Figure 3.4 shows a simple decision tree that classifies documents.

The node indicates attribute, the value of branch indicates the value of the attribute, and the leaf indicates the class. In order to classify a document, we start at the root of the tree, examine whether the document contains the attribute (keyword) or not and take the corresponding branch. The process continues until it reaches a leaf and the document is asserted to belong to the class corresponds the value of the leaf. This tree classifies web documents into  $T$  and  $F$ , and the web document, for example, that does not include “tablespoon”, does “recipe”, does not “home”, and does not “top” belongs to the class  $T$ .

We make the initial decision tree using an information gain measure[Quinlan, 1986b] for greedy search without using any pruning tech-

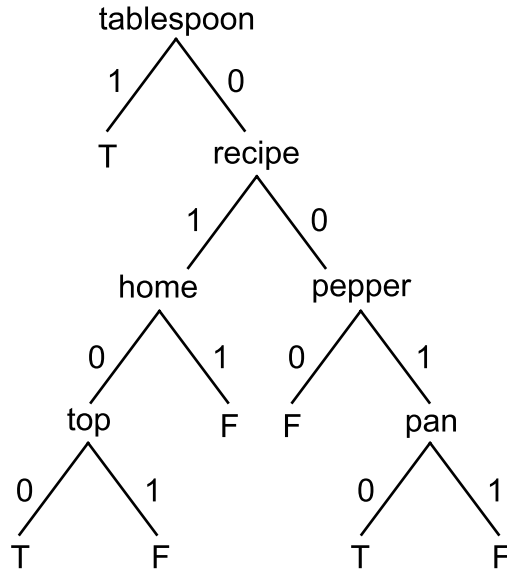


Figure 3.4: A simple decision tree

nique. If attribute  $A$  with values  $\{A_1, A_2, \dots, A_v\}$  is used for the root of the decision tree, it will partition an arbitrary collection  $C$  into  $\{C_1, C_2, \dots, C_v\}$  where  $C_i$  contains those objects in  $C$  that have value  $A_i$  of  $A$ . Let  $C_i$  contain  $t_i$  object of class  $T$  and  $f_i$  of class  $F$ . The expected entropy of the subtree for  $C_i$  is given by

$$I(t_i, f_i) = -\frac{t_i}{t_i + f_i} \log_2 \frac{t_i}{t_i + f_i} - \frac{f_i}{t_i + f_i} \log_2 \frac{f_i}{t_i + f_i}$$

The expected entropy of the tree  $A$  caused by knowing the value of attribute is then obtained as the weighted average of the entropies of subtrees

$$E(A) = \sum_{i=1}^v \frac{t_i + f_i}{t + f} I(t_i, f_i)$$

where the weight for the  $i$ th branch is the proportion of the objects in  $C$  that belong to  $C_i$ .

ID3 examines all candidate attributes and chooses the attribute  $A$  that minimize  $E(A)$  and then uses the same process recursively to form decision trees for the residual subsets  $C_1, C_2, \dots, C_v$ .

In our case, each keyword is used as an attribute whose value is 1 (when the document contains this keyword) or 0 (otherwise) and the number of attributes is large enough (several thousands) to make a tree that can classify all examples.

The algorithm for making a decision tree and extracting keyword spices from the tree is as follows:

1. Collect a set of web documents by submitting domain specific keywords to a general-purpose search engine.
2. Classify the web documents into the positive and negative examples.
3. Extract all keywords as attributes from the web documents.
4. Examine the values 1 or 0 of all the attributes (keywords) in the each web documents.
5. Make the decision tree using an information gain measure without any pruning technique.
6. Convert the learned tree into a set of positive rules by creating one rule for each path from the root node to each leaf nodes. This classifies positive web documents and is keyword spices.

In step 6, we make a set of only positive rules (rules whose conclusions are positive). Our aim is to make a Boolean expression query that specifies the domain documents and that can be entered into search engines; accordingly, we consider only positive rules. For each path in the inducted tree that ends in a positive result, we make a rule whose precondition is a conjunction of all keywords on the path.

Decision trees usually grow very large which triggers the over-fitting problem. Furthermore, commercial search engines cannot accept too-complex queries and so we have to simplify the inducted rules. There are several pruning as follows:

Table 3.2: The number of relevant and irrelevant documents

		Trials				
		1	2	3	4	5
Training set	relevant	275	267	286	277	299
	irrelevant	725	733	714	723	701
Validation set	relevant	290	298	279	288	266
	irrelevant	710	702	721	712	734

- Rule post-pruning
- Statistical testing
- Reduced-error pruning

In these approaches, the available data are split into two sets of examples: the *training set*, which is used to form the learned hypothesis, the *validation set*, which is used to evaluate the accuracy of this hypothesis over subsequent data. We paid no attention to whether the page is relevant or irrelevant and which keywords were input. Thus, each set was randomly composed of documents containing the input keywords. We performed 5 trials in which the sample pages were split randomly in this fashion. Table 3.2 shows the number of relevant and irrelevant documents in the training set and validation set of the 5 trials.

Figure 3.5 presents the decision tree in the first trial (The original attributes (keywords) are Japanese.). This tree has 45 nodes and the keyword spice extracted from the decision tree is the following:

(ingredients AND NOT *takada* AND NOT almost AND NOT trend AND NOT speciality AND NOT *wa* AND NOT goods AND NOT body) OR (tablespoon AND NOT question AND NOT ingredients) OR (boil AND NOT front AND NOT tablespoon AND NOT ingredients) OR (suitable AND NOT boil AND NOT tablespoon AND NOT ingredients) OR (proper

AND NOT suitable AND NOT boil AND NOT tablespoon AND NOT ingredients) OR (slice AND liking AND NOT proper AND NOT suitable AND NOT boil AND NOT tablespoon AND NOT ingredients) OR (taste AND NOT batter AND enough AND NOT liking AND NOT proper AND NOT suitable AND NOT boil AND NOT tablespoon AND NOT ingredients) OR (dressing AND NOT slice AND liking AND NOT proper AND NOT suitable AND NOT boil AND NOT tablespoon AND NOT ingredients) OR (drainer AND NOT enough AND NOT liking AND NOT proper AND NOT suitable AND NOT boil AND NOT tablespoon AND NOT ingredients) OR (*asatuki* AND NOT drainer AND NOT enough AND NOT liking AND NOT proper AND NOT suitable AND NOT boil AND NOT tablespoon AND NOT ingredients)

This keyword spice composed of 10 rules and 65 keywords is too complex to enter a search engine. Table 3.3 shows the number of nodes of the decision tree and the number of rules and keywords of the keyword spices from the decision tree.

Inducted trees are very large and after translating trees to rules we have more than 10 rules; the number of keywords in these rules exceeded 62. This number is too large to permit entry into commercial search engines. To discover the practical keyword spices we must prune the decision trees.

Table 3.3: The initial decision tree

	Trials				
	1	2	3	4	5
Nodes	45	55	47	49	49
Rules	10	15	13	15	10
Keywords	65	89	76	87	62

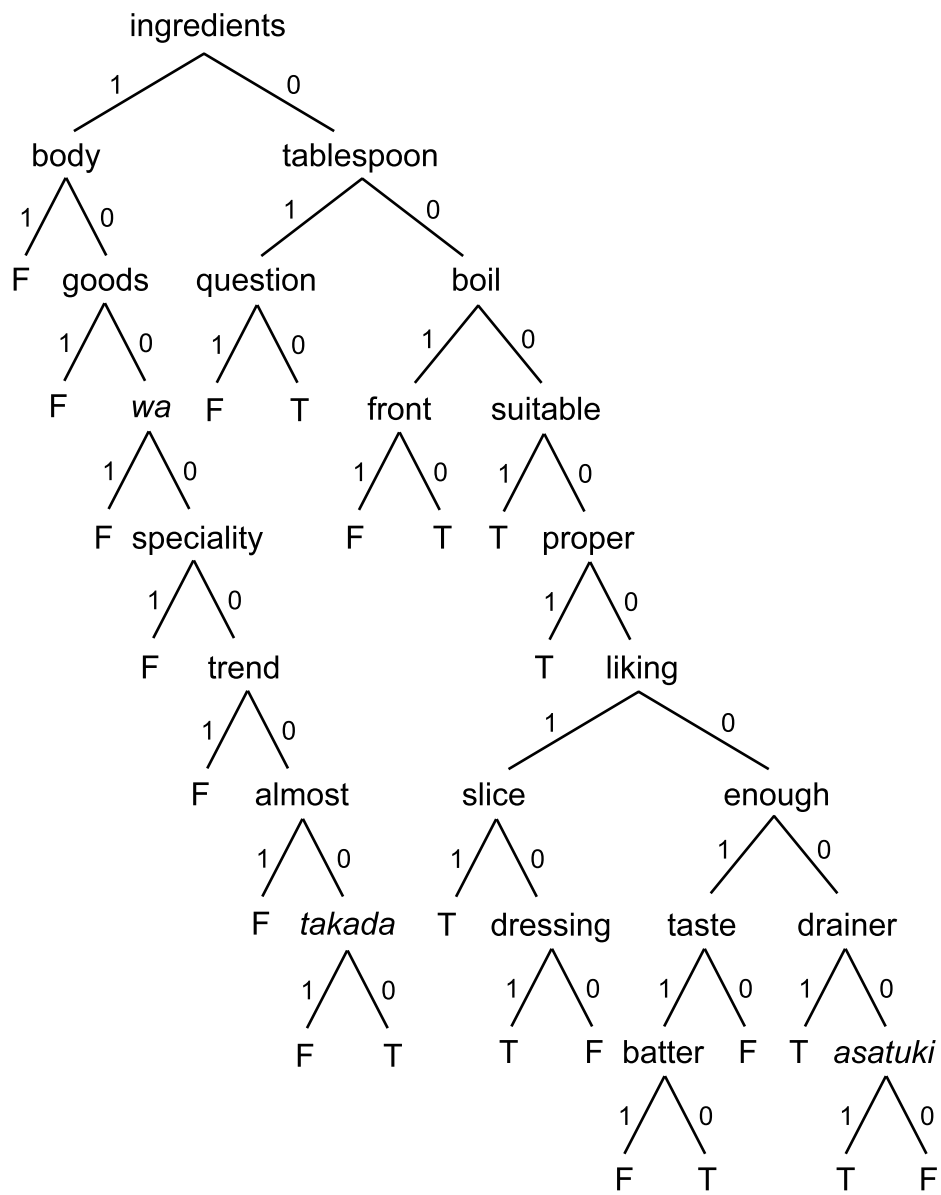


Figure 3.5: A decision tree of the training set

### 3.3.3 Rule Post-Pruning for Simplifying Keyword Spices

We adopt the algorithm based on rule post-pruning algorithm, which convert the decision tree to rules before pruning, because there are following advantages[Mitchell, 1997]:

- Converting the tree to rules enable us to distinguish among the different contexts in which a decision node is used. We can treat the same attribute tests that occur in different rules independently. In a tree, we have only two choices, removing the decision node completely or just leaving it.
- Converting the tree to rules enable us to remove any precondition keywords in the rules equally. In the tree, nodes near the root are more difficult to remove than the nodes near the leaves.

Rule post-pruning is regarded as one of the successful method for finding high accuracy hypotheses[Mitchell, 1997]. A variant of rule post-pruning is used by C4.5[Quinlan, 1993]. Original rule post-pruning consists of the following steps:

1. Make the decision tree that most accurately classifies the training set, without considering the overfitting problem.
2. Convert the tree into an equivalent set of rules by creating one rule for each path from the root node to a leaf node.
3. Simplify each rule by removing any preconditions as long as the removal improves the accuracy of the rule.
4. Sort the pruned rules by their accuracy and apply the rules in this order when classifying instances.

In information retrieval research, we normally use precision and recall for query evaluation. Precision is the ratio of number of relevant documents to the number of returned documents and recall is the ratio of the number

of relevant documents returned to the number of relevant documents in existence.

In our algorithm we use the harmonic mean of precision and recall as the criterion in the precondition keyword removal instead of the error rate generally used, and do not sort the pruned rules because we make the disjunction of the preconditions as the keyword spices and the sort is meaningless here.

### Simplifying rules

Here we let  $\mathcal{D}_{recipe}$  be the set of relevant documents classified by humans and  $\mathcal{D}_{rule}$  be the set of documents that the rule identify as being relevant in the validation set. Precision  $P_{validation}$  and recall  $R_{validation}$  over  $\mathcal{D}_{validation}$  are defined as follows:

$$P_{validation} = \frac{|\mathcal{D}_{rule} \cap \mathcal{D}_{recipe}|}{|\mathcal{D}_{rule}|}$$

$$R_{validation} = \frac{|\mathcal{D}_{rule} \cap \mathcal{D}_{recipe}|}{|\mathcal{D}_{recipe}|}$$

The harmonic mean of  $P_{validation}$  and  $R_{validation}$  over  $\mathcal{D}_{validation}$  is given as

$$F_r = \frac{2}{\frac{1}{R_{validation}} + \frac{1}{P_{validation}}}$$

We then remove the precondition keyword from the rule so as to maximize the increase in harmonic mean  $F_r$ . We repeat this process until there is no precondition that can be removed without decreasing the harmonic mean  $F_r$ .

The error rate  $E$  we usually use as the criterion of removal of the preconditions is defined as follows:

$$E = \frac{|\mathcal{D}_{rule} \cap \overline{\mathcal{D}_{recipe}}|}{|\mathcal{D}_{rule}|}$$

If we remove the precondition keyword from the rule,  $|\mathcal{D}_{rule} \cap \mathcal{D}_{recipe}|$  and  $|\mathcal{D}_{rule} \cap \overline{\mathcal{D}_{recipe}}|$  always increase. Here we let  $\delta |\mathcal{D}_{rule} \cap \overline{\mathcal{D}_{recipe}}|$  be

the increase of  $|\mathcal{D}_{rule} \cap \overline{\mathcal{D}_{recipe}}|$  and  $\delta |\mathcal{D}_{rule} \cap \mathcal{D}_{recipe}|$  be the increase of  $|\mathcal{D}_{rule} \cap \mathcal{D}_{recipe}|$  by removing the precondition keyword.

We prune each rule by removing any preconditions that result in reduction in the error rate  $E$ , which decreases when

$$\frac{|\mathcal{D}_{rule} \cap \overline{\mathcal{D}_{recipe}}|}{|\mathcal{D}_{rule}|} \geq \frac{\delta |\mathcal{D}_{rule} \cap \overline{\mathcal{D}_{recipe}}|}{\delta |\mathcal{D}_{rule} \cap \overline{\mathcal{D}_{recipe}}| + \delta |\mathcal{D}_{rule} \cap \mathcal{D}_{recipe}|}$$

We can write this expression as follows:

$$\delta |\mathcal{D}_{rule} \cap \overline{\mathcal{D}_{recipe}}| \leq \frac{|\mathcal{D}_{rule} \cap \overline{\mathcal{D}_{recipe}}|}{|\mathcal{D}_{rule} \cap \mathcal{D}_{recipe}|} \times \delta |\mathcal{D}_{rule} \cap \mathcal{D}_{recipe}|$$

The value of the first term of the right side is usually very low at the initial rule. To decrease the error rate  $E$ , therefore, we can remove the keywords that lead  $\delta |\mathcal{D}_{rule} \cap \overline{\mathcal{D}_{recipe}}|$  is very small and  $\delta |\mathcal{D}_{rule} \cap \mathcal{D}_{recipe}|$  is very large and there are few keywords that satisfy the condition in the rule. Furthermore, if the initial error rate is 0, we cannot remove any keywords even though the rule contains only a few relevant documents.

