Mining Web Functional Dependencies for Flexible Information Access

[†]Saverio Perugini
Department of Computer Science
University of Dayton, OH 45469–2160
E-mail: saverio@udayton.edu
Tel: (937) 229–4079
Fax: (937) 229–4000

Naren Ramakrishnan
Department of Computer Science
Virginia Tech, Blacksburg, VA 24061–0106
E-mail: naren@vt.edu

November 10, 2006

Abstract

We present an approach to enhancing information access through web *structure* mining in contrast to traditional approaches involving usage mining. Specifically, we mine the hardwired hierarchical hyperlink structure of websites to identify patterns of term-term cooccurrences we call *web FDs* (functional dependencies). Intuitively, a web FD '$x \rightarrow y$' declares that all paths through a site involving a hyperlink labeled x also contain a hyperlink labeled y. The complete set of FDs satisfied by a site help characterize (flexible and expressive) interaction paradigms supported by a site, where a paradigm is the set of explorable sequences therein. We describe algorithms for mining FDs, results from mining several hierarchical websites, and present several interface designs that can exploit such FDs to provide compelling user experiences.

[†] Contact author

1 Introduction

Web mining is an established area of research (Kolari & Joshi, 2004) that seeks to uncover patterns in site structure, usage patterns, and navigation schemas. The end effectors for web mining are primarily in electronic commerce, typically manifested as personalized experiences for users or better targeting of site content.

Our goal is to enhance browsing experiences for users (Narayan, Williams, Perugini, & Ramakrishnan, 2004; Perugini & Ramakrishnan, 2003) by developing interaction techniques that more directly address the user-site impedance mismatch. We differ from prior work (Chen, Park, & Yu, 1998; Eirinaki & Vazirgiannis, 2003; Kamdar & Joshi, 2005) in both the type of information we mine and the uses we find for the patterns mined. Specifically, we analyze site structure to recover persistent properties of the domain modeled, independent of user interactions. At the same time, we present ways to harness the results of data mining to realize more responsive dialogs between users and websites.

The basic approach adopted here is to think of site structure as exposing dialog completion paths, or interaction sequences that capture the sequential ordering of information inputs, en route to a target page. Patterns in these interaction sequences, called *web FDs*, expose important relationships which can be harnessed to create adaptive dialogs that situate the user's partial input in the context of the site, without disrupting the site's basic navigation schema. We show that web FDs are ubiquitous and suggest many natural information-seeking interfaces. We focus on primarily hierarchical sites as they present the greatest benefit for the techniques presented here.

2 Web Functional Dependencies

A *web functional dependency* (FD) of the form $x \rightarrow y$ declares something about the relationship between the terms x and y wrt their co-occurrence (or lack thereof) along the structure of a website. Intuitively, a web FD $x \rightarrow (\neg)y$ indicates that *all* (*no*) sequences involving x also involve y. We describe two classes of web FDs — negative and positive — which have complementary uses in information-seeking. Negative web FDs help prune a website, while positive web FDs suggest a way to conduct query expansion and approximate information retrieval. Each class can be further decomposed into path and leaf web FDs.

Positive Web FDs

A *positive-path web FD* $x \rightarrow y$, where x and y are terms, exists when the complete set of sequences containing x is a subset of the complete set of sequences containing y. The complete set of positive-path web FDs which hold in the website shown in Fig. 1 is

$$\begin{array}{rcl}
\{\text{2003} & \rightarrow & \{\text{Ford, Taurus}\}, \\
\text{2006} & \rightarrow & \{\text{Civic, Honda}\}, \\
\text{Accord} & \rightarrow & \text{Honda}, \\
\text{Camry} & \rightarrow & \text{Toyota}, \\
\text{Civic} & \rightarrow & \text{Honda}, \\
\text{Corolla} & \rightarrow & \text{Toyota}, \\
\text{Focus} & \rightarrow & \{\text{2005, Ford}\}, \\
\text{Taurus} & \rightarrow & \text{Ford}, \\
\{\text{2004, Ford}\} & \rightarrow & \text{Taurus}, \\
\{\text{2004, Honda}\} & \rightarrow & \text{Accord}\}
\end{array}$$

In Fig. 1, edge-labels model hyperlink labels (i.e., the text between `<a href="...">` and `</a>`) or, in other words, choices made by a navigator en route to a leaf. We refer to an edge-label as a *term* even though

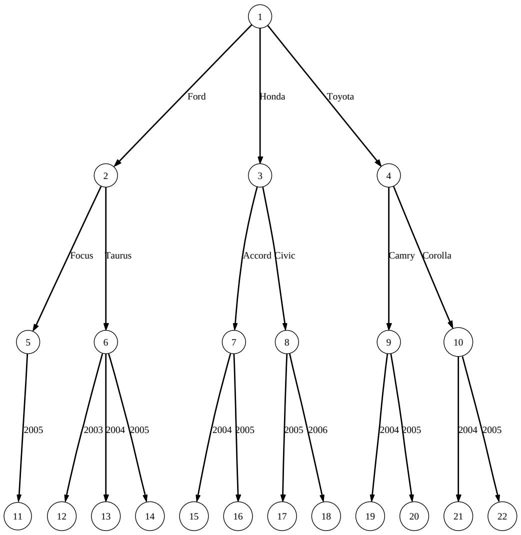


Figure 1: Example (DAG) model of a hierarchical automobile web directory with characteristics similar to those in Edmunds.com.

it may contain more than one word (i.e., any string of characters except space). For instance, 'side airbags' would be a term in our viewpoint if it were the anchor text for some hyperlink.

