

# A Novel Web-oriented Writing Environment using Objects' Facts Acquired from the Web

Naoki Yoshinaga  
Japan Society  
for the Promotion of Science  
6 Ichibancho, Chiyoda-ku,  
Tokyo, 102-8471, Japan  
n-yoshi@jaist.ac.jp

Kazumasa Nakamura  
Hitachi GP Corporation  
System Engineering, Ltd.  
2-4-18 Toyo, Koutou-Ku,  
Tokyo, 135-8633, Japan  
kazu@kazu59.jp

Kentaro Torisawa  
Japan Advanced Institute  
of Science and Technology  
1-1, Asahidai, Nomi,  
Ishikawa, 923-1292, Japan  
torisawa@jaist.ac.jp

## Abstract

*This paper presents a novel web-oriented writing environment that helps users describe their opinions on topics/events through weblogs, by showing facts related to the objects. Our writing environment automatically recognizes information needs for facts on aspects (e.g., "opening hours") of the objects (e.g., "J's cafe") by applying lexico-syntactic patterns to the snapshots of users' writings. It then attempts to extract facts that satisfy the information needs from the Web, and shows the extracted facts (e.g., "opening hours"- "10am-8:30pm") to the users. The users can thus refer to necessary facts without searching the web by themselves. We realized this environment for Japanese on a browser with AJAX, and evaluated how much useful information could be provided by our environment.*

## 1. Introduction

The recent Internet revolution has witnessed increasing opportunities for people to share their experiences with topics/events through weblogs. To provide accurate and distinctive information on the target, bloggers are likely to search for *facts* on various aspects of the objects to be described before and *during* writing (hereafter, *fact search*). For example, when you recommends a cafe you have visited, you may want to know the precise location and opening hours to help the readers visit. The bloggers may even find unexpected facts that remind them of other unique aspects of the objects to be mentioned (e.g., specialties other than the one they tried), which may stimulate their writings.

Fact search is thus essential for writing on the net, but it abruptly interrupts our writing and usually requires substantial time and effort, which prevents us from concentrating on writing. Assisting this fact search would give more chances for people to provide useful comments.

In this paper, we propose a web-oriented writing environment that assists users to write about topics/events by dynamically acquiring facts of the objects from the Web.

Our writing environment adopts an AJAX-based client-server architecture. The client editor sends snapshots of users' writings to the remote server at fixed intervals. The server applies lexico-syntactic patterns [2] to the given snapshots to recognize possible information needs for facts on aspects (e.g., "opening hours") of the objects (e.g., "J's bar"). It then extracts facts that answer the information needs from the Web (e.g., "opening hours"- "10am-8pm"), and returns them to the client. The user can thus refer to several facts at once without being interrupted by the search.

## 2. Web-oriented writing environment

In this section, we first show a possible scenario when we recommend a cafe object, and describe how the fact search interrupts our writing. We then describe the design and components of our web-oriented writing environment.

### 2.1. Possible scenario – recommend a cafe

When one recommends a cafe she has visited, she performs a fact search as in the following scenario (Figure 1).

**Recognizing information needs:** She first recognizes that she does not know exactly when the cafe opens, which would help the readers visit the cafe. Information needs for 'opening hours' of the cafe emerge.

**Performing fact search:** She next tries to search for the opening hours of the cafe, by reading through web pages and finding text fragments relevant to the facts needed (opening hours). She may need to read through other pages when the facts found seem insufficiently informative (e.g., '?-8:30pm') or even obsolete.

**Quoting the facts:** She quotes the most reliable fact and edits it in her writing style. When she finds other facts that are worth being mentioned (e.g., facts on 'menu'), she will next describe those aspects.