Here we let  $f_r$  be the denominator of the harmonic mean  $F_r$  that we use as the criterion. Since  $|\mathcal{D}_{rule}| = |\mathcal{D}_{rule} \cap \mathcal{D}_{recipe}| + |\mathcal{D}_{rule} \cap \overline{\mathcal{D}_{recipe}}|$ ,  $f_r$  is

$$\begin{aligned} f_r &= \frac{1}{R_{validation}} + \frac{1}{P_{validation}} \\ &= \frac{|\mathcal{D}_{rule}| + |\mathcal{D}_{recipe}|}{|\mathcal{D}_{rule} \cap \mathcal{D}_{recipe}|} \\ &= 1 + \frac{|\mathcal{D}_{rule} \cap \overline{\mathcal{D}_{recipe}}|}{|\mathcal{D}_{rule} \cap \mathcal{D}_{recipe}|} + \frac{|\mathcal{D}_{recipe}|}{|\mathcal{D}_{rule} \cap \mathcal{D}_{recipe}|} \end{aligned}$$

We can remove the keywords that lead to the decrease of  $f_r$  which causes  $F_r$  to increase. The third term of  $f_r$  always decreases because of increase of the denominator of the term. The second term of  $f_r$  decreases when

$$\frac{|\mathcal{D}_{rule} \cap \overline{\mathcal{D}_{recipe}}|}{|\mathcal{D}_{rule} \cap \mathcal{D}_{recipe}|} \geq \frac{\delta |\mathcal{D}_{rule} \cap \overline{\mathcal{D}_{recipe}}|}{\delta |\mathcal{D}_{rule} \cap \mathcal{D}_{recipe}|}$$

We can transform this expression into following way

$$\delta |\mathcal{D}_{rule} \cap \overline{\mathcal{D}_{recipe}}| \leq \frac{|\mathcal{D}_{rule} \cap \overline{\mathcal{D}_{recipe}}|}{|\mathcal{D}_{rule} \cap \mathcal{D}_{recipe}|} \times \delta |\mathcal{D}_{rule} \cap \mathcal{D}_{recipe}|$$

This condition is same that of the error rate  $E$  and we can remove few precondition keywords that cause the decrease of the second term of  $f_r$  as mentioned above. The value of  $f_r$ , however, decreases as far as the increase of the second term does not surpass the decrease of the third term. That is to say the removal of the keyword that results in the some increase of the irrelevant documents can be accepted and as a result we can make the rule to contain great many relevant documents by using the harmonic mean as the criterion.

Because converting the tree to rules enable us to remove any precondition keywords in the rules equally, we can remove the keywords that have little effect on the quality of the rule by the algorithm. For example, the rule “NOT ingredients AND tablespoon AND NOT question” which appears in the decision tree above contains only 15 relevant documents. Removing “NOT question” from the rule make it contain 15 relevant documents and an irrelevant document.

Although the keyword “NOT question” has little effect on the rule, we cannot remove the keyword from the rule by the harmonic mean. We, however, can remove another keyword that cannot be removed when we prune the decision tree as it is. In this case, we can remove “NOT ingredients” by the harmonic mean and obtain the rule “tablespoon and NOT question” which contains 135 relevant documents and 6 irrelevant documents. This removal of the keyword changes the effect of the keyword “NOT question” on the rule, and now we can remove “NOT question” by the harmonic mean that leads to 15 relevant documents and an irrelevant document increase. In this case, we cannot remove any keywords by the error rate.

The removal of the precondition keywords from the rule by the harmonic mean may appear to cause some problems. If the initial rule contains only a few relevant documents, the algorithm makes the rule that contains the very large number of irrelevant documents. For example, a rule “NOT in-

redients AND NOT tablespoon AND NOT boil AND suitable” which also appears in the decision tree above, contains only one relevant document. After removing keywords, the rule becomes “NOT boil” and this contains 263 relevant documents and 704 irrelevant documents. However, we can remove the rule from the keyword spices by the algorithm for simplifying a disjunction described below.

### Simplifying keyword spices

We make Boolean expression  $h$  by making a disjunction of all preconditions of these rules (i.e. we make a disjunctive normal form of keywords). This is the initial form of keyword spices but it is still too large to be entered to a search engine. Besides, the disjunction contains many irrelevant documents because of the problem as mentioned above. We, then, try to remove conjunctive components from  $h$  which makes the keyword spices simpler and we also use the harmonic mean  $F_s$  as the criterion.

Here we let  $\mathcal{D}_{recipe}$  be the set of relevant documents classified by humans and  $\mathcal{D}_{spice}$  be the set of documents that the keyword spices identify as being relevant in the validation set. Precision  $P_{validation}$  and recall  $R_{validation}$  over  $\mathcal{D}_{validation}$  are defined as follows:

$$P_{validation} = \frac{|\mathcal{D}_{spice} \cap \mathcal{D}_{recipe}|}{|\mathcal{D}_{spice}|}$$

$$R_{validation} = \frac{|\mathcal{D}_{spice} \cap \mathcal{D}_{recipe}|}{|\mathcal{D}_{recipe}|}$$

We remove the conjunctive components from the disjunctive normal form so as to maximize the increase in harmonic mean  $F_s$ . We repeat this process until there is no conjunction that can be removed without decreasing the harmonic mean  $F_s$ ;  $h$  is returned as the keyword spice for this domain.

We let  $f_s$  be the denominator of the harmonic mean  $F_s$  and it is given by

$$f_s = 1 + \frac{|\mathcal{D}_{spice} \cap \overline{\mathcal{D}_{recipe}}| + |\mathcal{D}_{recipe}|}{|\mathcal{D}_{spice} \cap \mathcal{D}_{recipe}|}$$

When this value decreases, we can remove the conjunctive component. Because we remove the conjunctions from the disjunction,  $|\mathcal{D}_{spice} \cap \mathcal{D}_{recipe}|$  and  $|\mathcal{D}_{spice} \cap \overline{\mathcal{D}_{recipe}}|$  always decrease.

Here we let  $\delta |\mathcal{D}_{spice} \cap \overline{\mathcal{D}_{recipe}}|$  be the decrease of  $|\mathcal{D}_{spice} \cap \overline{\mathcal{D}_{recipe}}|$  and  $\delta |\mathcal{D}_{spice} \cap \mathcal{D}_{recipe}|$  be the decrease of  $|\mathcal{D}_{spice} \cap \mathcal{D}_{recipe}|$  by removing the conjunctive components. We can remove the conjunctions when

$$\delta |\mathcal{D}_{spice} \cap \overline{\mathcal{D}_{recipe}}| \geq \frac{|\mathcal{D}_{spice} \cap \overline{\mathcal{D}_{recipe}}| + |\mathcal{D}_{recipe}|}{|\mathcal{D}_{spice} \cap \mathcal{D}_{recipe}|} \times \delta |\mathcal{D}_{spice} \cap \mathcal{D}_{recipe}|$$

This condition indicates that we can remove the conjunctive components that cause the large reduction of the irrelevant documents with the slight reduction of the relevant ones. In the disjunction of the preconditions of rules simplified by our algorithm, some conjunctions cover many irrelevant documents. These components, however, can be removed by the harmonic mean, because the other rules cover the most of relevant documents and the removal of the defective conjunctions do not cause the large reduction of the relevant documents. Then we can obtain the simple keyword spices composed of a few conjunctions.

### Pruning results

We applied our algorithm to the decision tree in first trial and at first we extracted the following rules from the tree.

- NOT *takada* AND NOT almost AND NOT trend AND NOT speciality AND NOT *wa* AND NOT goods AND NOT body AND ingredients
- NOT question AND tablespoon AND NOT ingredients
- NOT front AND boil AND NOT tablespoon AND NOT ingredients
- suitable AND NOT boil AND NOT tablespoon AND NOT ingredients

- proper AND NOT suitable AND NOT boil AND NOT tablespoon AND NOT ingredients
- slice AND liking AND NOT proper AND NOT suitable AND NOT boil AND NOT tablespoon AND NOT ingredients
- dressing AND NOT slice AND liking AND NOT proper AND NOT suitable AND NOT boil AND NOT tablespoon AND NOT ingredients
- NOT batter AND taste AND enough AND NOT liking AND NOT proper AND NOT suitable AND NOT boil AND NOT tablespoon AND NOT ingredients
- drainer AND NOT enough AND NOT liking AND NOT proper AND NOT suitable AND NOT boil AND NOT tablespoon AND NOT ingredients
- *asatuki* AND NOT drainer AND NOT enough AND NOT liking AND NOT proper AND NOT suitable AND NOT boil AND NOT tablespoon AND NOT ingredients

We removed the precondition keywords that result in the maximum increase in harmonic mean. After that the rules became as follows:

- NOT speciality AND NOT goods AND ingredients
- tablespoon
- NOT front
- NOT boil
- NOT boil
- slice AND liking AND NOT ingredients
- NOT slice

- NOT batter
- drainer AND NOT liking AND NOT tablespoon
- *asatuki* AND NOT liking

Next, we made a disjunction of all preconditions of these rules and removed conjunctive components from the disjunctive normal form so as to maximize the increase in harmonic mean. As the result we obtained

1. (ingredients AND NOT speciality AND NOT goods) OR (tablespoon)

This keyword spice composed of only 4 keywords and can be accepted by a commercial search engine. We applied our algorithm to the decision trees in other trials and discovered the keyword spices as follows:

2. (ingredients AND NOT Tokyo) OR (tablespoon)
3. (ingredients AND NOT goods AND NOT result) OR (tablespoon)
4. (ingredients AND NOT outbreak AND NOT goods) OR (seasoning)
5. (ingredients AND NOT season AND NOT description) OR (tablespoon)

Although each decision tree was different, we obtained almost same keyword spices that composed of the almost same number of keywords and contained the same keywords. We, therefore, consider that our algorithm could extract the specific keywords in the cooking domain.

We examined the effect of simplifying with *test set*; 307 relevant web documents and 771 irrelevant web documents, which is independent on the training set and the validation set. Table 3.4 presents precision and recall over the test set by using the keyword spices from the initial tree and the ones after our pruning algorithm. This table indicates that simplifying by our algorithm improves the performance of precision and recall in almost cases and the simple keywords spices after the pruning algorithm achieve the high precision and recall over the test set. This shows that our algorithm can effectively convert the initial decision tree to the keywords spices.

Table 3.4: Pruning results

		Trials				
		1	2	3	4	5
Initial	Precision	0.892	0.890	0.899	0.889	0.916
	Recall	0.912	0.899	0.928	0.909	0.919
After Pruning	Keywords	4	3	4	4	4
	Precision	0.908	0.888	0.908	0.864	0.904
	Recall	0.935	0.932	0.935	0.912	0.922

### 3.3.4 Comparison with Other Pruning Methods

#### Statistical testing

In preceding researches the chi-square test[Quinlan, 1986b] and the Fisher exact test[Winston, 1992] of nonparametric statistical tests are used for pruning a decision tree. These are used in designs having two independent groups and determine whether differences in the samples constitute convincing evidence of a difference in the processes applied to them.

In our case, the scores from two independent samples: 1(when the document contains the keyword) and 0(otherwise), all fall into one or the other of two mutually exclusive classes: T(when the document is relevant) and F(otherwise) (Table 3.5). Hence, we can use both of the chi-square test and the Fisher exact test, and write null hypothesis as follows.

**Null hypothesis.**  $H_0$  there is no difference in the score of documents contain a keyword and otherwise.

If this null hypothesis is rejected by the tests, we can prune the node that we analyze the data.

The chi-square test requires that the expected frequencies in each cell should not be too small. When they are too small, the test may not be properly or meaningfully used. While the Fisher exact test becomes computationally very tedious, if the smallest cell value in the contingency table

Table 3.5: 2×2 contingency table

Variable	Group		Combined
	0	1	
T	$A$	$B$	$A + B$
F	$C$	$D$	$C + D$
Total	$A + C$	$B + D$	$N$

is even moderately large.

Thus, we use two statistical tests according to the number of samples and expected frequencies of the node that we want to test whether it can be pruned or not [Siegel, 1988]. To find the expected frequency  $A'$  in  $A$ , multiply the two marginal totals common to a particular cell and then divide this product by the total number of cases  $N$ . Thus,

$$A' = \frac{(A + B)(A + C)}{N}$$

The  $df$  (degrees of freedom) for an  $r \times c$  contingency table may be found by

$$df = (r - 1)(c - 1)$$

$r$  = number of classifications (rows)

$c$  = number of groups (columns)

Thus the  $df = (2 - 1) \times (2 - 1) = 1$ .

The decision concerning the use of the tests is guided by the following:

1. When  $N > 50$ , use the chi-square test.

We compute the following:

$$\chi_0^2 = \frac{N(AD - BC)^2}{(A + B)(C + D)(A + C)(B + D)} \quad df = 1$$

If the value of  $\chi_0^2$  is greater than the one of  $\chi_\alpha^2$  ( $\alpha$  is significance level), we can reject the null hypothesis of no difference between the groups at chosen significance level  $\alpha$ , and cannot prune the node that we analyze the data.

2. When  $N$  is between 40 and 50, use the chi-square test with a correction. When  $N$  is between 20 and 40, the test is also used if all expected frequencies are 5 or more.

We compute the expression below incorporated a correction for continuity that markedly improves the approximation of the sampling distribution, and apply the chi-square test using  $\chi_y^2$  to the data.

$$\chi_y^2 = \frac{N(|AD - BC| - \frac{N}{2})^2}{(A + B)(C + D)(A + C)(B + D)} \quad df = 1$$

3. When  $N \leq 20$ , always use the Fisher exact test. When  $N$  is between 20 and 40, the Fisher exact test may be used if the smallest expected frequency is less than 5.

We can compute the probability  $P$  of observing a particular set of frequencies in a  $2 \times 2$  table by the following:

$$P = \frac{(A + B)!(C + D)!(A + C)!(B + D)!}{N!A!B!C!D!}$$

If we wish to apply the Fisher exact test of the null hypothesis to the data, we must sum the probability of its occurrence with the probability of the more extreme possible outcome with the same marginal totals. For example, if the smallest cell value is 2, then three exact probabilities must be determined by using the expression above and summed; if the smallest cell is 3, then four exact probabilities must be found and summed, etc. If the obtained probability  $P$  is greater than the chosen significance level  $\alpha$ , we cannot reject the null hypothesis and we can prune the node.

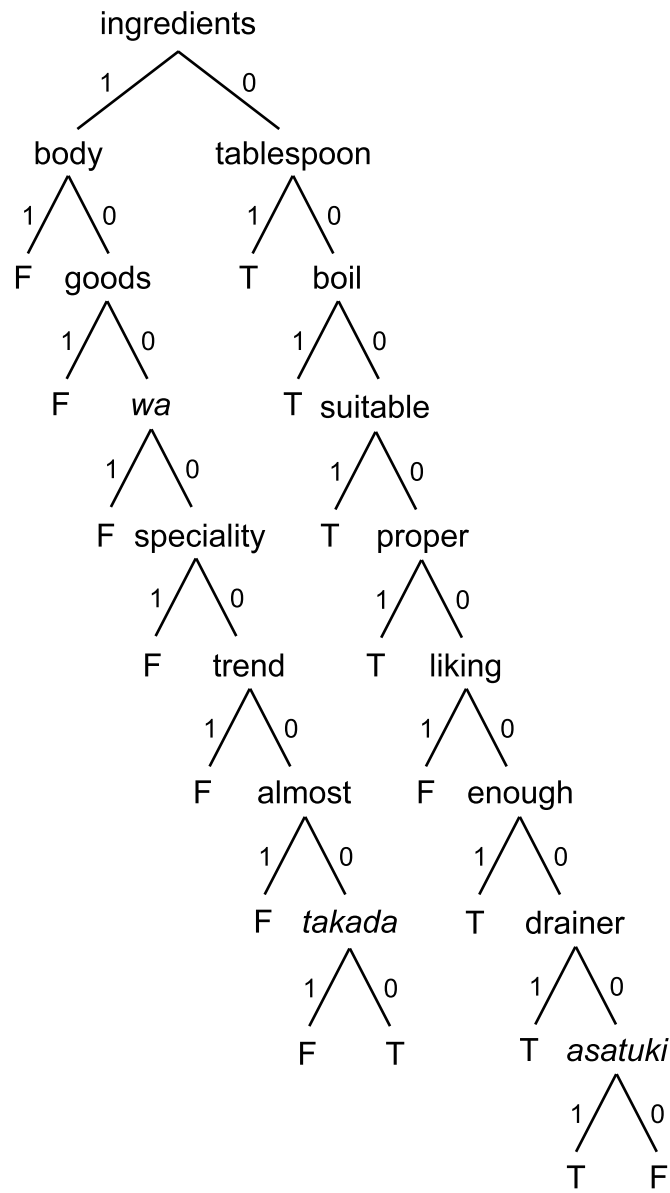


Figure 3.6: The decision tree pruned by statistical testing

Table 3.6: The number of nodes after statistical test pruning

	Trials				
	1	2	3	4	5
Initial	45	55	47	49	49
0.05	39	28	28	30	40
0.01	29	26	22	21	34

When we use the statistical testing, we do not use the validation set and can use all the web documents to make a decision tree and prune that. We, however, used only the training set to make a tree and prune that because of the comparison with our algorithm. We applied the statistical testing at the two significance level: 0.05 and 0.01 to the tree for pruning and the results are shown in Table 3.6.

