2005-07-01

Categorizing and Extracting Information from Multilingual HTML Documents

Yiu-Kai D. Ng
ng@cs.byu.edu

SeungJin Lim

Original Publication Citation

SeungJin Lim and Yiu-Kai Ng, "Categorization and Information Extraction of Multilingual HTML Documents." In Proceedings of the 9th International Database Engineering and Application Symposium (IDEAS'5), IEEE Computer Society, pp. 415-422, July 25, Montreal, Canada.

Categorizing and Extracting Information from Multilingual HTML Documents

SeungJin Lim
Department of Computer Science
Utah State University
Logan, Utah 84322, U.S.A.
Email: lim@cc.usu.edu

Yiu-Kai Ng
Computer Science Department
Brigham Young University
Provo, Utah 84602, U.S.A.
Email: ng@cs.byu.edu

Abstract

The amount of online information written in different natural languages and the number of non-English speaking Internet users have been increasing tremendously during the past decade. In order to provide high-performance access of multilingual information on the Internet, we have developed a data analysis and querying system (DatAQs) that (i) analyzes, identifies, and categorizes languages used in HTML documents, (ii) extracts information from HTML documents of interest written in different languages, (iii) allows the user to submit queries for retrieving extracted information in the same natural language provided by the query engine of DatAQs using a menu-driven user interface, and (iv) processes the user's queries (as Boolean expressions) to generate the results. DatAQs extracts information from HTML documents that belong to various data-rich, narrow-in-breadth *application domains, such as car ads, house rentals, job ads, stocks, university catalogs, etc. The average F-measure on identifying HTML documents written in a particular natural language correctly is 89%, whereas the F-measure on categorizing HTML documents belonged to the car-ads application domain is 94%.*

1 Introduction

During the past few years, the number of Internet users has increased tremendously, and non-English-speaking Internet users have outnumbered their English-speaking counterparts.[1] As a result, it is anticipated that the demand on accessing Web documents written in different (natural) languages increases proportionally. In this paper, we propose a data analysis and querying system (DatAQs) that (i) identifies the languages (i.e., encodings) in which HTML documents are written, (ii) extracts data of interest from the documents and categorizes them into different application domains using the Vector Space Model (VSM), and (iii) allows the user to submit a Boolean query interactively through an easy-to-use, menu-driven user interface to retrieve extracted data. Experimental results have demonstrated that the proposed system accomplishes the task with high accuracy.

DatAQs identifies the encoding of an HTML document using `charset`, UTF-8 encoding, and/or country code. The content of an HTML document is categorized by using VSM, a widely used information retrieval (IR) model. The proposed system query interface is easy to use and does not require the user to have extensive knowledge in formulating structured queries, such as SQL-based queries [8].

One may argue that existing commercial search engines, such as Google, Yahoo, and MSN Search, are adequate in processing multilingual Web queries. Queries generated by these search engines, however, are keyword-based, i.e., queries are created by using either a single or multiple keywords. Answers to these queries are Web documents in different languages; a portion of them might contain information relevant to the user's queries, but the user is required to scan through the retrieved documents to look for information of interest, which can be a tedious process. We handle this problem by (i) constructing a multilingual query engine that allows the user to submit (advanced) queries in different languages, such as English, Chinese, German, Japanese, and French, and (ii) processing the user's queries using data extracted from HTML documents written in different languages to generate answers to the queries. We succeed in extracting desired data, since our language analysis tool has high success rate (89%) and queries processed by our query engine come from application domains that include an abundant recognizable constants and information of common interest.

Compared with existing text categorization approaches that can be adopted for data extraction, we realize that machine-learning techniques are often used. [6] derive an automatic text-categorization approach using a number of instance-based learning and retrieval feedback techniques. [3] introduce a learning approach that integrate various per-

[1] See the statistical data compiled by Global Reach (http://global-reach.biz/globstats/) and others, such as China Internet Network Information Center (http://www.cnnic.net.cn), which support our claim.

Proceedings of the 9th International Database Engineering & Application Symposium (IDEAS'05)
1098-8068/05 $20.00 © 2005 **IEEE**

formance metrics into a text classifier. Instead of adopting the machine-learning paradigm, [5] propose a text categorization method based on distinct features, which are captured in the form of vectors, of each document class. [1] design different term association mining rules for constructing a classifier for text categorization. None of these categorization approaches, however, analyzes the encoding of a document for categorization and extracts information from categorized documents, which is our categorization approach that is simple and reliable.

We proceed to introduce DatAQs as follows. In Section 2, we present the detailed design of DatAQs, which includes components that analyze the encoding of Web documents (in Section 2.1), categorize HTML documents into different application domains based on the content of the documents (in Section 2.2), and process a user request formulated as a Boolean query through a query interface (in Section 2.3). In Section 3, we discuss the experimental results on categorization. In Section 4, we give a concluding remark.