We borrow the *splitting/combining* rule (Garcia-Molina, Ullman, & Widom, 2002) from relational database theory to simplify our presentation of these web FDs. For instance, '2003 $\rightarrow$ {Ford, Taurus}' expresses two individual web FDs in one expression: '2003 $\rightarrow$ Ford' and '2003 $\rightarrow$ Taurus.' However, the web FD '{2004, Ford} $\rightarrow$ Taurus' is *not* a simplification of the '2004 $\rightarrow$ Taurus' and 'Ford $\rightarrow$ Taurus' web FDs (neither are satisfied by the site), but rather states that all sequences through the site involving 2004 *and* Ford also involve Taurus. In total, 13 positive-path web FDs hold in the DAG shown in Fig. 1 (discounting trivial web FDs such as 'Civic $\rightarrow$ Civic', '{2005, Civic} $\rightarrow$ Civic', and '{2005, Civic} $\rightarrow$ Honda'), though we require only 10 expressions to present them.

A *positive-leaf web FD* $x \rightarrow y$ specifies that *all* of the leaves classified by sequences involving x also are classified by all sequences involving y. For instance, 'Accord $\rightarrow$ Honda' is a positive-leaf web FD satisfied by site depicted in Fig. 1. Positive-leaf web FDs are more general than positive-path web FDs in that all positive-path web FDs also are positive-leaf web FDs, but the reverse is only true in trees, e.g., such as that shown in Fig. 1, in which there is only one path from the root to each leaf. Thus, the complete set of positive-leaf web FDs which hold in Fig. 1 is the complete set of positive-path web FDs given above.

Notice also that a positive (-path or -leaf) web FD $x \rightarrow y$ does not necessarily mean that $y \rightarrow x$.

Negative Web FDs

A *negative-path web FD* $x \rightarrow \neg y$ indicates that *none* of the sequences through the site involving x involve y. Considering only negative-path web FDs with one term on the left, a total of 103 FDs are satisfied by the DAG shown in Fig. 1, though we require only 13 expressions to capture them:

Table 1: Two instances of a relation with schema $R(A_1, A_2)$. (left) Satisfies the database FD $A_1 \rightarrow A_2$ as well as 4 web FDs: $\{a \rightarrow c, b \rightarrow c, d \rightarrow e, e \rightarrow d\}$. (right) Does not satisfy the database FD $A_1 \rightarrow A_2$, but does satisfy 2 web FDs: $\{b \rightarrow c, e \rightarrow d\}$.

A_1	A_2
a	c
b	c
d	e

A_1	A_2
a	c
a	d
b	c
d	e

$$\begin{array}{rcl}
\{\text{2003} & \rightarrow & \neg\,\{\text{2004, 2005, 2006, Accord, Camry, Civic, Corolla, Focus, Honda, Toyota}\}, \\
\text{2004} & \rightarrow & \neg\,\{\text{2003, 2005, 2006, Civic, Focus}\}, \\
\text{2005} & \rightarrow & \neg\,\{\text{2003, 2004, 2006}\}, \\
\text{2006} & \rightarrow & \neg\,\{\text{2003, 2004, 2005, Accord, Camry, Corolla, Focus, Ford, Taurus, Toyota}\}, \\
\text{Accord} & \rightarrow & \neg\,\{\text{2003, 2006, Camry, Civic, Corolla, Focus, Ford, Taurus, Toyota}\}, \\
\text{Camry} & \rightarrow & \neg\,\{\text{2003, 2006, Accord, Civic, Corolla, Focus, Ford, Honda, Taurus}\}, \\
\text{Civic} & \rightarrow & \neg\,\{\text{2003, 2004, Accord, Camry, Corolla, Focus, Ford, Taurus, Toyota}\}, \\
\text{Corolla} & \rightarrow & \neg\,\{\text{2003, 2006, Accord, Camry, Civic, Focus, Ford, Honda, Taurus}\}, \\
\text{Focus} & \rightarrow & \neg\,\{\text{2003, 2004, 2006, Accord, Camry, Civic, Corolla, Honda, Taurus, Toyota}\} \\
\text{Ford} & \rightarrow & \neg\,\{\text{2006, Accord, Camry, Civic, Corolla, Honda, Toyota}\}, \\
\text{Honda} & \rightarrow & \neg\,\{\text{2003, Camry, Corolla, Focus, Ford, Taurus, Toyota}\}, \\
\text{Taurus} & \rightarrow & \neg\,\{\text{2006, Accord, Camry, Civic, Corolla, Focus, Honda, Toyota}\}, \\
\text{Toyota} & \rightarrow & \neg\,\{\text{2003, 2006, Accord, Civic, Focus, Ford, Honda, Taurus}\}\}.
\end{array}$$

A *negative-leaf web FD* $x \rightarrow \neg y$ specifies that *none* of the leaves classified by sequences involving x are classified by sequences involving y. Negative-leaf and -path web FDs also are equivalent in tree models of websites, e.g., such as that shown in Fig. 1. Further, any negative-leaf web FD satisfied by a site is a negative-path web FD satisfied by that site, though the reverse only holds in trees.

Notice further that negative (-path and -leaf) and positive (-path and -leaf) web FDs are not complements of each other, i.e., the presence of $x \rightarrow y$ does not necessarily imply the presence of $x \rightarrow \neg\,\{\mathcal{T} - y\}$, where $\mathcal{T}$ represents the complete set of terms in the site. Similarly, a site that satisfies $x \rightarrow \neg y$ may not satisfy $x \rightarrow \ \{\mathcal{T} - y\}$. However, it is true that any negative web FD $x \ \rightarrow \ \neg\, y$ considered here implies $y \rightarrow \ \neg\, x$.

