Learning Query Languages of Web Interfaces

André Bergholz
Xerox Research Centre Europe
6 chemin de Maupertuis, 38240 Meylan, France
aberghol@xrce.xerox.com

Boris Chidlovskii
Xerox Research Centre Europe
6 chemin de Maupertuis, 38240 Meylan, France
chidlovskii@xrce.xerox.com

ABSTRACT

This paper studies the problem of automatic acquisition of the query languages supported by a Web information resource. We describe a system that automatically probes the search interface of a resource with a set of test queries and analyses the returned pages to recognize supported query operators. The automatic acquisition assumes the availability of the number of matches the resource returns for a submitted query. The match numbers are used to train a learning system and to generate classification rules that recognize the query operators supported by a provider and their syntactic encodings. These classification rules are employed during the automatic probing of new providers to determine query operators they support. We report on results of experiments with a set of real Web resources.

Keywords

Hidden Web, search interface, query operators, learning

1. INTRODUCTION

Searching for relevant information is a primary activity on the Web. Often, people search for information using general-purpose search engines, such as Google or Yahoo!, which collect and index billions of Web pages. However, there exists an important part of the Web that remains unavailable for centralized indexing. This so-called "hidden" part of the Web includes the content of local databases and document collections accessible through search interfaces offered by various small- and middle-sized Web sites, including company sites, university sites, media sites, etc. According to the study conducted by BrightPlanet in 2000 [6], the size of the Hidden Web is about 400 to 550 times larger than the commonly defined (or "Visible") World Wide Web. This surprising discovery has fed new research on collecting and organizing the Hidden Web resources [1, 2, 15, 17, 19].

Commercial approaches to the Hidden Web are usually in the shape of Yahoo!-like directories which organize local sites belonging to specific domains. Some important examples of such directories are InvisibleWeb[1] and BrightPlanet[2] whose gateway site, CompletePlanet[3], is a directory as well as a meta-search engine. For each database incorporated into its search, the meta-search engine is provided with a manually written "wrapper", a software component that specifies how to submit queries and extract query answers embedded into HTML-formatted result pages.

Similar to the Visible Web, search resources on the Hidden Web are highly heterogeneous. In particular, they use different document retrieval models, such as Boolean or vector-space models. They allow different operators for the query formulation and, moreover, the syntax of supported operators can vary from one site to another. Conventionally, query languages are determined manually; reading the help pages associated with a given search interface, probing the interface with sample queries and checking the result pages is often the method of choice.

The manual acquisition of Web search interfaces has important drawbacks. First, the manual approach is hardly scalable to thousands of search resources that compose the Hidden Web. Second, the manual testing of Web resources with probe queries is often error-prone due to the inability to check results. Third, cases of incorrect or incomplete help pages are frequent. Operators that are actually supported by an engine may not be mentioned in the help pages, and conversely, help pages might mention operators that are not supported by the engine.

To overcome the shortcomings of the manual approach, we address the problem of acquiring the query languages of Web resources in an automatic manner. We develop a system that automatically probes a resource's search interface with a set of test queries and analyses the returned pages to recognize supported query operators. The automatic acquisition assumes the availability of the number of matches the resource returns for a submitted query. The match numbers are used to train a learning system and to generate classification rules that recognize the query operators supported by a provider and their syntactic encodings.

New technologies surrounding the XML syntax standard, in particular Web Services [18], establish a new basis for automatic discovery and information exchange and are becoming widely employed in corporate applications. However, this has yet to happen for thousands of public information providers. The question of when and how they will move toward open cooperation using Web Service technologies remains widely open [4]. Instead, the query-probing approach for acquiring supported operators does not assume any cooperation of Web providers; its only requirement is that they

SAC '04, March 14-17, 2004, Nicosia, Cyprus

provide an accessible interface and allow queries to be run.

This paper is organized as follows. In Section 2 we discuss the heterogeneity of Web interfaces; we formalize the problem and show its connection with the concept of learning by querying in Section 3. In Section 4 we design a classifier system for the automatic acquisition of a query language and investigate different aspects of the system. In Section 6 we review the prior art; in Section 5 we present experimental results to illustrate the performance of our system. Section 7 discusses open issues and Section 8 concludes the paper.

2. QUERYING WEB RESOURCES

Web resources vary considerably in the ways they retrieve relevant documents. In the theory of information retrieval, there exist at least five basic retrieval models, but only three of these models are visible on the Web, namely the Boolean, the extended Boolean and the vector-space models. In the Boolean query model, a query is a condition, which documents either do or do not satisfy, with the query result being a set of documents. In the vector-space model, a query is a list of terms, and documents are assigned a score according to how similar they are to the query. The query result is a ranked list of documents. A document in the query result might not contain all query terms. Finally, the extended Boolean model combines the advantages of both the Boolean and the vector-space query model. In this model, keywords can be preceded by special characters (like $'+'$ and $'-'$) requiring an obligatory presence or absence of a given keyword in a document. For example, the query $'+$information +provider$'$ will retrieve all documents containing both keywords and rank them according to some similarity function.

Analysis of information providers suggests that the majority of providers adopt one of the three basic models. Moreover, beyond query answers, many resources report the number of documents in their collections matching the query. If a resource deploys the (extended) Boolean model, the match number shows how many documents match the query. In the case of the vector-space model, the match number refers to documents containing at least one query term, thus being equivalent to the Boolean disjunction.

In the following, we develop an approach for automatic determination of query operators by reasoning on submitted queries and corresponding match numbers. Though this approach excludes resources that do not report match numbers, other ways of automatic detection of query operators appear even more problematic and difficult to implement. A method based on downloading answer documents and verifying the query against their content often fails, either for legal reasons, when the content of documents is unavailable or password-protected, or for technical reasons, when a query matches millions of documents and downloading even a part of them requires prohibitive time and network resources.