2 Our Multilingual System

In this section, we present the overall design of the major components in DatAQs, which include (i) a *document processing subsystem* that identifies the encoding of a Web document and extract information of interest from HTML documents and (ii) a *query processing subsystem* that evaluates user queries and retrieves the answers (i.e., in the form of data records) to the queries.

The document processing subsystem of DatAQs is an off-line component of DatAQs. Each Web document D submitted to this subsystem is analyzed and its encoding is determined. The subsystem further determines whether D belongs to one of the pre-defined application domains. These application domains contain information of common interest to common users, such as car ads, job ads, house rental, etc. Note that we are not building a general-purpose multilingual search engine. Instead, DatAQs analyzes and categorizes, if possible, HTML documents to a number of predefined application domains and extracts information from the documents belonged to one of the domains. The other major component of DatAQs, the *query processing subsystem*, evaluates a user query (as a Boolean query) created by using our menu-driven user interface to retrieve the records generated from extracted information that satisfy the selection criteria as specified in the query.

2.1 Identifying languages of Web documents

In the past, various approaches in identifying languages in multilingual documents have been presented. [9] introduces a vector space based categorizer for language identification; however, the accuracy of the categorizer depends on the size of input documents and the set of languages being considered, which do not apply to our language identification approach. [10] identifies the languages of on-line documents based on either visual characteristics or document image (characters). Our language identifier is more general, since it does not rely on character classes.

In order to categorize Web documents written in different languages, we first attempt to identify in which language a document is written. It is not a trivial task to identify the language of a document by parsing the document, unless the document indicates the language explicitly. This is because (i) what we obtain from parsing a document is a sequence of *code points*[2] representing characters, and (ii) code points cannot be used as discriminators for different languages, since the same code point can be used in a number of languages, potentially in every language. For example, the two-byte code points `E1h CAh` are all valid in Chinese, Japanese, and Korean. Furthermore, `E1h` and `CAh` each is individually valid in (nearly) any language. In general, *information* on the code set used in writing a document is more useful than code points found in the document in determining the written language of the document, since as mentioned earlier each code set contains a set of characters that can be used in more than one language. For example, code set `ISO-8859-1`, which is often called *Latin-1*, includes characters used in a number of Western European languages. We determine the targeted language of a document mainly by using its code set information.

While creating an HTML document, the author can explicitly specify the character encoding of the entire body of the document (global encoding for short), as well as the character encoding of an individual tag (local encoding for short). The global encoding can be specified in the head of the document by inserting a `META` tag with the `http-equiv` and `content` attributes. In identifying the language of an HTML document D, we first consider the `charset` information in a `META` tag in D, if it is available. Detailed discussion on our language identification approach for determining the encoding of a Web document can be found in [4].

2.2 Categorizing documents using application domains

Using a number of pre-defined application domains, we analyze and categorize HTML documents written in the

[2] A code point is the assigned integer value to a character in a particular character set for electronic processing on a computer. For example, characters `A` and `Z` in the `Latin` character set are assigned the integer values `41h` and `5Ah` in hexadecimal in the `US-ASCII` code set (or "encoding system"), respectively. Furthermore, the code point of a lowercase character is `20h` larger than its uppercase counterpart in the same code set. Since there are different encoding systems, e.g., `UCS-2` and `EBCIDIC`, the same character often has different code points in different code sets.

most commonly used natural languages that have been detected by our language identification approach. According to Global Reach (http://global-reach.biz/globstats/), these languages include English, Chinese (traditional and simplified), Japanese, Spanish, German, Korean, French, Russian, and Arabic. The entire process of categorizing and extracting information from HTML documents involves

1. Specifying a conceptual-model domain for each application domain of interest (See Section 2.2.1).

2. Retrieving a list of HTML documents from the Web and identifying those documents that contain data elements specified in an application domain. (See details in Section 2.2.2.) We are interested in data-rich, narrow-in-breadth application domains (i.e., domains defined by abundant recognizable constants with small scopes, such as ads), and we have observed that HTML documents belonged to these application domains often include multiple records.[3] (See Figure 1(a) for an example of a multiple-record, car-ads document.) Many Web documents, such as car ads and obituaries, belong to this category.

3. Computing the vector for the application domain $\vec{AV}$ using the Vector Space Model according to the constraints of the application domain that specify the expected number of objects for each object set of interest. (See detailed discussion in Section 2.2.3.)

4. Using a particular application domain AD, we identify objects for each object set (i.e., specified items of interest such as names, prices, locations, etc.) of the application that often appear within a relevant HTML document D, and we count how many we have for each occurrence of objects (defined in each object set) in the document. We then compute the vector $\vec{DV}$ to represent the document. (See Section 2.2.4.) We measure the degree of relevance between $\vec{DV}$ and $\vec{AV}$ of AD using VSM to determine whether D is relevant to AD. (See Section 2.2.5.)