The statistical test pruning resulted in the decrease of the number of nodes in the decision trees from more than 45 to less than 35. Figure 3.6 shows the decision tree after the statistical test pruning when  $\alpha = 0.01$  in the first trial, and the keyword spices extracted from the pruned tree as follows:

(ingredients AND NOT *takada* AND NOT almost AND NOT trend AND NOT speciality AND NOT *wa* AND NOT goods AND NOT body) OR (tablespoon AND NOT ingredients) OR (boil AND NOT tablespoon AND NOT ingredients) OR (suitable AND NOT boil AND NOT tablespoon AND NOT ingredients) OR (proper AND NOT suitable AND NOT boil AND NOT tablespoon AND NOT ingredients) OR (drainer AND NOT enough AND NOT liking AND NOT proper AND NOT suitable AND NOT boil AND NOT tablespoon AND NOT ingredients) OR (*asatuki* AND NOT drainer AND NOT enough AND NOT liking AND NOT proper AND NOT suitable AND NOT boil AND NOT tablespoon AND NOT ingredients)

Although the number of keywords is smaller than the initial keyword

spice, the keyword spice is still too large to be entered to a general-purpose search engine. In the statistical testing, additionally, if a cell value in  $2 \times 2$  contingency table is much larger than others, the null hypothesis is rejected and we cannot prune the node. In the initial decision trees made by the greedy search algorithm, the almost relevant or irrelevant documents fall into a certain deep leaf respectively. We, thus, cannot prune the many nodes and the decision trees remain still large.

### **Reduced-error pruning**

Reduced-error pruning[Quinlan, 1987] is a method that removes each node from the tree without converting it to rules. When a decision node in the tree is removed, the subtree rooted at the node is changed to a leaf node. The most common class of the examples under the subtree is assigned to the class of the new leaf node. The node whose removal most increases the accuracy of the decision tree over the validation set is choosed and removed. This process is repeated until there is no node that can be removed without decreasing the accuracy of the tree.

The effect of reduced-error pruning of the initial decision tree of trial 1 is illustrated in Figure 3.7. Through the pruning step, the number of nodes of the tree and the error rate over the validation set decrease. Although the error rate over the training set increases, the error rate over the test set, which is independent from the training set and validation set, decreases and we confirm that the tree is pruned effectively.

Figure 3.8 presents the decision tree after reduced error pruning. This decision tree has 11 nodes and is so simple compared to the initial decision tree that has 49 nodes. The keyword spices extracted from this tree is

- (ingredients AND NOT goods AND NOT body) OR (tablespoon AND NOT question AND NOT ingredients)

This keyword spice is so small as to be entered to a commercial search engine. Table 3.7 shows the result of reduced error pruning in all trials and the result of our algorithm to compare with.

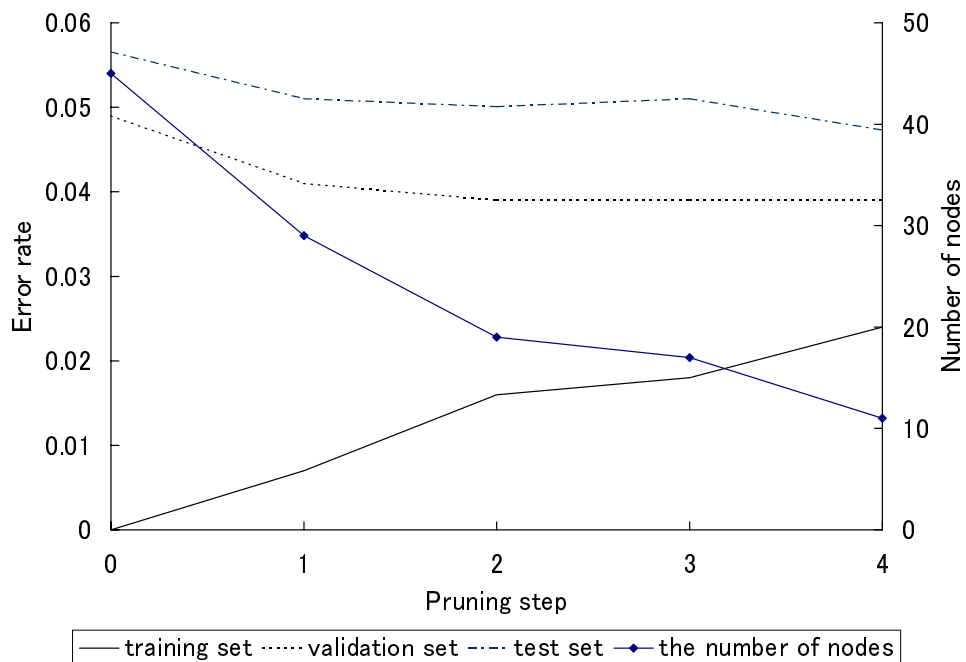


Figure 3.7: Effect of reduced-error pruning

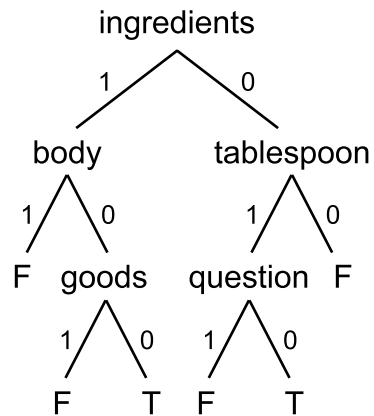


Figure 3.8: The decision tree pruned by reduced-error pruning

Table 3.7: Pruning results by reduced error pruning

		Trials				
		1	2	3	4	5
Reduced error pruning	Nodes	11	5	9	25	17
	Rules	2	2	2	7	6
	Keywords	6	3	5	34	20
	Precision	0.913	0.886	0.913	0.873	0.887
	Recall	0.922	0.935	0.928	0.919	0.945
Our algorithm	Keywords	4	3	4	4	4
	Precision	0.908	0.888	0.908	0.864	0.904
	Recall	0.935	0.932	0.935	0.912	0.922

The important point to note is the results of trial 4 and trial 5. The decision tree showed in Figure 3.8 is small and the keyword spices extracted from the tree is also simple, while the decision trees of trial 4 and 5 remain too large and the keyword spices are also too complex. The reason is that we cannot remove the nodes near the root of the tree that have more extensive effect on the tree than the nodes near the leaf. Table 3.7 indicates that there is no difference between the performance of the reduced error pruning and the one of our algorithm, and in our algorithm the problem above does not occur because of converting to rules before pruning. This indicates that our algorithm is more effective than the reduced error pruning algorithm to discover the keyword spices.

### 3.3.5 Keyword Spice Extraction Algorithm

The following summarizes our algorithm for extracting keyword spices.

0. Generate input keywords according to some estimate of distribution  $p(k)$  and collect web pages that contain keyword  $k$  and classify them into positive and negative examples by hand.
1. Split the examples into two disjoint subsets, the training set,

$\mathcal{D}_{training}$ , for generating the initial decision tree and the validation set,  $\mathcal{D}_{validation}$ , for pruning.

2. Make the initial decision tree from  $\mathcal{D}_{training}$  using an information gain measure without any pruning technique.
3. Convert the learned tree into a set of positive rules by creating one rule for each path from the root node to each leaf node: this classifies positive examples.
4. **For each** rule  $r$  **do**

**Repeat**

- Remove the precondition keyword that results in the maximum increase in the harmonic mean

$$F_r = \frac{2}{\frac{1}{R_{validation}} + \frac{1}{P_{validation}}}$$

of precision measure  $P_{validation} = |\mathcal{D}_{rule} \cap \mathcal{D}_{recipe}| / |\mathcal{D}_{rule}|$  and recall measure  $R_{validation} = |\mathcal{D}_{rule} \cap \mathcal{D}_{recipe}| / |\mathcal{D}_{recipe}|$  where  $\mathcal{D}_{recipe}$  is the set of relevant documents classified by human and  $\mathcal{D}_{rule}$  is the set of documents which  $r$  classify relevant in  $\mathcal{D}_{validation}$ .

**Until** until there is no keyword that can be removed without decreasing the harmonic mean  $F_r$ .

**End**

5. Make a disjunctive normal form of Boolean expression  $h$  by making a disjunction of all preconditions of the positive rules.
6. **Repeat**
  - Remove the conjunctive component from the disjunctive normal form  $h$  that results in the maximum increase in the harmonic mean

$$F_s = \frac{2}{\frac{1}{R_{validation}} + \frac{1}{P_{validation}}}$$

of precision measure  $P_{validation} = |\mathcal{D}_{spice} \cap \mathcal{D}_{recipe}| / |\mathcal{D}_{spice}|$   
and recall measure  $R_{validation} = |\mathcal{D}_{spice} \cap \mathcal{D}_{recipe}| / |\mathcal{D}_{recipe}|$   
where  $\mathcal{D}_{recipe}$  is the set of relevant documents classified by  
human and  $\mathcal{D}_{spice}$  is the set of documents which  $h$  classify  
relevant in  $\mathcal{D}_{validation}$ .

**Until** until there is no conjunction that can be removed without de-  
creasing the harmonic mean  $F_s$ .

**Return**  $h$

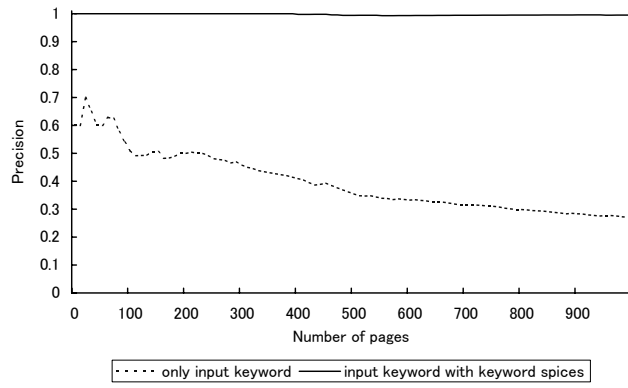
## 3.4 Experiments

### 3.4.1 Experimental Settings

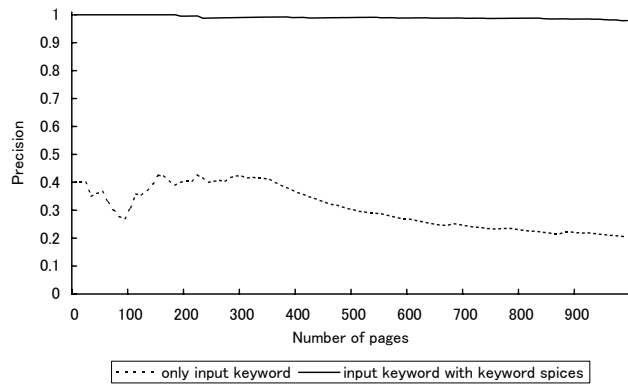
To evaluate the performance of our keyword spice method, we conducted realistic tests in the cooking domain with external commercial search engines. For the experiment we chose the keywords of (*butaniku* (pork) and *horenso* (spinach) and *ebi* (shrimp)) which were not used to generate the keyword spices and used the major Japanese web search engine “Goo” and evaluated the keyword spices “(ingredients AND NOT speciality AND NOT goods) OR (tablespoon)” discovered in the first trial. Then we forwarded the queries containing only keywords and the queries with the keyword spices to “Goo” and compared the results.

### 3.4.2 Precision

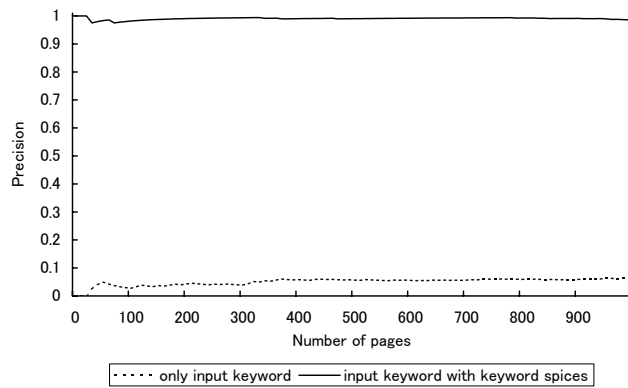
Figure 3.9 compares the precision values for the queries containing only keywords and the queries with keyword spices for the three input keywords. We checked up to the top 1000 pages as ranked by the search engine Goo. In general, as the number of pages viewed increases, the precision with query-only input decreases or stays low, while the precision of queries with keyword spices stays high. Table 3.8 presents the average precision of the top 1000 returned results. Precision is higher than 97% for all queries.



(a) query "pork" forwarded to goo



(b) query "spinach" forwarded to goo



(c) query "shrimp" forwarded to goo

Figure 3.9: Precision of queries forwarded to Goo

Table 3.8: Average precision over the index of Goo

Query	The input query	The query with with keyword spices
pork	0.271	0.995
spinach	0.205	0.979
shrimp	0.063	0.986

Table 3.9: Estimated recall of the queries with keyword spices over the index of goo

Query	$Reldoc_{index}$	$Reldoc_{spice}$	Estimated recall
pork	10728	10084	0.940
spinach	4744	4126	0.870
shrimp	5868	5728	0.976

### 3.4.3 Estimated Recall

It is easy to achieve high precision if we do not address recall, but keeping both high is rather difficult. The recall of a query is much harder to calculate than the precision because  $\mathcal{D}_t$ , the set of all relevant documents in the web, is unknown. We estimated  $\mathcal{D}_t$  from the results returned from a general-purpose search engine. Most search engines show the total number of documents that matched the query. We can calculate the estimated number of relevant documents in the search engine's index ( $Reldoc_{index}$ ) by using the average precision of the query for the top 1000 returned documents.

$$\begin{aligned}
 Reldoc_{index} &\simeq \\
 &(\text{The number of documents matched with the input query}) \\
 &\times (\text{Average precision of the input query})
 \end{aligned}$$

The number of relevant documents found with the spice-extended query

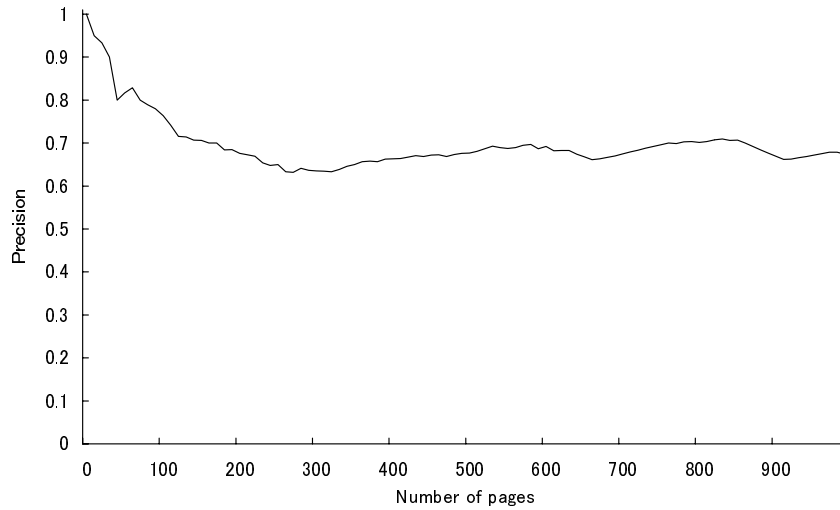


Figure 3.10: Precision of the query “pork AND salt”forwarded to Goo

can be calculated in the same way.

$$R_{\text{el}doc_{\text{spice}}} \simeq$$

(The number of documents matched with the query  
with keyword spices )

×(Average precision of the query with keyword spices)

It is reasonable to use  $R_{\text{el}doc_{\text{index}}}$  because we have no consistent way of finding web pages that are not linked to any general-purpose search engine. We estimate the recall of a spice-extended query as follows

$$R \simeq \frac{R_{\text{el}doc_{\text{spice}}}}{R_{\text{el}doc_{\text{index}}}}$$

Table 3.9 shows the estimated recall values of different spice-extended queries over the index of Goo.

