1. Introduction

Automatic knowledge extraction from such a very large document corpus as the Web is one of the hottest research topics in the domain of Artificial Intelligence and Database technologies. This chapter introduces my object-oriented and the existing methods to extract semantic (e.g., hyponymy and meronymy) and sensory (e.g., visual and aural) knowledge from the Web, and compares them by showing several experimental results. My object-oriented semantic knowledge extraction is based on property inheritance(s) and property aggregation, and repeatedly improves the extracted results of both hyponymy and meronymy relations. Meanwhile, my object-oriented sensory knowledge extraction is improved by utilizing the extracted hyponymy and meronymy relations. Finally, this chapter introduces my Sense-based Object-name Search (SOS) to enable users to identify the concrete name of a target object which they do not know only by inputting its hyponym (class-name) and some sensory descriptions, as an application system to utilize the Web-extracted semantic and sensory knowledge.

2. Concept Hierarchy Extraction from the Web

Concept hierarchies, such as hyponymy (is-a) and meronymy (has-a) relations between words, are very fundamental as semantic knowledge for various natural language processing systems. For example, query expansion in information retrieval (Hattori, Tezuka, Ohshima, Oyama, Kawamoto, Tajima & Tanaka, 2007; Hattori et al., 2006; Mandala et al., 1998), question answering (Fleischman et al., 2003), machine translation, object information extraction by text mining (Hattori, Tezuka & Tanaka, 2007), and so forth. Also, I have been developing Sense-based Object-name Search (SOS) for a name-unknown object by its class-name (e.g., “bird”) and feature descriptions (e.g., “blue wings”) as one of the application systems that utilize hyponymy relations (e.g., isa(“kingfisher”, “bird”) = 1) as a basic knowledge (Hattori & Tanaka, 2009).

While the WordNet (2009) (Miller et al., 1990) and Wikipedia (2009) (Völkel et al., 2006) etc. are being manually constructed and maintained as lexical ontologies at the cost of much time and effort, many researchers have tackled how to extract concept hierarchies from very large corpora of text documents such as the Web not manually but automatically (Caraballo, 1999; Hearst, 1992; Kim et al., 2006; Morin & Jacquemin, 2004; Ruiz-Casado et al., 2007; Sanderson &

---

*Presently with School of Computer Science, Tokyo University of Technology.
Croft, 1999; Shinzato & Torisawa, 2005). However, their methods are mostly based on lexico-
syntactic patterns as sufficient but not necessary conditions of concept hierarchies. Therefore,
they can achieve high precision but only low recall when using stricter patterns (e.g., “x such
as y”), or they can achieve high recall but only low precision when using looser patterns (e.g.,
“y is a/an x”).

To achieve high recall and not low precision, I first propose a basic method to extract hy-
ponymy relations (i.e., subordinate concepts for a target concept) from the Web based on sin-
gle “Property Inheritance” from the target concept to its hyponym candidate as a necessary
and sufficient condition of hyponymy (Hattori et al., 2008).

Next, I try to make my basic method more robust by two approaches: not single but “Multi-
ple Property Inheritances” and “Property Aggregation” (Hattori & Tanaka, 2008). The former
approach is to utilize the other semantic relations surrounding the subordinate relation (hy-
ponymy) between a target concept and its hyponym candidate, i.e., superordinate relations
(hypernymy) and coordinate relations (including synonymy and antonymy), and to improve
a weighting of hyponymy extraction by using multiple property inheritances not only from
the target concept to its hyponym candidate but also between the other pairs of concepts (e.g.,
from a hypernym of the target concept to its hyponym candidate and/or from the target con-
cept to a coordinate concept of its hyponym candidate). The latter approach is to improve a
weighting of property (meronymy) extraction by using property aggregation to each target
concept from its typical hyponyms.

The remainder of this section is organized as follows. Section 2.1 introduces my basic hy-
ponymy extraction based on single “Property Inheritance”. Section 2.2 and 2.3 describe my
improved hyponymy extraction based on “Multiple Property Inheritances” and “Property
Aggregation” respectively. Section 2.4 shows several experimental results to evaluate my pro-
posed hyponymy extractions.

2.1 Hyponymy Extraction based on Single Property Inheritance

I introduce my basic method to extract hyponymy relations (i.e., subordinate concepts for
a target concept) from the Web not only by using lexico-syntactic patterns as sufficient but
not necessary conditions of hyponymy but also by using single “Property Inheritance” as its
necessary and sufficient condition (PI-based hyponymy extraction as shown in Fig. 1).

Basic Assumption

Let C be the universal set of concepts (words). I assume that a concept \( y \in C \) is a hyponym
of a concept \( x \in C \) if and only if the set of properties that the concept \( y \) has, \( P(y) \), completely
includes the set of properties that the concept \( x \) has, \( P(x) \), and the concept \( y \) is not equal
(equivalent) to the concept \( x \):

\[
isa(y, x) = 1 \iff P(y) \supseteq P(x) \text{ and } y \neq x,
\]

\[
P(c) = \{ p_i \in P | \text{has}(p_i, c) = 1 \},
\]

where \( P \) stands for the universal set of \( N \) properties,

\[
P = \{ p_1, p_2, ..., p_N \},
\]

and \( \text{has}(p_i, c) \in \{0, 1\} \) indicates whether or not a concept \( c \in C \) has a property \( p_i \in P \),

\[
\text{has}(p_i, c) = \begin{cases} 1 & \text{if a concept } c \text{ has a property } p_i, \\ 0 & \text{otherwise.} \end{cases}
\]
In other words,

\[
\text{isa}(y, x) = \begin{cases} 
1 & \text{if } \sum_{\forall p_i \in P} \text{has}(p_i, x) \cdot \text{has}(p_i, y) = \sum_{\forall p_i \in P} \text{has}(p_i, x) \cdot \text{has}(p_i, x), \\
0 & \text{if } \sum_{\forall p_i \in P} \text{has}(p_i, x) \cdot \text{has}(p_i, y) < \sum_{\forall p_i \in P} \text{has}(p_i, x) \cdot \text{has}(p_i, x),
\end{cases}
\]

(5)

\[\text{P}(c) = (\text{has}(p_1, c), \text{has}(p_2, c), ..., \text{has}(p_N, c)).\]

(6)

It is very essential for hyponymy extraction based on the above assumption of hyponymy and property inheritance to calculate the binary value \(\text{has}(p_i, c) \in \{0, 1\}\) for any pair of a concept \(c \in C\) and a property \(p_i \in P\) accurately. However, it is not easy, and I can use only the continuous value \(\text{has}^*(p_i, c) \in [0, 1]\) in the below-mentioned method. Therefore, I suppose that the ratio of the number of properties that a concept \(y \in C\) inherits from a target concept \(x \in C\) to the number of properties that the target concept \(x\) has,

\[
\frac{\sum_{\forall p_i \in P} \text{has}^*(p_i, x) \cdot \text{has}^*(p_i, y)}{\sum_{\forall p_i \in P} \text{has}^*(p_i, x) \cdot \text{has}^*(p_i, x)}
\]

(7)

can measure how suitable the concept \(y\) is for a hyponym of the target concept \(x\), as an approximation of whether or not the concept \(y\) is a hyponym of the target concept \(x\), \(\text{isa}^*(y, x)\).

