Toward Automatic Generation of Web Directories

Satoshi Sato
JAIST / PRESTO, JST
Asahidai 1-1, Tatsunokuchi, Nomi
Ishikawa, 923-1292, JAPAN
sato@jaist.ac.jp

Madoka Sato
Kanazawa Gakuin University
Suemachi 10, Kanazawa
Ishikawa, 920-1392, JAPAN
madoka@kanazawa-gu.ac.jp

Abstract

A way of constructing and editing a large directory of the World Wide Web automatically is to implement a powerful classifier that assigns an appropriate category to every web page. In this paper, we present another way—to automate editing of a link collection for each category. We present a system that generates a link collection from a category word such as *aquarium* and *zoo* without any human interaction.

Keywords: the World Wide Web, web directory, automated editing

1 Introduction

The World Wide Web (WWW) is now the popular resource when we look for certain information. Two kinds of tools help us find the web pages that contain the desired information: search engine and web directory. A search engine is fully automated. In contrast, a web directory is edited manually: Human editors review web pages, and assign an appropriate category to each page. A question arises here: Can computer create a large directory of WWW like Yahoo? The final goal of our study is to answer this question.

One possible procedure to create a web directory is as follows.

Procedure 1

1. Create a classification tree (concept tree) manually. A node of the classification tree corresponds to a classification category.
2. Collect web pages (by using a web robot).
3. Assign an appropriate category to each web page (Figure 1).
4. Generate a link collection for each category from web pages that are classified into the category.

Classification step (step 3) is the key of this procedure. A powerful classifier that assigns an appropriate category to every web page is required to automate this step. As far as we know, nobody has succeeded to implement such a powerful classifier yet.

We present another procedure.

Procedure 2

1. Create a classification tree (concept tree) manually.
2. Generate a link collection for each category by collecting web pages that are relevant to the category (Figure 2).

To automate step 2 in Procedure 2 is much easier than to automate step 3 in Procedure 1 because of two reasons.

- It does not require a powerful classifier; it requires only a binary classifier, which determines whether a web page is relevant to a category.
- Different programs can be used for generating link collections of different types of categories.

In this paper, we present a system that generates a link collection from a given category word, such as *aquarium* and *zoo*, without any human interaction. A generated link collection is organized regionally and hierarchically, so it can be viewed as a small web directory of the specific category—it can be a part of a large directory of WWW.

2 A Directory Example

Here, we show a directory example, which was generated from a word suizokkan(*aquarium*). The directory consists of two kinds of pages: table-of-contents page and digest page of an instance. Figure 3 shows the table-of-contents page of the directory[1]. This page lists aquarium names ordered by region. Each aquarium name has a hyperlink to the digest page of that aquarium, which is also generated automatically by the system. Figure 4 is the digest page of *Okhotsk Aquarium*; it consists of the name, the address, the phone

[1]The original page is only in Japanese, but we insert English translation for understandings.

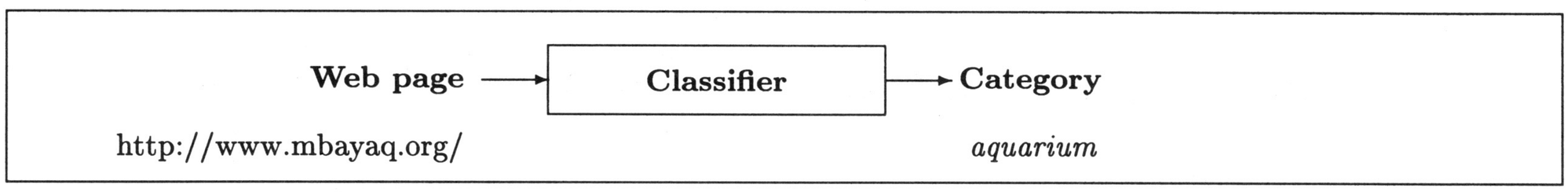

Figure 1: Classifier approach. A classifier assigns an appropriate category to each web page.