2.1 Query language model

A query language of a Web provider includes a set of basic operators and the way of combining the operators to get complex queries. Basic operators have different arities, in particular, the *default* term processing and the *unary* and *binary* operators. The default processing refers primarily to *case sensitivity* in this paper, but we could also refer to whether the query term is treated as a complete word or as a substring in a possible matching document. Unary operators include the *Stem*-operator, which replaces a query term with its lexem; binary operators include the Boolean operators $\wedge$ (conjunction), $\vee$ (disjunction), and $\neg$ (negation)[1] and the operator *Phrase* which requires the adjacency of all terms in a document.

Some other operators, like substring matching or word proximity operators have been studied in various systems, however the six query operators mentioned above are by far the ones most frequently supported by Web interfaces. In the following, we develop a method to cope with the operator set $O = \{Case, Stem, \wedge, \vee, \neg, Phrase\}$. Issues relevant to the possible extension of set O with other operators are delegated to Section 7.

2.2 Query interpretation

Web providers are queried by filling their search forms with *query strings*. CGI or JavaScript code linked to the query form *interprets* the query strings according to certain rules. These rules allow *syntactic encodings* for the supported query operators. If correctly interpreted, the query is executed on the document collection before a (full or partial) answer is reported to the user.

Unfortunately, the same query operator may be encoded differently by different providers. For example, the Boolean conjunction is often encoded as $'$A AND B$'$, $'$A B$'$, or $'+$A +B$'$, where A and B are query terms. Worse, two providers can interpret the same query string differently. For example, query string $'$A B$'$ can be interpreted as a Boolean conjunction, Boolean disjunction, or *Phrase*.

EXAMPLE 1. *To illustrate the problem, consider the query string $q = 'Casablanca\ AND\ Bogart'$. On Google, $'AND'$ is interpreted as the Boolean conjunction, that is, $i_{Google}('Casablanca\ AND\ Bogart') = 'Casablanca' \wedge 'Bogart'$. As a result, query q matches 24,500 pages at Google, as opposed to 551,000 for query $q_1 = 'Casablanca'$ and 263,000 for $q_2 = 'Bogart'$. On the Internet Movie Database (IMDB) (*`http://www.imdb.com/.`*), $'AND'$ is taken literally and all terms in a query are implicitly OR-connected. Therefore, the IMDB interprets query q as follows: $i_{IMDB}('Casablanca\ AND\ Bogart') = 'Casablanca' \vee 'AND' \vee 'Bogart'$. The query returns 12,020 matches documents on IMDB, as opposed to only 22 for $q_1 = 'Casablanca'$ and 4 for $q_2 = 'Bogart'$.*

If we investigate an unknown query language, then Example 1 shows that observing match numbers for probe queries can provide a good insight into the supported operators. However, no definitive decision appears possible from the three queries above q, q_1, q_2. An accurate decision on supported operators/syntaxes will require probing the provider with other queries and comparing all match numbers in order to confirm or reject various hypotheses.

EXAMPLE 2. *As in Example 1, let us compare match numbers for the queries $q = 'Casablanca\ AND\ Bogart'$, $q_1 = 'Casablanca'$, and $q_2 = 'Bogart'$. For Google, the fact that q matches less documents than any of q_1 and q_2, favors the Conjunction-hypotheses, but is still insufficient to exclude other hypotheses, like that of Phrase. Probing Google with query $q_3 = 'Bogart\ AND\ Casablanca'$ returns the same number of matched documents as q. This (most likely) discards the Phrase-hypothesis, but not the hypothesis $'Casablanca' \wedge 'AND' \wedge$*

[1]Negation is a binary operator in Web query languages and its interpretation is given by 'AND NOT', that is, A $\neg$ B is a synonym for A $\wedge\neg$ B (the latter using the unary $\neg$).

'Bogart'. To exclude this one, even more queries should be sent to Google, like q_4='Casablanca AND', and so on. Similarly in IMDB, the fact that query q matches more documents than q_1 and q_2 suggests that q is processed as a disjunction, but it can not tell whether 'AND' is taken literally or ignored. A deeper analysis requires further probing IMDB with, for example, queries q_4='Casablanca AND' or q_5='Casablanca Bogart' to compare their match numbers to the ones of previous queries and decide about the 'AND'.

Our approach to the automatic acquisition of Web query languages formalizes and generalizes the idea described in Examples 1 and 2. We build a learning system that trains a number of classifiers with data from manually annotated sites to automatically determine supported operators and their syntaxes at a new site. The training data from annotated sites includes an ensemble of *test queries* together with the corresponding match numbers.

3. PROBLEM DEFINITION

Assume an information provider P supports some or all query operators in O; these operators form a set O_P, $O_P \subseteq O$ and allow us to compose a set of complex queries $Q(O_P)$. For any operator $o_i \in O_P$, P accepts one or more syntactical encodings, s_{i1}, s_{i2}, ... The set $\{s_{ij}\}$ of accepted syntaxes for $o_i \in O_P$ is denoted S_i. The *interpretation* I_P of operator set O_P is defined as $I_P = \{(o_i, s_{ij}) | o_i \in O_P, s_{ij} \in S_i\} = \{(o_i, S_i) | o_i \in O_P\}$. Interpretation I_P is *monovalued* if each operator has at most one syntax, i.e, $|S_i| = 1$ for all $o_i \in O_P$. I_P is *multivalued*, if it allows multiple syntaxes for at least one operator, i.e., $\exists o_i \in O_P$ such that $|S_i| > 1$. In Google, the Boolean conjunction can be encoded by both 'AND' and '$\sqcup$' (whitespace). Therefore, for any query terms A and B, both query strings 'A B' and 'A AND B' are interpreted as A $\wedge$ B. I_{Google} contains ($\wedge$,'AND') and ($\wedge$, '$\sqcup$') and is a multivalued interpretation.