The high value of recall (higher than 86%) indicates that our method filters out only non-relevant documents and lose little information in the search process.

These excellent performance of precision and recall show that the keyword spice method is effective method for building the domain-specific web search engines.

To compare these results with the example in Introduction, Figure 3.10 shows the results of submitting the query “pork AND salt” to Goo. The average precision and estimated recall for the top 1000 returned documents are 0.674 and 0.871, respectively. This shows that the disjunction above discovered by our algorithm yields a great improvement in search performance.

### **3.5 Application: A Cooking Recipe Search Engine**

In this section we introduce a domain-specific web search engine we built using keyword spices. As described in preceding sections, we attempted to build a cooking recipe search engine as an example. Figure 3.11 shows the cooking recipe search engine we have built. In this system, we add the keyword spice “(ingredients AND NOT speciality AND NOT goods) OR (tablespoon)” to the user’s query and forward to a general-purpose search engine “Goo”.

This system searches the web documents as follows:

1. Accept a user’s request in natural language.  
(ex. “I want to eat the pork dish.”)
2. Extract keywords (noun) by morphemic analysis.  
(ex. “pork” and “dish”)
3. Remove the unnecessary keywords and leave only ingredient keywords.  
(ex. “pork”)

### Query in Natural Language

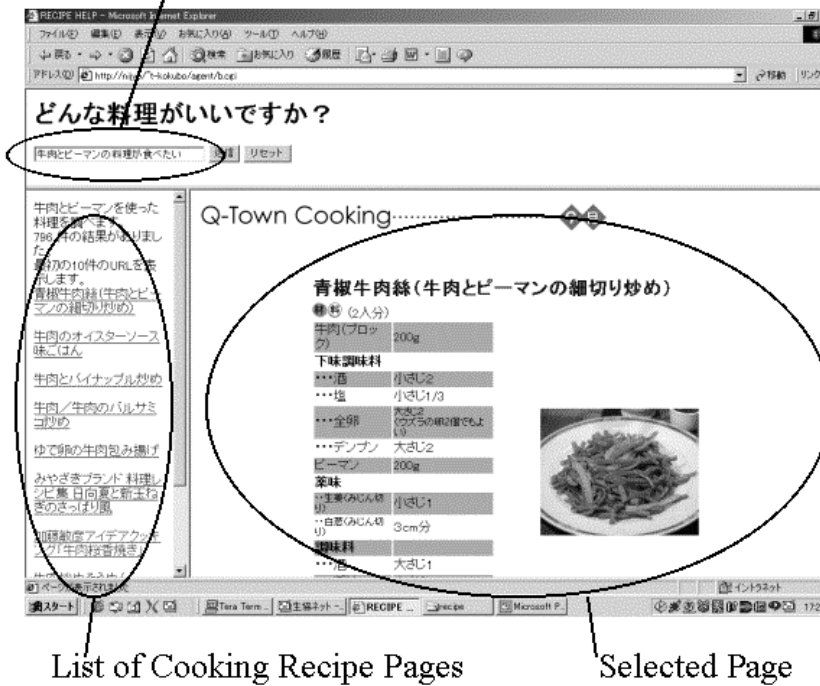


Figure 3.11: A cooking recipe search engine

4. Add the keyword spices to the remaining keywords.
5. Forward the query to Goo.
6. Display the list of results returned from Goo in the left part of the web browser and display the top ranked page in the right part of it.

Almost of the returned web pages from this system is relevant, even if we input a message that contains keywords other than ones we used in the evaluation. We consider that this system provides one of the most superior level of service for searching web documents in the cooking domain.

## 3.6 Summary

We have proposed a novel method for domain specific web searches that is based on the idea of keyword spices; Boolean expression that are added to the user's input query to improve the search performance of commercial search engines. This method allows us to build domain-specific search engines without any domain heuristics. We described a practical learning algorithm to extract powerful but comprehensive keyword spices. This algorithm turns complicated initial decision trees to small Boolean expressions that can be accepted by search engines. Our experiments with an external general-purpose search engine yielded good results. For three different keywords in the field of cooking, precision was higher than 97%. High estimated recall (higher than 86%) over the search engine's index was also confirmed.

We built a search engine in the cooking domain as an example but we are also developing search service for other domains such as restaurant pages and personal homepages.

In this chapter, we used input keywords selected by humans to make training examples. To be more comprehensive we need some criteria with which input keywords can be selected. It is sufficient to make examples based on the distribution of user's input query  $p(k)$ . We are planning to open our recipe search system to the public through the web and we will obtain the value of  $p(k)$  afterwards. We will study how the input keywords used to form the training examples affect the performance of the system.



# Chapter 4

## Applying Association Rules to Information Navigation

### 4.1 Introduction

In the previous chapter, we have dealt with the problem of how to build a search engine that only returns web pages that are relevant to a certain domain. We have placed the emphasis on returning high quality results for user's initial query. (This type of search is called "ad-hoc search" in the literature of information retrieval.) However, even when the problem of selecting only domain pages from the web is settled, the problem of finding the pages that most satisfy a user's request among collected domain pages still remains. In this chapter, we describe technologies for solving this problem.

Usually the user cannot obtain the most relevant page after the first search even if he uses a highly effective domain-specific web search engine. Thus, the user has to examine the search results and modify the query so as to obtain pages that are more relevant.

Therefore, through interaction with the user, the system supports the user to refine the query incrementally for improving search results. In the field of information retrieval, many researches on interactive search have been carried out. In *relevance feedback*[Salton and Buckley, 1990] the system presents the results for the initial query and the user judges each doc-

ument relevant or irrelevant. The system then modifies the query based on the user's feedback. By repeating this process, the query is incrementally refined so as to obtain more relevant results. However, the query is modified as a whole and it cannot handle each condition in the query separately. In addition the system appears to be a "black box" to the user and the user cannot comprehend the process of query modifications. In the web many domain-specific web search systems provide sophisticated search functions. For example, the information search system that can handle the combination of keyword search, hyperlinks, and geographical relations of pages, has been developed[Hiramatsu and Ishida, 2001], and a user can make a query like "Chinese restaurants opening on Sunday within 200 meters from Ginkakuji temple." It is difficult for a user to form and modify these complex queries appropriately using traditional information retrieval systems like relevance feedback systems.

Therefore, we need an agent interface that supports a user to formulate a complex search query interactively. As the number of internet users with cellular phones grows, supporting this demand will become important because it is tedious work for a user to specify many search conditions through a tiny screen of a mobile terminal. We propose an interactive method that presents an initial search result to the user and gradually navigates him/her to better information. The system has to navigate a user to appropriate search conditions in a short time. To do this the agent can use the dependence of search conditions in a certain domain. Each search condition that a user can specify is not usually independent each other. For example, in the domain of finding restaurants, we can possibly hold a party in a restaurant that have a hall. The agent can utilize the dependence like this for navigating the user to better information.

In this chapter, we introduce a method for using association rules as domain-specific navigation heuristics. Association rules are popular in the data-mining research field[Agrawal and Srikant, 1994], and they also have been applied to a general-purpose web search engine[Kawano and Hasegawa, 1998]. We consider that association rules are more effective in domain specific web search, where the mutual dependence among keywords

appears more peculiar.

Using association rules has following advantages.

- Association rules extracted from the domain documents reveal the hidden relationship between keywords in the domain, which is usually cannot hit on by the user. They can compensate the weakness of the user's ability in formulating queries.
- Association rules are based on the occurrence of keywords in the domain. Therefore, for an effective navigation, the system can estimate the amount of matched pages with the query before the modification.
- Rules in the explicit form are comprehensible to the user. Therefore, they are suitable for the interactive search where the system explains the effects of query refinements to the user.

The quality of navigation heuristics is a key to the success of the search. Therefore, in this chapter, we describe pruning algorithms for extracting high quality association rules.

The remainder of this chapter is organized as follows: Section 4.2 presents definition of association rules in the context of information retrieval and describes how to apply association rules to information navigation. We describe algorithms for pruning association rules in Section 4.3. Section 4.4 evaluates our method and related researches are given in Section 4.5.

## **4.2 Information Navigation using Association Rules**

### **4.2.1 Association Rules**

Mining association rules is a well-researched topic in the literature of data mining. The following is a formal definition of an association rule [Agrawal and Srikant, 1994]. Let  $\mathcal{I} = \{i_1, i_2, \dots, i_m\}$  be a set of items. Let  $\mathcal{D} = \{t_1, t_2, \dots, t_n\}$  be a set of transactions, where each transaction  $t_j$  is a set

of items such that  $t_j \subseteq \mathcal{I}$ . We say that a transaction  $t_j$  contains a set of some items  $X$  if  $X \subseteq t_j$ . An association rule is an implication of the form  $X \implies Y$ , where  $X \subset \mathcal{I}, Y \subset \mathcal{I}, X \cap Y = \emptyset$ . Here we define support of itemset  $X$  as a ratio of transactions containing  $X$  to  $\mathcal{D}$ . The rule  $X \implies Y$  is evaluated by its confidence and support defined as follows:

$$\text{confidence}(X \implies Y) = \frac{\text{support}(X \cup Y)}{\text{support}(X)}$$

and

$$\text{support}(X \implies Y) = \frac{\text{support}(X \cup Y)}{|\mathcal{D}|}$$

We can apply the above definition of association rule to information retrieval if we regard Web pages and words as transactions and items respectively[Kawano and Hasegawa, 1998].

Fig. 4.1 presents the case when the most of the pages containing a keyword set  $X$  also contain another keyword set  $Y$ . In this case, the association rule  $X \implies Y$  holds.

Let  $X = \{\text{“pork”}, \text{“soup”}\}, Y = \{\text{“noodle”}\}$  and the confidence and the support has following values:

$$\begin{aligned} \text{confidence}(X \implies Y) &= 0.83 \\ \text{support}(X \implies Y) &= 0.1 \end{aligned}$$

This means that if web pages contain keywords “pork” and “soup”, 83% of these pages contain keyword “noodle”. and pages that contain these all three word “pork”, “soup” and “noodle” account for 10% of all pages. Our proposal is to use these association rules to information navigation by software agents. In the following subsections, we describe a method that applies association rules between keywords to information navigation.

## 4.2.2 Using Association Rules in Interactive Web Search

We are aiming for a interaction between a user and an agent as follows:

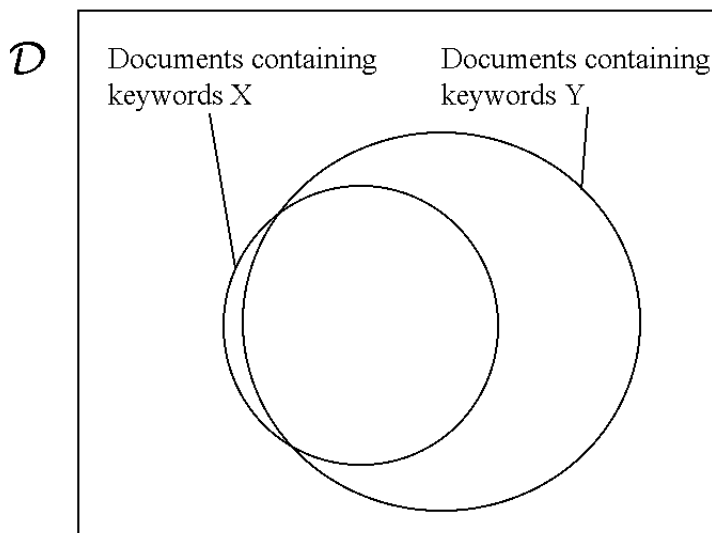


Figure 4.1: Keyword Associations

User: I'm looking for a restaurant for a banquet.  
 Agent: How about sukiyaki for a banquet?  
 User: Sounds good. Tell me one.  
 Agent: How about Daimonji-tei.  
 User: Well... show me another one.  
 Agent: How about Ginkakuji-tei. This has sukiyaki in menu too and I think you can hold a banquet.  
 User: Any other restaurants?  
 Agent: Shirakawa-ya is an Japanese restaurant and has a hall. I think you can hold a banquet in this restaurant.  
 User: Can we have hot-pot dishes in this restaurant?  
 Agent: Yes.  
 User: Please give me the details.  
 Agent: This restaurant is near Hyakumanben crossing...

To realize the above interaction, the agent is required to have domain-specific knowledge like "If a restaurant has a hall, a banquet can be held

in the restaurant.” We present a method that extracts association rules as domain-specific knowledge from the Web. The agent uses this knowledge to interact with the user.

### 4.2.3 Search Aspects in Domain-Specific Web Search

The objects that are targets of an information search have several attributes considered during the search processes. For example, when we search some restaurants, we consider location, menu, atmosphere, service etc. We call these attributes *search aspects*.

If we treat information retrieval as a problem of satisfying search aspects, users’ search target can be expressed as pairs of aspects and their values. Let us consider the situation when we search for some place for a banquet in Gion area. This is interpreted as a problem of finding some Web pages that satisfy the constraint [location]=”Gion” and [situation]=”banquet”.

Let  $\mathcal{A}$  be a set of aspects in some application domain, and  $D(A)$  be a set of possible values that are assigned to each aspect  $A \in \mathcal{A}$ . User’s information needs are given as a set of assignments of values to aspects:

$$A_i = x_i \quad (A_i \in \mathcal{A}, x_i \in D(A_i))$$

The agent tries to satisfy as much aspects through dialogue with the user using association rules.

For example, let us consider the situation where a query {”sukiyaki”, ”banquet”, ”Gion”} is given. ”sukiyaki”, ”banquet” and ”Gion” are belong to the aspects [menu], [situation] and [location] respectively. Here we assume several association rules described in Fig. 4.2 hold.

After the first search, following three cases can be occur,

1. There are moderate number of pages that satisfy the three aspects.
2. There are no pages that satisfy the three aspects.
3. There are too many number of pages that satisfy the three aspects.

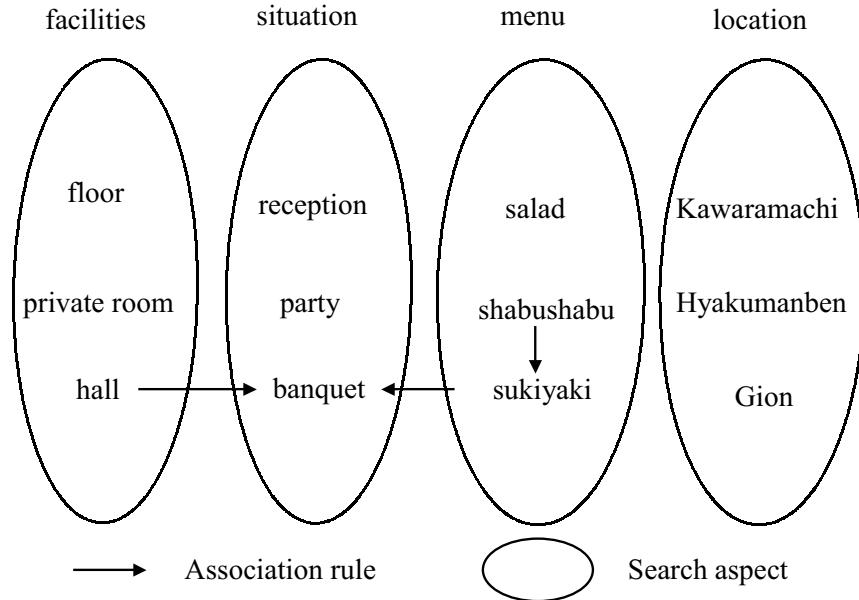


Figure 4.2: Applying association rules to a query

In the first case the agent just shows one of the pages to the user. In the second case the agent relaxes the search condition and finds some results using association rules. The agent explains to the user how he has obtained these results. In the last case the agent asks the user to take into account another search aspect to reduce search results. The agent finds candidates of additional search aspects using association rules.