3 Reasoning with Web FDs

Relationship between Web FDs and Database FDs

Web FDs can be viewed as both generalizations (in one sense) and specializations (in another) of database FDs. In relational database theory, an FD of the form $A \rightarrow B$ means that all tuples which agree on their value(s) for A *must* agree on their value(s) for B. Here, A and B are attributes of the relational schema, and therefore, we refer to such FDs as *schema FDs*. In contrast, the left and right sides of a web FD constitute values rather than attributes. Therefore, we say that web FD are *value FDs*. This caveat implies that web FDs are less restrictive than database FDs. Specifically, web FDs can exist between values of two attributes A_1 and A_2 of a relation instance R even if no database FDs between A_1 and A_2 are satisfied by R. For instance, consider the two instances in Table 1 of the relation with schema $R(A_1, A_2)$. The schema FD $A_1 \rightarrow A_2$ holds in the instance on the left. In addition, several value FDs hold as well: $\{a \rightarrow c, b \rightarrow c, d \rightarrow e, e \rightarrow d\}$. However, while the schema FD $A_1 \rightarrow A_2$ does not hold in the instance on the right, notably some value FDs are still satisfied by that instance: $\{b \rightarrow c, e \rightarrow d\}$.

In the above example of Table. 1 (right), observe that the web FDs $a \rightarrow c$ and $d \rightarrow e$ are *not* satisfied. The violation of the first web FD is an example that violates the underlying database FD ($A_1 \rightarrow A_2$) as well. However the reason the second does not hold in more subtle. Here d is a legal value for either of the attributes A_1 and A_2 and it is not generally true that all sequences involving d also involve e. Nevertheless, it is true that all sequences involving d *as a value of* A_1 also involve e *as a value of* A_2. Therefore, since web FDs are silent on the relationship between schema and attribute values, their requirements can be more stringent than database FDs.

The above discussion reiterates that web FDs can be simultaneously more restrictive and less restrictive, in different aspects, than database FDs. We can extend the complete set of inference rules for reasoning with database FDs, known as *Armstrong's axioms* (*reflexivity*, *augmentation*, and *transitivity*) (Garcia-Molina et al., 2002), from relational database theory to reason about web FDs as well, by focusing on relationships between values rather than attributes. Moreover, this deductive apparatus is also *sound* and *complete* for web FDs (outside the scope of this paper).

The emphasis on values rather than attributes also implies that there are more ways for a web FD to be trivial. In a database, an FD $A \rightarrow B$ is *trivial* if the right side (B) is a subset of the attributes on the left (A). Similarly, in the web context, a (positive) FD $A \rightarrow B$ can be trivial for this reason and, in addition, when the number of interaction sequences supporting the left side is zero. Moreover, in the database world, an FD $A \rightarrow B$ is said to be *nontrivial* if at least one of the B's is not among the A's and *completely nontrivial* if none of the B's is also one of the A's (Garcia-Molina et al., 2002). The first of these definitions also applies to web FDs. For example, the web FD 'Accord $\rightarrow \{$Accord, Honda$\}$' is nontrivial. However, we strengthen the definition of completely nontrivial in the web setting by augmenting it: a web FD $A \rightarrow B$ is *completely nontrivial* if none of the B's is also one of the A's and if no proper subset S of A functionally determines B. The database community calls the attributes in $A - S$ *extraneous* (Silberschatz, Korth, & Sudarshan, 2006). For instance, the web FD '$\{$2005, Accord$\} \rightarrow$ Honda' is not completely nontrivial in the site shown in Fig. 1 since the site also satisfies the 'Accord $\rightarrow$ Honda' web FD. Notice that above we only show the completely nontrivial web FDs satisfied by the site in Fig. 1. As we will see when we discuss uses of web FDs, we are only interested in all completely nontrivial web FDs.

Relationship between Web FDs and Association Rules

The ideas between web FDs and association rules are more aligned if we view interaction sequences as transactions and terms as items. In this modeling, web FDs are simply association rules with a confidence of 1 (Agrawal, Imielinski, & Swami, 1993). However, we hasten to add that, later in this paper, we will view web FDs in the context of a real website interaction, so that the terms are time-indexed with an arrival stamp. Hence, the use of terms in a web FD will imply an ordering based on relationships between their arrival time in an interaction sequence (whereas the items of a transaction are unordered). Furthermore, we will relax the confidence threshold to less than 1 and suggest potential uses in information retrieval. In addition, we will discuss term-similarity metrics, beyond the containment of one set of sequences in another, which yield web FDs bearing little similarity to association rules. Also, note that association rules in web mining are predominantly mined from web usage logs and, as a result, have been used in web modeling primarily to study and understand site usage patterns.

4 Mining Web Functional Dependencies

Recall a web FD of the form $x \rightarrow Y$, where x is an edge label (Y may represent an edge label or several edge labels), exists when the complete set of sequences containing x is a subset of the complete set of sequences containing (all of the terms in) Y. We call these *single-step web FDs* and those of the form $X \rightarrow Y$, where X is a set of edge labels, *multi-step web FDs*. Notice that since a set may contain only one element, every single-step FD is a multi-step FD. Here, a *sequence* is an ordered list of terms from the root of a website to a leaf.

To mine all multi-step web FDs, we build a multi-level, recursive index and walk it in a recursive, depth-first fashion. While recursing deep into the index, we repeatedly identify single-step web FDs on subgraphs of the website and when backing out of the recursion, we grow those single-step web FDs into multi-step web FDs.

Algorithm 1 illustrates the functions for mining multi-step positive-path, web FDs from a DAG representation of a website. The function $build{-}index$ in Algorithm 1 builds the multi-level, recursive index. Each entry in the index is a (key, set of sequences, multi-level recursive index) triple. The key is a term. The set of sequences are those which contain the key term. The third component of the triple is a multi-level, recursive index of a site consisting only of the sequences from the second component (of this triple) after removing the key term from each (see function $selectSequences$). Thus, the third component points to a multi-level, recursive index of a reduced version of the website. Each index contains exactly one entry for each term in the site. The worst case running time of $build{-}index$ is potentially $O(|\mathcal{T}|!)$, where $\mathcal{T}$ is the complete set of terms in the website (but see discussion later about how it is more instructive to study algorithmic complexity in output-sensitive terms and why this is practical). Table 2 shows the index resulting from applying the $build{-}index$ function to the DAG shown in Fig. 1. Notice that the recursive nature of the index is evident in the third component of every element. Since positive web FDs can only exist between terms which co-occur, this index helps us prune the search space.