And then, the concept \(y\) would be considered to be a hyponym of the target concept \(x\) when the ratio is enough near to one (or greater than a threshold value), while the concept \(y\) would be considered to be not a hyponym of the target concept \(x\) when the ratio is not near to one (or less than a threshold value).
Method

When a target concept \( x \in C \) is given, my basic hyponymy extraction based on single property inheritance executes the following three steps to extract its hyponyms from the Web. First, a set of candidates for its hyponyms, \( C(x) \), is collected from the Web as exhaustively as possible. Second, the continuous value \( \text{has}^*(p_i, c) \) for each pair of a property \( p_i \in P \) and a concept \( c \) (the target concept \( x \) or its hyponym candidate \( y \in C(x) \)) is calculated by using Web search engine indices. That is, the property vector of each concept \( c \), \( P^*(c) \), is obtained,

\[
P^*(c) = (\text{has}^*(p_1, c), \text{has}^*(p_2, c), ..., \text{has}^*(p_N, c)).
\]

(8)

Last, the continuous value \( \text{isa-PI}^*_n(y, x) \) for each pair of the target concept \( x \) and its hyponym candidate \( y \) is calculated based on single inheritance of the top \( n \) typical properties of the target concept \( x \) to its hyponym candidate \( y \), and then a set of its top \( k \) hyponym candidates ordered by their weight would be outputted to the users.

Step 1. Hyponym Candidate Collection

A set of hyponym candidates of the target concept, \( C(x) \), is collected from the Web as exhaustively as possible and enough precisely. If \( C(x) \) is set to the universal set of concepts, \( C \), its recall always equals to 1.0 (the highest value) but its precision nearly equals to 0.0 (too low value). Meanwhile, if \( y \in C(x) \) is collected from some sort of corpus of documents by using too strict lexico-syntactic pattern such as “\( y \) is a kind of \( x \)”, its precision is enough high but its recall is too low in most cases. Therefore, I use not too strict but enough strict lexico-syntactic pattern of hyponymy to collect the set. Any noun phrase \( y \) whose lexico-syntactic pattern “\( y \) is a/an \( x \)” exists at least once in the title or summary text of the top 1000 search results by submitting a phrase “is a/an \( x \)” as a query to Yahoo! Web Search API (2008) is inserted into \( C(x) \) as a hyponym candidate of the target concept \( x \).

Step 2. Property Extraction

Typical properties (meronyms) \( p_i \) of each concept \( c \) (the target concept \( x \) or its hyponym candidate \( y \in C(x) \)) are extracted as precisely as possible from the Web by using an enough strict lexico-syntactic pattern “\( c \)'s \( p_i \)” as a sufficient condition of meronymy. The continuous value \( \text{has}^*(p_i, c) \) of a property \( p_i \) for each concept \( c \) is defined as follows:

\[
\text{has}^*(p_i, c) := \frac{\text{df}(["c\'s\ p_i"])}{\text{df}(["c\'s"])},
\]

(9)

where \( \text{df}([q]) \) stands for the number of documents that meet a query condition \( q \) in such a corpus as the Web. In the after-mentioned experiments, I calculate it by submitting each query to Yahoo! Web Search API (2008).

Note that \( \text{has}^*(p_i, c) \) is not a binary value \( \{0, 1\} \) but a continuous value \([0, 1]\), so \( \text{has}^*(p_i, c) \) cannot indicate whether or not a concept \( c \) has a property \( p_i \) but it is supposed to indicate how typical the property \( p_i \) is of the concept \( c \).

Step 3. Hyponym Candidate Weighting

To filter out noisy hyponym candidates of the target concept, my basic hyponymy extraction based on single Property Inheritance, i.e., how many properties a hyponym candidate
$y \in C(x)$ inherits from the target concept $x$, assigns the following continuous weight to each hyponym candidate:

$$
\text{isa-PI}_n(y, x) := \frac{\sum_{p_i \in P_n(x)} \text{has}^*(p_i, x) \cdot \text{has}^*(p_i, y)}{\sum_{p_i \in P_n(x)} \text{has}^*(p_i, x) \cdot \text{has}^*(p_i, x)} \quad (10)
$$

where $P_n(x)$ stands for a set of the top $n$ typical properties of the target concept $x$ ordered by their weight $\text{has}^*(p_i, x)$ that is calculated in Step 2.

Note that if $n = N$, i.e., $P_n(x) = P_n(y) = P$, I cannot decide which $x$ or $y$ is subordinate to the other because of $\text{isa-PI}_n(y, x) = \text{isa-PI}_n(x, y)$. So, I must set $n$ to a number which is less than $N$. In the after-mentioned experiments, I use 1 to 20 as $n$ to evaluate its effect.

Meanwhile, an existing hyponymy extraction based on lexico-Syntactic Patterns defines the following weighting:

$$
\text{isa-SP}(y, x) := \text{df}(\text{“y is a/an x”}). \quad (11)
$$

I use it as a baseline to validate my proposed hyponymy extractions in the after-mentioned experiments.

---

**Fig. 2.** Surrounding concepts in hyponymy extraction based on property inheritance.
2.2 Hyponymy Extraction based on Multiple Property Inheritances

My previous hyponymy extraction is based on single property inheritance from a target concept to a hyponym candidate. One approach to make it more robust is to utilize the other semantic relations surrounding the subordinate relation (hyponymy) \( \text{isa}(y, x) \) between the target concept \( x \) and the hyponym candidate \( y \), i.e., superordinate relations (hypernymy) and coordinate relations (including synonymy and antonymy), as shown in Fig. 2. I propose a improved method to extract hyponymy relations (i.e., subordinate concepts for a target concept) from the Web based on not single but “Multiple Property Inheritances”.