The problem here is that she may need to spend much time to browse and read through several pages to retrieve

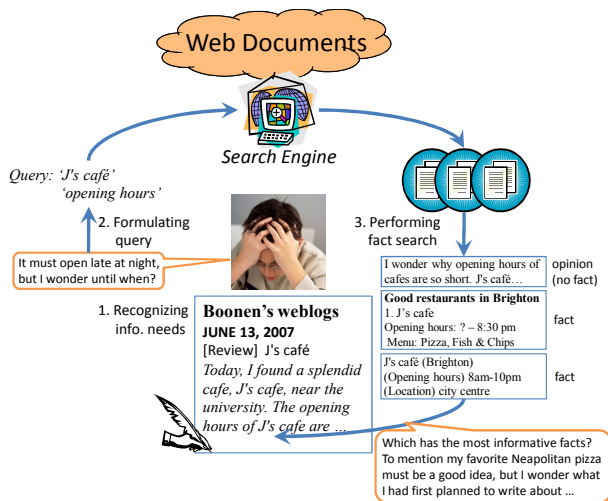


Figure 1. Possible scenario of fact search

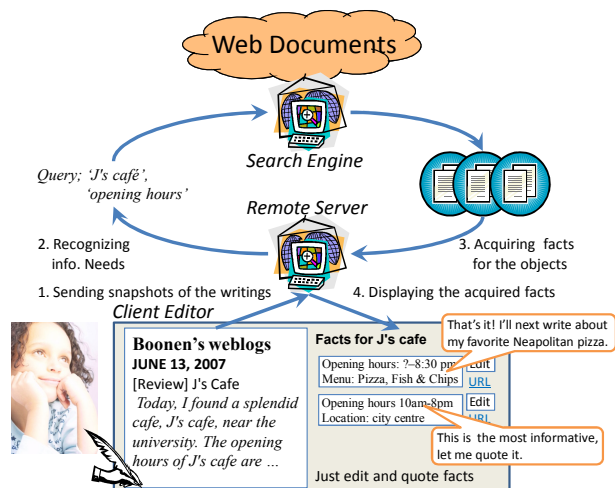


Figure 2. Computer-aided fact search

the facts needed, because a single web page does not always include reliable facts for the desired aspects. Also, by reading several pages, she may find facts of other aspects worth being mentioned, but she may be distracted by the excess of information. Fact search is thus likely to occupy considerable time in describing our opinions about objects in the web, which seriously interrupts the thoughts we first had in mind.

## 2.2. Design of our writing environment

To let people concentrate on essential writing, we automate the process in the above possible scenario. Figure 2 depicts our AJAX-based writing environment. It works for the above situation in the following way:

1. A user writes fragments of comments, “*Today, I found a splendid café, J's café, near the university. The opening hours of J's café are*”, on the client editor. The client sends this snapshot to the remote server.
2. The remote server next recognizes from the snapshot her information needs for an aspect of an object, and formulates a search query to retrieve web pages to acquire facts for the information needs.
3. The server then acquires the facts from the web pages. When facts needed are acquired, the server sends the facts to the client (e.g., “*?-8:30pm*” and “*10am-8pm*”) along with facts for other aspects of the object.
4. The client finally shows the acquired facts on the editor screen. She can keep writing by referring to the facts.

By the writing environment performing fact search on behalf of the user, the user can glance through facts that are acquired from several web pages. This significantly reduces the user's workload in searching for facts, and lets the user concentrate on the writing itself. The key feature

of our environment is to recognize users' information needs automatically from snapshots of their writings. This may allow the users to notice that the information they had for the objects are insufficient when the environment could show more reliable facts written by others. Also, detecting information needs before they are recognized by the user allows the environment to spend more time searching for the facts.

In the following, we describe how to recognize users' information needs, and then explain a method of acquiring facts for the recognized information needs.