In the following subsections, we will describe the behavior of agents in the last two cases.

#### 4.2.4 Association Rules to Satisfy Search Aspects

Web pages are not well formatted and even if the pages do not contain any keyword of some search aspect, it does not necessarily mean that the page

is irrelevant. For example, restaurants whose web pages do not contain keyword “banquet” may still be suitable for a banquet. If we find some keyword like “hall” we can assume it is possible to hold a banquet in that restaurant. In short, we can satisfy the aspect [situation] by using an association rule “hall”  $\implies$  “banquet”.

The agent search association rules in the following order:

1. Association rules whose precondition and postcondition keywords already appeared in the query.
2. Association rules from a keyword in some aspect that does not yet appeared in the query to the keyword that already appeared in the query.
3. Association rules from another keyword in the same aspect that is already appeared in the query.

Let  $A_1 = x_1 \wedge, \dots, \wedge A_n = x_n$  be a present query. In the first case we apply a rule  $x_j \implies x_i (1 \leq i \leq n, 1 \leq j \leq n)$  and remove a condition  $A_i = x_i$  and retry the search. When the initial query is composed of three aspect [situation]=”banquet”  $\wedge$  [menu]=”sukiyaki”  $\wedge$  [location]=”Gion” as described in Fig. 4.2 and association rule “sukiyaki”  $\implies$  ”banquet” holds, we can rewrite the query to [menu]=”sukiyaki”  $\wedge$  [location]=”Gion”. The agent will explain to the user “There is a restaurant called Higashiyama-tei. I think you can hold a banquet because this restaurant has sukiyaki in its menu”.

In the second case the agent uses the association rule  $x \implies x_i$  where  $x$  is a keyword belong to a new aspect. The agent removes the condition  $A_i = x_i$  from the query and adds a new condition  $A_{n+1} = x$ . For example if the input query is [situation]=”banquet”  $\wedge$  [location]=”Gion”, the agent use association rule “hall”  $\implies$  ”banquet” and search using new query [facility]=”hall”  $\wedge$  [location]=”Gion”. The explanation of agent will be like “There is a restaurant called Gion-ya. You can hold a banquet because it has a hall”.

In the last case the agent uses an association rule  $x \implies x_i$  where  $x \in D(A_i)$  and modify the condition to this aspect to  $A_i = x$ . Given same initial search query as in the first case, the agent applies association rule “shabushabu” $\implies$  “sukiyaki” and change the keyword in [menu]. The agent explains to users : “There is Yasaka-tei at Gion. You can hold a banquet here. There is shabushabu in the menu. So I think you can eat sukiyaki too.”

#### 4.2.5 Association Rules to Propose New Search Aspects

When the result by initial query is too large, the agent has to add new search condition and reduce the number of search results. By adding a new search aspect to the query, the search result can be reduced. The agent use association rules to propose a new search aspect to be considered. Let  $A_1 = x_1, \dots, A_n = x_n$  be the present query, the agent applies an association rule  $x \implies x_i$ , where  $x$  is a keyword belongs to an aspect that has not appeared in the query, and proposes to add a new search aspect

$$A_{n+1} = x$$

to the query. For example, when there are too many results returned by the query [situation]=“banquet” and [menu]=“sukiyaki” and the rule “hall” $\implies$  “banquet” holds, the agent can propose additional query condition [facility]=“hall”. In this case the agent will ask a user “Is it nice if there is a hall in the restaurant?”

### 4.3 Pruning Association Rules

The quality of association rules is a key to the success of information navigation. When we extract association rules from web documents, there are many rules that seem to be meaningless. So we propose following methods to extract association rules that are useful to information navigation.

### 4.3.1 Keyword Clustering for Pruning Association Rules

In web pages there are many words that are not related to meaning of web pages because the web pages has navigational parts (for example, header or footer of web pages). These words are co-occur with high frequency and we can remove these words by finding cluster of keywords in the graph of association rules between keywords. Strongly connected components (SCCs) of digraph  $G$  are a set of maximal subgraph  $G' = (V', E')$  where every pair of nodes  $u, v \in V'$  has at least one directed path over  $E'$ [Harary *et al.*, 1965].

Our method of pruning association rules using keyword clustering is described as follows(here  $c$  is the minimum cluster size and  $\alpha(\simeq 1)$  is the minimum confidence in the cluster):

1. From the set of association rules pick out the rules whose confidence is higher than  $\alpha$  and construct a digraph of keywords.
2. From this digraph extract SCCs whose degree is larger than  $c$  as keyword clusters.
3. Let  $K_{cluster}$  be a set of keywords used in these clusters. From the original rule set remove the rules whose precondition or postcondition contain keywords appeared in  $K_{cluster}$  (as described in Fig. 4.3).

### 4.3.2 Statistical Tests for Pruning Association Rules

In this subsection we remove rules that are not statistically significant. Let us consider the case where 30% of the whole documents  $\mathcal{D}$  contain keyword  $x$  and 100% of the documents contain keyword  $y$ . In this case the association rule  $x \implies y$  holds with confidence 1.0, but we cannot conclude any relation in the occurrence of keyword  $x$  and  $y$ . In other words, we cannot use the rule in information navigation when the following relationship holds:

$$\text{confidence}(x \implies y) \simeq \text{support}(y)$$

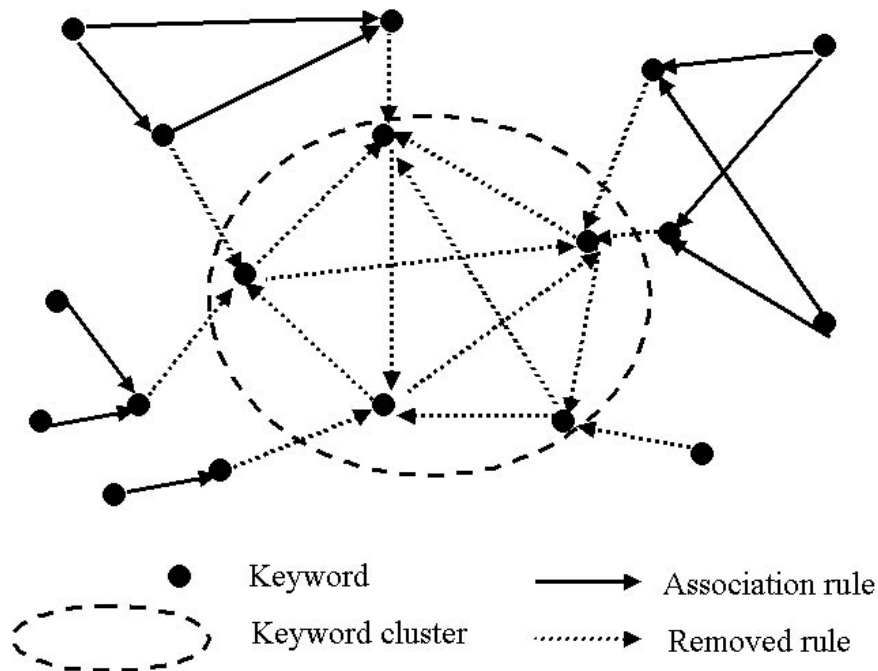


Figure 4.3: Pruning association rules by keyword clustering

To validate the significance of rules we use  $\chi^2$  test in statistics to the occurrence of keyword  $x$  and  $y$  in  $\mathcal{D}$ . This is a same idea of  $\chi^2$  pruning in decision tree learning [Quinlan, 1986a]. In [Srikant and Agrawal, 1995]  $\chi^2$  test was applied to association rules in POS database and they cannot confirm effect of this method. In this chapter, however, we can confirm the effect of  $\chi^2$  method in web documents.

Let us consider the case where the population is categorized to  $A_1, \dots, A_m$  and  $B_1, \dots, B_n$  by two attributes  $A, B$  respectively. In a sample of size  $N$  the observed frequency of each category is  $a_1, \dots, a_m$  and  $b_1, \dots, b_n$ . When the observed frequency of the class of “ $A_i$  and  $B_j$ ” is

Table 4.1: Keyword co-occurrence with no significance

	Containing “menu”	Not containing “menu”
Containing “time”	488	273
Not containing “time”	178	75

$x_{ij}(i = 1, \dots, m; j = 1, \dots, n)$  (where  $\sum_{j=1}^n x_{ij} = a_i, \sum_{i=1}^m x_{ij} = b_j$ ),

$$\chi_0^2 = \sum_{i=1}^m \sum_{j=1}^n \frac{(x_{ij} - a_i b_j / N)^2}{a_i b_j / N} \quad (4.1)$$

obeys  $\chi^2$  distribution with degree of freedom  $(m - 1)(n - 1)$ .

Here we consider the example of Table 4.1. In this case the association rule with confidence (“time”  $\implies$  “menu”) = 0.64 holds. To confirm the significance of this rule we set up a null hypothesis “In the document set  $\mathcal{D}$  the occurrence of keyword “menu” and “time” is independent.”

By Table 4.1 and equation (4.1) we obtain the value  $\chi_0^2 = 0.84$ . Here the degree of freedom is  $(2 - 1)(2 - 1) = 1$ . If we let the significant level 0.05,  $\chi_{0.05}^2 = 3.84 > 0.84$  and we cannot reject the null hypothesis. In short we cannot say there is any significance in co-occurrence of keywords “time” and “menu”. We remove these rules to increase the effect of information navigation.

## 4.4 Experiments

### 4.4.1 Experimental Settings

Our aim is to develop an information agent that supports people’s daily lives. Thus we take the domain of finding restaurants in Kyoto as an example. We collect about 1000 web pages of restaurants in Kyoto by hand and pick out nouns as keywords using morphological analysis. We add 120 domain-specific words to the dictionary of the morphological analysis system.

We obtained following dataset:

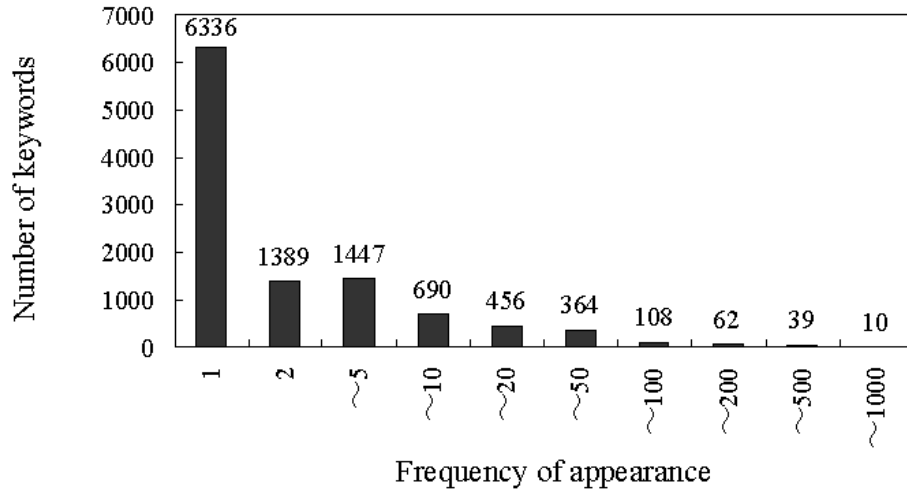


Figure 4.4: Distribution of keyword appearance

**Number of html documents:** 1014

**Number of keywords:** 10901

**Occurrence of the keywords in the documents:** 76243

**Average number of keywords in each document:** 76

In Fig. 4.4 we present distribution of keyword occurrence. It shows following characteristics:

- More than half of all keywords occur in only one document.
- A few keywords occur very frequently in the documents.

We extract 30397 association rules with 50% minimum confidence and 0.5% minimum support. For example the keyword derived from the precondition keyword “Italy” is presented below\*.

---

\*Original keywords are Japanese

“Kyoto” “dish” “time” “opening” “menu”  
“store” “holiday” “reservation” “address” “station”  
“pasta” “budget” “inquiry” “information” “phone”  
“average” “question” “gourmet” “Shiga”  
“copyright” “report”

#### 4.4.2 Pruning Association Rules

We can reduce the number of association rules to 30% (9179) using the clustering method. In the case of keyword “Italy” following rules remained:

“Italy”  $\implies$  “dish”

“Italy”  $\implies$  “reservation”

“Italy”  $\implies$  “pasta”

We applied statistical method with significant level 0.01 to the above results and obtain 3651 rules (12% of initial rules).

In the research applied statistical test to association rules in POS databases [Srikant and Agrawal, 1995] they reported that only 1% of rules can be removed. In the domain of information retrieval there are a few high frequent words described in Fig. 4.4. These word co-occurred with many other keywords and this results in many association rules that have no relation in meaning. We think the method of statistical test is effective because of the above reason.

In the example of “Italy” we obtain following final results.

“Italy”  $\implies$  “dish”

“Italy”  $\implies$  “pasta”

We can find only rules that seemed to be meaningful.

Table 4.2: Search aspects

Aspect	Number of keywords	Example of keywords
1.menu	284	“sukiyaki” “salad” “wine” “oyster”
2.style	27	“full course” “kaiseki” “lunch box”
3.situation	24	“party” “dinner” “banquet” “reception”
4.atmosphere	38	“cheerfully” “casual” “traditional” “elegance”
5.facility	30	“private room” “hall” “terrace” “Japanese room”
6.country	22	“Europe” “Italy” “China” “Japan”
7.user	12	“group” “lady” “couple” “family”

### 4.4.3 Effectiveness of Association Rules in Information Navigation

In our future work we are planning to extract keywords that related to each search aspect automatically from web documents. In this chapter, however, we selected keywords with high occurrence and categorized to each aspect by hand as described in Table 4.2 .

In Table 4.3 we present the number of extracted association rules for each aspect. Rows and columns of the table represent precondition and postcondition of rules respectively. We can find a large dispersion in the number of extracted rules among aspects. For the aspect [atmosphere] or [user] we can extract seldom rules.

As an example we present association rules from [facility] to [situation].

“hall”  $\implies$  “banquet”  
 “terrace”  $\implies$  “cool”  
 “hotel”  $\implies$  “banquet”  
 “terrace”  $\implies$  “dinner”

Table 4.3: Number of extracted rules for each aspect

	1	2	3	4	5	6	7
1.menu	158	61	17	2	10	7	0
2.style	16	16	0	1	2	0	0
3.situation	3	9	20	3	5	0	4
4.atmosphere	1	2	0	1	0	2	0
5.facility	9	1	5	1	1	0	0
6.country	6	1	0	0	0	3	0
7.user	0	0	0	0	0	0	2

“Japanese room”  $\implies$  “banquet”

To evaluate the effectiveness of these rules, we have to consider the distribution of input query by users and it can be estimated by conducting real-world experiments. However, we can obtain following inspection from above examples.

- We can successfully extract association rules that can be understood by humans.
- The number of extracted rules is small. For example there are 24 keywords in the precondition [situation] but only 3 rules can be found.

We think the sparseness of the data causes the small number of extracted rules. By regarding synonyms as a same word using external thesaurus[Srikant and Agrawal, 1995] we can solve this problem.

In the following we pick out typical rules and consider the merits and demerits when the agent uses these rules.

**1. “spaghetti”  $\implies$  “pasta”([menu]  $\implies$  [menu])**

This rule represents the concept hierarchy of words. Given a keyword which has higher concept the agent can find documents that contains a lower level

of concept hierarchy using this rule. This rule has an effect like keyword expansion using thesaurus.

**2. “Italy” $\implies$ “Italian” ([country] $\implies$ [country])**

This shows a relation of synonyms. This rule also can be used as a thesaurus.

**3. “shabushabu” $\implies$ “sukiyaki”([menu] $\implies$ [menu])**

These two keywords are not synonyms but have related meaning. The agent can use this rule to satisfy the search aspect but it may also result in failure of navigation.