**Basic Assumption**

I assume that if a concept \( y \in C \) is a hyponym of a concept \( x \in C \), then the set of properties that the concept \( y \) has, \( P(y) \), completely includes not only the set of properties that the concept \( x \) has, \( P(x) \), but also the set of properties that \( x \)'s hypernym \( x_h \) has, \( P(x_h) \), and the concept \( y \) is not equal (equivalent) to the concept \( x \):

\[
\text{isa}(y, x) = 1 \iff P(y) \supseteq P(x) \quad \text{and} \quad P(y) \supseteq P(x_h) \quad \text{and} \quad y \neq x, \tag{12}
\]

where \( \text{Hypernym}(x) \) stands for a set of superordinate concepts (hypernyms) of a concept \( x \). I also assume that if a concept \( y \in C \) is a hyponym of a concept \( x \in C \), then not only the set of properties that the concept \( y \) has, \( P(y) \), but also the set of properties that \( y \)'s coordinate concept \( y_c \) has, \( P(y_c) \), completely includes the set of properties that the concept \( x \) has, \( P(x) \), and the concept \( y \) is not equal (equivalent) to the concept \( x \):

\[
\text{isa}(y, x) = 1 \iff P(y) \supseteq P(x) \quad \text{and} \quad P(y_c) \supseteq P(x) \quad \text{and} \quad y \neq x, \tag{13}
\]

where \( \text{Coordinate}(y, x) \) stands for a set of coordinate concepts of a concept \( y \) sharing a concept \( x \) as their superordinate concept.

**Method**

When a target concept \( x \in C \) is given, my improved hyponymy extraction based on multiple property inheritances executes the following four steps to extract its hyponyms from the Web. First, a set of candidates for its hyponyms, \( C(x) \), is collected from the Web as exhaustively as possible. Second, hypernyms of the target concept, \( x_h \), or coordinate concepts of each hyponym candidate, \( y_c \), is extracted from the Web as precisely as possible. Third, the continuous value \( \text{has}^*(p_i, c) \) for each pair of a property \( p_i \in P \) and a concept \( c \) is calculated by using Web search engine indices. Last, the continuous value \( \text{isa-PIH}_n^*(y, x) \) or \( \text{isa-PIC}_n^*(y, x) \) for each pair of the target concept \( x \) and its hyponym candidate \( y \in C(x) \) is calculated based on multiple property inheritances not only from the target concept \( x \) to its hyponym candidate \( y \) but also from \( x \)'s hypernym \( x_h \) to its hyponym candidate \( y \) or from the target concept \( x \) to \( y \)'s coordinate concept \( y_c \) as shown in Fig. 3 or 4.

**Step 1. Hyponym Candidate Collection**

(Just all the same as Step 1 in Section 2.1.)
Fig. 3. Hyponymy and multiple property inheritances from not only a target concept $x$ but also its superordinate concept (hypernym) $x_h$ to its subordinate candidate $y$ (PIH-based).

**Step 2. Hypernym or Coordinate Extraction**

There are some methods to extract hypernymy or coordinate relations (Ohshima et al., 2006). Because my improved method requires at least one hypernym of the target concept or one coordinate term of each hyponym candidate as precisely as possible, I use not recall-conscious but precision-conscious extraction. A hypernym $x_h$ of the target concept $x$ is extracted by using stricter lexico-syntactic pattern “$x_h(x)$ such as $x$ is a/an $x_h$”, and a coordinate term $y_c$ of a hyponym candidate $y \in C(x)$ is extracted by using a pair of enough strict lexico-syntactic patterns “$y$ or $y_c$” and “$y_c$ or $y$” (Hattori, Tezuka, Ohshima, Oyama, Kawamoto, Tajima & Tanaka, 2007).

**Step 3. Property Extraction**

(Just all the same as Step 2 in Section 2.1.)

**Step 4. Hyponym Candidate Weighting**

A weight for each hyponym candidate $y$ of the target concept $x$ is defined based on two Property Inheritances not only from the target concept $x$ to the hyponym candidate $y$ but also from a Hypernym $x_h$ of the target concept to the hyponym candidate $y$:

$$\text{isa-PIH}_n(y, x) := (1 - \alpha) \cdot \text{isa-PI}_n(y, x) + \alpha \cdot \text{isa-PI}_h(y, x_h),$$

(14)

where $\alpha \in [0, 1]$ stands for a certain combination parameter.

Another weight is also defined based on two Property Inheritances not only from the target concept $x$ to the hyponym candidate $y$ but also from the target concept $x$ to a Coordinate concept $y_c$ of the hypernym candidate:

$$\text{isa-PIC}_n(y, x) := (1 - \beta) \cdot \text{isa-PI}_n(y, x) + \beta \cdot \text{isa-PI}_n(y_c, x),$$

(15)

where $\beta \in [0, 1]$ stands for a certain combination parameter.
Fig. 4. Hyponymy and multiple property inheritances from a target concept $x$ to not only its subordinate candidate $y$ but also $y$’s coordinate concept $y_c$ (PIC-based).

### 2.3 Meronymy Refinement based on Property Aggregation

Another approach to make my hyponymy extraction based on “Property Inheritance(s)” more robust is to improve a method of property extraction (Step 2 in Section 2.1 and Step 3 in Section 2.2). I propose a improved method to extract hyponymy relations (i.e., subordinate concepts for a target concept) from the Web by adding “Property Aggregation” as a necessary and sufficient condition of hyponymy.

#### Basic Assumption

Let $Y \subset C$ be a set of concepts. I assume that all concepts of $\forall y \in Y$ are hyponyms of a concept $x \in C$ if and only if the product set of any set of properties that the concept $y$ has, $\bigcap P(y)$, completely includes the set of properties that the concept $x$ has, $P(x)$, and the set of concepts $Y$ does not include the concept $x$:

$$
\forall y \in Y, \ \text{isa}(y, x) = 1 \iff \bigcap_{y \in Y} P(y) \supseteq P(x) \text{ and } x \notin Y. \quad (16)
$$

In other words,

$$
\text{has}(p_i, x) = \begin{cases} 
1 & \text{if } \sum_{c \in C} \text{isa}(c, x) \cdot \text{has}(p_i, c) = \sum_{c \in C} \text{isa}(c, x), \\
0 & \text{if } \sum_{c \in C} \text{isa}(c, x) \cdot \text{has}(p_i, c) < \sum_{c \in C} \text{isa}(c, x). \end{cases} \quad (17)
$$

It is also essential for property extraction based on the above assumption of meronymy and property aggregation to calculate the binary value $\text{isa}(c, x) \in \{0, 1\}$ for any pair of concepts accurately. However, it is not easy, and I can use only the continuous value $\text{isa}^*(c, x)$ in the below-mentioned method. Therefore, I suppose that the ratio

$$
\frac{\sum_{c \in C} \text{isa}^*(c, x) \cdot \text{has}^*(p_i, c)}{\sum_{c \in C} \text{isa}^*(c, x)}, \quad (18)
$$
can measure how suitable a property $p_i$ is for a concept $x$, as an approximation of whether or not the concept $x$ has the property $p_i$, has*($p_i$, $x$).