We distinguish between ambiguous and unambiguous interpretations. A pair of distinct operator encodings (o_i, s_{ij}) and (o_k, s_{kl}) is *ambiguous* if the two operators have the same syntax: $o_i \neq o_k$ but $s_{ij} = s_{kl}$. An interpretation I_P is ambiguous, if it contains at least one ambiguous pair of encodings. An interpretation I is *unambiguous*, if for any pair of encodings (o_i, s_{ij}) and (o_k, s_{kl}) in I, $o_i \neq o_k \Rightarrow s_{ij} \neq s_{kl}$.

Ambiguous interpretations can be observed with Web providers that interpret query strings dynamically, when the final decision depends on results of the query execution with different retrieval models[2]. However, the major part of Web providers interpret query strings unambiguously and our method copes with unambiguous interpretations only. Further discussion on ambiguous interpretations is in Section 7.

Like with the query operators, we select the most frequent syntaxes on the Web, S = { Default[3], '$*$', '$\sqcup$', 'AND', '+', 'OR', 'NOT', '$-$', '""' (quote marks)}. Like set O, these syntaxes have been selected after verification of hundreds of Web providers. Set S is easily extendable to alternative syntaxes, like ones employed by non-English providers. For example, French providers may use 'ET' for the Boolean conjunction and 'OU' for the disjunction.

The theoretical framework for the query language acquisition is derived from the learning of an unknown concept by querying [5]. Assume that provider P supports the basic operators in O; complex queries composed from the basic operators form a set $Q(O)$. For the document collection at P, query $q \in Q(O)$ constrains a subset $P(q)$ of documents matching q. An abstract query $q \in Q(O)$ is mapped into a textual string with a mapping $M : O \rightarrow 2^S$ that defines (possibly multiple) syntaxes for operators in O. The mapping of a complex query q is denoted $m(q)$, the set of mapped queries is denoted $Q(S) = Q(M(O))$.

The sets O and S are assumed to be known, whereas the mapping M is unknown. We are given an oracle that can be queried with a mapped query $m(q) \in Q(S)$ on the size of subset $P(q)$, $oracle(m(q)) = |P(q)|$. By observing the oracle's responses to queries, the learning system should produce a hypothesis on the mapping M, which should be as close as possible to the correct one.

The identification of the mapping M may be simple under certain circumstances. Below we show an example of reconstruction when O_P includes a particular subset of operators and the oracle is noiseless.

EXAMPLE 3. *Let O include the three Boolean operators ($\wedge, \vee$ and $\neg$) and Phrase. Then, for a given syntax set S, any unambiguous mapping $M : O \rightarrow 2^S$ can be exactly identified if the oracle is noise-less[4]. In such a case, subset sizes returned by the oracle fit the Boolean logic on sets.Indeed, when querying the oracle with terms A and B and syntaxes from S, the disjunction is distinguishable from other operators by the fact that it constrains bigger subsets in a collection than any of terms does:*

$$|A \vee B| \geq |A|, |A \vee B| \geq |B| \quad (1)$$

Furthermore, among three other operators, the conjunction is recognized by its commutativity:

$$|A \wedge B| = |B \wedge A| \quad (2)$$

Finally, the difference between negation and phrases is detected by the basic equation linking three Boolean operators:

$$|A \vee B| = |A \neg B| + |A \wedge B| + |B \neg A| \quad (3)$$

Sizes of subsets constrained by the Boolean operators satisfy the disequation (1) and equations (2), (3) for any pair of A and B, so one can easily design a learning system that exactly identifies an unambiguous mapping M after only a few probing queries.

Unfortunately, easy identification of the mapping M is rather an exception on the real Web, where few if any of the assumptions made in Example 3 become true. First, any change in the operator set O_p makes the exact reconstruction less obvious. If the conjunction and/or disjunction are not supported, then the size of A $\wedge$ B (or A $\vee$ B) is unavailable and equation (3) cannot help distinguish negation from phrases. In cases like this, the identification of supported syntaxes requires an analysis of the semantic correlation between query terms A and B and guessing on their co-occurrence in (unknown) document collections.

[2]Citeseer at `http://citeseer.nj.nec.com/cs` is an example of ambiguous interpretation. By default, it interprets 'A B' as a conjunction; however if A $\wedge$ B matches zero documents, the query is interpreted as disjunction.

[3]'Default' refers to the absence of any syntax; it assumes the processing of plain terms.

[4]Oracle noiseless assumes the pure Boolean logics, with no query preprocessing, like the stopword removal.

Second, Web query interfaces that play the role of oracles and return sizes of subsets constrained by queries $m(q) \in Q(S)$ are rarely noiseless. When probing interfaces with test queries, the match numbers may violate equations (2) and (3). Most violations happen because converting query strings into queries on collections hides the stop-word removal and term stemming. It is not clear, whether queries like $'$A AND B$'$ are interpreted as one (A is a stopword), two, or three terms. Moreover, for the performance reasons, real match numbers are often replaced by their estimations which are calculated using various collection statistics [13], without the real retrieval of documents matching the query.

4. LEARNING SYSTEM

To automatically determine supported query operators, we reduce the overall problem to a set of classification tasks, where each task is associated with recognizing a specific query operator or syntax, and where some standard learning algorithms like SVM, k-nearest neighbors or decision trees can be applied. To build the classifiers, we collect and annotate a set of Web providers. We develop a set of test queries and probe all selected providers with the test queries. We train the classifiers with query matches for test queries. For any new provider, we first probe it with the test queries. Query matches returned by the provider upon test queries are used to automatically classify operators and syntaxes and produce an unambiguous interpretation for P.