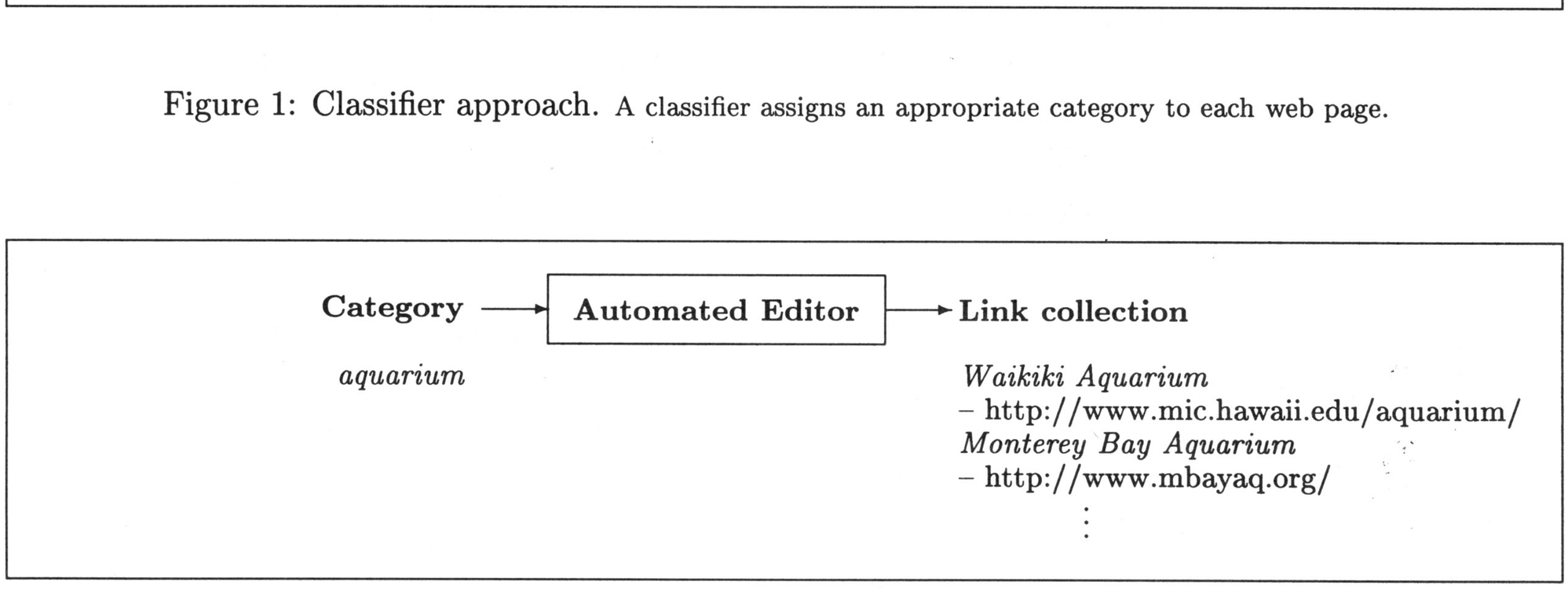

Figure 2: Our approach. An automated editor creates a link collection for each category.

Figure 3: The table-of-contents page of aquarium directory. Aquariums are classified by geographical region.