## 2.3. Recognition of users' information needs using lexico-syntactic patterns

To detect users' information needs, we should know the intention in their writings. Researchers in the field of natural language processing (NLP) have found that certain types of facts are likely to be described in some syntactic patterns [1, 6]. For example, assume the following sentences:

— The *specialties* of **J's café's** are Neapolitan pizza and organic beer. Although the *size* of **the Neapolitan pizza** is usually less than —

The strings in bold face are objects, while the strings in italic face are *attributes* (the term used to refer to aspects of objects in the field of NLP), which are followed by the facts on the attributes. We can see that the same syntactic pattern is used for describing objects, attributes, and facts; namely, “the A of O [is |are] V” where A is an attribute, O is an object and V is a fact (attribute's value). Such syntactic patterns are called *lexico-syntactic patterns* [2] in the field of NLP, and have been widely used for extracting certain knowledge such as object-attribute-fact relations [1, 6, 7] from sentences. We make use of the patterns used for extracting object-attribute-fact relations in order to recognize user's intentions for writing facts on attributes of the objects, because these patterns are often used to describe facts.

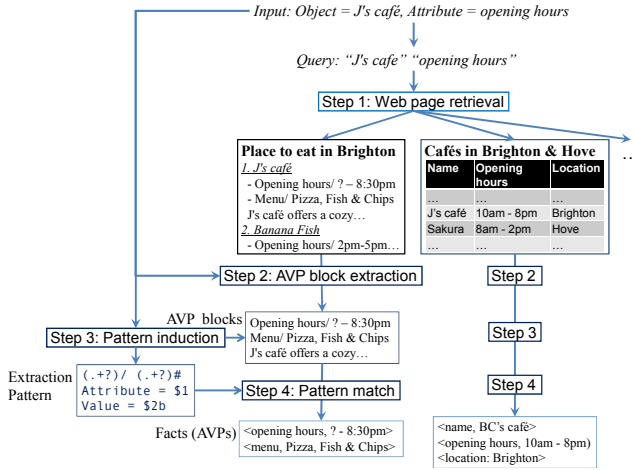


Figure 3. Fact Retrieval Engine

In order to recognize information needs for facts related to objects in Japanese, we exploit the existing lexico-syntactic patterns [6, 7] for acquiring facts:

O の A は (The A of X [is |was])

where O is a proper noun and A is a noun. By using this pattern, we identify information needs for an aspect (or an attribute, A) of an object (O). In the case when the lexico-syntactic patterns fail to detect information needs, we prepare an optional interface for a user to directly specify an object and its aspects for which the user wants facts.

## 2.4. Automatic fact retrieval from the Web

In this section, we briefly describe a method of acquiring facts for the information needs that are recognized by the lexico-syntactic patterns. We hereafter denote facts for attributes of objects as the attributes' value, and refer to an attribute-value pair as AVP (e.g., (open hours, 10am-8pm)).

To effectively acquire values for the input attribute of the object, we exploit facts represented in layouts such as lists or tables in the web (Figure 3). Because experts on the target objects who know facts on several attributes of the objects are likely to use layouts in order to concisely describe several facts, we will be able to acquire more reliable facts from the layouts than normal sentences written by novices, as well as acquire facts for other attributes of the object.

Our AVP acquisition consists of the following steps. We first retrieve web pages that are likely to include an AVP for a given object and attribute (Step 1). We next enumerate regions that include objects' AVPs (AVP blocks) from each retrieved page (Step 2). We then induce patterns to extract each AVP from each block (Step 3). We finally apply the induced patterns to the blocks and acquire AVPs from them (Step 4). In the following, we describe each step in detail.

**Step 1. Collecting pages that include AVPs:** We use a commercial search engine with a query consisting of the

object and the attribute to collect web pages that are likely to include an AVP for the input attribute of the object.

**Step 2. Identifying AVP blocks:** In Step 2, we identify layouts (tables or lists) that describe the objects' AVPs in the web pages retrieved in Step 1. The pages may even include layouts that describe AVPs for other objects (e.g., facts for "Banana Fish" and "Sakura" in Figure 3). We investigated where objects and their AVPs appear in web pages, and hypothesize the positional relations between objects and layouts that describe the AVPs for the objects as follows:

**Hypothesis I:** AVPs for a given object are likely to appear concentrated in a small number of regions (AVP blocks) enclosed by HTML block tags. An AVP block describing an object should include attribute words, and also include or closely follow the object name.