To achieve a good level of classification accuracy, we investigate different aspects of the learning system including the target function, probe queries, data preparation, and feature encoding and selection.

4.1 Target function

Due to the multivalued relationships between query operators and syntaxes, the target function for our learning system has two alternatives, one for the direct mapping M and the other one for the inverted mapping M^{-1}:

- T_1: $O \rightarrow 2^S$. T_1 targets the unknown mapping M; it assigns zero or more syntaxes to each operator in O. T_1 builds a multi-value classifier for every $o_i \in O$, or alternatively, a set of binary classifiers for all valid combinations (o_i, s_j), $o_i \in O, s_j \in S(o_i)$.
- T_2: $S \rightarrow O$. T_2 targets the inverted mapping M^{-1}; it assigns at most one operator to every syntax $s_j \in S$.

Either target function gets implemented as a set of classifiers, *operator classifiers* for T_1 or *syntax classifiers* for T_2. Classifiers are trained with match numbers for probe queries from annotated providers. For a new provider P, either function produces a hypothesis $I^T(P)$ that approximates the real interpretation I_P. The major difference between T_1 and T_2 is that the former can produce ambiguous interpretations, while the output of T_2 is always unambiguous. Indeed, two operator classifiers with T_1 can output the same syntax leading to ambiguity, while each classifier in T_2 outputs at most, one operator for one syntax. In experiments we tested both functions, though when building the learning system we put an emphasis on T_2, which is free of ambiguity.

To build syntax classifiers for the target function T_2, we should consider beyond "good" classification cases for the operators in O and include some "real-world" cases where providers process syntaxes in S literally or simply ignore them. For certain providers, it is difficult to find any valid interpretation. In the learning system, we extend the set of possible interpretations of syntaxes in S by three more cases, $O' = O \cup \{Ignored, Literal, Unknown\}$. Syntaxes in S have different alternatives for their interpretation; below we revisit some syntaxes and report possible matches in O' as they are specified in the learning system.

Default : Case sensitivity for query terms: possible values are case-insensitive (*Case*) or case-sensitive (*Literal*).

$'$*$'$: This unary operator can be interpreted as *Stem*, when $i(\text{A*}) = Stem(\text{A})$, *Ignored* when $i(\text{A*}) = i(\text{A})$, and *Literal*, when $'$A*$'$ is accepted as one term.

$'\sqcup'$: Whitespace is often a default for another syntax in S. Three possible interpretations include the Boolean conjunction when $i('\text{A B}') = \text{A} \wedge \text{B}$, the Boolean disjunction when $i('\text{A B}') = \text{A} \vee \text{B}$, and *Phrase* when $i('\text{A B}') = Phrase$ (A,B).

$'$AND$'$: Three alternatives here are the conjunction when $i('\text{A AND B}') = \text{A} \wedge \text{B}$, *Ignored*, when $'$AND$'$ is ignored and the interpretation goes with the whitespace meaning, $i('\text{A AND B}') = i('\text{A B}') = M^{-1}('\sqcup')$ (A, B), and *Literal* when $i('\text{A AND B}') = M^{-1}('\sqcup')$ (A,$'$AND$'$,B).

$'$“ “$'$ (Quote marks): Two possible interpretations are *Phrase*, when $i('\text{“A B“}') = Phrase(\text{A,B})$, and *Ignore* when quote marks are ignored and terms are interpreted with the whitespace, $i('\text{“A B“}') = i('\text{A B}') = M^{-1}('\sqcup')$ (A, B).

A similar analysis is done for the syntaxes $'+'$, $'$OR$'$, $'$NOT$'$ and $'-'$. Additionally, all syntaxes for binary operators can be labeled as *Unknown*.

4.2 Probing with test queries

To train syntax classifiers for T_2, we collect data from annotated sites by probing their interfaces and extracting the match numbers. Probing has a fairly low cost, but requires a certain policy when selecting terms for test queries to provide meaningful data for the learning. We define a set R of *model queries* that contain syntaxes in S and parameter terms A and B, which are later bound with real terms.

We form the set R by first selecting well-formed queries that contain all syntaxes we want to classify. Second, we add queries that are term permutations of previously selected queries, for example the permutation $'$B A$'$ for query $'$A B$'$. Finally, we add model queries that are not well-formed, but appear helpful for building accurate classification rules. Below, the set R of model queries is illustrated using the pair of terms A and B; model queries are split into three groups containing one, two or three words:

- One word queries: $'$A$'$,$'$B$'$,*UpperCase*(A), $'$A*$'$, *Stem*(A).
- Two word queries: $'$A B$'$,$'$B A$'$,$'$"A B"$'$,$'$"B A"$'$,$'$+A +B$'$, $'$+B +A$'$, $'$A -B$'$, $'$A AND$'$, $'$A OR$'$, $'$A NOT$'$.
- Three word queries: $'$A AND B$'$, $'$B AND A$'$, $'$A OR B$'$, $'$B OR A$'$, $'$A NOT B$'$, $'$B NOT A$'$.

In total, the set R is composed of 22 model queries, all in lower case, except *UpperCase* (A), which is an upper case of term A. Six queries in R are permutations of other queries

and three queries are (purposely) not well-formed. These queries 'A AND', 'A OR', 'A NOT' are selected to help detect *Literal*-cases for 'AND', 'OR', 'NOT'.

Probe queries are obtained from the model queries by replacing parameters A and B with specific query terms, like 'knowledge' and 'retrieval'. These 22 probe queries form a *probe package* denoted $R_{A,B}$. For a provider P, probe queries together with corresponding match numbers form the *elementary feature set* $F^0_{A,B} = \{(m(q_i), oracle(P(q_i))), w(q_i) \in R_{A,B}\}$. Query terms are selected from a generic English vocabulary with all standard stopwords excluded. One site can be probed with one or more probe packages, all packages using different term pairs (A,B).