**Method**

When a target concept $x \in C$ is given, my improved hyponymy extraction based on property inheritance(s) and property aggregation executes the first step and repeats the other steps cyclically to extract its hyponyms from the Web. First, a set of candidates for its hyponyms, $C(x)$, is collected. Second, the continuous value has*($p_i$, $c$) for each pair of a property $p_i \in P$ and a concept $c$ is calculated by using Web search engine indices. That is, the typical properties of each concept $c$ (the target concept $x$ or its hyponym candidate $y \in C(x)$) are extracted.

Third, the continuous weight $\text{isa-PIA}^*(0)(y, x)$ for each pair of the target concept $x$ and its hyponym candidate $y \in C(x)$ is calculated based on property inheritance from the original property vector of the target concept $x$, $P^0(x)$, to the original property vector of its hyponym candidate $y$, $P^0(y)$. Next, only the property vector of the target concept $x$, $P^1(x)$, is re-calculated based on property aggregation from its hyponym candidates with their weights, and the continuous weight $\text{isa-PIA}^*(1)(y, x)$ for each pair of the target concept $x$ and its hyponym candidate $y$ is re-calculated based on property inheritance from the new property vector of the target concept $x$, $P^1(x)$, to the original property vector of its hyponym candidate $y$, $P^0(y)$. Subsequently, the method repeats several times in the same way. Finally, a set of its top $k$ hyponym candidates ordered by their weight would be outputted to the users.

**Step 1. Hyponym Candidate Collection**

(Just all the same as Step 1 in Section 2.1.)

**Step 2. Property Extraction and Improvement**

Typical properties (meronyms) $p_i$ of each concept $c$ (the target concept $x$ or its hyponym candidate $y \in C(x)$) are extracted and repeatedly improved based on property aggregation. The
0-th (original) continuous weight of a property \( p_i \) for each concept \( c \) (i.e., how typically a concept \( c \) has a property \( p_i \)) is defined by using the document frequency of the lexico-syntactic pattern “\( c \)’s \( p_i \)” just as Step 2 in Section 2.1:

\[
\text{has}^{(0)}(p_i, c) := \frac{\text{df}([“c\’s \, p_i”])}{\text{df}([“c”])}.
\]  

(19)

The \( m \in \{1, 2, \ldots\} \)-th continuous weight of a property \( p_i \) for only the target concept \( x \) is defined as follows:

\[
\text{has}^{(m)}(p_i, x) := (1 - \gamma) \cdot \text{has}^{(m-1)}(p_i, x) + \gamma \cdot \frac{\sum_{y \in \mathcal{C}(x)} \text{isa-PI}_n^{(m-1)}(y, x) \cdot \text{has}^{(0)}(p_i, y)}{\sum_{y \in \mathcal{C}(x)} \text{isa-PI}_n^{(m-1)}(y, x)}.
\]  

(20)

where \( \gamma \in [0, 1] \) stands for a certain combination parameter.

**Step 3. Hyponym Candidate Weighting**

My improved hyponymy extraction based on Property Inheritance and Aggregation assigns the \( m \in \{0, 1, 2, \ldots\} \)-th weight to each hyponym candidate \( y \) just as Step 3 in Section 2.1:

\[
\text{isa-PI}_n^{(m)}(y, x) := \frac{\sum_{p_i \in P^{(m)}_n(x)} \text{has}^{(m)}(p_i, x) \cdot \text{has}^{(0)}(p_i, y)}{\sum_{p_i \in P^{(m)}_n(x)} \text{has}^{(m)}(p_i, x) \cdot \text{has}^{(m)}(p_i, x)},
\]  

(21)

where \( P^{(m)}_n(x) \) stands for the \( m \)-th set of the top \( n \) typical properties of a concept \( x \) ordered by their weight \( \text{has}^{(m)}(p_i, x) \) that is calculated in Step 2. Note that

\[
\text{isa-PI}_n^{(0)}(y, x) = \text{isa-PI}_n^*(y, x).
\]  

(22)

**2.4 Experiment**

I show several experimental results to evaluate my proposed methods of extracting hyponymy relations (i.e., subordinate concepts for a target concept) from the Web based on “Property Inheritance(s)” and “Property Aggregation” by comparing them with a traditional lexico-syntactic pattern based hyponymy extraction.

First, I applied my basic hyponymy extraction based on single Property Inheritance (i.e., \( \text{isa-PI}_n^*(y, x) \) defined in Section 2.1) and a traditional one based on such a lexico-Syntactic Pattern as “\( y \) is a/an \( x \)” (i.e., \( \text{isa-SP}(y, x) \) defined in Section 2.1) to 25 kinds of target concepts as follows.

**Animals:** “amphibian”, “bird”, “fish”, “mammal”, and “reptile”.

**Plants:** “flower”, “fruit”, “herb”, “tree”, and “vegetable”.

**People:** “actor”, “novelist”, “prime minister”, “singer”, and “soccer player”.

**Products:** “furniture”, “magazine”, “musical instrument”, “stationery”, and “vehicle”.

**Places:** “capital city”, “hot spring resort”, “shrine”, “temple”, and “world heritage site”.

www.intechopen.com
Fig. 6 compares my PI-based and the SP-based hyponymy extractions with regard to the average of area under PR (Precision-Recall) curve (Davis & Goadrich, 2006) for all of 25 concepts. My PI-based hyponymy extraction is almost completely superior to the SP-based one, and gains the most, +0.0586 (+11.75%), when \( n = 17 \).