We process each occurrence of the object name in the page from first to last, and iteratively collect innermost blocks enclosed by block tags such that the blocks include the input attribute preceded by the appearance of the object name as candidate AVP blocks.

**Step 3. Inducing AVP Extraction Patterns:** We then induce patterns for extracting AVPs from each AVP block collected in Step 2. To design the induction procedure, we made the following assumption:

**Hypothesis II:** In an AVP block, an attribute immediately precedes its value, and other AVP immediately follow those values. When attributes in an AVP block are emphasized by some HTML tags, braces, prefixes or suffixes, the same symbols are used to emphasize other attributes in that block.

Following this hypothesis, our procedure induces a regular expression as a generic pattern for extracting attributes and their values. The procedure first applies patterns used for extracting attributes from layouts [8] to the AVP block, and utilizes the matched strings that include the input attribute to identify HTML tags or symbolic cues that are used to emphasize attributes in the block. By replacing the input attribute in this string with a wildcard that matches the shortest string, we acquire a regular expression for extracting attributes in this block. A value for an attribute is a string that spans from immediately after the corresponding attribute to just before another attribute. By putting a special character '#' before the attributes in the block, we obtain a generic pattern for extracting attributes and values. Note that this pattern can extract AVPs for attributes other than the input attribute (e.g., menu in Figure 3).

**Step 4. Acquiring AVPs from AVP blocks:** We iteratively acquire AVPs from each AVP block by applying the extraction pattern induced for the block in Step 3 to the block. We merge all the AVPs acquired from the blocks and return the resulting AVPs. When the patterns extract the same attributes twice, we stop the acquisition from the page and return the acquired AVPs until that point. This is because when the same attribute appears twice in the page, it usually expresses an AVP for different objects.

Steps 3 and 4 assumed that the AVP block is described by a list. When we fail to recognize AVPs from AVP blocks enclosed by `th` or `td` tags (which express a cell in a table), we attempt to recognize AVPs from the whole table. We detect a row or column that represents attributes based on the position of the input object and attribute. We omit further details due to space limitations.

### 3. Experiments

We implemented our writing environment for Japanese using AJAX technology on a browser, and evaluated the usefulness of our environment. As we described in section 2.3, our writing environment returns facts for users' information needs which are recognized by lexico-syntactic patterns. We investigated objects and their attributes that often appear in lexico-syntactic patterns in major Japanese blog sites (articles), and regard them as possible information needs during writing. We then evaluated how many object-attribute pairs (information needs) we could retrieve correct facts about among those object-attribute pairs to show the utility of our writing environment. We also measured the efficiency of our fact retrieval engine to examine whether it can be used as a real-time application.

#### 3.1. Experimental settings

In this section, we explain how we built a testset of object-attribute pairs (information needs), and then describe the specification of the testset in detail.

We obtained class-object pairs from hyponymy relations automatically acquired from the definition sentences in Wikipedia [5]. We sorted hypernyms in the hyponymy relations according to the number of the hyponyms, and selected the top-1,000 hypernyms to obtain major hypernyms with a large number of hyponyms. We then obtained class-object relations from these hyponymy relations by assuming hypernyms as classes and by eliminating their hyponyms X when a hit count<sup>1</sup> of “*ど*の X (which X)” was larger than 0 because generic concepts appear in this pattern [4].

We next acquired at most ten attributes for each class in the class-object relations by using an existing attribute acquisition method [8].<sup>2</sup> We calculated a hit count of “O の A (the A of O)” for each pair of an object O and an attribute A in each class in seven major Japanese blog sites,<sup>3</sup> in order to obtain *major* attributes (aspects) that were often referred to by people. We regarded attributes whose hit counts of “O の A” was larger than 0 for at least three objects in the class as the major attributes, and randomly chose 39 classes and at most five major attributes for them. Then, we randomly chose at most five objects for the selected attributes, from