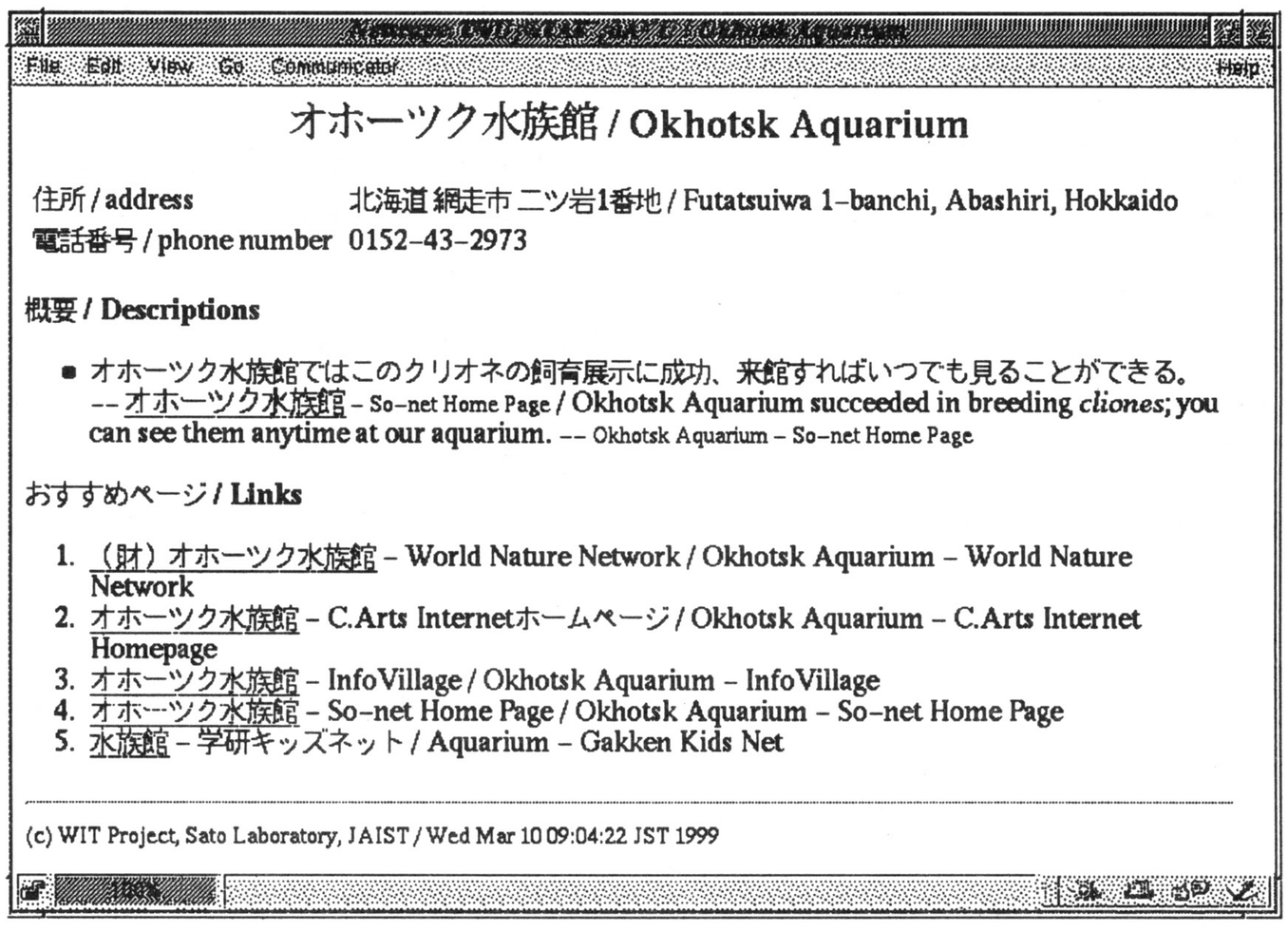

Figure 4: The digest page of *Okhotsk Aquarium.* This page shows digest information and five links to the existing web pages of the aquarium.

Category	*Instance (Place)*	*URL*
aquarium	Waikiki Aquarium (Honolulu, HI)	http://www.mic.hawaii.edu/aquarium/ http://www.hisurf.com/explorer/aquarium.html ⋮
	Monterey Bay Aquarium (Monterey, CA)	http://www.mbayaq.org/ http://members.aol.com/roynlinda/aquarium.htm ⋮
	⋮	

Figure 5: Fundamental structure behind a directory. To create a web directory, we have to fill columns in the table.