To probe the sites with test queries, we bind model queries in R with query terms. To obtain meaningful training data, query terms should not be common stopwords, such as 'and' or 'the'. As the term co-occurrence in a provider's document collection is unknown, we select pairs with different degrees of semantic correlation. Here, the term pairs fall into three categories:

- C_1: terms that form a phrase (such as A='information' and B='retrieval');
- C_2: terms that do not form a phrase but occur in the same document ('knowledge' and 'wireless');
- C_3: terms that rarely occur in the same document (such as 'cancer' and 'wireless').

These three categories can be expressed through term co-occurrence in some generic document collection P_G. We re-use our query probing component to establish criteria for term selection for the three categories. A pair of terms (A, B) is in category C_1 (*phrase co-occurrence*) if the match number for $Phrase(A, B)$ is comparable with the conjunction A $\wedge$ B, that is $\frac{|P_G(Phrase(A,B))|}{|P_G(A \wedge B)|} > \alpha$, for some threshold $0 < \alpha < 1$. A term pair (A, B) is in category C_2 (*high co-occurrence*) if the terms are not co-occurred in a phrase, but their conjunction is comparable with either A or B, $\frac{|P_G(A \wedge B)|}{min\{|P_G(A)|,|P_G(B)|\}} > \beta$, for some $0 < \beta < 1$. If pair (A,B) does not fit the conditions for categories C_1 and C_2, then it is in category C_3 (*low co-occurrence*). For our experiments, we have selected Google as generic document collection G and set the values of α and β both to 0.01.

4.3 Elementary features

Match numbers for probe queries in $F^0_{A,B}$ represent *elementary features* that can be directly used to train classifiers. Unfortunately, this often leads to poor results. The reason is that Web resources considerably differ in size and, therefore, the query matches from different resources are of different magnitude and thus hardly comparable. A query may match millions of documents on Google, but only a few at a small local resource. To leverage the processing of query matches from resources of different size, we develop two alternative methods for the feature encoding.

In the first approach, we normalize the query matches in F^0 by the maximum number of matches for the two basic queries 'A' and 'B'. We thus obtain features F^1 with values mostly between 0 and 1 (except for queries related to the Boolean disjunction). The second approach F^2 to the feature encoding, uses the "less-equal-greater"-relationship between any two probe queries in a probe package. This produces a three-value feature for each pair of test queries.

4.4 Feature selection

The refinement of raw features produces l=22 refined real value features with F^1 and $\frac{l*(l-1)}{2} = 231$ three-value features with F^2. The basic approach is to train each classifier with the entire feature set F^0, F^1 or F^2. However, because of the noise in the data, building accurate classifiers may require a lot of training data. To control the amount of training data and enhance the quality of classification rules, we proceed with two methods of feature selection. First, we distinguish between relevant and irrelevant features for a given classifier and remove irrelevant ones. Second, beyond the direct feature filtering, we use prior knowledge and classify new syntaxes using previously classified ones.

Removing irrelevant features. The definition of relevant features requires establishing syntactical dependencies between model queries in R and semantic relationships between syntaxes in S. Model query $r_i \in R$ *syntactically depends* on model query r_j if r_i includes syntaxes present in r_j. Syntaxes s_i and s_j in S are *semantically related* if they *can* be interpreted with the same operator in O.

We define *the relevant feature set* F_i for syntax s_i as containing three parts, $FS(s_i) = FS_i = FS^0_i + FS^1_i + FS^2_i$. FS^0_i simply contains all model queries $r_j \in R$ that involve syntax s_i, for example $FS^0('AND')$= { 'A AND B', 'B AND A', 'A AND'}. Next, FS^1_i contains model queries for syntactically dependent syntaxes. Actually, FS^1_i contains the two model queries 'A B' and 'B A' for all binary syntaxes. Finally, FS^2_1 contains the model queries for semantically related syntaxes. For example, $FS^2('AND') = FS^0('+')$, and vice versa, $FS^2('+')$=$FS^0('AND')$.

Use of prior knowledge. Beyond removing irrelevant features, it is possible to benefit from the dependencies between syntaxes established in Section 4.1. For example, the *Literal*-cases for 'OR' and 'AND' depend on the interpretation of whitespaces. The classification of 'AND' as *Literal* becomes simpler when the system already knows that, for example, '$\sqcup$' is interpreted as conjunction. To use the prior knowledge, we alter the training and classification process. We impose an order on the syntaxes in S. When training or using syntax classifiers, we use the classification results of previous syntaxes.

We convert the syntax set in the ordered list S^O= (Default, '$*$', '$\sqcup$', '""', 'AND', '+', 'OR'', 'NOT', '$-$') and impose the order on how the classifiers are trained and used for the classification. In the prior knowledge approach, the feature set used to train the classifier for syntax $s_i \in S^O$ will include the classifications of all s_j preceding s_i in S^O.

Removing irrelevant features and using prior knowledge are two independent methods for feature selection and can be applied separately or together. This allows us to consider four feature selection methods for training classifiers and classifying new sites:

1. Full feature set, $Ffs_i = F$, where F is a selected feature encoding, F^0, F^1 or F^2;

2. Relevant feature set, $Rfs_i = FS_i$;

3. Prior knowledge features, $PKfs_i$=$F \cup M^{-1}(s_j), j < i$.

4. Relevant prior knowledge feature set $RPKfs_i = FS_i \cup M^{-1}(s_j), j < i$.

5. EXPERIMENTAL EVALUATION

To run experiments, we collected and annotated 36 Web sites with search interfaces. All sites report the match num-

bers for user queries and unambiguously interpret their query languages. Selected sites represent a wide spectrum of supported operator sets. For each site, we annotated all supported operators and their syntaxes. For the extraction of the match numbers from HTML pages we used the Xerox IWrap wrapper toolkit [7, 12]. Out of 36 providers, only 4 support monovalued interpretations; in the other 32 cases, at least one operator has two or more syntaxes.