Fig. 7 and Table 1 show the average PR curves and the recall per precision by my PI-based \((n = 17)\) and the SP-based hyponymy extractions respectively. My PI-based hyponymy extraction is almost superior to the SP-based one, but is sometimes a little inferior in the high recall range (about from 0.9 to 1.0). My PI-based hyponymy extraction cannot salvage any candidate concept \( c \) whose \( \text{df}(["c"]') \) is equal to zero as a hyponym of a target concept. For example, most scientific names such as “panthera leo” and “allium cepa” cannot be salvaged, while most common names such as “lion” and “onion” corresponding to them can be salvaged. Therefore, I could refine my proposed extractions if I can utilize the equivalent relations between scientific names and their common names in some handcrafted databases or automatically extract the equivalent relations from the Web.

<table>
<thead>
<tr>
<th>Precision</th>
<th>0.80</th>
<th>0.75</th>
<th>0.70</th>
<th>0.65</th>
<th>0.60</th>
<th>0.55</th>
<th>0.50</th>
<th>0.45</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP-based</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0083</td>
<td>0.0294</td>
<td>0.2649</td>
<td>0.7511</td>
<td>1.0000</td>
</tr>
<tr>
<td>PI-based ((n = 17))</td>
<td>0.0088</td>
<td>0.0305</td>
<td>0.0883</td>
<td>0.2160</td>
<td>0.4441</td>
<td>0.6659</td>
<td>0.8195</td>
<td>1.0000</td>
</tr>
</tbody>
</table>

Table 1. Recall per precision by SP-based and PI-based hyponymy extractions.

Fig. 6. Area under precision-recall curve by Syntactic Pattern (SP) based vs. single Property Inheritance (PI) based hyponymy extractions.
Second, I also applied my improved hyponymy extractions based on “Multiple Property Inheritances” not only from a target concept $x$ to its hyponym candidate $y$ but also from a Hypernym $x_h$ of the target concept to the hyponym candidate $y$ (i.e., isa-PIH$_n^*(y, x)$ defined in Section 2.2), and not only from a target concept $x$ to its hyponym candidate $y$ but also from the target concept $x$ to a Coordinate concept $y_c$ of the hyponym candidate (i.e., isa-PIC$_n^*(y, x)$ defined in Section 2.2) to the same 25 kinds of target concepts.

Fig. 8 compares my improved PIH-based and my basic PI-based hyponymy extractions with regard to average of area under PR curve per combination parameter $\alpha$. Note that my basic PI-based hyponymy extraction is equivalent to my improved PIH-based one when $\alpha = 0.00$. Unfortunately, my improved PIH-based hyponymy extraction is almost inferior to my basic PI-based one. Therefore, I should not take into account the additional property inheritance from a hypernym $x_h$ of a target concept $x$ to its hyponym candidate $y$.

Fig. 9 and 10 compare my improved PIC-based and my basic PI-based hyponymy extractions with regard to average of area under PR curve per combination parameter $\beta$ and per number of properties $n$ respectively. Note that my basic PI-based hyponymy extraction is equivalent to my improved PIC-based one when $\beta = 0.00$. Fortunately, my improved PIC-based hyponymy extraction is superior to my basic PI-based one and gains the most, +0.0073 (+1.31%), and it is almost completely superior to the SP-based one and gains the most, +0.0659 (+13.22%), when $n = 17$ and $\beta = 0.40$. And Fig. 9 shows that I should not set the combination parameter $\beta$ too great, i.e., take into account too much the additional property inheritance from a target concept $x$ to a coordinate concept $y_c$ of its hyponym candidate $y$. 

Fig. 7. Average precision-recall curves by Syntactic Pattern (SP) based vs. single Property Inheritance (PI) based hyponymy extractions.
Subsequently, I also applied my improved hyponymy extraction based on both single Property Inheritance and Property Aggregation (i.e., isa-PIA$_n^m(x,y)$ defined in Section 2.3) to the same 25 kinds of target concepts.

Fig. 11 and 12 compare my improved PIA-based and my basic PI-based hyponymy extractions with regard to average of area under PR curve per combination parameter $\gamma$ and per number of repeats $m$ respectively. Note that my basic PI-based hyponymy extraction is equivalent to my improved PIA-based hyponymy extraction when $\gamma = 0$. My improved PIA-based hyponymy extraction is superior to my basic PI-based one and gains the most, $+0.0071$ ($+1.28\%$), and it is superior to the SP-based one and gains the most, $+0.0657$ ($+13.18\%$), when $n = 17$, $\gamma = 0.45$, and $m = 1$. But it is slightly inferior to my improved PIC-based one.

Last, I also applied my combined hyponymy extraction based on both multiple Property Inheritances with Coordinate and Property Aggregation (PICA), i.e., by combining both my PIC-based and PIA-based methods, to the same 25 kinds of target concepts. Fortunately, my combined PICA-based hyponymy extraction is the best among my proposed ones. It is superior to my basic PI-based one and gains the most, $+0.0222$ ($+3.99\%$), and it is superior to the SP-based one and gains the most, $+0.0808$ ($+16.21\%$), when $n = 4$, $\beta = 0.30$, $\gamma = 0.90$, and $m = 11$.

Fig. 13 shows the average PR curves for all of 25 concepts by my basic PI-based ($n = 17$), my improved PIC-based ($n = 17$, $\beta = 0.40$), my improved PIA-based ($n = 17$, $\gamma = 0.45$, $m = 1$), and my combined PICA-based ($n = 4$, $\beta = 0.30$, $\gamma = 0.90$, $m = 11$) hyponymy extractions. My combined PICA-based hyponymy extraction is almost the best in the low recall range (about from 0.00 to 0.65), while my improved PIC-based hyponymy extraction is almost the
Fig. 9. Average of area under precision-recall curve per $\beta$ by single Property Inheritance (PI) based vs. multiple Property Inheritances with Coordinate (PIC) based hyponymy extractions.

best in the high recall range (about from 0.65 to 0.90). In the future, I try to make more robust hyponymy extraction in the whole recall range by sampling from both worlds (PICA-based and PIC-based).

Table 2 summarizes the experimental results for comparison of my proposed hyponymy extractions. They show that my basic hyponymy extraction based on single property inheritance (PI-based) is superior to a traditional hyponymy extraction based on such a lexico-syntactic pattern as “$y$ is a/an $x$”, that one of my improved hyponymy extractions based on multiple property inheritances (not PIH-based but PIC-based) is superior to my basic PI-based one as well as the traditional SP-based one, that my improved hyponymy extraction based on both property inheritance and property aggregation (PIA-based) is superior to my basic hyponymy extraction based on only property inheritance but is slightly inferior to my improved PIC-based one, and that my combined hyponymy extraction based on both multiple property inheritances with coordinate and property aggregation (PICA-based) is superior to the others.