2.2.1 Representations of application domains

Prior to categorizing HTML documents into different application domains, we must first define the application domains. An application domain, which can be created by anyone who is knowledgeable in the domain and notations used in capturing constraints in the domain, is required to be defined in just one of the natural languages, which is then translated into other languages automatically by DatAQs.

[3] A document is a *multiple-record* document if it contains a sequence of similar chunks of information about the object sets, each of which contains a set of objects and each object is a constant such as '1999' as a year, '15000' as a price, etc., defined in an application domain [11].

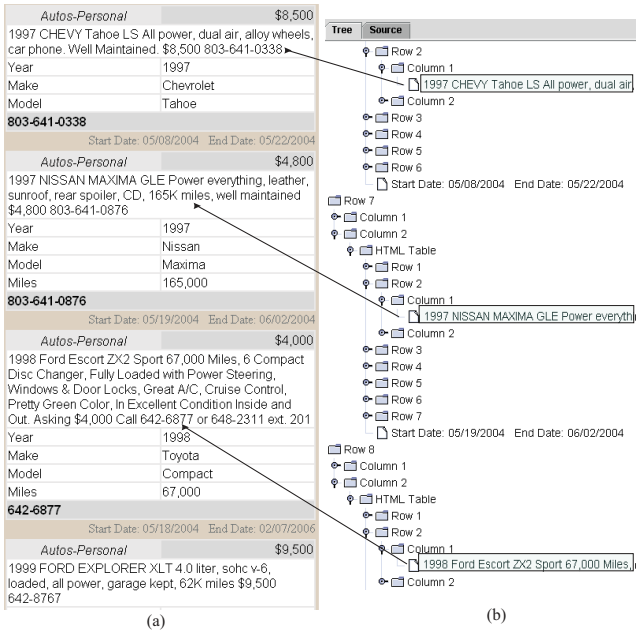
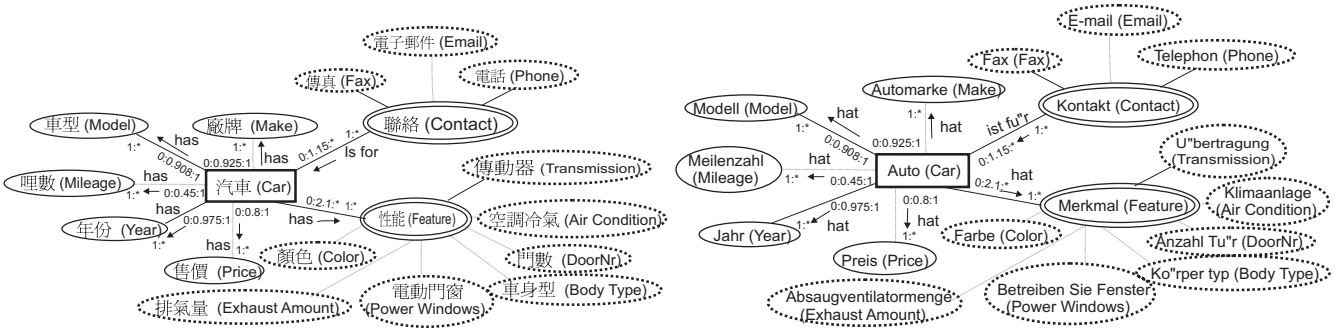


Figure 1. (a) Multiple car ads retrieved from http://classified.aikenstandard.com on May 19, 2004 and (b) its semantic hierarchy

For example, we first designed an English car-ads application domain, which was then translated into Chinese, German, and other natural languages. (See Figure 2 for an example.) In the graphical representation of an application domain, the rectangle denotes the application domain, a single-edge oval denotes an object set, a double-edge oval denotes a multiple-valued object set, and a dotted-edge oval denotes one of the possible object sets of the corresponding multiple-valued object set. Based on our past experience, we realize that it takes one or two dozen person-hours to define an application domain [11], which includes identifying the synonyms of objects and possible values that can be assigned to the objects. In the past, we have tried applications in car ads, obituaries, job ads, house rental, course listing, etc. The textual representation of an application domain are specified by using regular expressions in Perl.