**4. “croquette” $\implies$ “salad” ([menu] $\implies$ [menu])**

“salad” is a keyword that occurred in many pages. So the rule that seems to be meaningless is extracted. In the information navigation, however, this rule will not be used many times because the keyword in the postcondition has large frequency and we will have little chance to apply these rules.

**5. “hall” $\implies$ “banquet” ([facility] $\implies$ [situation])**

These two keywords belong to different aspects. This type of rule cannot be derived from a thesaurus and can be discovered only by analysing web pages. The agent uses this rule to satisfy search aspects or propose a new aspect in information navigation.

**6. “terrace” $\implies$ “dinner” ([facility] $\implies$ [situation])**

This is similar to case No. 5 above. However, when the agent uses this rule to find an additional aspect, it will specialize the query too much and the possibility of failure in information navigation is high. Thus these types of rules have to be used after the confirmation of a user during the dialogue.

We regard the rules like 1,2,3 and 5 in the above examples as effective rules and present the ratio of effective rules in Table 4.4. The ratio of

Table 4.4: Rate of effective rules

	1	2	3	4	5	6	7
1.menu	0.33	0.28	0.59	0.00	0.30	1.00	-
2.style	0.00	0.88	-	0.00	0.00	-	-
3.situation	0.00	0.33	0.70	0.33	0.80	-	0.25
4.atmosphere	0.00	0.50	-	1.00	-	0.50	-
5.facility	0.00	1.00	0.80	0.00	0.00	-	-
6.country	1.00	1.00	-	-	-	0.33	-
7.user	-	-	-	-	-	-	1.00

effective rules is dispersed among aspects. The agent has to consider this value and select rules that seemed to have a high possibility of success for effective information navigation.

## 4.5 Related Work

In the research of information retrieval, many methods called query expansion are proposed[Baeza-Yates and Ribeiro-Neto, 1999]. These method are automatically add some keywords to the query. Additional keywords are selected using external thesaurus or automatically developed thesaurus from the document set[Xu and Croft, 1996][de Campos *et al.*, 1998].

Mondou[Kawano and Hasegawa, 1998] is a search engine that uses association rules for proposing additional keywords to users. The user selects keywords from the list and adds them to the query for narrow down the search topic.

Several domain-specific information systems that interactively support users to obtain better information have been developed. Automated Travel Assistant[Linden *et al.*, 1997] is a system that proposes a initial result and refine the result thorough the interaction with the user. This work also regards user's information needs as pairs of attributes and their values. Their work treated air flight databases.

ExpertClerk[Shimazu, 2001] is an agent that supports shoppers' buying process in a Web shop using natural language conversation. They surveyed human sales manuals for human salesclerks and implemented their conversation techniques into the agent. The agent asks questions of users for narrowing down the matched results based on the effectiveness of questions measured by their entropy. It also shows three contrasting examples to the user and explain their characteristics.

These two systems are based on relational databases, but their methods for information navigation will be applicable to web searches.

## 4.6 Summary

This chapter presented a new interface to domain-specific web search.

Our information navigation agent presents an initial search result to the user and gradually navigates him/her to better information.

We described following two navigation methods using association rules of the keywords:

- Automatic modification of the query to satisfy search aspects
- Proposing a new search aspect to be considered

The key to the success of the navigation is the quality of rules. Thus we have developed two algorithms for selecting effective association rules as follows:

- Removing rules based on strongly connected components of the keyword graph
- $\chi^2$  test for the significance of keyword co-occurrence in the rules



# Chapter 5

## Cooperative Information Agents for Supporting Daily Life

### 5.1 Introduction

In this chapter, we explore the idea of constructing cooperative information agents and applying them to digital cities, which are the platform for supporting people's daily life. We present Digital City Kyoto[Ishida, 2001]\*, in which we are heavily participating, as the first trial of agent application. First, we survey various architectures for information agents on the Internet. We also introduce various digital cities including our Digital City Kyoto. We then propose a new architecture for cooperative information agents.

The crucial characteristics of the architecture are as follows:

1. We define two types of agents. The *front-end agents* determine and refine users' uncertain goals. The *back-end agents* extract and organize useful information from incomplete information sources on the WWW. Both types of agents opportunistically cooperate through a blackboard.
2. To realize a natural and intellectual appearance, we incorporate the methodology of social psychology with the technologies of animation

---

\*<http://www.digitalcity.gr.jp/>

and 3D graphics in designing the front-end agents.

3. The back-end agents use the planning technology in AI, especially technology of real-time planning with incomplete information to satisfy users' requirement communicating with users.

Front-end agents act *humanly* and back-end agents act *rationally*. [Russell and Norvig, 1995] Front-end agents are gentle and kind to various users in digital cities who may be not good at using computers. Back-end agents behave smartly and efficiently to deal with the hard problems that occur in digital cities. The cooperation of these two types of agents is a key feature for user-friendly and useful information systems.

## **5.2 Cooperative Information Agents on the Internet**

### **5.2.1 Architectures for Information Agents on the Internet**

Some research has regarded information gathering in terms of a cooperative task of humans and agents, not just as a search engine task.

Letizia [Lieberman, 1995] is an earliest realization of this type of agent. It discovers the user's interest from his past behavior. The agent fetches the pages linked from the current page and recommend pages that match with the user's known interests. Letizia increases the opportunity of encountering interesting pages even when the user is searching for other information. The idea that a person's interest is persistent underlies this research. Letizia runs in the user's computer and learns the user's interest. WebWatcher [Joachims *et al.*, 1997], on the other, is executed on the server side and recommends links based on the behaviors of other users who visited the site in the past. WebWatcher is like a guide working in the museum and giving tours to visitors. At the beginning of a tour, a user is asked to enter a phrase representing

his interest. When the user follows some hyperlink, his phrase is added to the description of that link. The description of each link is represented as a feature vector that is used to guide information retrieval. As phrases in the system accumulate, WebWatcher hones its ability to recommend appropriate links to visitors.

The above agents have a rather simple behavior: suggesting relevant links to users. Other types of information agents can make plans to gather relevant information. Information integration is an interdisciplinary research field influenced by artificial intelligence and database systems. Information integration systems usually consist of *wrappers*, *domain models*, and *mediators*. Wrappers extract information from the WWW using machine learning technology.[Perkowits *et al.*, 1997; Muslea *et al.*, 1999] Domain models are written in knowledge representation languages and specify relationship among information sources in the Internet.[Knoblock *et al.*, 1998; Levy *et al.*, 1996] After wrappers and domain models are constructed, a task of answering a user's query is treated as a planning problem by a mediator.

Planning for information gathering differs from classical planning problems. In many cases, goals and actions are intended to get information, not to change the world state. The plans must handle information that is unknown at the time of planning but available during execution. Recent planners can handle these issues.[Etzioni and Weld, 1994] Planning for information gathering on the Internet must be flexible enough to cope with unexpected errors. Sage[Knoblock, 1995] is a planner that tightly integrates planning and execution. Sage can interleave planning and executing. The agent monitors execution of actions and can recover from errors. It can redraw plans given new goals without interrupting current actions.

For developing practical information agents, it is necessary to compose many kinds of functional blocks or to link many agents. Examples of these types of agent architectures are given below.

BIG(resource Bounded Information Gathering)[Lesser *et al.*, 1998] integrates different kinds of problem solvers, such as a top-down scheduler and an opportunistic planner. An opportunistic planner[Hayes-Roth, 1985] processes the most promising hypothesis at a given time considering newly ac-

quired information. The planner receives action schedules from the scheduler and performs information gathering and processing tasks. Note that the planner can also request the scheduler to remake plans considering new information.

Sycara and Zeng[Sycara and Zeng, 1996] presented a multi-agent architecture consisting of: (1) *Interface agents*, which interact with users to acquire information about the task and present the results to them; (2) *Task agents*, which receive tasks from interface agents and make plans for the goals and perform actions; and (3) *Information agents*, which retrieve information from information sources or other agents to meet the demands of task agents.

Specifying reusable architectures is an efficient way to develop information agents that consist of many components, Decker *et al.*[K. Decker and Williamson, 1997] presented reusable architectural building blocks that supported the specification of agents' behaviors. A set of agent behaviors, such as responding to queries, monitoring information sources, advertising their services, and replicating themselves, is specified. Kuwabara *et al.*[Kuwabara *et al.*, 1995] introduced an inheritance mechanism for multi-agent coordination protocols the used the protocol description language called Agentalk. It enables designers of agents to define new interaction protocols incrementally by extending existing protocols.

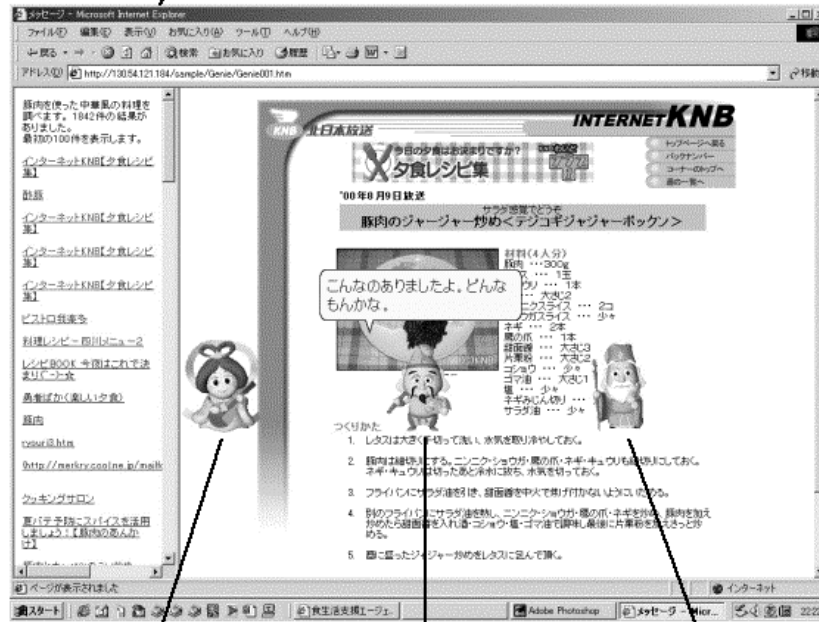
Our proposal of cooperative information agents for digital cities is based on the ideas leaned from the above research.

## **5.2.2 Multi-agent system: Venus & Mars**

Venus & Mars(Virtual Environment for Novice User Support & Multi-Agent Recommendation System)[Yamada *et al.*, 2000] is a multi-agent system that aims to support novice user to search information in the domain of dietary life from the WWW. Figure 5.1 shows a screenshot of the system.

There are 3 agents and each agent operates independently of the other agents' program. One of the agents catches up a user's request through the dialog in natural language and the other agents search information from

## Results of Cooking Recipe Search



Personal Agent   Recipe Search Agent   Healthcare Agent

Figure 5.1: Venus & Mars (provided by Teruhiro Yamada)

their respective information source such as search engines and the WWW databases. Two of the agents use search engines as the information source and they can search specific information by adding the keyword spices of the domain they want to search to the user's query.

### 5.3 Digital Cities

In this section we introduce the concept of digital cities and the activities in Digital City Kyoto.

The Internet is used not only for research and business but also for ev-

eryday urban life. While the Internet makes research and business global, life remains inherently local[Ishida *et al.*, 1999]. The proposed concept of digital cities was to build an arena in which people in regional communities could interact and share knowledge, experience, and mutual interests[Ishida, 2001].

Various digital cities have been created around the world. In the US, America Online (AOL) has developed a series of digital cities.<sup>†</sup> Each AOL digital city collects the tourist and shopping information of the corresponding city. The success of AOL digital cities shows that people need regional information services for their everyday life.

In Europe, the European Digital Cities Conference has been held annually since 1994 to discuss a wide variety of topics including “the role of cities, towns, and regions in the deployment of advanced telematics solutions within the development of the Information Society.” As an example of the experiments performed, Digital City Amsterdam[van den Besselaar and Beckers, 1998] was started in 1994. This city was built as a platform for various community networking systems and thus strongly focuses on social interaction among digital citizens. We were able to see how social interaction increased in the digital city. Though Digital City Amsterdam succeeded in introducing the city metaphor to regional information services, there is no direct mapping between digital and physical Amsterdam.

The Helsinki Arena 2000 Project[Linturi *et al.*, 2000]<sup>‡</sup> started in 1996, under the initiative of the Helsinki telephone company (HPY). The goal of the project was building the next generation metropolitan network. In parallel to the development of high-speed networks, a trial of building a 3D virtual city is underway. The virtual Helsinki is the face of this project, and the entire city of Helsinki is currently under construction in virtual space. This virtual city will be a human interface to new broadband network services.

In October 1998, the Digital City Kyoto Project started to develop a so-

---

<sup>†</sup><http://www.digitalcity.com/>

<sup>‡</sup><http://www.hel.fi/infocities/>

cial information infrastructure for Kyoto. Digital City Kyoto presents different city metaphors, a 2D map[Hiramatsu *et al.*, 2000] and a 3D virtual space, which are easily understood by non-technical people. WEB information is collected and linked to the 2D/3D city. Real-time sensory data from the physical city is also mapped to the digital city. People can get information related to the physical city such as traffic, weather, parking, shopping, and sightseeing. Currently the development of cooperative information agents is planned as the next stage of the project.

Digital City Kyoto also encourages social interaction among residents and tourists. Even if we build a beautiful 3D space, if there no one visits the city, the city will not be very attractive. Similarly, even if we have a virtual town where people often visit to chat, if there is no connection to the corresponding physical city, this town cannot be an information infrastructure for the city.

The design policies set up for Digital City Kyoto are summarized as follows[Ishida *et al.*, 1999]. The first design policy was to make it *real* by establishing a strong connection to physical Kyoto. Our digital city is not an imaginary city that exists only in cyberspace. Instead, our digital city complements the corresponding physical city, and provides an information center for everyday life for actual urban communities. We think “digital” and “physical” make things “real.” For example, in computerized organizations, such as universities and advanced companies, we cannot figure out their activities without accessing their networks (E-mail, WEB and so on). As in those organizations, digital activities will become an essential part of urban life in the near future. The second design policy was to make the digital city *live* by dynamically integrating WEB archives and real-time sensory information created in the city. We will not produce contents nor select them. We will provide a tool for viewing and reorganizing vivid regional information created by the people of the city.

Ishida propose the system architecture of digital cities as follows[Ishida *et al.*, 1999]. Fig. 5.2 shows the three layer model for designing digital cities. The first layer is called the *information layer*; in it WWW archives and real-time sensory data are integrated and reorganized using the city

metaphor. A geographical database is used to integrate those types of information.

The second layer is called the *interface layer*; in it 2D maps[Hiramatsu *et al.*, 2000] and 3D virtual spaces provide an intuitive view of digital cities. The animation of moving objects such as avatars, cars, busses, trains, and helicopters demonstrate dynamic activities in the cities. If the animation reflects real activities of the corresponding physical city, each moving object can become a media for social interaction: you may want to click on the object to communicate with it.

The third layer is called the *interaction layer*; it allows residents and tourists to interact with each other. Community computing experiments[Ishida, 1998b; Ishida, 1998a] especially agent/multiagent technologies are applied to encourage interactions in digital cities. Isbister[Isbister, 2000] developed an tour guide agent as a prototype of social agents for digital cities. By providing local information and stories, we hope the tour guide will encourage dialogue and relationships among those participating, both during and after the tour, and will encourage visitors to reach out to and communicate with the local users of the Digital City Kyoto.