Table 1. A testset of the object-attribute pairs

class	attributes	# obj, # pairs
女優 (actress)	映画 (film), ドラマ (drama), 趣味 (hobby), テーマ (theme), 監督 (director)	5 25
シングル (CD single)	C D, 作詞 (lyricist), テーマ (theme), 編曲 (arranger), 順位 (rank),	5 25
タレント (talent)	写真集 (photo book), 出身地 (hometown), 本名 (autonym), 出演 (appear- ance), 映画 (film appearance)	5 25
アルバム (CD album)	C D, 価格 (price), 作詞 (lyricist), 作曲 (composer), プロフィール (profile),	5 25
ピアニスト (pianist)	ピアノ (piano), 演奏 (performance), 音楽 (music)	5 15
アニメ (animation)	音楽 (music), キャラクターデザイン (character design), 脚本 (story), 製 作 (producer), ジャンル (genre),	5 25
声優 (dubbing artist)	画像 (picture), キャラクター (character), プロフィール (profile), 代表作 (magnum opus)	5 20
芸人 (comedian)	プロフィール (profile)	5 5
ブランド (brand)	特徴 (feature), 価格 (price), バッグ (bag), カラー (color), ポイント (point)	5 25
料理 (cooking)	作り方 (recipes), 材料 (ingredient)	5 10
法律 (law)	目的 (objective), 罰則 (penalty), 定義 (definition), 検討 (inspection)	5 20
都市 (city)	人口 (population), 平均気温 (ave. temperature), サイズ (size), 価格 (price), プロフィール (profile)	5 25
レース (race)	予選 (prelim), 決勝 (final match), タイヤ (tire), コース (course)	5 20
ドラマ (drama)	主題歌 (theme song), 原作 (original), 監督 (director), 脚本 (writer), 出演 (starring)	5 25
神社 (shrine)	社務所 (office), 主祭神 (main god worshiped), 手水舎 (wash-pot), 名称 (name), 社殿 (pavilion)	5 25
企業 (company)	本社 (head office), 代表取締役 (CEO), 社名 (name), 売上高 (sales amount), 電話番号 (phone)	5 25
オペラ (opera)	音楽 (music), 作曲 (composer), 台本 (script), 演奏 (performance), 原作 (original), 合唱 (chorus)	5 25
鉱物 (mineral)	結晶 (crystal), 原石 (raw stone)	5 10
人物 (person)	性格 (character), 年齢 (age), プロフィール (profile), 趣味 (hobby), 身長 (height)	5 25
武器 (weapon)	攻撃 (attack), 価格 (price)	5 10
戦争 (war)	映画 (film)	5 5
番組 (program)	出演 (starring), 内容 (content), 画像 (picture), 音楽 (music), 放送時間 (airtime)	5 25
劇場 (theater)	場所 (location), 会場 (venue), 出演 (starring), 電話番号 (phone), 電話 (phone)	5 25
メーカー (manufacturer)	価格 (price), 商品名 (goods name)	5 10
植物 (plant)	別名 (alias), 学名 (nomenclature), 和名 (Japanese name), 原産地 (place of origin), 価格 (price)	5 10
アーティスト (artist)	C D, 音楽 (music)	5 10
文書 (document)	内容 (content)	5 5
ソフトウェア (software)	ダウンロード (download), バージョン (ver.), 価格 (price)	5 15
公園 (park)	場所 (location), トイレ (bathroom)	5 10
器具 (instrument)	サイズ (size), 価格 (price)	5 10
制度 (institution)	目的 (objective), 内容 (content)	5 10
行事 (event)	会場 (venue), 時間 (schedule)	5 10
菓子 (sweet stuff)	賞味期限 (expiry date)	5 5
パン (bread)	作り方 (recipe)	3 3
路線 (route)	駅名 (station name)	5 5
警察署 (police station)	電話 (phone), 管轄区域 (district boundary)	5 10
台風 (typhoon)	最低気圧 (least atmospheric pressure), 中心気圧 (core atmospheric pressure)	5 10
資格 (qualification)	試験内容 (examination contents)	4 4
バス (bus)	時刻表 (timetable)	4 4
<b>total</b>		<b>191 611</b>

objects whose hit counts of “O の A” were larger than 0 for at least one attribute A in the class. Table 1 shows the attributes used to generate the testset of object-attribute pairs. The total number of the object-attribute pairs was 611.