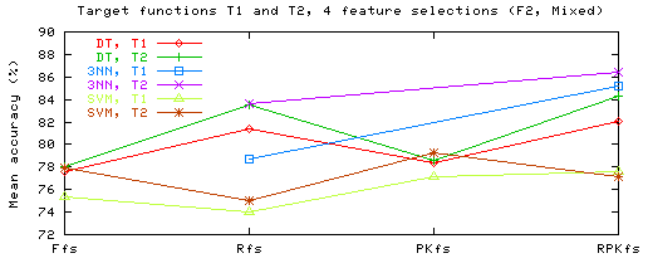

Figure 1: T_1 and T_2 target functions.

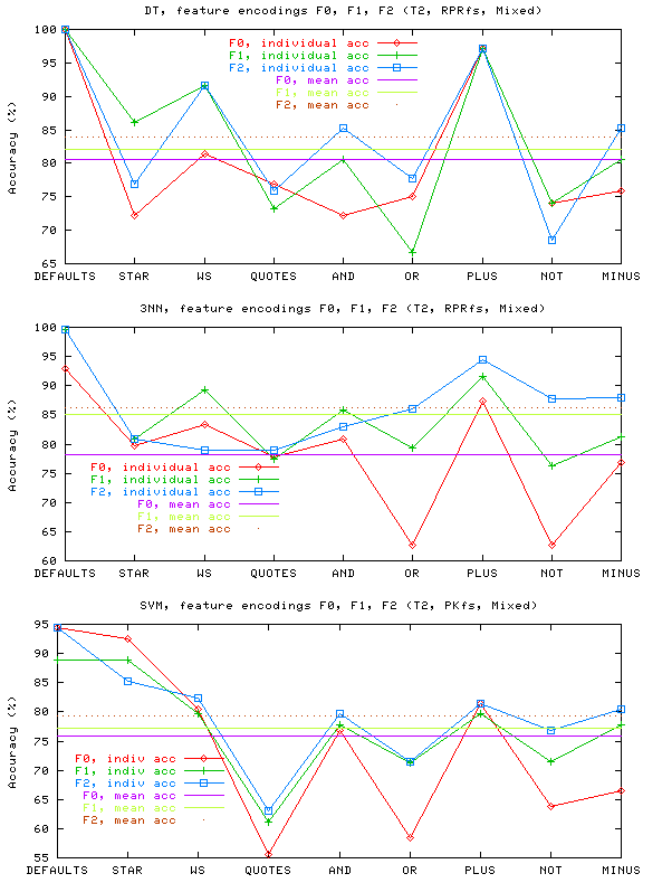

Figure 2: Three feature encodings for DT, KNN and SVM.

5.1 Experimental framework

In all experiments we estimate the classification accuracy for the individual operators in O (with T_1) and the syntaxes in S (with T_2). We also estimate the mean accuracy for the target functions T_1 and T_2. Experiments are conducted using the cross-validation method. 36 annotated sites are split into N=9 groups, S_1, S_2,..., S_N. We run N experiments; in experiment i, classifiers are trained with the groups S_1,...,S_{i-1}, S_{i+1},...,S_N and then tested with sites from group S_i. Accuracy (precision) values over N experiments are averaged for each operator/syntax classifier and form the *individual accuracies*. The average of individual accuracies over O/S gives the *mean accuracy*.

We test the learning system by varying the system parameters introduced in Section 4. We train and test classifiers with three different learning algorithms: decision trees from Borgelt's package (DT), k-nearest neighbors algorithm (KNN), and support vector machines (SVM)[5]. The following list recalls the test parameters and possible options.

1. *Target function*: T_1 includes $|O'|$=9 operator classifiers; multivalued interpretations are implemented as classifications with subsets of $|O'|$. For T_2, the system includes $|S|$=9 syntax classifiers.

2. *Feature encoding*: The three different feature encodings (see Section 4.3) include the raw match numbers given by F^0, the normalized match numbers given by F^1, and three-value feature comparison given by F^2.

3. *Feature selection*: The four methods presented in Section 4.4 include Ffs (full feature set), Rfs (relevant feature set), PKfs (prior knowledge feature set) and RPKfs (relevant prior knowledge feature set).

4. *Term selection*: We test three term selection categories, C_1, C_2 and C_3 introduced in Section 4.2. Additionally, we test the mixture of the three categories, when three term pairs are used to probe a site, i.e. one term pair from each category C_1, C_2 and C_3.

Experiments have been run for all parameter combinations; most combinations achieve mean accuracy superior to 60%. The four system parameters appear to be uncorrelated in their impact on the classification results. To figure out the most interesting ones, we determine overall "winners" for each parameter, except for the learning algorithm. The winners are T_2 target function, F^2 feature encoding, and *Mixed* term selection. $RPKfs$ feature selection behaves best for DT and KNN and $PKfs$ feature selection is the winner for SVM. We report more detail below.

5.2 Experimental Results

Decision trees are induced by the Borgelt's software; they are then pruned using the confidence level (p=0.5) pruning method. In SVM, linear kernel functions have been used. For the KNN method, we report results for k=3 which behaves better that k=1, 5 and 10. Because the implementation of the KNN algorithm cannot process large sets of features, we were not able to test the Ffs and PKfs feature selection methods.

All three learning algorithms show a similar performance. 3NN slightly outperforms DT and SVM for the "winner" combination (86.42% against 84.26% and 79.74%), however it is often less accurate with other parameter combinations.