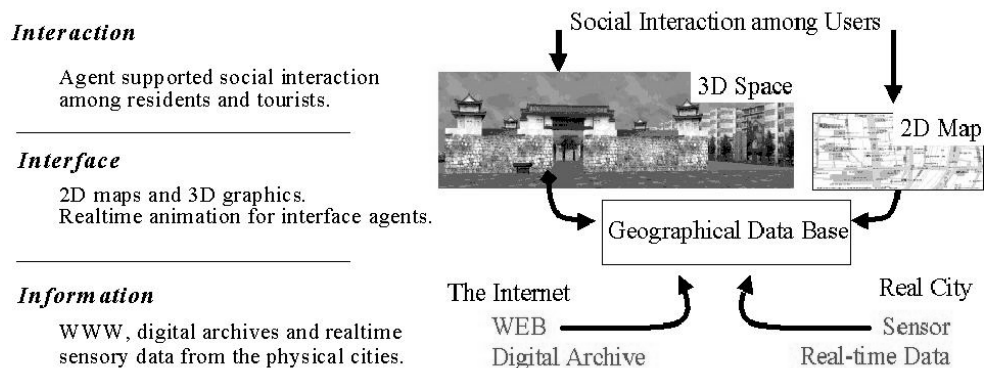


Figure 5.2: The Three Layers Model for Digital Cities

## 5.4 Cooperative Information Agents for Digital Cities

### 5.4.1 Architecture for Cooperative Information Agents

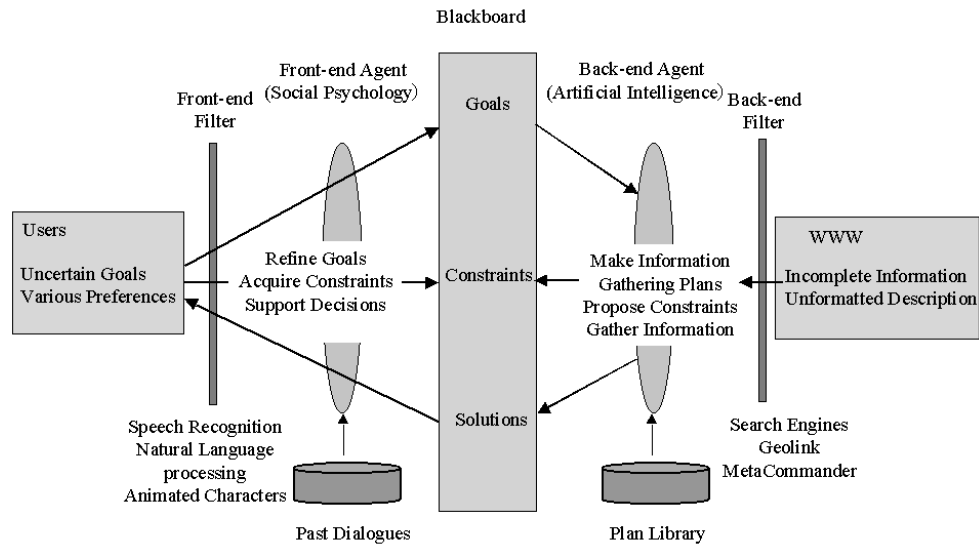


Figure 5.3: Architecture for Cooperative Information Agents for Digital Cities

The architecture for using cooperative information agents in digital cities is shown in Fig. 5.3. The agents' environment is uncertain and incomplete in terms of users' goals and information sources in WWW. Agents are required to satisfy the users' obscure goals using unorganized information from the WWW.

There are four main technical challenges. First, we need to develop dialogue interfaces. Developing general-purpose natural language interfaces and voice interfaces are hard technical problems in their own right. Second, the agents must support the decision making process of users. Users do not always give clear goals to the agents. The agents have to discover the users'

goals and support them in making their goals clearer. Third, we have to develop opportunistic problem solving systems. To accomplish users uncertain goals using Internet information, the agents will show related information to users and watch their reactions. Flexible real-time planning [Ishida, 1997] is indispensable to achieving such interactive behavior. Last, there is a problem of how to filter the vast and unorganized information available on the WWW.

To develop cooperative information agents for digital cities, we have to overcome all these problems, but we have to remember that digital cities evolve continuously through interaction with users. We cannot spend many years in the laboratory apart from users. We have to incrementally develop the agents while providing services to users. The agent architecture is required to support such incremental development. As is shown in Fig. 5.3, we propose an architecture for cooperative information agents that consists of four components: *front-end filter*, *front-end agent*, *back-end agent*, and *back-end filter*. The basic concepts of this architecture are:

1. We envisage different agents for different purposes instead of trying to make an all-round agent. For example, we may develop a shopping agent and make a guide agent. This specialization reduces the difficulty of handling the uncertainty and incompleteness of the agent's environment.
2. The system has two types of agents, front-end agents responsible for interaction with users and back-end agents responsible for information gathering, and integrates them using a blackboard. The front-end agents support users' decision making while back-end agents make plans for information gathering. Both types of agents maintain past cases and plans in databases, and refine users' uncertain goals by presenting this information to users.
3. The front-end and back-end filters are introduced to clarify the agent's uncertain environment. The front-end filter utilizes speech recognition and natural language processing tools. It also uses animated

characters like Microsoft's Peedy Agent or Extempo's<sup>§</sup>. By specifying the environment and constraining the content of the dialogue, it compensates the agent's imperfect natural language understanding and converges on the user's goals. The back-end filter organizes the information on the Internet, utilizing tools like Keyword Spice [Oyama *et al.*, 2001], GeoLink [Hiramatsu and Ishida, 2001] or MetaCommander [Kitamura *et al.*, 1996]. GeoLink matches URLs to geographic longitude and latitude, and integrates and re-organizes information on the WWW. MetaCommander is a script-based WWW tool to collect, arrange, store, and dispatch WWW data automatically.

Thus, the cooperative information agents for the digital cities employ a multi-agent architecture consisting of various specialized front-end and back-end agents. One front-end agent is assigned to each user, and usually runs on the user's terminal. The back-end agents are developed for the services provided in the digital city and executed on the server side. Communication between these agents is implemented using a blackboard system.

## 5.4.2 Front-end Agents

Let us consider the following example.

Today is a fine day in April. John has finished his work and he is free in the afternoon. He visits the Digital City web site. He asks his front-end agent in the Digital City, "Could you recommend something interesting?" The agent recommends a jazz session at a club because the agent knows he likes music. But the weather is so beautiful he tells the agent, "But I want to go out right now!" The agent knows the cherry blossoms are beautiful right now and suggests going to see cherry blossoms. John is interested and says, "That sounds good." The agent knows

---

<sup>§</sup><http://www.extempo.com/>

many places to visit. In order to list several good places near him, the agent asks John where he is now.

To interact with users, the front-end agent recognizes the user's goals, refines the goals, and supports the user's decision making. The functions of a front-end agent are listed below.

**Determining the user's uncertain goals:** in many cases, the user does not know exactly what he/she really wants and so cannot give enough information about the goal to the agent. The front-end agent must discern the user's goals using his/her past behaviors or other environmental situations like the season or ongoing events.

**Refining user's goals:** after a rough outline of the user's goals is obtained, the agent starts to refine them through conversation with the user. For example, the user may have decided what to do, but does not know where, when, and how to do it. Considering partial solutions proposed by the back-end agent, the front-end agent constrains the user's goals. If necessary, the agent can ask the user to supply the missing information.

**Supporting decisions:** The front-agent agent presents solutions to the user based on his/her preferences. The agent lists alternatives and describes their advantages and disadvantages. If a conflict is discovered in the user's demands, the agent helps him/her to resolve it.

In refining the user's goal or supporting his/her decisions, the front-end agent uses social psychology to guide the user comfortably without imposing unnecessary mental load on the user.

A front-end agent writes the user's goals on a blackboard. Because of the uncertainty in the user's goals, the agent writes several hypotheses. These goals are used by the backend agents to make plans. Other information provided by back-end agents, such as information about occasional events, weather, and traffic may also be utilized.

The communication abilities of the front-end agents are augmented by the front-end filter, which supports speech recognition and natural language conversation. A front-end agent identifies an individual user and records their conversation. This information is accumulated and used by the front-end agent to discern the user's goal and preferences.

### **5.4.3 Back-end Agents**

Let us consider the following example.

Today is Saturday. John has a dinner appointment with Mary. However, he first needs to finish writing a paper at his university and mail it from a post office. He asks an agent on his mobile phone, "Please find the nearest post office that is open today?" While a back-end agent is searching, John calls Mary. She says she cannot wait because she is hungry and will meet him at the restaurant. John now asks the agent, "Please provide a restaurant guide. Are there any good Italian restaurants within one kilometer of the post office? I would prefer one with a large wine selection."

The user's goals as indicated in the above questions are clear, and the back-end agents can satisfy them. There are many Web pages in Digital City Kyoto. These pages are linked together by hyperlinks. A GIS stores urban building information with geographical attributes. If we can use not only the hyperlinks in a Web space but also the links representing geographical relations, we can put inquiry services, such as the one in the above scenario, into practice.

To connect back-end agents to the digital city, we use a program interface to a GIS. Hiramatsu and Ishida proposed an augmented Web space that is obtained by dynamically extending a Web space with generic links, which represent geographical relations in a physical city [Hiramatsu and Ishida, 2001]. Though various kinds of links can be defined as a generic link, they focus on generic links based on geographical information. This extension

is triggered by users' queries, which contain geographical evaluation functions. The generic links are created dynamically and therefore do not explicitly appear on Web pages. This system called GeoLink[Hiramatsu and Ishida, 2001] works as a back-end filter mentioned above. It provides a query language extended SQL with a notion for path expressions including generic links. The back-end agents use this language as a program interface to the Digital City.

The main functions of a back-end agent in the digital city include:

**Utilizing multiple information sources:** The back-end agents in a digital city gather comprehensive information about the city through the use of GeoLink as well as other domain-specific web search engines and directory services. They maintain metalevel knowledge about information sources, such as accessibility, quality of information, or structure of information, and use this knowledge for efficiently accessing the information sources.

**Automatic acquisition of the knowledge for information gathering:** A back-end agent automatically acquires knowledge of the domain assigned to the agent and generates information gathering operators that combine various information sources. For example, a back-end agent analyzes associations of keyword frequency and hierarchical concept trees, and generates information-gathering operators that combine keyword-based search engines with directory services.

**Flexible real-time planning:** The back-end agents make plans for information gathering using the acquired operators. In the WWW, services and contents are frequently changed and updated. The network environment always changes and services that were available a few minutes ago suddenly become unavailable. Thus, the agent monitors the execution of the information-gathering plan and remakes the plan if it fails.

**Proposing additional constraints for user's goals:** In the case of searching for information on the Internet, users seldom get information rel-

evant to their requirements from the first queries. They are presented with either too much irrelevant information or no results at all. Since the back-end agent utilizes domain knowledge for information gathering, it can propose additional search conditions to increase the possibility of finding relevant information.

In the research to date, information agents are presumed to know the user's goals clearly. This requires the user to interact through domain-specific and fixed interfaces. In a digital city, however, there are many kinds of services and various types of users who may dislike direct manipulation interfaces. The back-end agents cooperate with front-end agents in making the user's goals clear. Thus, back-end agents must offer an opportunistic planner. The agents recognize the goals on the blackboard written by the front-end agents, gather information relevant to them, and write the information on the blackboard.

Cooperative information agents have, so far, been designed to provide services to individual users. Other than discerning the user's goals and refining them, our front-end agent can act as a social agent in supporting interaction among several users. The social agent can support group decision making when users in a group have to decide the destination of their group tour.

## **5.5 Summary**

We have discussed the various roles and characteristics of information agents in digital cities. We have proposed cooperative information agents for digital cities. To support the user, the agents estimate the user's goals, refine them where necessary, gather relevant information available on the Internet, present solutions to the user, and support his/her decision making. We adopt a blackboard architecture to realize cooperation among these agents. To reduce uncertainty in the agents' environment, we utilize many kinds of front-end and back-end filters such as animated characters and domain-specific web search engines.



# Chapter 6

## Conclusion

### 6.1 Contributions

In this thesis, we have discussed roles and virtues of domain-specific web search systems and described various technologies to build such systems. The major contribution is summarized into the following respects.

- A new method for building domain-specific web search engines using the idea of “keyword spice”

In this thesis we described a new method that improves search performance by adding the domain-specific keywords, called *keyword spices*, to the user’s input query; the modified query is then forwarded to a general-purpose search engine. Existing domain-specific search engines adopt one of the following architectures, building specialized indices that point to only domain pages or filtering the results of general-purpose search engines to display only domain documents to the user. The former requires web-crawlers to collect domain documents and local databases to store indices to pages, and consumes network bandwidth. The latter needs human effort to write domain-specific rules to filter documents. Attempts to build these rules automatically by machine learning run into a barrier of collecting training examples because randomly sampled pages include few domain

pages. The keyword spice method enables us to build domain-specific search engines without requiring specific facilities and burden networks. This method also solves the problem in preparing training examples. In keyword spice method we need consider only those web pages that contain the user's input query keywords; not all web pages. Thus, we can limit the scope of sampling to the pages that contain input keywords, which results in increasing the ratio of positive examples.

- A machine learning algorithm for extracting keyword spices

We describe a machine learning algorithm, which is a type of decision-tree learning algorithm that can extract keyword spices as a disjunctive normal form of keywords from web documents. Using decision-tree learning, the keyword spices can be effectively discovered and we can apply the algorithm to other domains with ease. Decision trees, however, usually grow very large and the keyword spices from the trees are too complex to be accepted by commercial search engines. Therefore, we have to simplify the inducted rules. We adopted the algorithm based on rule post-pruning, converting the decision tree to rules before pruning. In the algorithm, we use the harmonic mean of precision and recall as the criterion to remove the keywords and rules and that well keeps the balance of precision and recall.

- Development of a cooking recipe search engine

We attempted to build a search engine in the cooking domain as a real application and we discovered keyword spices composed of only 4 keywords by our algorithm. To demonstrate the value of the keyword spices, we conduct experiments with a commercial search engine "Goo" and the results showed the high value of precision. Although it is difficult to calculate the recall of a query in the WWW, we contrived the new method to estimate the recall from the results returned by a general-purpose search engine and confirmed the high

value of recall.

- A new method for information navigation using association rules

We proposed a method for supporting the user to find relevant information from the collection of web documents in a certain domain. User interface of existing search engines, which display long list of results to the users, is designated for users in front of desktop computers, but is not suitable for mobile users. Therefore, we propose a method in which the system presents an initial search result to the user and gradually navigates him/her to better information. For effective navigation we invent navigation methods that can treat each attribute of search target independently using association rules of keywords.

- Algorithms for pruning association rules

The key to the success of the navigation is the quality of rules. Association rules extracted from web pages by use of existing rule discovery algorithm include many rules that are not useful to information retrieval. Thus, we propose two methods for selecting effective rules. The first is eliminating association rules that contain keywords derived from the formats of web pages. Strongly connected components of the graph derived from association rules between keywords are calculated and the keywords in the component larger than specified size are regarded as candidates for removal. The second is removing the association rules that only mean accidental co-occurrence of keywords. We applied  $\chi^2$  test in statistics to examine statistical significance of association rules. We applied these methods to the restaurant domain and confirmed their effects.

- A multi-agent architecture for supporting information access using domain-specific web search systems.

We propose an architecture for developing information agent on the infrastructures that support people's daily life like digital cities. We

do not construct the agent from scratch but combine various domain-specific information agents. The front-end agents that handle interaction with users and the back-end agents that gather information from the web cooperate to support users. The front-end agents use the front-end filters, such as natural language interface and animated characters, to clarify user's uncertain goals. The back-end agents use the back-end filters, such as domain-specific search engines, to reduce the heterogeneity of web information.

## 6.2 Future Directions

We conclude this thesis with the list of possible future research issues of domain-specific web search systems.

- Using pre-organized information in the web.

In the keyword spice method, it is necessary to prepare training examples classified by human and it is a barrier to building search engines for various domains. We can use web pages already classified into existing web directories as training examples in order to reduce the cost of building domain-specific search engines.

- Using information sources out of the web

In this research, we tried to extract rules for search using the information that can be obtained from the web. However, it is reasonable idea to use existing external information sources for building domain-specific search system at low cost. For example, several thesauruses are provided electronically these days. Using this enable us to extract effective rules from the smaller size of training examples.