Figure 2 shows the graphical representation of the car-ads application domain, whereas Figure 3 shows a portion of the *textual* representation of the domain. The textual representation includes the declaration of object sets, relationship sets, and cardinality constraints of objects (between lines 1 and 8 in Figure 3), in addition to a number of lines for defining the data frames that define the constant values and/or synonyms of objects (between lines 9 and 41). The name of an object, which is denoted by either a single-edge or double-edge oval in a graphical representation, is treated as an *index term* in the vector representation of an application domain or an HTML document. (See, as an exam-



(a) Car-ads application domain in Chinese (Traditional)

(b) Car-ads application domain in German

Figure 2. The graphical representations of the car-ads application domain in Chinese (Traditional) and German

```
1. Car [-> object];
2. Car [0:0.975:1] has Year [1:*];
3. Car [0:0.925:1] has Make [1:*];
4. Car [0:0.908:1] has Model [1:*];
5. Car [0:0.45:1] has Mileage [1:*];
6. Car [0:2.1:*] has Feature [1:*];
7. Car [0:0.8:1] has Price [1:*];
8. Contact [1:*] is for Car [1:1.15:*];
9. Year matches [4]
10.   constant { extract "\d{2}";
11.     context "([^\$\d]|^)[4-9]\d[^,\dkK]";
12.     substitute "^" -> "19";
13.     },
14.   { extract "\d{2}";
15.   context "([^\$\d]|^)[4-9]\d,[^\d]";
16.   substitute "^" -> "19";
17.   },
18.   { extract "\d{2}";
19.    context "\b'[4-9]\d\b";
20.    substitute "^" -> "19";
21.   },
22.   { extract "\d{4}";
23.    context "([^\$\d]|^)(19[4-9]\d|200\d)[^,\dkK]";
24.   },
25.   { extract "\d{4}";
26.   context "([^\$\d]|^)(19[4-9]\d|200\d),[^\d]";
27.   };
28. end;
29. Make matches [10]
30.   constant { extract "\bacura\b"; },
31.            { extract "\balfa(\s*romeo)?\b"; },
32.            { extract "\bamc\b"; },
33.            { extract "\bam\s*general\b"; },
34.            { extract "\baudi\b"; },
35.            ...
36. end;
37. ...
38. Mileage matches [8]
39. ...
40. keyword  "\bmiles\b", "\bmi[\.]\b", "\bmileage\b";
41. end;
42. ...}
```

Figure 3. The textual representation of the car-ads application domain

ple, the index term *Make* in the graphical representation in Figure 2 or in the textual representation with a `constant` declaration on lines 29-36 in Figure 3.)

A line connecting boxes in the graphical representation of an application domain denotes a *relationship set* among the linked object sets. A labeled reading-direction arrow next to a relationship-set line, along with the names of the connected object sets, yields the name of the relationship set. *Car has Model* in Figure 2, for example, names the relationship between object sets *Car* and *Model*. In the textual representation, we use this name to represent both the relationship set and the connected object sets. For example, line 4 in Figure 3 defines the relationship set between *Car* and *Model*.

The `min:max` or `min:ave:max` constraint specified next to the connection from object set O_1 to object set O_2 in a graphical representation is the *participation constraint* of O_1 in a relationship set R with O_2. `Min`, `ave`, and `max`, denote the minimum, average, and maximum number of times an object in O_1 participating in R with O_2, respectively, and '*', one of the possible maximum values, designates an unknown but finite maximum number of times an object in an object set can participate in a relationship set. Participation constraints are captured in the textual representation of an application domain as well. In the textual representation of the car-ads domain, the *participation constraints* are listed from line 2 to line 8 in Figure 3. The participation constraint [0:0.975:1] on *Car has Year*, as shown on line 2 in Figure 3, specifies that a car ad should have at most one occurrence of the manufacture year, but it could have none, and on the average 97.5% of car ads have a manufacture year. An application domain designer can obtain participation constraints of an application domain by examining relevant (Web) documents extracted from different different sources, such as different regions covering the United States. From each of these documents one can simply count by hand and determine the minimum, average, and maximum values for each object set in each relationship set and normalized the values for the application domain being designed.

In an application domain, objects (i.e., *constants*) in an object set are expressed by regular expressions in its textual representation, each of which is embedded by two double quotes. Each of these regular expressions (in Perl) are prefixed by the string `extract` and is located after the label `constant`. Furthermore, the values of *context keywords*, which are used for detecting the existence of correspond-

ing relevant constants, in the data frame of an object set are also expressed by regular expressions, each of which is also inserted between two double quotes. Different notations in a regular expression are used to define the values of a constant or a context keyword. For example, the expression "\bacura\b" (on line 30 in Figure 3) defines the constant string "$acura$," which stands for *Acura*, a car make. The expression "\bmiles\b" (on line 40 in Figure 3) defines one of the context keywords of *Mileage*.

2.2.2 Using IR models and application domains for document categorization

According to [2], the most significant problem in information retrieval is to find the set of relevant documents, which are sought, amongst a large collection of documents, by the user. In order to achieve high recall and precision ratios [4] on retrieving multilingual HTML documents relevant to a particular application domain AD, we (i) represent each HTML document D by selected constants as defined in AD that appear in D and (ii) determine the relevance of D to AD using the Vector Space Model (VSM).