[5]Available at `http://fuzzy.cs.uni-magdeburg.de/borgelt/software.html`, `http://www.dontveter.com/nnsoft/nnsoft.html`, `http://svmlight.joachims.org/`, respectively.

Target functions and feature selection. The target functions T_1 and T_2 implement alternative approaches to the query language acquisition; T_1 uses operator classifiers while T_2 uses syntax classifiers. As seen in Section 4, T_2 has an advantage over T_1 because it avoids multivalued classification and outputs only unambiguous interpretations, while the output of T_1 should be further tested for unambiguity. Thus we have built the learning system for T_2. Series of experiments conducted with T_1 and T_2 confirm the superiority of T_2. As operator classifiers in T_1 are trained independently, their combined output does not guarantee unambiguity. Unlike T_2, high accuracy of individual classifiers may not be translated into global good accuracy, because one misclassification may produce an ambiguous interpretation and undermine the good performance of other classifiers.

In practice, we test the output of operator classifiers of T_1 and discard those that form ambiguous interpretations. This gives a 2% to 10% drop in the mean accuracy. Figure 1 plots mean accuracies for T_1 and T_2 for all feature selection methods (with fixed F_2 feature encoding and $Mixed$ term selection) and the three learning methods (only Rfs and RPKfs could be measured for 3NN algorithm). Within feature selection methods, keeping relevant features spurs the performance of DT and 3NN better than the prior knowledge, with their combination being the winner. For SVM, instead, adding prior knowledge to the full feature set is the best choice. In the following, all reported experiments refer to the target function T_2.

Feature encoding. Previous figures compared the mean accuracies. We unfold the mean value and plot individual accuracies for the syntaxes in S. Figure 2 plots accuracy values for the three feature encoding methods (for T_2-$RPKfs$-$Mixed$ combination for DT and 3NN and T_2-$PKfs$-$Mixed$ combination for SVM). As the figure shows, the pair-wise comparison F_2 performs best with respect to the raw and normalized match numbers.

Term selection. We complete the analysis of system parameters by testing four methods of term selection. They include categories C_1, C_2 and C_3 and $Mixed$. Figure 3 plots mean accuracies for all learning algorithms and four term selection methods, giving $Mixed$ as the winner.

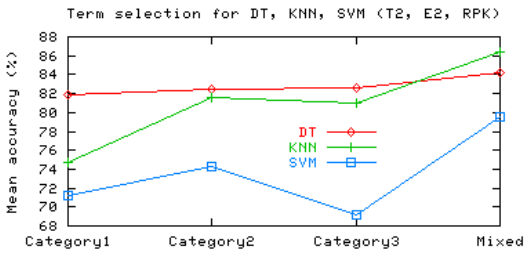

Figure 3: Four term selection methods for DT, KNN, and SVM.

5.3 Bias in training data

Among the syntaxes in S, all methods show only little difference for the unary operators $Default$ and $Case$. Among the syntaxes for binary operators, certain ($'\sqcup'$, $'$AND$'$ and $'+'$) are easier to detect than others ($'$" "$'$, $'$OR$'$ and $'$NOT$'$). However, this phenomenon is not linked to the nature of the operators or their syntaxes, but rather can be explained by the bias in training data. In Table 1, we unfold the individual accuracies and show results for each case $(s, o), s \in S, o \in O'$ in the annotated data. Each non-empty cell in Table 1 reports the occurrence of the case (in brackets) and its classification accuracy. We can observe a definitive bias of high accuracy for more frequent cases; instead, rare cases have a very low accuracy. This explains good results for $'\sqcup'$, $'$AND$'$ and $'+'$, where occurrences are fairly split between two main cases. For other syntaxes instead, the high error ratio for rare cases decreases the individual accuracy.

6. RELATED WORK

The Hidden Web has emerged as a research trend and different research groups have started to address various problems relevant to organizing Hidden Web resources [15, 14, 17, 19, 20]. One focus is crawling; [17] presents a task-specific and human-assisted approach to the crawling of the Hidden Web. The crawler searches for Hidden Web resources relevant to a specific domain. The identification is achieved by selecting domain-specific keywords from a fuzzy set and assigning them to elements of HTML forms; the resource is judged relevant if it returns valid search results.

Another important task is the classification of Hidden Web resources. [15, 14] and [19] have developed approaches to this problem based on query probing. Moreover, [15] makes use of the number of documents matching a submitted conjunction query, as does our approach. Instead of query languages, they use match numbers to reason about the relevance of a provider for a given category.

Originally, the query probing has been used for the automatic language model discovery in [9], it probed documents in a collection to analyze the content of the result pages. [14] extends the work in [15] to the problem of database selection by computing content summaries based on probing. Once query language interfaces are understood, meaningful query transformation becomes possible. [11] describes one way of transforming a front-end query into subsuming queries that are supported by the sources and a way to filter out incorrectly returned documents. In [16], interaction with online-vendors is automated.

In the close domain of meta-searching, the declaration of a search resource's query features is often coupled with methods of converting/translating meta-search queries into the resource's native queries. Considerable research effort has been devoted to minimizing the possible overheads of query translation when the meta-search and search resource differ in supporting basic query features [11]. In all these methods, the manual discovery of the query features is assumed.

In information mediation systems that query Web resources to materialize views on hidden data [20], one approach is to reconstruct an image of a resource's database. Because of a restricted Web interface, a critical problem is the entire or partial reconstruction of the database image without the unnecessary overload of the Web servers. [8] builds efficient query covers that are accessible through nearest-neighbor interfaces for the specific domain of spatial databases.

7. OPEN QUESTIONS

The experiments have raised a number of open questions that require further research. Below we list some of them.