Traversal of this index, and thus the discovery of all multi-step web FDs, is an induction on the size of the index.

BASIS: A site with an empty index satisfies no multi-step web FDs. Since our algorithm is not linear-recursive, the basis is reached often, i.e., whenever we encounter an index entry of the form (x, $\{s_1, s_2, \ldots, s_n\}$, nil).

INDUCTION: When we encounter an index entry with a non-nil third component, we check, as described above, if a single-step web FD exists between the key of the triple (i.e., its first component) and the key of the first entry of the index in its third component. If a dependency exists, we add it to a list of web FDs local to this recursive instance of the problem and then recurse with the entire sub-index. When the most recent recursive call returns, we have a list of all of the multi-step web FDs from that problem instance. To achieve the complete set of multi-step web FDs for the prior instance we simply have to prepend the key term of the index entry from the prior instance to each multi-step FD of the current instance.

Intuitively, our approach involves progressively removing terms, one at a time, from the original site, i.e., removing it from every sequence containing it. We can think of this new set of sequences as constituting a new site S'. We then can compute the complete set of single-step web FDs satisfied S'. By adding the term removed to the lhs of each of the resulting (single-step) web FDs, we have identified a *portion* of the two-step web FDs, i.e., those with only two terms on the lhs. For example, the following is the set of sequences from Fig. 1 involving 'Ford' (with the term 'Ford' then removed from those): $\{\prec$Focus, 2005$\succ$, $\prec$Taurus, 2003$\succ$, $\prec$Taurus, 2004$\succ$, $\prec$Taurus, 2005$\succ\}$. The complete set of single-step web FDs satisfied by only these sequences is $\{$2003 → Taurus, 2004 → Taurus$\}$. We can now add the term 'Ford' to the lhs of each to attain a portion of the two-step web FDs from Fig. 1: $\{\{$2003, Ford$\}$ → Taurus, $\{$2004, Ford$\}$ → Taurus$\}$. Since the single-step web FD '2003 → Taurus' is satisfied by the original site, the web FD '$\{$2003, Ford$\}$ → Taurus' is trivial (i.e., it adds no new information) and can be safely omitted. Therefore, we have a portion of the complete set of two-step web FDs from the original site: '$\{$2003, Ford$\}$ → Taurus'.

Functions $mine{-}multi{-}step{-}webFDsHelper$ and $mine{-}multi{-}step{-}webFDs$ in Algorithm 1 illustrate how to mine multi-step positive-path web FDs from a multi-level, recursive index using this procedure.

Table 3 shows the multi-step positive-path web FDs resulting from applying $mine{-}multi{-}step{-}webFDs$ to the DAG shown in Fig. 1, which involves building and using the index shown in Table 2. The function $walk$ collects the web FDs encapsulated within this data structure into a bag. Walking the data structure shown in Table 3 results in 45 web FDs and 17 of them are duplicates (e.g., '$\{$Ford, 2003$\}$ → Taurus' and '$\{$2003, Ford$\}$ → Taurus'). The function $uniq$ removes duplicates from the resulting bag. Furthermore, we can safely reduce the 28 unique web FDs to 13 FDs by removing all web FDs which are not completely nontrivial (e.g., '$\{$2004, Accord$\}$ → Honda' and '$\{$2005, Accord$\}$ → Honda'). The function $simplify$ further prunes out all webs FDs which are not completely nontrivial. Notice that these simplifications of the complete set of web FDs is *not* analogous to computing a minimal basis of a set of relational database FDs using a canonical cover algorithm (Silberschatz et al., 2006). A minimal basis of a set of FDs is desired for each relation in a relational database to reduce the number of FDs which must be checked for satisfaction by the relation after an update to it. In stark contrast, here we desire *all* completely nontrivial web FDs, not just a minimal set from which the complete set follows, for reasons related to our application of them in a web user interface discussed in section 6.

Algorithm 1 is sound and complete. It is levelwise like the *Apriori* algorithm from association rule mining (Agrawal & Srikant, 1994) and, hence, its complexity can be studied in similar terms. The number of web FDs (say p) in a site is potentially exponential in its size but, just as in traditional data mining algorithms, the number of web FDs reduces dramatically as we drill through the levels. This makes Algorithm 1 to be output-sensitive with complexity $O(p)$. The section below covering the uses of web FDs discusses how their discovery is practical in a web user interface as the user traverses through the levels.

Identifying the complete set of negative web FDs satisfied by a site is straightforward (see Algorithm 2). The rhs of negative web FDs with lhs x is the difference between the entire set of terms in the site and those terms with which x co-occurs (in some sequence). Note that the rhs of a negative web FD is not simply the complement of terms that occur in its positive counterparts: for a term to be used in the rhs of a positive web FD, it must co-occur in *all* sequences involving the lhs, whereas for a term to be used in the rhs of a negative web FD, it must not co-occur in *any* of the sequences involving the lhs. Algorithm 2 is linear in the total number of terms in the website.