In VSM a document D (an application domain AD, respectively) can be represented as an n-dimensional vector using the selected n (index) terms whose numbers of occurrences of constants defined in the corresponding object sets in D (AD, respectively) yield the coefficients of the vector. We define a *term* in D as an object set. In our categorization approach, the *weight* of a term (i.e., an object set) in the document vector ($\vec{DV}$) of D is determined by the *frequencies* of its corresponding constants appeared in D. The weight of a term in $\vec{DV}$ is independent of the weights of other terms in $\vec{DV}$. Selected terms of D must be carefully chosen to represent D using the *term weighting system*. We adopt the *term frequency* of a term as the *term weight* in D. VSM can be adopted to rank a given set of HTML documents according to various degrees of similarities among the documents and a particular application domain, which is also represented as an n-dimensional vector structure, the same type of vector of a matching document.

2.2.3 Application domain vector ($\vec{AV}$) construction

Given a set of HTML documents, we proceed to find out which documents in the set belong to which application domain of interest, or belong to none. Given an application domain AD, we know the terms and their corresponding weights by using the predefined constants in the domain. Using these terms and their weights, we are able to construct the vector $\vec{AV}$ of AD. The construction process of $\vec{AV}$ includes identifying (i) the names of coefficients of $\vec{AV}$, which are index terms (i.e., object sets) in AD, and (ii) the coefficients of $\vec{AV}$. We determine the *coefficients* of $\vec{AV}$ by using the *participation constraints* of the object sets in AD to which constants belong. If the *average* participation constraint of index term t exists, then the coefficient of t is the average. Otherwise, if max is not '*', then the coefficient of t is max; by default, the coefficient of t is min.

Consider the car-ads application domain file in Figure 3 again. The constraint "Car[0: 0.925:1] has $Make$[1:*]" on line 3 indicates the anticipated occurrence of "Make" in a car ad. Since the *average* is 0.925, it is expected that 92.5% of car ads include the `make` of a car. The vector $\vec{AV}$ of the car-ads application domain is constructed according to all the participation constraints defined in the car-ads application domain, and its components include <0.975, 0.925, 0.908, 0.45, 2.1, 0.8, 1.15> which indicates the number of occurrence of *Year, Make, Model, Mileage, Feature, Price, and Contact*, respectively, in a car ad.

2.2.4 Document vector ($\vec{DV}$) construction

After defining the domain vector $\vec{AV}$ for a particular application domain AD, which is constructed only once, we construct the document vector $\vec{DV}$ to represent each HTML document D and determine whether D is relevant to AD. The order of the coefficients in $\vec{DV}$ corresponds to the order of the coefficients in $\vec{AV}$. We assign each coefficient of $\vec{DV}$ according to the weight of its corresponding object set, which is determined by the number of occurrences of the objects belonged to the object sets appeared in D, the object sets (i.e., index terms) as specified in $\vec{AV}$, and compute the normalized coefficients of $\vec{DV}$. As discussed earlier, HTML documents belonged to application domains that we are interested in often include multiple records, which occur abundantly on the Web. In general, the number of occurrences of constants defined in an index term (i.e., object set) in each of these HTML documents is closely related to the size of the document, since the occurrences of the index term increase as the number of records in the document increases, which is the *normalization factor* in IR. In order to remove the size dependency in the document, we *normalize* the number of occurrences of each index term in the document by the estimated number of records in the document. Thus, for the car-ads application domain, if the numbers of occurrences of (the constant values of) *Year*, *Make*, *Model*, *Mileage*, *Feature*, *Price*, and *Contact* in an HTML document D are 22, 4, 12, 2, 7, 14, and 84, respectively, then the corresponding vector of D is $\vec{DV}$ = <22, 4, 12, 2, 7, 14, 84>. This vector should be normalized by the estimated number of car ads (i.e., records) in D. The estimated number of car ads is calculated by dividing the *length* of the document vector of D, (i.e., $|\vec{DV}|$) by the *length* of the car-ads domain vector $\vec{AV}$ (i.e., $|\vec{AV}|$). For the car-ads application domain, the corresponding $\vec{AV}$ is <0.975, 0.925,

[4] *Precision* is the percentage of retrieved documents that are relevant, whereas *recall* is the percentage of relevant documents that are retrieved.

0.908, 0.45, 2.1, 0.8, 1.15>, which are the coefficients of the index terms *Year, Make, Model, Mileage, Feature, Price*, and *Contact* as shown in Section 2.2.3. Since $|\vec{AV}| = \sqrt{0.975^2 + 0.925^2 + 0.908^2 + 0.45^2 + 2.1^2 + 0.8^2 + 1.15^2} = 3.03$, and $|\vec{DV}| = \sqrt{22^2 + 4^2 + 12^2 + 2^2 + 7^2 + 14^2 + 84^2} = 89.16$, the normalized vector of D is $\vec{DV}_{\text{norm}} = \frac{<22,4,12,2,7,14,84>}{89.16/3.03}$ = <0.75, 0.13, 0.41, 0.07, 0.24, 0.48, 2.85>.