#### 3.2. Experimental results

We then acquired facts for each object-attribute pair using a method described in Section 2.4. We collected top-15 web pages for each pair using the Yahoo API. 14.97 pages (in total 9,148 pages) were on average downloaded for each object-attribute pair.

For each web page, a human subject was asked to judge if the system could extract correct facts for the given object-attribute pair. The subject judged facts as correct when he could find an association between the extracted facts and the object name in the page. Table 2 shows the results of fact retrieval for the object-attribute pairs. The column titled CORR shows the number of object-attribute pairs for which we successfully acquired correct facts for the attribute from at least one web page, the column titled CORR\* shows the number of those for which we acquired correct facts with some irrelevant strings for the attribute from at least one web page. The column FAIL shows the number of those for which we failed to obtain correct facts for the attribute.

<sup>1</sup> Yahoo API (<http://developer.yahoo.co.jp/>) is used to obtain hit counts.

<sup>2</sup> We also ran a simple supervised classifier to filter out implausible attributes. We omit details here since it is irrelevant to the main argument.

<sup>3</sup> [blogs.yahoo.co.jp](http://blogs.yahoo.co.jp), [fc2.com](http://fc2.com), [blog.goo.ne.jp](http://blog.goo.ne.jp), [blog.livedoor.jp](http://blog.livedoor.jp), [ameblo.jp](http://ameblo.jp), [yaplog.jp](http://yaplog.jp), and [plaza.rakuten.co.jp](http://plaza.rakuten.co.jp): these sites had more than 100,000 users in June, 2007 (cf., <http://www.blogfan.org/service/>)

We acquired correct facts for 244 (CORR: 230, CORR\*: 14) (40%) object-attribute pairs. We should note that even when we exclude facts extracted from the pages in Wikipedia (<http://ja.wikipedia.org/>), we acquired correct facts for 240 (39%) object-attribute pairs. This means that Wikipedia was not the only source for acquiring facts from the Web.

We observed that the coverage varied greatly from one class to another. Object-attribute pairs in some classes did not have concise facts that could be written in layouts on the web. For example, most of the attributes for pianist, race, mineral, artist, document, institution, bread, bus were too abstract to obtain objective facts from layouts. Also, most of the attributes for war and manufacturer were meaningless. For 519 object-attribute pairs in 29 classes other than these classes, we acquired correct facts for 233 (45%) object-attribute pairs. We conclude that our writing environment is helpful for writing about objects in certain classes.

We then evaluated the efficiency in acquiring facts for the object-attribute pairs. In order to acquire facts for each object-attribute pair, the remote server took on average 12.8 sec. (total 7843.4 sec) for downloading top-15 web pages and 2.8 sec. (total 1686.8 sec.) for extracting facts from them. Although our fact retrieval engine needed around 15.6 (= 12.8 + 2.8) sec. to acquire facts for the object-attribute pairs, we can decrease this to 2.8 sec. if we could implement our fact retrieval engine on a search engine, which can handle web documents without downloading.

We have also observed the number of object-attribute pairs for which we acquired correct facts from at least one page among the top-5 pages retrieved by the Yahoo API (Table 2). The number of object-attribute pairs for which we could acquire correct facts reduced from 244 (CORR: 230, CORR\*: 14) to 153 (CORR: 140, CORR\*: 13) while the timing needed also reduced from 12.8 sec. to 4.7 sec. (3.8 sec. for downloading, 0.9 sec. for extracting facts). This difference suggests that we will be able to obtain more facts by increasing the number of web pages to be processed when the coverage is more important.

Finally, to estimate the number of facts (AVPs) acquired for attributes other than the input attributes, we calculated the number of facts that our system returns for the 244 object-attribute pairs for which we acquired correct facts for the input attribute. The system extracted on average 14.3 correct facts for each object-attribute pair. This means that our writing environment will give more chances for users to write opinions about various aspects of the target objects.