Notice that while *Armstrong's axioms* hold in the web FD context, a closure algorithm (involving applications of them) (Garcia-Molina et al., 2002) is insufficient to discover all web FDs satisfied by a site. This is because a closure algorithm must start with a *basis* set of FDs to determine the complete set of FDs (Garcia-Molina et al., 2002). Algorithm 1 discovers all web FDs and is thus functionally equivalent to discovering a basis set of FDs followed by computing the closure of that basis.

```
Name: $build{-}index$
Type: set of sequences → Index {a list of (term, set of sequences, Index) triples}
Parameters: set of sequences $S$
  if $S$ = nil then
    return nil
  else
    $\mathcal{T} \leftarrow getTerms(S)$ {returns the set of terms in $S$}
    for $i \leftarrow 0$ to $|\mathcal{T}|$ do
      $S^{'} \leftarrow selectSequences(\mathcal{T}[i],\ S)$ {selects all sequences from $S$ containing the term $\mathcal{T}[i]$, removes $\mathcal{T}[i]$ from
      each sequence in that set, and returns the resulting set}
      $index[i] \leftarrow (\mathcal{T}[i],\ S^{'},\ build{-}index(S^{'}))$
    end for
  end if
  return $index$ {a list of (term, set of sequences, Index) triples}

Name: $mine{-}multi{-}step{-}webFDsHelper$
Type: Index → multi-step-webFDs {a list of (term, set of terms, multi-step-webFDs) triples}
Parameters: Index $index$
  for $i \leftarrow 0$ to $|index|$ do
    $multi{-}step{-}webFDs[i](\#1) \leftarrow index[i](\#1)$
    if $index[i](\#3)$ = nil then
      $multi{-}step{-}webFDs[i](\#2)$ = nil
      $multi{-}step{-}webFDs[i](\#3)$ = nil
    else
      for $j \leftarrow 0$ to $|index[i](\#3)|$ do
        if $index[i](\#2) \subset index[i](\#3)[j](\#2)$ then
          $multi{-}step{-}webFDs[i](\#2) \leftarrow multi{-}step{-}webFDs[i](\#2) \cup index[i](\#3)[j](\#1)$
        end if
        $multi{-}step{-}webFDs[i](\#3) \leftarrow$
        $multi{-}step{-}webFDs[i](\#3) \cup mine{-}multi{-}step{-}webFDsHelper(index[i](\#3)[j](\#1))$
      end for
    end if
  end for
  return $multi{-}step{-}webFDs$ {a list of (term, set of terms, multi-step-webFDs) triples}

Name: $mine{-}multi{-}step{-}webFDs$
Type: DAG → multi-step_webFDs {a set of completely nontrivial multi-step web FDs}
Parameters: DAG $D$
  $S \leftarrow sequencize(D)$ {returns the set of sequences from $D$}
  $index \leftarrow build{-}index(S)$
  $multi{-}step{-}webFDsIntermed1 \leftarrow mine{-}multi{-}step{-}webFDsHelper(index)$ {a list of (term, set of
  terms, multi-step-webFDs) triples}
  $multi{-}step{-}webFDsIntermed2 \leftarrow walk(multi{-}step{-}webFDsIntermed1)$ {a bag of multi-step web FDs}
  $multi{-}step{-}webFDsIntermed3 \leftarrow uniq(multi{-}step{-}webFDsIntermed2)$ {a set of multi-step web FDs}
  $multi{-}step{-}webFDs \leftarrow simplify(multi{-}step{-}webFDsIntermed3)$ {a set of completely nontrivial multi-
  step web FDs}
  return $multi{-}step{-}webFDs$ {final set of completely nontrivial multi-step webFDs}
```

Algorithm 1: Mine completely nontrivial multi-step positive-path web FDs, simplified for purposes of presentation.

Table 2: Multi-level, recursive index of the sample automobile website in Fig. 1.

Ford	{11, 12, 13, 14}	Focus	{11}	2005	{11}	nil
		Taurus	{12, 13, 14}	2003	{12}	nil
				2004	{13}	nil
				2005	{14}	nil
		2003	{12}	Taurus	{12}	nil
		2004	{13}	Taurus	{13}	nil
		2005	{11, 14}	Focus	{11}	nil
				Taurus	{14}	nil
Honda	{15, 16, 17, 18}	Accord	{15, 16}	2004	{15}	nil
				2005	{16}	nil
		Civic	{17, 18}	2005	{17}	nil
				2006	{18}	nil
		2004	{15}	Accord	{15}	nil
		2005	{16, 17}	Accord	{16}	nil
				Civic	{17}	nil
		2006	{18}	Civic	{18}	nil
Toyota	{19, 20, 21, 22}	Camry	{19, 20}	2004	{19}	nil
				2005	{20}	nil
		Corolla	{21, 22}	2004	{21}	nil
				2005	{22}	nil
		2004	{19, 21}	Camry	{19}	nil
				Corolla	{21}	nil
		2005	{20, 22}	Camry	{20}	nil
				Corolla	{22}	nil
Focus	{11}	Ford	{11}	2005	{11}	nil
		2005	{11}	Ford	{11}	nil
Taurus	{12, 13, 14}	Ford	{12, 13, 14}	2003	{12}	nil
				2004	{13}	nil
				2005	{14}	nil
		2003	{12}	Ford	{12}	nil
		2004	{13}	Ford	{13}	nil
		2005	{14}	Ford	{14}	nil
Accord	{15, 16}	Honda	{15, 16}	2004	{15}	nil
				2005	{16}	nil
		2004	{15}	Honda	{15}	nil
		2005	{16}	Honda	{16}	nil
Civic	{17, 18}	Honda	{17, 18}	2005	{17}	nil
				2006	{18}	nil
		2005	{17}	Honda	{17}	nil
		2006	{18}	Honda	{18}	nil
Camry	{19, 20}	Toyota	{19, 20}	2004	{19}	nil
				2005	{20}	nil
		2004	{19, 21}	Toyota	{19, 21}	nil
		2005	{20, 22}	Toyota	{20, 22}	nil
Corolla	{21, 22}	Toyota	{21, 22}	2004	{21}	nil
				2005	{22}	nil
		2004	{21}	Toyota	{21}	nil
		2005	{22}	Toyota	{22}	nil
2003	{12}	Ford	{12}	Taurus	{12}	nil
		Taurus	{12}	Ford	{12}	nil
2004	{13, 15, 19, 21}	Ford	{13}	Taurus	{13}	nil
		Honda	{15}	Accord	{15}	nil
		Toyota	{19, 21}	Camry	{19}	nil
				Corolla	{21}	nil
		Taurus	{13}	Ford	{13}	nil
		Accord	{15}	Honda	{15}	nil
		Camry	{19}	Toyota	{19}	nil
		Corolla	{21}	Toyota	{21}	nil
2005	{11, 14, 16, 17, 20, 22}	Ford	{11,14}	Focus	{11}	nil
				Taurus	{14}	nil
		Honda	{16, 17}	Accord	{16}	nil
				Civic	{17}	nil
		Toyota	{20, 22}	Camry	{20}	nil
				Corolla	{22}	nil
		Focus	{11}	Ford	{11}	nil
		Taurus	{14}	Ford	{14}	nil
		Accord	{16}	Honda	{16}	nil
		Civic	{17}	Honda	{17}	nil
		Camry	{20}	Toyota	{20}	nil
		Corolla	{22}	Toyota	{22}	nil
2006	{18}	Honda	{18}	Civic	{18}	nil
		Civic	{18}	Honda	{18}	nil