2.2.5 Document relevance

We determine the relevance of a Web document D to an application domain AD by measuring the similarity of their corresponding vectors $\vec{DV}_{\text{norm}}$ and $\vec{AV}$, which is defined as $sim(D, AD) = \vec{DV}_{\text{norm}} \bullet \vec{AV} / (|\vec{DV}_{\text{norm}}| \times |\vec{AV}|)$, where $\bullet$ is the dot product operation. A threshold for $sim(D, AD)$, which is determined by a set of training and another set of test documents, is adopted to determine the "acceptance" similarity, i.e., relevance, between D and AD. For car-ads documents, the threshold is set to be 0.83 using thousands of training and text Web documents.

2.2.6 Record extraction

Having determined a multiple-record HTML document that is relevant to a particular application domain, we use the semantic hierarchy construction tool proposed in [7] for extracting information from the document. (See, as an example, a portion of the semantic hierarchy, as shown in Figure 1(b), of the multiple car-ads HTML document, as shown in Figure 1(a).) The construction tool of semantic hierarchies, which present only the hierarchies of data, not tags, in HTML documents, is fully automated [7]. Data components of each record retrieved from an HTML document are stored in a record in the database of DatAQs.

2.3 Query processing subsystem

The processor of our query processing subsystem is an on-line component of our multilingual system, DatAQs, which handles the user's queries in real time. In the query processing subsystem, the query processor handles user queries that are formulated as Boolean queries. The front-end query interface is a menu-driven user interface that allows the user to (i) specify the natural language in which an input query and its retrieved results should be written in and (ii) choose the category (e.g., car ads, job ads, house rentals, obituary, etc.) to look up information of interest.

2.3.1 Query formulation

Records extracted by DatAQs from HTML documents can be retrieved by the user through the user interface of DatAQs, which accepts the user's inputs and transforms the

<Query> := <Logical_Expre>
<Logical_Expre> := <Logical_Expre> 異或 <Logical_Expre> |
<Logical_Expre> 或 <Logical_Expre> |
<Logical_Expre> 和 <Logical_Expre> |
(<Logical_Expre>) | Predicate | 非 Predicate
Predicate := Attribute Comparison_op Value
Attribute := [\u4E00 - \u9FFF]+
Comparison_op := < | ≤ | = | ≠ | ≥ | >
Values := [\u4E00 - \u9FFF]+ | ([a – z]+( )*[0 – 9]*)+ | ([a – z]*( )*[0 – 9]+)+ |
[0 – 9]+.[0 – 9]+ [\u4E00 - \u9FFF]+

Notation:
| : OR; + : one or more; * : zero or more; [-]: range; "非": NOT; "和": AND; "或": OR; "異或": XOR; [\u4E00 - \u9FFF]: Range of Chinese characters in Unicode

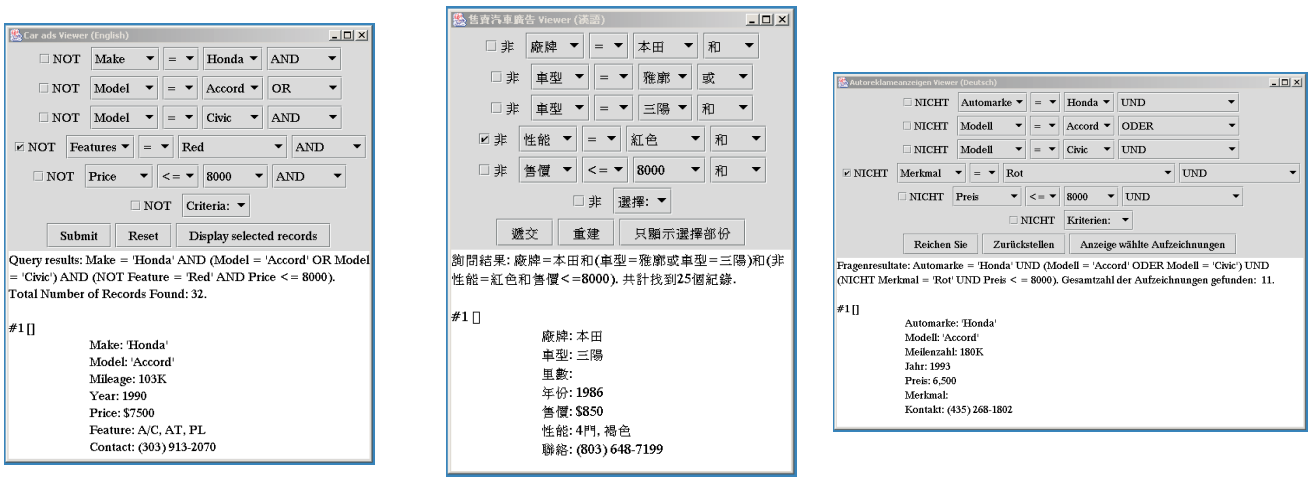
Figure 4. The BNF notation of the (Traditional) Chinese query language

inputs into a Boolean query using the language that the user has chosen earlier, since the same Boolean query language is available in multiple natural languages. For example, consider the Boolean query language specified in Chinese in Figure 4. The German version of the language can be constructed by replacing 異或 (i.e., XOR) by XODER, 或 (i.e., OR) by ODER, 和 (i.e., AND) by UND, 非 (i.e., NOT) by NICHT, and $[\backslash u4E00 - \backslash u9FFF]^+$ by $[\backslash u0000 - \backslash u007F]^+$.