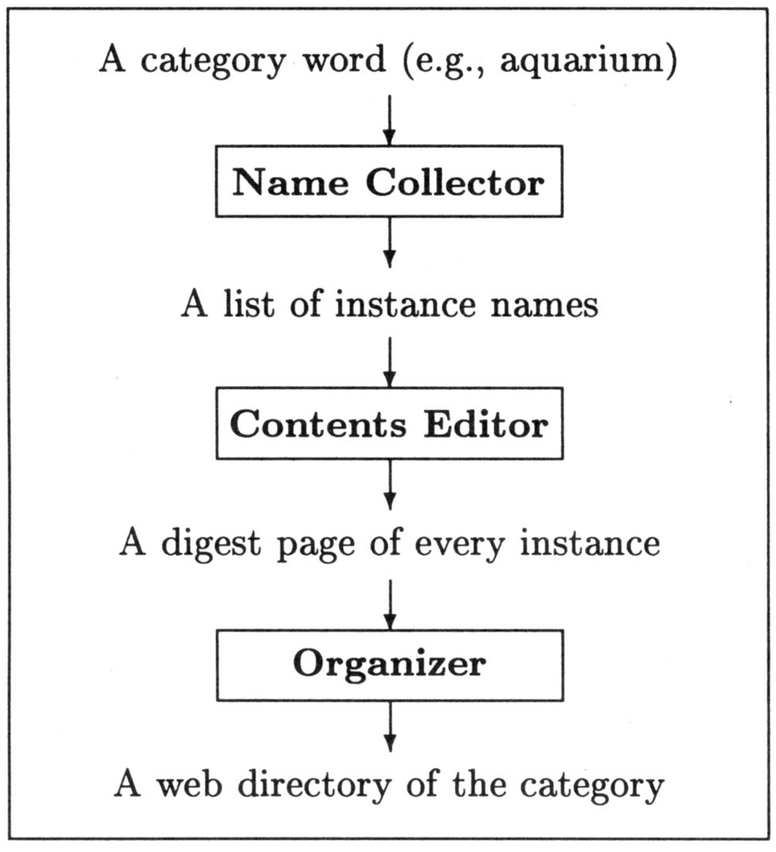

Figure 6: The system configuration. From a given category word (e.g., aquarium), the system generates a web directory.

number, and the five links to the existing web pages of *Okhotsk Aquarium*.

The fundamental structure behind the directory is a table like Figure 5. This table clarifies the problems we have to solve: among them, three major problems are as follows.

1. How do we collect the proper names (instances) of the category (e.g., aquariums)?
2. How do we collect the web pages that are relevant to the specific instance (e.g., *Waikiki Aquarium*)?
3. How do we generate reliable information from unreliable materials (web pages)?

There is no perfect solution to each problem: we conbine partial and practical solutions to realize the system.

3 The System – Automated Editor

Figure 6 shows the outline of the system. The system consists of three major modules: **name collector**, **contents editor**, and **organizer**. From a given category word (e.g., aquarium), the name collector collects the proper names (i.e., the instances of the category). For every instance, the contents editor collects web pages that are relevant to the instance, extracts pre-defined important information from these pages, and generates a digest page of the instance. The organizer generates the regionally-classified table-of-contents of the directory. Though the current system handles only Japanese text, we use English examples for explanation in this section.

1. Waikiki Aquarium
2. Sea Life Park Hawaii
3. Steinhart Aquarium
4. Monterey Bay Aquarium
5. Sea World of California

Figure 7: A link collection of aquariums. We can infer #2 and #4 are also aquariums.

3.1 Name Collector

The first step for generating a web directory of a specific category is to collect proper names (instances) of the category. For example, from the category word *aquarium*, the system collects aquariums' names (e.g., *Waikiki Aquarium*, *Sea Life Park Hawaii*, *Monterey Bay Aquarium*).