Table 3: Data structure storing the complete set of multi-step web FDs satisfied by the sample automobile website in Fig. 1.

Ford	nil	Focus	{2005}	nil
		2003	{Taurus}	nil
		2004	{Taurus}	nil
Honda	nil	2004	{Accord}	nil
		2006	{Civic}	nil
Toyota	nil	nil		
Focus	{Ford, 2005}	Ford	{2005}	nil
		2005	{Ford}	nil
Taurus	{Ford}	2003	{Ford}	nil
		2004	{Ford}	nil
		2005	{Ford}	nil
Accord	{Honda}	2004	{Honda}	nil
		2005	{Honda}	nil
Civic	{Honda}	2005	{Honda}	nil
		2006	{Honda}	nil
Camry	{Toyota}	2004	{Toyota}	nil
		2005	{Toyota}	nil
Corolla	{Toyota}	2004	{Toyota}	nil
		2005	{Toyota}	nil
2003	{Ford, Taurus}	Ford	{Taurus}	nil
		Taurus	{Ford}	nil
2004	nil	Ford	{Taurus}	nil
		Honda	{Accord}	nil
		Taurus	{Ford}	nil
		Accord	{Honda}	nil
		Camry	{Toyota}	nil
		Corolla	{Toyota}	nil
2005	nil	Focus	{Ford}	nil
		Taurus	{Ford}	nil
		Accord	{Honda}	nil
		Civic	{Honda}	nil
		Camry	{Toyota}	nil
		Corolla	{Toyota}	nil
2006	{Honda, Civic}	Honda	{Civic}	nil
		Civic	{Honda}	nil

```
Name: $mine{-}negative{-}path{-}webFDs$
Type: DAG → negative-path-webFDs {a list of (term, set of terms) pairs}
Parameters: DAG $D$
  $S \leftarrow sequencize(D)$ {returns the set of sequences from $D$}
  $\mathcal{T} \leftarrow getTerms(S)$ {returns the set of terms from $S$}
  $term{-}termIndex \leftarrow build{-}term{-}term{-}index(\mathcal{T},\ S)$
  for $i \leftarrow 0$ to $|\mathcal{T}|$ do
    $negative{-}path{-}webFDs[i](\#1) \leftarrow \mathcal{T}[i]$
    $negative{-}path{-}webFDs[i](\#2) \leftarrow \mathcal{T} - term{-}termIndex[i]$
  end for
  return $negative{-}path{-}webFDs$ {a list of (term, set of terms) pairs}
```

Algorithm 2: Mine negative-path web FDs, simplified for purposes of presentation.

Table 4: Structural characteristics of sites mined for web FDs. Key: Nl-x=non-leaf crosslink, L-x=leaf crosslink, F=faceted,T-n=total nodes, Nl-n=non-leaf nodes, L=leaf nodes, and S=sequences. ODP is not a DAG because some of its crosslinks create cycles.

Website Name	**URL**	**Type**	**Nl-x?**	**L-x?**	**Depth**	**F?**	**#T-n**	**#Nl-n**	**#L=#S**	**#Terms**	**μ Fan-out**
Project Vote Smart	vote-smart.org	tree	×	×	[4]	√	856	316	540	116	2.71
Citidel DL	citidel.org	DAG	×	√	[2–4]	×	6,066	551	5,515	4,715	11.00
Epicurious Recipes	epicurious.com	DAG	×	√	[2–7]	√	86,488	26,102	60,386	127	3.31
Edmunds Autos	edmunds.com	tree	×	×	[6]	√	15,826	11,326	4,500	5,195	1.40
ODP (Home)	dmoz.org	graph	√	√	[2–8]	×	2,059	493	2,026	1,667	5.11

5 Results

We mined web FDs in five hierarchical-oriented sites using an implementation[1] of the algorithm presented in the previous section. We chose these sites for the variety of structural characteristics they provide as well as their differences in size (see Table 4). Before providing the details of the web FDs mined from each site, we briefly discuss various characteristics of each dataset, how and when we collected them, and their structural differences.

We collected each dataset, except that from ODP (Open Directory Project at dmoz.org), by conducting a depth-first crawl of the site and extracting each hyperlink label during the traversal. ODP, on the other hand, makes its entire hierarchical hyperlinked structure in RDF[2] format available for free download through http://rdf.dmoz.org/rdf/structure.rdf.u8.gz. We collected the Edmunds dataset in August 2005, the Citidel and Epicurious datasets in December 2005, and the PVS dataset in March 2006. We extracted the dataset for ODP from the RDF structure file downloaded in November 2005. The dataset from PVS represents the entire Congressional hierarchy and that from Epicurious represents the entire recipe browsing classification. Citidel indexes documents through four popular classification schemes and the dataset we collected represents the unity of the complete hierarchy for each of the four through a common root and an edge to the root of each individual scheme labeled with the name of the scheme to which it links (e.g., CCS – ACM Computing Classification System). We mined web FDs between all four schemes and not just within each individual scheme. The Edmunds dataset contains a sequence for each automobile from 1990–2004 with attributes year, make, model, size, price, and fuel efficiency (miles per gallon). Due the immense size of ODP (i.e., approximately 690,000 nodes, over 218,000 distinct terms, and a maximum depth of 14), we only mined web FDs from the *Home* category.