As future directions, I have to invent a method for parameter optimization, and I will tackle how to construct time-dependent and/or space-dependent concept hierarchies for context-aware applications in mobile and ubiquitous computing environments.
3. Visual Description Extraction from the Web

Sensory knowledge of objects, such as visual descriptions/images and audio descriptions/clips, is very useful for us in various situations. For example, object-name search by sensory information when we encounter a name-unknown species of bird, insect, or plant and want to know its concrete name to search the Web for its more detail documents in mobile and ubiquitous computing environments, and sensory information search by object-name when we visit a unacquainted area and want to know the appearance information of navigated landmarks rather than their concrete names in car/walk navigation systems (Fig. 14). One approach to solve these problems is to manually construct databases of relationships between object-names and their sensory information. There are already good databases for only specific domains (e.g., pictorial books of only animals or plants), but they are not general.

<table>
<thead>
<tr>
<th>Method</th>
<th>AuPR</th>
<th>vs. SP-based</th>
<th>vs. PI-based</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP-based</td>
<td>0.4984</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PI-based ((n=17))</td>
<td>0.5570</td>
<td>+0.0586(+11.75%)</td>
<td></td>
</tr>
<tr>
<td>PIC-based ((n=17, \beta=0.40))</td>
<td>0.5643</td>
<td>+0.0659(+13.22%)</td>
<td>+0.0073(+1.31%)</td>
</tr>
<tr>
<td>PIA-based ((n=17, \gamma=0.45, m=1))</td>
<td>0.5641</td>
<td>+0.0657(+13.18%)</td>
<td>+0.0071(+1.28%)</td>
</tr>
<tr>
<td>PICA-based ((n=4, \beta=0.30, \gamma=0.90, m=11))</td>
<td>0.5792</td>
<td>+0.0808(+16.21%)</td>
<td>+0.0222(+3.99%)</td>
</tr>
</tbody>
</table>

Table 2. Comparison of my proposed hyponymy extractions with regard to area under precision-recall curve (AuPR).
Fig. 11. Average of area under precision-recall curve per $\gamma$ by Property Inheritance (PI) based vs. Property Inheritance and Aggregation (PIA) based hyponymy extractions.

By mining the Web as another approach, I try to construct much larger database for general objects from various domains. This section introduces my visual description extraction to extract typical appearance descriptions for each target object-name from the Web by using text mining techniques and concept hierarchy knowledge, i.e., hyponymy (is-a) and meronymy (has-a) relations (Hattori, Tezuka & Tanaka, 2007). Section 3.1 analyzes and models lexico-syntactic patterns of visual appearance descriptions. Section 3.2 describes my visual description extraction in detail. Section 3.3 shows several experimental results to evaluate my proposed visual description extraction.

3.1 Pattern Analysis and Modeling
There are the following lexico-syntactic patterns of appearance descriptions for a target object, which consisting of its concrete object-name (e.g., “kingfisher”), visual-modifiers, and class/component-names:

1. “(visual-modifier) (object-name)”
   - e.g., “beautiful kingfisher”
2. “(object-name) is/are (visual-modifier)”
   - e.g., “kingfisher is very pretty”
3. “(object-name) is a/an (visual-modifier) (class-name)”
   - e.g., “kingfisher is a small bird”
4. “(object-name) is a/an (class-name) with (visual-modifier) (component-name)”
   - e.g., “kingfisher is a bird with brilliantly colored feathers”
5. “(object-name) has/have (visual-modifier) (component-name)”
   - e.g., “kingfisher has a large beak”

I have formalized the most simplified model of an appearance description for a general object, as a triplet of the name of object itself, a name of its class or component, and a visual modifier describing its visual characteristic:

\[
\text{appearance} = (\text{object}, \ \text{visual-modifier}, \ \text{class}/\text{component}).
\] (23)

For example, a sentence “A kingfisher is a small bird.” can be simplified to one triplet of an object-name, a visual modifier, and a modified class-name of the object, (kingfisher, small, bird). For another example, a sentence “A kingfisher has a short blue tail and a long bill.” can be simplified to three triplets of an object-name, a visual modifier, and a modified component-name of the object, (kingfisher, short, tail), (kingfisher, blue, tail), and (kingfisher, long, bill). This model is based on the observation that an object is usually perceived as an aggregation of components with specific visual characteristics. If the visual modifier describes the whole of the object itself, such as in the case of “A kingfisher is colorful.”, then the class would be the name of the object itself, that is, the simplified triplet would be (kingfisher, colorful, kingfisher).
3.2 Method

I describe in detail a method to extract typical appearance descriptions for each target object-name by mining a very large corpus of documents such as the Web. The goal of my proposed method is to collect a set of pairs \((\text{visual } v_i, \text{ class/component } c_i)\) that correctly describes the appearance of the given object \(o\). I call these pairs, \(V-C\) pairs,

\[
o \Longrightarrow \{(v_1, c_1), (v_2, c_2), \ldots, (v_n, c_n)\}. \tag{24}
\]

Moreover, I also aim to rank these \(V-C\) pairs in the order of some weight \(w_i\), which indicates the suitability of each \(V-C\) pair as an appearance description of the target object \(o\),

\[
o \Longrightarrow \{(v_1, c_1, w_1), (v_2, c_2, w_2), \ldots, (v_n, c_n, w_n)\}. \tag{25}
\]

When a object-name \(o\) is given, my proposed method executes the following three steps and then outputs its typical appearance descriptions as several \(V-C\) pairs ordered by their weight.

**Step 1. Visual Modifier Dictionary Construction**

I have manually collected a set of visual modifiers as a basic data set for my proposed method. It consists of 617 words that describe color, shape, size, and surface material of objects. The composition of the set is shown in Table 3. The dictionary contains many (maybe too many) words referring to size, such as “short”, “long”, “small”, “big”, “high”, “low”, “many”, “much”, “few”, and “little”. Although they are important for describing the appearance in some occasions, they have also caused some noise in the results.
Step 2. Class/Component-Name Collection for Visual Modifiers

To collect the names of classes/components for a target object, first, my proposed method crawls Web pages described about only the target object, by submitting the name of the target object as a query to Google Web Search (2007) which is a conventional keyword-based Web search engine. To make the search results more accurate for the target object, my system retrieves only Web pages that contain the name of the target object in the title, by submitting not [“o”] but [intitle:“o”] as a query to Google Web Search. Henceforth, I use \( D(o) \) as the set of crawled documents relevant to each target object-name \( o \) obtained from \( D \), the set of all documents of a corpus such as the Web.