A simple heuristic rule is used here: **inference from head noun**. A proper name often has a category word as the head noun. For example, *Waikiki Aquarium* has the word *aquarium* as the head noun (the final word). Therefore, we can infer that *Waikiki Aquarium* is an aquarium, even if we know nothing about it[2]. Based on this rule, the system collects the proper names that have the category words as the head noun. However, this method fails to find *Sea life Park Hawaii*, because it does not contain the word *aquarium*. To complement this method, we use another method that uses existing link collections. Let us imagine that the system found the enumerating list (Figure 7) in a web page. From head nouns, the system infers that item #1, #3, and #4 are aquariums. From this inference result, the system infers that this is a list of aquariums, and thus the rest (i.e., #2 and #5) can be also aquariums. We call this **inference from list**.

We have implemented the name collector based on the above two methods. The current program collects not only names but also their URLs by collecting anchors[3] that contain the category words. The procedure of the name collector is as follows:

1. A category word (e.g., aquarium) is given.
2. Obtain the URLs of the pages that contain the category word by using search engines. Currently the system uses two search engines, Goo (www.goo.ne.jp) and Infoseek (www.infoseek.co.jp), and obtains 200 URLs

[2]Note that the inference is not always correct. There are several counter examples: *Tama Aquarium* and *Joyo Aquarium* are pet shops of tropical fish, not aquariums.

[3]In this paper, we use the word *anchor* as a unit that consists of the anchor tags (`<a>` and `</a>`) and the anchored string (the words between `<a>` and `</a>`). For example, "`<a href=http://www.jaist.ac.jp>JAIST</a>`" is an anchor, and *JAIST* is the anchored string.

from the search results.

3. Obtain pairs of <name, URL> by applying the following procedure to every URL obtained at step 2.
   (a) Download the HTML source of the URL and extract anchors that contain the category word as the head noun.
   (b) If there is a list that is presumably a list of the category's instances, also collect anchors of items that do not contain the category word (based on the inference-from-list method).
   (c) Convert every anchor obtained at (a) and (b) into a pair of <name, URL>.
4. Merge two pairs if they have the same URL or name.
5. Output the list of <(name, ...), (URL, ...)>.

Step 4 is to obtain one set of names for one instance, because an instance are often described by more than one names (i.e., the official name and several nicknames). For example, *Monterey Bay Aquarium* is sometimes referred as *Monterey Aquarium*. In case two anchors have the same URL and different anchor strings, the system infers that both strings may be the names of the same instance.

3.2 Contents Editor

From a set of names of an instance, the contents editor generates a digest page of the instance. Figure 4 shows an example of a digest page, which was generated from the proper name *Okhotsk Aquarium*. The digest page consists of the name, the address, the phone number, the short description, and the link collection of the aquarium.

The procedure of the contents editor is as follows:

1. An instance is given. It is represented by a set of proper names and a set of URLs (i.e., an element of the output of the name collector, <(name, ...), (URL, ...)>).
2. Collect web pages that are relevant to the instance.
3. Select web pages for further steps.
4. Extract summary information (i.e., name, address, city code, phone number, and short description) from every selected page.
5. Determine the summary information of the instance by integrating extracted information.
6. Generate a web page for the instance.

At the collection step (step 2), the system collects the web pages that are relevant to the instance by using the given proper names. The system uses two rules to judge relevancy.

a. Judgment by title
   If a page's title is one of the given proper names, the page is relevant to the instance.
b. Judgment by anchor string
   If a page is referred by the anchor whose string is one of the given proper names, the page is relevant to the instance. For example, if the system finds the anchor `<a href=http://www.mbayaq.org/>Monterey Bay Aquarium</a>`, the system judges that `http://www.mbayaq.org/` is a relevant page to *Monterey Bay Aquarium*.

Currently, we use Infoseek for collecting web pages based on the first rule, because Infoseek provides the *title search*, which can retrieve the pages whose titles contain the given keywords. We also use the URLs that are provided by the name collector, because they are always relevant to the instance according to the second rule.