We can draw several distinctions between hierarchical websites. For instance, some sites are modeled as trees, where there is only one path from the root to any leaf (i.e., a node which is the source of no edge). Others are modeled by non-tree DAGs (directed acyclic graphs), where there may be more than one path from the root to any leaf owing to the presence of *crosslinks* (sometimes called *symbolic links*). A crosslink is a special type of hyperlink which makes a directed connection between a webpage along one path through a site to a page along another path. Crosslinks are typically employed in human-compiled directories of websites, such as Yahoo! or ODP which are popular hubs to web resources. In these sites, crosslinks, whose labels end with @, are used to provide multiclassification, shortcuts deeper into the directory, and backlinks out of sub-categories or, in short, to make information access flexible. The *Arts* category of ODP has a crosslink labeled 'Museums@' whose target is a page in the *Reference* category. We distinguish between a *non-leaf crosslink*, whose target is a non-leaf node (i.e., the source of some edge) and a *leaf crosslink*, whose

[1] We implemented the multi-step web FD mining algorithm in Standard ML using less than 600 lines of code.

[2] Resource Description Framework, a data interchange format commonly used to describe web metadata for machine processing.

target is a leaf. This distinction is motivated by sites, especially those whose pre-leaf nodes represent a flat list of search results (e.g., Citidel, Epicurious, and ODP), which use crosslinks originating from nodes at the pre-leaf level to classify terminal objects through multiple sequences.

Since Citidel uses four tree-structured classification schemes it, as shown in Table 4, has only leaf crosslinks, i.e., only the documents themselves which it indexes may have more than one parent. Similarly, Epicurious only has leaf crosslinks. ODP has both non-leaf and leaf crosslinks. Thus, the leaf nodes of these sites have more than one parent. Leaves in Citidel, Epicurious, and ODP correspond to documents, recipes, and websites, respectively, while nodes at their pre-leaf levels correspond to sets of these items. Links into these leaves are labeled with the particular document title, recipe name, or website name/URL and this label is typically a key for the sequence. Thus, when mining web FDs in these sites, we consider sequences from the root down to only the pre-leaf level in order to foster a fair basis for comparison to the web FDs mined in the other sites considered here.

The hierarchies of the sites we mined also vary in depth. We define the *level* of an edge-label as the depth of the source of the edge it labels. The *minimum depth* of a DAG D is the level of an edge-label in D which is less than or equal to the level of all other edge-labels in D. In other words, minimum depth is the minimum number of clicks required to access a leaf, i.e., the depth of that leaf. The *maximum depth* of a DAG D is the level of an edge-label in D which is greater than or equal to the level of all other edge-labels in D. In other words, maximum depth is the maximum number of clicks required to access a leaf, i.e., again, the depth of that leaf. The column labeled *depth* in Table 4 provides the minimum and maximum depth (or, in other words, depth range or sequence-length range) of each site; in some sites, each sequence has a consistent length (see PVS and Edmunds) and, thus, require only a single number to represent depth. The sample website given Fig. 1 has a consistent sequence length.

We can also distinguish sites based on the relationship of the hyperlink labels on each webpage of the hierarchy. In *faceted* sites (Hearst et al., 2002), all of the hyperlink labels on a single webpage belong to only one facet of information assessment. The facets of PVS are ≺state, branch, party, seat/district≻ while those of Epicurious are ≺main ingredient, preparation method, cuisine, season/occasion, course/meal, dish, special considerations≻. Moreover, notice that the sample website illustrated in Fig. 1 is faceted (≺make, model, year≻ are its facets). In some sites however, such as Citidel and ODP, all of the hyperlink labels on each page have no discernible relationship and thus lack the concept of a facet. We called these sites *unfaceted*. Notice from Table 4 that a faceted site does not imply a consistent sequence length (see Epicurious).

Faceted sites present the opportunity to compress dependencies between values of facets (i.e., traditional web FDs) using dependencies between facets themselves. Essentially, web FDs become synonymous with database FDs. For example, we can say that the site in Fig. 1 and Edmunds satisfy FDs of the form $model \rightarrow make$ because a total, onto function exists between the values of the facets model and make, where model is the domain and make is the codomain.

The values for the number of nodes, non-leaf nodes, leaf nodes, and terms in Table 4 capture the volume of each site. Each term count does not include duplicates. Again, note that we define a term as a hyperlink label (e.g., 'Information Storage and Retrieval' is one term in Citidel) and this viewpoint is reflected in our terms counts. Lastly, we define average fan-out as the total fan-out (i.e., the total number of nodes minus one) divided by the total number of non-leaf nodes in the site.

Table 5 provides statistics on the number of completely nontrivial (positive- and negative-path) web FDs mined from each site. The column labeled '#Pos. web FDs' captures the number of completely nontrivial positive-path web FDs with only one term on the rhs. The column labeled '#Neg. web FDs' gives the number of negative-path web FDs with only one term on both the lhs and rhs. We provide the average,

Table 5: Statistics on the number of completely nontrivial (positive- and negative-path) web FDs mined from each website.

Website	#Pos. web FDs	#Neg. web FDs	μ #Neg. web FDs	σ^2 #Neg. web FDs	σ #Neg. web FDs
Project Vote Smart	715	11,556	99.62	342.15	18.50
Citidel DL	13,266	1,798,016	191,353.48	1.42×10^{12}	1,189,953.43
Epicurious Recipes	41,653	4,932	42.52	306.29	17.50
Edmunds Autos	49,304	26,919,082	232,061.05	2.31×10^{12}	1,518,303.92
ODP (Home)	6,188	2,765,444	23,840.03	7,070,405,517.77	84,085.70

variance, and standard deviation of the number of negative web FDs involving the same term on the lhs in the columns labeled 'μ #Neg. web FDs','σ^2 #Neg. web FDs', and 'σ #Neg. web FDs', respectively. These are better motivated later.