Stopwords. In tests, common English stopwords were excluded from probing. However, the set of stopwords is

Operators	Syntaxes								
	default	′∗′	′␣′	′“”′	′AND′	′OR′	′+′	′NOT′	′−′
Case	97.6(20)								
Stemming		41.1(9)							
Conjunction			92.9(15)		95.2(27)		100(16)		
Disjunction			100(19)			87.8(17)			
Negation								91.0(15)	94.9(26)
Phrase			0(1)	90.4(28)					
Literal	100(16)				79.6(7)	91.7(11)		69.2(14)	
Ignored		79.9(27)		18.5(4)	3.7(1)	55.6(4)	100(19)	25.0(4)	81.0(7)
Unknown			0(1)	4.1(4)	0(1)	19.4(4)	0(1)	0(3)	0(3)

Table 1: Classification accuracy and occurrence for all syntax+interpretation cases (DT,T_2,F_2,RPKfs,$Mixed$).

often domain-dependent; this should be taken into account when generating test queries. A more difficult case is when a resource treats terms as stopwords in a certain context. For example, Google accepts the term "WWW" when it is queried alone and ignores it when the term is conjuncted with other terms. Such query-dependent treatment of stopwords is considered as noise in the current system.

Acquiring other operators. We have addressed the set of most frequently used query operators. Other operators defined by existing document retrieval models, like proximity operators, can be added to the operator set and processed in a similar manner. Two remarks concerning less frequent operators are that their syntactical encodings may vary even more than for Boolean operators, and, more importantly, finding sufficient training data to build reliable classifiers may be technically difficult.

Query composition. The next issue is the manner in which basic query operators are combined to form complex queries. The most frequent manner on the Web is the use of parentheses or a certain operator priority. How to detect this remains an open problem at this point.

Ambiguous interpretations. Recognizing ambiguous interpretations is the most difficult problem. One example is Citeseer, which interprets whitespaces as conjunction by default, but switches to disjunction if the conjunction query matches no documents. Some other Web providers behave in the same or a similar manner. We will need to extend the learning system to include a possibility of triggering the retrieval model as a function of the oracle answers.

8. CONCLUSION

We have addressed the problem of automatic recognition of operators and syntaxes supported by query languages of Web resources. We have developed a machine learning approach based on reformulation of the entire problem as a set of classification problems. By introducing various refined solutions for the target function, feature encoding, and feature selection, we have achieved 86% mean accuracy for the set of the most frequent operators and syntaxes. Further improvement in the accuracy is possible with better preparation of annotated sites, but this is limited because of the complexity of the a-priori unknown operator composition and the noise produced by the hidden query preprocessing.

9. REFERENCES

[1] The InvisibleWeb, http://www.invisibleweb.com/.
[2] BrightPlanet, http://www.brightplanet.com/.
[3] CompletePlanet, http://www.completeplanet.com/.
[4] G. Alonso. Myths around web services. *IEEE Bulletin on Data Engineering*, 25(4):3–9, 2002.
[5] D. Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1987.
[6] M. K. Bergman. The Deep Web: Surfacing hidden value. *Journal of Electronic Publishing*, 7(1), 2001.
[7] D. Bredelet and B. Roustant. Java IWrap: Wrapper induction by grammar learning. Master's thesis, ENSIMAG Grenoble, 2000.
[8] S. Byers, J. Freire, and C. T. Silva. Efficient acquisition of web data through restricted query interfaces. In *Proc. WWW Conf.*, China, May 2001.
[9] J. P. Callan, M. Connell, and A. Du. Automatic discovery of language models for text databases. In *Proc. ACM SIGMOD Conf.*, pp. 479–490, June 1999.
[10] C.-C. K. Chang and H Garcia-Molina. Approximate query translation across heterogeneous information sources. In *Proc. VLDB Conf.*, pp. 566–577, Cairo, Egypt, September 2000.
[11] C.-C. K. Chang, H. Garcia-Molina, and A. Paepcke. Boolean query mapping across heterogeneous information sources. *IEEE TKDE*, 8(4):515–521, 1996.
[12] B. Chidlovskii. Automatic repairing of web wrappers by combining redundant views. In *Proc. of the IEEE Intern. Conf. Tools with AI*, USA, November 2002.
[13] L. Gravano, H. Garcia-Molina, and A. Tomasic. Gloss: Text-source discovery over the internet. *ACM TODS*, 24(2):229–264, 1999.
[14] P. G. Ipeirotis and L. Gravano. Distributed search over the hidden web: Hierarchical database sampling and selection. In *Proc. VLDB Conf.*, pp. 394–405, Hong Kong, China, August 2002.
[15] P. G. Ipeirotis, L. Gravano, and M. Sahami. Probe, count, and classify: Categorizing hidden-web databases. In *Proc. ACM SIGMOD Conf.*, pp. 67–78, Santa Barbara, CA, USA, May 2001.
[16] M. Perkowitz, R. B. Doorenbos, O. Etzioni, and D. S. Weld. Learning to understand information on the internet: An example-based approach. *Journal of Intelligent Information Systems*, 8(2):133–153, 1997.
[17] S. Raghavan and H. Garcia-Molina. Crawling the hidden web. In *Proc. VLDB Conf.*, pp. 129–138, Rome, Italy, September 2001.
[18] D. Tsur. Are web services the next revolution in e-commerce? In *Proc. VLDB Conf.*, pp. 614–617, Rome, Italy, September 2001.
[19] W. Wang, W. Meng, and C. Yu. Concept hierarchy based text database categorization. In *Proc. Intern. WISE Conf.*, pp. 283–290, China, June 2000.
[20] R. Yerneni, C. Li, H. Garcia-Molina, and J. Ullman. Computing capabilities of mediators. In *Proc. ACM SIGMOD Conf.*, pp. 443–454, PA, USA, June 1999.