In DatAQs, each query interface includes a number of check boxes and sets of drop-down menus (see, as an example, our English query interface for the car-ads application domain in Figure 5(a).) The set of check boxes, which is specified under the leftmost column, allows the user to negate a search term (e.g., year $\neq$ 2002), and the next four sets of drop-down menus are used for (i) choosing attributes (i.e., index terms) as specified in the pre-defined application domain, (ii) selecting a comparison operation (i.e., $<$, $\leq$, $>$, $\geq$, =, or $\neq$), (iii) specifying an attribute value of the chosen attribute, and (iv) indicating logical operators "AND," "OR," and "XOR" (where "AND" is the default value) applied to predicate(s) constructed by the choices of (i), (ii), and (iii) specified earlier. The list of attribute values is determined dynamically based on the selected attribute. For example, in the car-ads application domain, if the attribute "Make" is selected, the corresponding list of values includes "Toyota," "Honda," "Ford," etc., as defined in the domain. Furthermore, our query engine combines predicates by attaching user-chosen logical operators in between and inserts parentheses into the query expression to dictate the evaluation order. If necessary, the user is inquired to determine the order of evaluating the AND, OR, and XOR operators.

2.3.2 Query interfaces

In each query interface of our query engine, there are two panels, the *query panel* that occupies the top half and the *answer panel* that occupies the bottom half of the interface window (see Figures 5(a)-5(c)). The *query panel* allows the user to create queries. Besides the check boxes and drop-



(a) Query Interface in English (b) Query Interface in Traditional Chinese (c) Query Interface in German

Figure 5. Car-ads interfaces in different languages of our multilingual querying system

down menus, the query panel also includes three buttons: *submit, reset*, and *display selected records*. These buttons allow the user to (i) submit a new query, (ii) clean up the inputs, which is either a current or previous query, appearing in the query panel, and (iii) display only selected data records retrieved by a previously evaluated query using the corresponding check box attached to the left of each record, respectively. The second panel, the *answer panel*, is designed for displaying (selected) query answers after the *submit* (*display select records*, respectively) button is clicked (after a query has been evaluated, respectively). At the top of the answer panel, the submitted query is displayed as a Boolean expression in its infix notation and the total number of records retrieved by the query is shown, which are followed by each of the retrieved record listed one after another as answers to a query. (See, as examples, the bottom half of Figures 5(a)-5(c).)

We have developed different query interfaces in different languages for different application domains. These query interfaces have the same components (i.e., check boxes and drop-down menus), and the layout of each component (i.e., its relative location in the interface) of each application domain written in different languages is the same. The differences among different query interfaces are the languages used and values specified under the drop-down menus that are determined by their corresponding application domains.

3 Experimental results of categorization

We evaluate the performance of our multilingual system, DatAQs, on categorizing HTML documents into various application domains by calculating the precision, recall, and F-measure ratios using a collection of training set and test set documents. The training set documents determined the threshold values to be set for assigning documents to various application domains, if possible, and verified the correctness of our categorization approach (i.e., our VSM, application domains, etc.), whereas the test set documents confirmed the appropriateness of threshold values and assisted us to make necessary adjustments to the categorization approach. In order to verify the robustness of our approach, we collected the training and test documents that are both "broad" and "narrow" in content. The content should be *broad* because we want the system to handle different kinds of documents. The content should be *narrow* because we would like to catch subtle differences between similar documents. In this section, we present our experimental results on extracting multiple car-ads records from Chinese HTML documents. We chose to discuss the performance of our categorization and record-extraction tool on Chinese car ads because (i) almost everyone needs a car for transportation and (ii) Chinese Internet users are the largest non-English speaking Internet user group, according to Global Reach (http://global-reach.biz/globstats/), and are the fastest growing Internet user group, according to the survey performed by China Internet Network Information Center (http://www.cnnic.net.cn).