At the selection step (step 3), the system selects the URLs for the further steps. First, the system tries to access every URL, and filters out unaccessible URLs. Second, the system checks duplication of the pages by simple heuristic rules. For example, when there are `http://.../` and `http://.../index.html` in the URL list, the system compares two HTML sources of these. If these two sources are the same, the system filters out one of them.

At the extraction step (step 4), the system extracts several pieces of information from every URL in the list obtained at step 3. Currently, the system extracts five types of information: the name, the address, the city code[4], the phone number, and the short description (summary) of the instance. The methods that were developed in our previous study on automated editing [6, 7] are used for information extraction and summary extraction.

The determination step (step 5) is crucial, because the contents of the web pages are not always correct. For example, a web page denotes that the phone number of *Tokyo Tower Aquarium* is 03-3434-8833, but another page denotes that it is 03-3433-5111. Which one should we trust? There is no dominant solution to the problem, so we use a majority vote as a practical solution. First, the system determines the city code of the instance by a majority vote. After the vote, the pages that support the different code are filtered out.

[4] A city code is an ID of a city or town in Japan. It is a 5-digit code: the first 2 digits denotes what prefecture the city or town belongs to.

Second, the system determines the phone number: the pages that support the different phone number are also filtered out. Third, the system determines the name and the address in the same way, but filtering process is not used because (1) there can be several names for an instance, and (2) there are various ways to represent the same address[5]. Finally the system accumulates the extracted descriptions from remaining pages.

Sometimes, the contents editor fails to obtain necessary information about the instance. The current system filters out the instance when it fails to determine the city code of the instance[6].

3.3 Organizer

The organizer has two tasks: identical check and generation of the table-of-contents.

The contents editor obtains the name, the address, the city code, and the phone number of each instance. If two instances have different names but the same address, they may be identical. Currently the system determines two instances are identical in case (1) they have the same city code, and (2) they have the same name, the same address, or the same phone number. The system merges two instances into one, if they are identical.

After the number of instances to be presented is determined, the system generates the table-of-contents of the web directory. The table-of-contents is organized regionally and hierarchically. Japan can be divided into 8 regions, 47 prefectures, and 3427 cities and towns; the city code denotes what region and prefecture it belongs, so the system can easily classify every instance into the correct region. The system determines the depth of the table-of-contents—the number of levels of the hierarchy—based on the number of the instances to be presented. In case the number is under one hundred, the system uses only the first level (i.e., 8 regions).

4 Preliminary Evaluation

Currently we are testing the system for three category words: suizokkan (aquarium), dobutsuen (zoo), and bijyutsukan (art museum).

Table 1 shows the number of items in the generated web directories. It is difficult to judge whether an item belongs to the specific category or not, because boundary between categories is sometimes vague. For example, should we classify *Atagawa Tropical and Alligator Garden* into aquarium? How about *Video Reality Aquarium*? In order to avoid the difficulty of judgment, we decided to consult printed guidebooks of each category. In this table, the class *is* means that an item is in the guidebook. The class *prob.* (probably) means that an item is not in the guidebook but probably it is an instance of the category (by judgment from its web pages). The class *not* means that an item is not an instance of the category. We have checked the generated directories of aquariums and zoos, but have not checked the directory of art museums yet. The *total* shows the number of items of each web directory, and the *est.* (estimate) shows the estimate number of instances existing in Japan according to the guidebooks.

Table 1: The number of items of the generated directories.

	Is	Prob.	Not	Total	Est.
Aquarium	44	1	2	47	129
Zoo	60	1	2	63	130
Art museum				286	1560

Table 2: The number of items on Yahoo Japan's pages

	Is	Probably	Total
Aquarium	18	3	22
Zoo	20	0	20

Table 2 shows that the number of instances listed on the Yahoo Japan's pages: the aquarium page lists only 22 aquariums and the zoo page lists only 20 zoos. They are presented by alphabetic order, and are not classified by region; it does not fit to users' convenience.