- Using hyperlinks between web pages

In this research, we do not use hyperlink, which is a major characteristic of the web. Currently there are no major search engines that

can use a hyperlink between pages in a query, but some prototypical systems that can accept path regular expressions to specify link between pages have been developed[Konopnicki and Shmueli, 1995; Mendelzon *et al.*, 1997]. In this research, we assume the topic is described in a single page. This is true for the domain of cooking recipes but does not hold in all other domains. For example, some web sites of restaurants consist of several pages, such as pages for describing menu, location, and business hours. For these domains, we can improve recall by treating several linked pages together. To treat relations between pages, we have to expand the hypothesis space from Boolean logic to the first-order logic. Tools for inductive logic programming like FOIL[Quinlan, 1990] can be used for extracting rules of the first-order logic.

- Supporting multi-lingual web search

We used web documents written in Japanese in experiments described in this thesis. There are many languages used in the web and some search engines have started to support multi-lingual access to the web. For example, Google sometimes returns English pages even if user input Japanese keywords. The proposed methods in this research are language-independent and can be directly applied to other languages than Japanese if the tools for extracting keywords from the web are provided. However, extracted keyword spaces or association rules in another language may be different from direct translation of those discovered in Japanese because of cultural difference between communities using the two languages. Studying the differences of rules to describe domains in various languages is itself an interesting research issue. In practice, we can support multi-lingual web search by preparing rules in different languages.

- Utilizing XML and Semantic Web

XML(Extensible Markup Language) is regarded as the successor to HTML for writing web pages. In XML, we can define new tags by

ourselves. When XML comes into wide use, we can extract more precise query refinement rules by referring these tags in web documents.

Many people are making an effort for research and standardization on the semantic web, which is an attempt to make the web understandable by software. It is unrealistic to build a single ontology that can be applied to all domains. Thus, an domain-specific approach that builds many small ontologies for each domain and interoperating them, will be adopted. If the semantic web becomes a reality and ontologies for various domains are provided, we can build more sophisticated domain specific search engines or information agents that can handle multiple domains.

# Bibliography

- [Agrawal and Srikant, 1994] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In *Proceedings of the 20th VLDB Conference*, Santiago, Chile, September 1994.
- [Baeza-Yates and Ribeiro-Neto, 1999] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, 1999.
- [Bharat and Broder, 1998] K. Bharat and A. Broder. A technique for measuring the relative size and overlap of public web search engines. In *Proceedings of the 7th International World Wide Web Conference.*, 1998.
- [Brin and Page, 1998] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. In *WWW7*, 1998.
- [Cho *et al.*, 1998] Junghoo Cho, Hector García-Molina, and Lawrence Page. Efficient crawling through URL ordering. *Computer Networks and ISDN Systems*, 30(1–7):161–172, 1998.
- [Cohen, 1998] William W. Cohen. A web-based information system that reasons with structured collections of text. In *Agents'98*, pages 116–123, 1998.
- [Craven *et al.*, 1998] Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew McCallum, Tom Mitchell, Kamal Nigam, and Seán Slattery. Learning to extract symbolic knowledge from the world wide web. In *AAAI-98*, pages 509–516, 1998.

- [de Campos *et al.*, 1998] Luis M. de Campos, Juan M. Fernández, and Juan F. Huete. Query expansion in information retrieval system using a bayesian network-based thesaurus. In *Proceedings of 14th Conference on Uncertainty in Artificial Intelligence*, pages 53–60, Wisconsin, USA, 1998.
- [Etzioni and Weld, 1994] O. Etzioni and D. Weld. A softbot-based interface to the internet. *CACM*, 37(7):72–76, 1994.
- [Etzioni, 1996] Oren Etzioni. Moving up the information food chain: Deploying softbots on the world wide web. In *AAAI-96*, pages 1322–1326, 1996.
- [gvu, 1998] Gvu’s 10th www user survey. [http://www.cc.gatech.edu/gvu/user\\_surveys/survey-1998-10/](http://www.cc.gatech.edu/gvu/user_surveys/survey-1998-10/), 1998.
- [Harary *et al.*, 1965] Frank Harary, Robert Z. Norman, and Dorwin Cartwright. *Structural Models: An Introduction to the Theory of Directed Graphs*. John Wiley & Sons, 1965.
- [Hayes-Roth, 1985] Barbara Hayes-Roth. A blackboard architecture for control. *Artificial Intelligence*, 26(3):251–321, 1985.
- [Hiramatsu and Ishida, 2001] Kaoru Hiramatsu and Toru Ishida. An augmented web space for digital cities. In *Proceedings of the 2001 Symposium on Applications and the Internet (SAINT-2001)*, pages 105–112, 2001.
- [Hiramatsu *et al.*, 2000] Kaoru Hiramatsu, Kenji Kobayashi, Ben Benjamin, Toru Ishida, and Junichi Akahani. Map-based user interface for digital city kyoto. In *Proc. 10th Annual Internet Society Conference (INET2000)*, Yokohama, Japan, July 2000.
- [Howe and Dreilinger, 1997] Adele E. Howe and Daniel Dreilinger. SAVVYSEARCH: A metasearch engine that learns which search engines to query. *AI Magazine*, 18(2):19–25, 1997.

- [Isbister, 2000] Katherine Isbister. A warm cyber-welcome: Using an agent-led group tour to introduce visitors to kyoto. In Toru Ishida and Katherine Isbister, editors, *Digital Cities: Experiences, Technologies and Future Perspectives*, Lecture Notes in Computer Science 1765, pages 397–406. Springer-Verlag, 2000.
- [Ishida *et al.*, 1999] Toru Ishida, Jun ichi Akahani, Kaoru Hiramatsu, Katherine Isbister, Stefan Lisowski, Hideyuki Nakanishi, Masayuki Okamoto, Yasuhiko Miyazaki, and Ken Tsutsuguchi. Digital city kyoto: Towards a social information infrastructure. In M. Klusch, O. Shehory, and G. Weiss, editors, *Cooperative Information Agents III*, Lecture Notes in Artificial Intelligence 1652, pages 23–35. Springer-Verlag, 1999.
- [Ishida, 1997] Toru Ishida. *Real-Time Search for Learning Autonomous Agents*. Kluwer International Series in Engineering and Computer Science 406. Kluwer Academic Publishers, 1997.
- [Ishida, 1998a] Toru Ishida, editor. *Community Computing and Support Systems*. Lecture Notes in Computer Science 1519. Springer-Verlag, 1998.
- [Ishida, 1998b] Toru Ishida, editor. *Community Computing: Collaboration over Global Information Networks*. John Wiley and Sons, 1998.
- [Ishida, 2001] Toru Ishida. Digital city kyoto: Social information infrastructure for everyday life. *Communications of the ACM (CACM)*, 2001. (to appear.).
- [Jansen *et al.*, 1998] B. Jansen, A. Spink, J. Bateman, and T. Saracevic. Real life information retrieval: A study of user queries on the web. *SIGIR FORUM*, 32(1):5–17, 1998.
- [Joachims *et al.*, 1997] T. Joachims, D. Freitag, and T. Mitchell. Web-watcher: A tour guide for the world wide web. In *Proc. 15th International Joint Conference on Artificial Intelligence*, pages 770–775, Nagoya, Japan, August 1997.

- [K. Decker and Williamson, 1997] K. Sycara K. Decker, A. Pannu and M. Williamson. Designing behaviors for information agents. In *Proc. 1st International Conference on Autonomous Agents*, pages 404–412, February 1997.
- [Kawano and Hasegawa, 1998] Hiroyuki Kawano and Toshiharu Hasegawa. The structure of mondou – web search engine with textual data mining. In *Proc. of Twelfth International Conference on Systems Engineering*, pages 373–378, 1998.
- [Kitamura *et al.*, 1996] Y. Kitamura, H. Nakanishi, T. Nozaki, T. Miura, and T. Ishida. Metaviewer and metacommander: Applying www tools to genome informatics. In *Proc. 7th Workshop on Genome Informatics*, pages 137–146, 1996.
- [Knoblock *et al.*, 1998] Craig A. Knoblock, Steven Minton, Jose Luis Ambite, Naveen Ashish, Pragnesh Jay Modi, Ion Muslea, Andrew G. Philpot, and Sheila Tejada. Modeling web sources for information integration. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 211–218, Madison, WI, USA, 1998.
- [Knoblock, 1995] C. A. Knoblock. Planning, executing, sensing, and replanning for information gathering. In *Proc. 14th International Joint Conference on Artificial Intelligence*, pages 1686–1693, Montreal, Canada, August 1995.
- [Konopnicki and Shmueli, 1995] D. Konopnicki and O. Shmueli. W3ql: A query system for the world wide web. In *Proc. of the 21th Intl. Conf. on Very Large Databases*, pages 54–65, 1995.
- [Kuwabara *et al.*, 1995] K. Kuwabara, T. Ishida, and N. Osato. Agentalk: Describing multiagent coordination protocols with inheritance. In *Proc. 7th IEEE International Conference on Tools with Artificial Intelligence*, pages 460–465, 1995.

- [Lawrence and Giles, 1998] Steve Lawrence and C. Lee Giles. Inquirus, the neci meta search engine. In *Seventh International World Wide Web Conference*, pages 95–105. Elsevier Science, 1998.
- [Lawrence and Gilg, 1999] Steve Lawrence and C. Lee Gilg. Accessibility of information on the web. *Nature*, 400:107–109, 1999.
- [Lesser *et al.*, 1998] V. Lesser, B. Horling, F. Klassner, A. Raja, T. Wagner, and S. XQ. Zhang. Big: A resource-bounded information gathering agent. In *Proc. 15th National Conference on Artificial Intelligence*, pages 539–546, Madison, Wisconsin, July 1998.
- [Levy *et al.*, 1996] Alon Y. Levy, Anand Rajaraman, and Joann J. Ordille. Query-answering algorithms for information agents. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, Portland, Oregon, USA, 1996.
- [Lieberman, 1995] H. Lieberman. Letizia: An agent that assists web browsing. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 924–929, Montreal, Canada, August 1995.
- [Linden *et al.*, 1997] Greg Linden, Steve Hanks, and Neal Lesh. Interactive assessment of user preference models: The automated travel assistant. In *Proc. 6th International Conference on User Modeling (UM97)*, pages 67–78, Sardinia, Italy, June 1997.
- [Linturi *et al.*, 2000] R. Linturi, M. Koivunen, and J. Sulkanen. Helsinki arena 2000 - augmenting a real city to a virtual one. In Toru Ishida and Katherine Isbister, editors, *Digital Cities: Experiences, Technologies and Future Perspectives*, Lecture Notes in Computer Science 1765, pages 84–97. Springer-Verlag, 2000.
- [McCallum *et al.*, 1999] Andrew McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. A machine learning approach to building domain-specific search engines. In *IJCAI-99*, pages 662–667, 1999.

- [Mendelzon *et al.*, 1997] Alberto O. Mendelzon, George A. Mihaila, and Tova Milo. Querying the world wide web. *Int. J. on Digital Libraries*, 1(1):54–67, 1997.
- [Mitchell, 1997] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.
- [Muslea *et al.*, 1999] I. Muslea, S. Minton, and C. A. Knoblock. A hierarchical approach to wrapper induction. In *Proc. 3rd International Conference on Autonomous Agents*, pages 190–197, Seattle, Washington, May 1999.
- [Oyama *et al.*, 2001] Satoshi Oyama, Takashi Kokubo, Toru Ishida, Teruhiro Yamada, and Yasuhiko Kitamura. Keyword spices: A new method for building domain-specific web search engines. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01)*, pages 1457–1463, 2001.
- [Perkowits *et al.*, 1997] M. Perkowits, R. B. Doorenbos, O. Etzioni, and D. S. Weld. Learning to understand information on the internet: An example-based approach. *Journal of Intelligent Information Systems*, 8:133–153, 1997.
- [Quinlan, 1986a] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.
- [Quinlan, 1986b] J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.
- [Quinlan, 1987] J. Ross Quinlan. Simplifying decision trees. *International Journal of Man-Machine Studies*, 27:221–234, 1987.
- [Quinlan, 1990] J. Ross Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.
- [Quinlan, 1993] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

- [Russell and Norvig, 1995] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, 1995.
- [Salton and Buckley, 1990] Gerard Salton and Chris Buckley. Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, 41(4):288–297, 1990.
- [Selberg and Etzioni, 1997] E. Selberg and O. Etzioni. The MetaCrawler architecture for resource aggregation on the Web. *IEEE Expert*, 12(1):11–14, 1997.
- [Shakes *et al.*, 1997] Jonathan Shakes, Marc Langheinrich, and Oren Etzioni. Dynamic reference sifting: a case study in the homepage domain. In *Proceedings of the 6th International World Wide Web Conference (WWW6)*, pages 189–200, 1997.
- [Shaw Jr. *et al.*, 1997] W. M. Shaw Jr., Robert Burgin, and Patrick Howell. Performance standards and evaluations in ir test collections: Cluster-based retrieval models. *Information Processing & Management*, 33(1):1–14, 1997.
- [Shimazu, 2001] Hideo Shimazu. Expertclerk: Navigating shoppers buying process with the combination of asking and proposing. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01)*, pages 1443–1450, 2001.
- [Siegel, 1988] Sidney Siegel. *NONPARAMETRIC STATISTICS*. McGRAW-HILL, 1988.
- [Silverstein *et al.*, 1998] C. Silverstein, M. Henzinger, H. Marais, and M. Moricz. Analysis of a very large altavista query log. SRC Technical Note 1998-014, DEC Systems Research Center, 1998.
- [Srikant and Agrawal, 1995] Ramakrishnan Srikant and Rakesh Agrawal. Mining generalized association rules. In *Proceedings of the 21st VLDB Conference*, Zurich, Swizerland, 1995.

- [Sycara and Zeng, 1996] K. Sycara and D. Zeng. Coordination of multiple intelligent software agents. *International Journal of Cooperative Information Systems*, 5(2&3):181–211, 1996.
- [van den Besselaar and Beckers, 1998] P. van den Besselaar and D. Beckers. Demographics and sociographics of the digital city. In Toru Ishida, editor, *Community Computing and Support Systems*, Lecture Notes in Computer Science 1519, pages 109–125. Springer-Verlag, 1998.
- [Winston, 1992] Patoric Henry Winston. *Artificial Intelligence*. ADDISON-WESLEY, third edition edition, 1992.
- [Xu and Croft, 1996] Jinxi Xu and W. Bruce Croft. Query expansion using local and global document analysis. In *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '96)*, pages 4–11, Zurich Switzerland, August 1996.
- [Yamada *et al.*, 2000] T. Yamada, T. Kokubo, and Y. Kitamura. Information retrieval from www based on multi-agents (in janapnese). In *MACC2000*, Okinawa, December 2000.

# Publications

## Major Publications

### Journals

1. Satoshi Oyama and Toru Ishida, "Applying Association Rules to Information Navigation," IEICE Transactions on Information and Systems, Vol.J84-D-I, No.8, pp.1266-1274, 2001, (in Japanese).
2. Satoshi Oyama, Kaoru Hiramatsu and Toru Ishida, "Cooperative Information Agents for Digital Cities," International Journal of Cooperative Information Systems, Vol.10, No.1&2, pp.197-215, 2001.

### International Conference

1. Satoshi Oyama, Takashi Kokubo, Toru Ishida, Teruhiro Yamada and Yasuhiko Kitamura, "Keyword Spices: A New Method for Building Domain-Specific Web Search Engines," International Joint Conference on Artificial Intelligence (IJCAI-01), pp.1457-1463, 2001.

## Other Publications

### Workshops

1. Satoshi Oyama, Makoto Takema and Toru Ishida, "Development of a Community Information Sharing Platform," The Ninth Workshop on

Multi-Agent and Cooperative Computation (MACC2000),2000, (in Japanese).

2. Satoshi Oyama and Toru Ishida, "Applying Association Rules to Information Navigation," Workshop on Software Agent and its Applications (SAA2000),2000, (in Japanese).

## **Convention**

1. Satoshi Oyama, Takashi Kokubo and Toru Ishida, "Extracting a Boolean Expression of Keywords for Domain-Specific Web Search," The 15th Annual Conference of Japanese Society for Artificial Intelligence, 2001 (in Japanese).

## **Article**

1. Satoshi Oyama, Kaoru Hiramatsu and Koichi Yamada, "Digital City Kyoto," bit, Vol.33, No.4, pp.8-12, 2001, (in Japanese).