## 4. Conclusion

We have proposed a web-oriented writing environment that helps users to describe opinions about objects by dynamically acquiring facts of various aspects (attributes) of the objects from the web. Experimental results showed that our method acquired correct facts for 244 (40%) of 611 open domain object-attribute pairs when we processed 15 web pages for each pair, and we acquired on average 14.3 facts for each pair. These results suggest that our writing

**Table 2. Experimental results of fact retrieval for the object-attribute pairs**

class	# pairs	top-5 pages			top-15 pages		
		CORR	CORR*	FAIL	CORR	CORR*	FAIL
actress	25	2	0	23	4	0	21
CD single	25	5	2	18	12	1	12
talent	25	6	0	19	6	0	19
CD album	25	12	0	13	14	0	11
pianist	15	0	0	15	0	1	14
animation	25	12	1	12	20	0	5
dubbing artist	20	2	0	18	4	0	16
comedian	5	0	0	5	0	0	5
brand	25	6	0	19	8	1	16
cooking	10	1	0	9	3	0	7
law	20	5	0	15	7	0	13
city	20	0	0	20	1	0	18
race	25	0	0	20	1	0	19
drama	25	14	1	10	14	2	9
shrine	25	5	0	20	5	0	20
company	25	12	2	11	16	1	8
opera	25	9	1	15	15	2	8
mineral	10	0	0	10	0	0	10
person	25	1	0	24	3	0	22
weapon	10	2	0	8	3	0	7
war	5	0	0	5	0	0	5
program	25	1	0	24	1	0	24
theater	25	7	1	17	11	1	13
manufacturer	10	3	1	6	7	0	3
plant	25	12	1	12	16	2	7
artist	10	0	0	10	0	0	10
document	5	0	0	5	0	0	5
software	15	2	0	13	3	1	11
park	10	1	0	9	2	0	8
instrument	10	5	1	4	8	0	2
institution	10	0	0	10	0	0	10
event	10	1	1	8	2	1	7
sweet stuff	5	2	0	3	2	0	3
bread	3	0	0	3	0	0	3
route	5	1	0	4	1	0	4
police station	10	4	0	6	5	0	5
typhoon	10	3	1	6	5	0	5
qualification	4	2	0	2	2	0	2
bus	4	0	0	4	0	0	4
total	611	140	13	458	230	14	367

environment is helpful for writing about objects on the net.

We will use object search [8, 3] to collect web pages for objects, and exploit on-line databases for some classes (e.g., amazon.com for book objects) or other methods of acquiring facts from sentences [6] to improve the coverage of fact retrieval. We will employ other lexico-syntactic patterns to recognize users' information needs.

## References

- [1] A. Almuhareb and M. Poesio. Attribute-based and value-based clustering: An evaluation. In *Proc. EMNLP*, 2004.
- [2] M. A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proc. COLING*, 1992.
- [3] Z. Nie, Y. Ma, S. Shi, J.-R. Wen, and W.-Y. Ma. Web object retrieval. In *Proc. WWW*, 2007.
- [4] A. Sumida, K. Shinzato, and K. Torisawa. Concept-instance relation extraction from simple noun sequences using a search engine on a web repository. In *Proc. ISWC workshop on Web Content Mining with Human Language Technologies*, 2006.
- [5] A. Sumida and K. Torisawa. Hacking Wikipedia for hyponymy relation acquisition. 2007. in submission.
- [6] T. Takahashi. *Computation of Semantic Equivalence for Question Answering*. PhD thesis, Nara Institute of Science and Technology, Nara, Japan, 2005.
- [7] K. Tokunaga, J. Kazama, and K. Torisawa. Automatic discovery of attribute words from web documents. In *Proc. IJCNLP*, 2005.
- [8] N. Yoshinaga and K. Torisawa. Finding specification pages according to attributes. In *Proc. WWW*, 2006.