Table 3 shows the accuracy of the determined information (except summary description[7]) of the confirmed 44 aquariums. The system made only two errors: one name, and one phone number—the accuracy is very high.

5 Discussion

5.1 Generality and Applicability

We have not completely finished the preliminary evaluation, but the current result is promising. Tentatively, we have run the system for several other categories, such as hakubutsukan (museum), kuuko (airport), shokubutsuen (botanical

[5]There are an official description and several unofficial descriptions for an address in Japan. For example, the first part of the official JAIST's address is Asahidai 1-chome 1-banchi but we usually write it as Asahidai 1-1.

[6]Our address extractor is designed to extract addresses only in Japan. Therefore, the instances of other countries are filtered out.

[7]Evaluation of the extracted descriptions (summaries) is not easy, so we have not performed the systematic evaluation yet. However, we have an impression that the current method may be too simple to generate the effective explanatory descriptions: rewriting [8] is necessary.

Table 3: The accuracy of the determined information

	OK	NG
Name	43	1
City code	44	0
Address	44	0
Phone number	43	1

garden), kagakukan (science museum), and keibajyo (horse racing track), and it seems that the system works on these categories except kuuko (airport)[8].

Table 4 shows a summary of the generality and applicability of the current system and its components. We think that the architecture of the system—first collecting instance names, then editing contents for each instance—is applicable to many other types of categories, such as *people* (writer, movie star, etc.) and *merchandise* (computer, automobile, etc). The ability of automatic collection of instance names is crucial for some categories: If the system does not have such ability, we have to collect instance names of the category and put them into the classification tree as subcategories manually in advance; it is impractical if the number of instances are large and changeable.

The current name collector, unfortunately, is not powerful enough. First, the *inference from head noun* is a reliable heuristic but it does not work on categories such as department store: none of Neiman-Marcus, Macy's, Bloomingdale's, and Nordstrom has an explicit expression of department store in its name. Second, the *inference from list* works only when majority of the items in the list are already classified into the category. We need a more powerful method for name collection.

Other two modules, the contents editor and the organizer, are designed on the assumption that the address can be used as an identifier of each instance. Therefore, they work only on *place*-type categories. It is natural that *information extractor*, *identical check*, and *organization scheme* are specific to the category type. For a different category type, e.g., *people*, we should extract a different set of information, and use a different identifier and a different organization scheme.

A variant of the contents editor works as an *address finder*, because it finds the address, the phone number, and URLs from a given name. We have implemented an address finder *Doko*, which is a modified (and faster) version of the contents editor and works on line.

[8]Many web pages of airports have no address information. We think that it is because transportation information is much more important and crucial, and the address is less important and omittable.

5.2 Related Works

The Clever project [5, 1, 2] is the most similar work to ours. Their primary goal is to investigate the second generation of search engines that output a small number of authoritative documents (*hubs* and *authorities*), and their HITS algorithm [5] can be used for generating a link collection for a given topic (category) [2]. Their method works well on many topics such as *cheese*, *alcoholism*, and *Zen Buddhism*, on which our system may not work. The important difference between their work and ours is the number of links to be output. They focus on generating a small number (typically ten) of authoritative links of the given topic or category. In contrast, we focus on generating a directory that consists of more than a hundred links, where sub-classification and organization is necessary.

The Scorpion project [11] takes the traditional approach, i.e., Procedure 1 described in Section 1. This project focuses on building the automatic subject recognition tool—a powerful classifier that assigns appropriate codes of Dewey Decimal Classification to each web page. There are two reasons why we do not take this traditional approach. First, we prefer *active* search to *passive* classification. Our approach actively collects web pages that are relevant to a category. In contrast, the traditional approach passively classifies the given web pages that are collected by someone. Because collecting all web pages in WWW preparatory to classification is very difficult and impractical, the active approach is better than the passive one in case we make a directory of a specific category. Second, we now focus on generating rich directories of specific categories, because they are desirable and valuable for many of WWW users. We do not believe that the traditional classification schemes such as Dewey Decimal Classification and Nippon Decimal Classification fit such directories; a special-purpose (i.e., category-specific) classification scheme works much better.