3.1 Experimental results on Chinese car-ads application domain

We used 212 documents to form a training set and 211 documents to form a test set for evaluating the performance of our categorization approach on Chinese car-ads application domain. All the documents were retrieved from var-

Domain	*Correct*	*FP*	*FN*	*Prec*	*Rec*	*F-M*
Car-ads	68	2	4	97%	94%	95%
House ads	44	4	2	91%	96%	93%
Average				94%	95%	94%

F-M(easure) = 2 / ($\frac{1}{Prec(ision)} + \frac{1}{Rec(all)}$)

Table 1. Experimental results of information extraction on two application domains

ious Web sites located in Mainland China, Taiwan, Hong Kong, U.S.A., Canada, etc., on the Internet, after they have been classified as Chinese documents using our language identification approach. And both the training set and test set contain three types of documents—Chinese relevant car-ads documents, irrelevant car-ads documents, and closely related but irrelevant car-ads documents that include motorcycle for-sale ads, motor-boat ads, new car-model ads, etc. Out of the 211 documents in the test set, which includes car-ads and non-car-ads documents, 68 were correctly categorized as car-ads. The precision, recall, and F-measure on the Chinese car-ads application domain are 97%, 94%, and 95%, respectively (see Table 1).

3.2 Observations of the Chinese car-ads experimental results

In this experiment, we notice that f(alse)p(ositives), i.e., irrelevant documents that are treated as relevant documents, are caused by those closely related but irrelevant documents such as motorcycle sale, car rental, etc. Some of these documents are difficult to be distinguished from real car-ads documents because of closely related data values (such as "Honda" who makes both cars and motorcycles). Some of the f(alse)n(egatives), i.e., relevant documents that are treated as irrelevant documents, are caused by documents that include dealer ads for several dozen cars. This, by itself, is not a problem, but there are some complications that do cause problems. Some of these ads contain neither mileage information nor feature information about the cars. Missing mileage and feature information is generally not a problem, but since there are more cars mentioned in dealer ads than in all the rest of the car ads together, missing mileage and feature information adds up. Furthermore, the phone number is factored out of each individual ad but there is only one phone number for all dealership cars, which causes the relevant ratio used by our VSM to drop significantly.

4 Conclusions

We have proposed a data analysis and querying system, DatAQs, that can determine document languages with high accuracy and categorize HTML documents into different application domains, which include an abundant recognizable constants and information of common interest, with high precision and recall ratios. Data in categorized HTML documents are extracted and stored as records in an underlying database according to the (index) terms defined in the corresponding application domains and can be retrieved by user queries, which are created by using a menu-driven user interface available in different natural languages provided by DatAQs. We have conducted different experiments to evaluate the performance of DatAQs by categorizing multilingual HTML documents, which include English, Chinese, Japanese, German, French, etc., and extracting car-ads and others (e.g., house-rental) HTML documents written in Chinese. The average F-measures of our categorization approach and extracting car and house-rental ads from Chinese HTML documents are 89% and 94%, respectively. DatAQs has been implemented in Java using JPad Pro 4.5 and Google Web APIs on a Pentium 4 Windows XP PC.

References

[1] M. Antonie and O. Zaiane. Text Document Categorization by Term Association. In *Proceedings of IEEE Intl. Conf. on Data Mining*, pages 19–26, 2002.

[2] F. Crestani, M. Lalmas, and C. van Rijsbergen. *Information Retrieval: Uncertainty and Logics*. Kluwer Academic Publishers, Massachusetts, 1998.

[3] S. Gao, W. Wu, C.-H. Lee, and T.-S. Chua. A Maximal Figure-of-Merit Learning Approach to Text Categorization. In *Proceedings of ACM Conference on Research and Development in Information Retrieval*, pages 174–181, 2003.

[4] N. Gustafson, S. Lim, and Y.-K. Ng. Categorization of Web Documents Using Character Encodings. http://faculty.cs.byu.edu/˜dennis/papers.html, 2005.

[5] T. Kawatani. Topic Difference Factor Extraction Between Two Document Sets and Its Application to Text Categorization. In *Proceedings of ACM Conf. on Research & Development in Information Retrieval*, pages 137–144, 2002.

[6] W. Lam, M. Ruiz, and P. Srinivasan. Automatic Text Categorization and Its Application to Text Retrieval. *IEEE Trans. on Knowledge & Data Engineering*, 11(6):865–879, 1999.

[7] S. Lim and Y.-K. Ng. An Automated Change-Detection Algorithm for HTML Documents Based on Semantic Hierarchies. In *Proceedings of the 17th International Conference on Data Engineering (ICDE'01)*, pages 303–312, 2001.

[8] G. Mihaila. WebSQL - An SQL-like Query Language for the World Wide Web. Master's thesis, Univ. of Toronto, 1996.

[9] J. Prager. Linguini: Language Idenification for Multilingual Documents. *Management Info. Sys.*, 16(3):71–102, 2000.

[10] A. Spitz. Determination of the Script and Language Content of Document Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(3):235–245, 1997.

[11] Q. Wang and Y.-K. Ng. An Ontology-Based Binary-Categorization Approach for Recognizing Multiple-Record Web Documents Using a Probabilistic Retrieval Model. *Information Retrieval*, 6(3-4):295–332, December 2003.