Next, the parser scans through the collected Web pages and finds phrases that contain a visual modifier in the dictionary. Words that immediately follow a visual modifier in the crawled Web documents are considered as candidates for class/component-names of the target object. There are, however, many irrelevant words on the candidate list. I apply the following ranking techniques to refine the results.
<table>
<thead>
<tr>
<th>Type</th>
<th>Number</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color</td>
<td>192</td>
<td>aeruginous, amber, amethyst, antique, apricot, ..., wheat, white, wine, wisteria, yellow</td>
</tr>
<tr>
<td>Shape</td>
<td>143</td>
<td>antisymmetric, aquiline, arc, asymmetric, ..., vertical, wavy, wedge-shaped, winding</td>
</tr>
<tr>
<td>Texture</td>
<td>119</td>
<td>abrasive, allover, argyle, banded, belted, ..., veined, velvety, watermarked, wet, zebra-stripe</td>
</tr>
<tr>
<td>Size</td>
<td>53</td>
<td>abundant, average, big, bold, brief, broad, ..., thick, thin, tiny tremendous, trivial, vast, wide</td>
</tr>
<tr>
<td>Surface material</td>
<td>110</td>
<td>acrylic, adobe, alloy, aluminum, asphalt, bamboo, ..., vinyl, waxy, wire, wood, wooden, woolly</td>
</tr>
<tr>
<td>Total</td>
<td>617</td>
<td></td>
</tr>
</tbody>
</table>

Table 3. Manually-constructed set of visual modifiers.

Step 3. V-C Pair Weighting

After obtaining a mixture of good and bad pairs of Visual modifier and Class/Component-name (V-C pairs), my system evaluates their weight to offer the users them ranked according to their significance for the target object. I present four kinds of weighting methods.

**Simple weighting** is a very simple approach that evaluates each weight of a V-C pair \((v_i, c_i)\) for the target object \(o\) by the number of Web documents in \(D(o)\) that contain the adjacent phrase “\(v_i\ c_i\)” (i.e., a visual modifier \(v_i\) immediately followed by a class/component \(c_i\)):

\[
\text{weight}^{\text{simple}}_o(v_i, c_i) := \text{df}_o(“v_i\ c_i”),
\]

where \(\text{df}_o([“p”])\) stands for the number of Web documents within \(D(o)\) that contain a phrase \(p\), i.e, by submitting \([\text{intitle:} “o” & “p”]\) as a query to Google Web Search. Because the weighting method considers a word that frequently appears after a visual modifier to be a class/component-name automatically (maybe too illogically), it is vulnerable to a compound word that starts with a visual modifier but is not an appearance description of the target object, e.g., “high school” and “yellow pages”. Therefore, the below weighting methods try to cope with this problem.

**Summation based weighting** is a more refined approach that evaluates each weight of a V-C pair \((v_i, c_i)\) for the target object \(o\) by multiplying the simple weight by the significance of a class/component \(c_i\) for the target object \(o\):

\[
\text{weight}^{\text{sum}}_o(v_i, c_i) := \text{weight}^{\text{simple}}_o(v_i, c_i) \cdot \text{weight}_o(c_i),
\]

\[
\text{weight}_o(c_i) := \sum_{v_j \in V} f_o(v_j, c_i),
\]
\[ f_o (v_j, c_i) := \begin{cases} 1 & \text{if } df_o (["v_j c_i"]) > t^{sum}, \\ 0 & \text{otherwise.} \end{cases} \]  

where \( t^{sum} \) stands for a threshold value, which I set to 1. \( f_o (v_j, c_i) \) is a boolean function that indicates whether or not there is a meaningful co-occurrence as the phrase “\( v_j c_i \)” in the crawled Web documents \( D(o) \) for the target object \( o \), and \( \text{weight}_o (c_i) \) means that the number of variations of visual modifiers that have a meaningful co-occurrence with a candidate class/component \( c_i \) of the target object in \( D(o) \).

**Syntactic Pattern based weighting** is to filter the above-mentioned problematical V-C pairs by using a lexico-syntactic pattern “\( c_i \) is/are \( v_i \)”:

\[ \text{weight}_{sp}^o (v_i, c_i) := \begin{cases} \text{weight}_{sum}^o (v_i, c_i) & \text{if } \frac{df(["c_i is/are v_i"])}{df(["v_i c_i"])} > t^{sp}, \\ 0 & \text{otherwise.} \end{cases} \]  

where \( t^{sp} \) stands for a threshold value, which I set to \( 10^{-4} \), and \( df(["p"] \) stands for the number of documents that contains the phrase \( p \) within \( D \), the set of all the documents of a corpus, not just the sampled documents \( D(o) \) (i.e., by submitting \(["p"] \) to Google Web Search). The formula is based on an observation for a set phrase “\( v_i c_i \)” that \( df(["v c_i"]) \) is too great but \( df(["c_i is/are v_i"]) \) is too small or nearly equal to 0. For example, in the case of (kingfisher, red, legs), both “legs are red” and “red legs” appear with a high frequency in all the documents and maybe also in \( D("kingfisher") \). Therefore, this V-C pair (red, legs) is considered as an appearance description in general. On the other hand, although the phrase “high school” appears at a certain high rate in \( D("kingfisher") \), this V-C pair (high, school) is not considered as an appearance description in general because of low frequency of the phrase “school is high”, and thus should not always be considered as an appearance description of the target object “kingfisher”.

**Concept Hierarchy based weighting** is to restrict \( c_i \) of a V-C pair \((v_i, c_i)\) to the target object-name \( o \), its class-name (hyponym) or component-name (meronym) by utilizing concept hierarchy knowledge:

\[ \text{weight}_{ch}^o (v_i, c_i) := \begin{cases} \text{weight}_{simple}^o (v_i, c_i) & \text{if } c_i \text{ is the object } o \text{, its class or component,} \\ 0 & \text{otherwise.} \end{cases} \]  

3.3 Experiment

I show several experimental results to evaluate my proposed method of extracting typical appearance descriptions for a target object-name from the Web. I performed experiments on a set of 20 kinds of target object-names, consisting of four typical categories that the users might encounter in their daily lives. Each set has five objects as follows.