In the context of information food chain [4], our system can be classified into *information carnivores*, but we take a different approach from softbots such as MetaCrawler [10] , Ahoy! [9], and ShopBot [3] in the following point: Our system generates the *prêt-à-porter* style information package for typical users, whereas the softbots generate the *hauto couture* answer to each user's query.

Table 4: Generality and Applicability

Architecture	applicable to many categories
Name Collector	
inference from head noun	reliable heuristic but limited applicability
inference from list	general heuristic but sometimes not powerful enough
Contents Editor	
relevancy judgment	general heuristic but sometimes not powerful enough
information extraction	specific to *place*
credit assignment	general heuristic (majority vote)
Organizer	
identical check	specific to *place* (address)
organization scheme	specific to *place* (geographical)

6 Concluding Remarks

In this paper, we have presented the system that generates web directories of specific categories automatically. We have not completely finished the preliminary evaluation, but the current result seems promising.

An important remaining issue is classification of the pages that are relevant to each instance—the current system simply enumerates the pages. The system should detect the types of the pages (e.g., official page or private visiting report), evaluate the values of the pages, and order them by their types and values.

There is still a long way to the final goal: automatic generation of a large directory of WWW. Intensive study of generating link collections of other types of category is required. We have already started studying methods for three types of categories: *people* (writer, movie star, etc.), *merchandise* (computer, automobile, etc.), and *region* (Tokyo, Kyoto, etc.). An advantage of our approach is that we can obtain a lot of small web directories of specific categories in the course of the study. These directories themselves are useful and valuable to WWW users.

References

[1] Soumen Chakrabarti, Byron Dom, Prabhakar Raghavan, Sridhar Rajagopalan, David Gibson, and Jon Klenberg. Automatic resource compilation by analyzing hyperlink structure and associate text. In *Proceedings of the 7th World-Wide Web conference*, 1998.

[2] Soumen Chakrabarti, Martin van den Berg, and Byron Dom. Focused crawling: a new approach to topic-specific Web resource discovery. In *Proceedings of the 8th World-Wide Web conference*, 1999.

[3] Robert B. Doorenbos, Oren Etzioni, and Daniel S. Weld. A scalabel comparison-shopping agent for the World-Wide Web. In *Proceedings of the First International Conference on Autonomous Agents*, 1997.

[4] Oren Etzioni. Moving up the information food chain. *AI Magazine*, 18(2), 1997.

[5] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. IBM research report, RJ 10076, IBM, 1997.

[6] Madoka Sato. *Automated Editing of Electronic Texts.* PhD thesis, School of Information Science, Japan Advanced Institute of Science and Technology, 1997.

[7] Madoka Sato and Satoshi Sato. Natural language processing in the automated editing system for the USENET articles. In *Proceedings of Natural Language Processing Pacific Rim Symposium 1997*, pages 261–266, 1997.

[8] Satoshi Sato and Madoka Sato. Rewriting saves extracted summaries. In *Intelligent Text Summarization, Technical Report, SS–98–06*, pages 76–83. American Association for Artificial Intelligence, 1998.

[9] Jonathan Shakes, Marc Langheinrich, and Oren Etzioni. Dynamic reference shifting: a case study in the homepage domain. In *Sixth International World Wide Web Conference*, 1997.

[10] Erik Sleberg and Oren Etzioni. The MetaCrawler architecture for resource aggregation on the Web. *IEEE Expert*, 12(1), 1997.

[11] Roger Thompson, Keith Shafer, and Dinae Vizine-Goetz. Evaluating Dewey concepts as a knowledge base for automatic subject assignment. In *Second ACM International Conference on Digital Libraries*, 1997.