**Birds:** “Jungle Myna”, “Kingfisher”, “Shoebill”, “Snowy Owl”, and “Sun Conure”.

**Flowers:** “Edelweiss”, “Japanese Cherry”, “Lavender”, “Lily of the Valley”, and “Sunflower”.

**Products:** “InterCityExpress”, “PS3”, “TGV”, “ThinkPad”, and “Wii”.

**Landmarks:** “Big Ben”, “Leaning Tower of Pisa”, “Statue of Liberty”, “Taj Mahal”, and “Tokyo Tower”.

www.intechopen.com
I applied the above-mentioned four kinds of weighting methods to obtain typical appearance descriptions appropriate for the 20 kinds of target object-names. Fig. 15 to 18 compares the four kinds of weighting methods with regard to the top $k$ average precision for each category. These graphs illustrate that the Concept Hierarchy (CH) based weighting is substantially better than the others. Note that the top $k$ average precision for products is much worse than the other categories. Last, Fig. 19 compares the four kinds of weighting methods with regard to the top $k$ average precision in the total for all categories. The graph also illustrates that the Concept Hierarchy (CH) based weighting is substantially better than the others and keeps about 0.70 average precision in the top 10 ranks. So, concept hierarchy knowledge (e.g., extracted from the Web in Section 2) is very useful for appearance description extraction from the Web as well as the other natural language processing systems.

As one future direction, I will have to evaluate the recall as well as the precision of my proposed weighting methods. Because only the top $k$ average precision of the Syntactic Pattern (SP) based weighting becomes obviously lower when $k$ becomes greater, the SP-based weighting seems to be unable to achieve higher recall than the other weighting methods.

**4. Application: Sense-based Object-name Search**

This section introduces my Sense-based Object-name Search (SOS) to enable users to identify the concrete name of a target object which they do not know only by inputting its hyponym (class-name) and some sensory descriptions, as an application system to utilize the Web-extracted semantic and sensory knowledge. When we move around in the real world such as town or nature, we usually encounter various real objects (e.g., products/buildings or animals/plants) and want information about some among them. In recent years, with the advances in mobile/ubiquitous computing devices and mobile Web search engines, we have become able to search the Web for information anywhere at any time in our daily lives.
When we search the Web for information about a target object by submitting a keyword-based query to such a conventional Web search engine as Google and Yahoo!, the precision and recall of the search results depend a great deal on whether or not we have known exactly the concrete name of the target object. If we know the concrete name of the target object, we could get not bad search results only by submitting it without any modification to a mobile Web search engine. However, we do not always know the concrete name of any target object that we have encountered in the real world and wanted information about. If we do not know the concrete name of the target object, we have no alternative but to compose a keyword-based query by its ambiguous features except its concrete object-name. For example, its class-name (i.e., hypernym), its visual appearance (e.g., color or shape), and its spatio-temporal information (e.g., place or season). And we could not get enough good search results only by submitting its class-name, visual appearance and/or spatio-temporal descriptions to the same mobile Web search engine.

Let’s suppose that two users encounter a beautiful bird at a riverside and want to search the Web for information about the target bird (shown in Fig. 20). If one of them knows exactly “Kingfisher” which is a concrete name of the bird, she could get good (possibly not bad) search results only by submitting a concrete query [“Kingfisher”]. Meanwhile, if the other user does not know the concrete name of the bird, he might have no alternative but to compose an ambiguous query [“blue bird”] by its class-name “bird” and its visual appearance “blue”. And he would get too bad search results which rarely include information about the target bird named by “Kingfisher” only by submitting the ambiguous query, because there are various kinds of not only blue birds but also objects except “Kingfisher”. The precision of our keyword-based Web search results depends a great deal on whether or not we have known exactly the concrete name of a target object.

Therefore, for a mobile user trying to search the Web for information about the target object whose concrete name s/he has not known, an object-name search that helps her/him to identify the concrete name of the target object by its ambiguous features is one of the most essential services for mobile/ubiquitous Web searches. Meanwhile, there have been many researches...
on Generic Object Recognition that recognizes a target object in its photographs with its category name (Duygulu et al., 2002), and the accuracy has improved greatly. But it is still very difficult to recognize the target object with its concrete name. I have been developing an object-name search system by a class-name, visual appearance and/or spatio-temporal descriptions (Hattori & Tanaka, 2009), as the first step to Sense-based on Object-name Search (SOS). When the user inputs a class-name, visual appearance and/or real-world context descriptions, my proposed system executes the following three steps to return not only concrete object-names ranked by her/his specification but also their typical images, visual appearance and spatio-temporal descriptions. First, my system converts from the user-given class-name to its concrete object-names by using Web-extracted semantic knowledge in Section 2. Second, each concrete object-name is assigned some kind of weight to, which is calculated by co-occurrence frequency of the object-name with the user-given descriptions in the Web. Next, sensory knowledge such as typical appearance descriptions of concrete object-names ranked by their weight is extracted from the Web in Section 3. And then the user can also modify her/his original specification repeatedly by using their typical features as a useful reference. Finally, the user could identify more concrete name of the target object by its ambiguous features, and get better search results by submitting the concrete object-name as a keyword-based Web query than by submitting its ambiguous features.

5. Conclusion

Automatic knowledge extraction from such a very large document corpus as the Web is one of the hottest research topics in the domain of Artificial Intelligence and Database technologies. This chapter has introduced my object-oriented and the existing methods to extract semantic (e.g., hyponymy and meronymy) and sensory (e.g., visual and aural) knowledge from the Web, and compares them by showing several experimental results. My object-oriented semantic knowledge extraction is based on property inheritance(s) and property aggregation, and repeatedly improves the extracted results of both hyponymy and meronymy relations. Meanwhile, my object-oriented sensory knowledge extraction is improved by utilizing the ex-
Fig. 20. Sense-based object-name search by a class-name, visual appearance and/or spatio-temporal descriptions.

extracted hyponymy and meronymy relations. The effectiveness of my proposed object-oriented approach for object-knowledge extractions from the Web has been shown by several experimental results. Finally, this chapter has introduced my Sense-based Object-name Search (SOS) to enable users to identify the concrete name of a target object which they do not know only by inputting its hyponym (class-name) and some sensory descriptions, as an application system to utilize the Web-extracted semantic and sensory knowledge.

6. References


This book presents a unique and diversified collection of research work ranging from controlling the activities in virtual world to optimization of productivity in games, from collaborative recommendations to populate an open computational environment with autonomous hypothetical reasoning, and from dynamic health portal to measuring information quality, correctness, and readability from the web.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

© 2010 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License, which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited and derivative works building on this content are distributed under the same license.