Given our experimental results (Table 5), we draw no conclusion about the relationship between the number of sequences through a site and the number of positive web FDs it satisfies. This re-affirms that web FDs are a function of the term-term co-occurrences along the sequences of the site and an artifact of the underlying domain rather than simply of the number of sequences through the site. Notice that the (#sequences, #positive web FDs) point provided by the Edmunds dataset appears to be an outlier wrt number of sequences vs. number of positive web FDs. The Edmunds dataset contains only 4,500 sequences, yet satisfies nearly 50,000 positive web FDs. Without that point, the number of sequences and positive web FDs are correlated linearly with r^2=0.95. Similarly, we draw no conclusion about the relationship between the number of sequences through a site and the number of negative web FDs it satisfies. When analyzing the relationship between the number of sequences through a site and the number of negative FDs it satisfies, the (#sequences, #negative web FDs) data points provided by Epicurious and Citidel appear to be outliers. The Epicurious dataset contains over 60,000 sequences, but less than 5,000 negative web FDs. On the other hand, the Citidel dataset contains less than 6,000 sequences, yet satisfies nearly 2 million negative web FDs. As we will see later, terms in sites with many negative FDs, and particularly a high average number of negative FDs per term, can be used as a basis to perform a great deal of website pruning. Without those points, the number of sequences and negative web FDs are linearly correlated with r^2=0.92. Overall, our results indicate that web FDs occur naturally in a variety of sites across the web. Moreover, our results show that the presence (not to be confused with volume or length) of FDs is not affected by the distinctions between these sites.

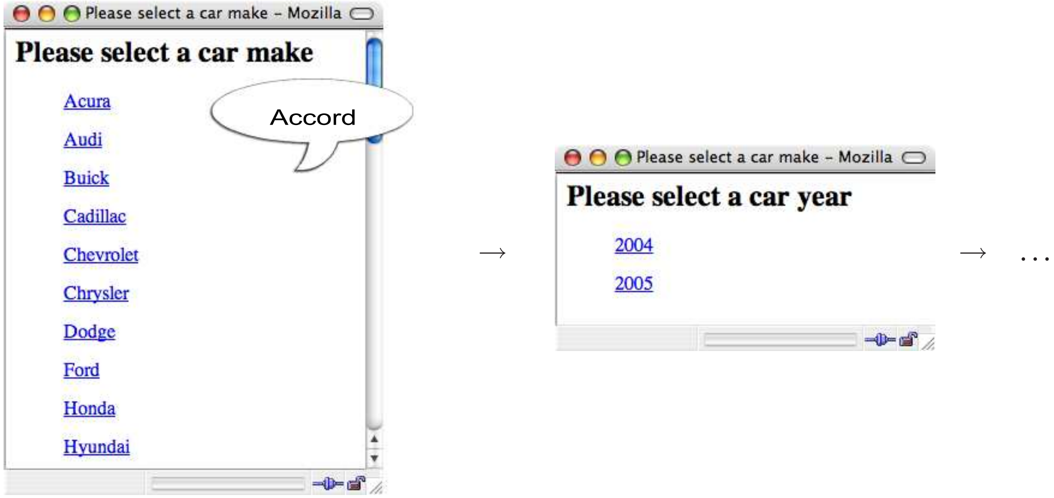
6 Uses of Web Functional Dependencies in Information Seeking

Web FDs serve multiple uses, from enhancing interaction experiences of users, to helping communicate the structure of a website by capturing relationships between site facets. We now showcase some of the more promising of these uses.

6.1 Real-time Query Expansion

Implicit Expansion

One use of web FDs is as a means to conduct query expansion while a user is searching or browsing a web hierarchy. We use *out-of-turn interaction*, a technique for integrating browsing and searching a web hierarchy, to illustrate this use of web FDs. Out-of-turn interaction permits the user, at any point while browsing a hierarchy, to supply terms corresponding to their information need. These terms are then matched


Please select a car make – Mozilla

Please select a car make

Acura
Audi
Buick
Cadillac
Chevrolet
Chrysler
Dodge
Ford
Honda
Hyundai

Please select a car make – Mozilla

Please select a car year

2004
2005

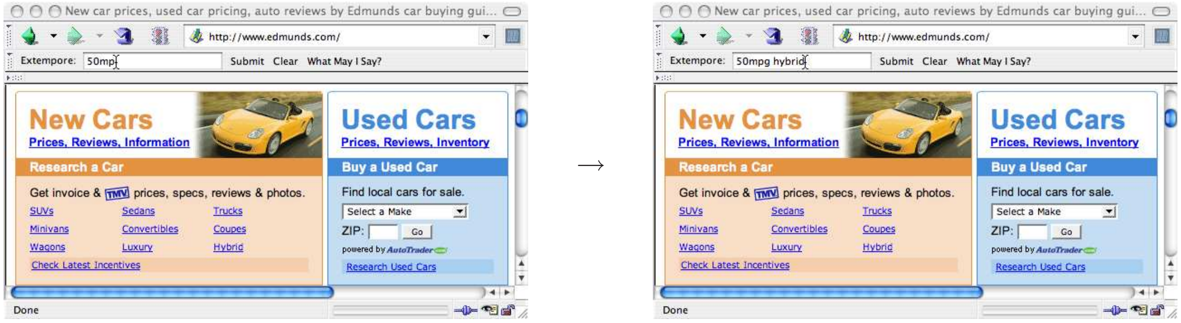
Figure 2: Use of an out-of-turn interaction interface through a voice-enabled browser. (left) The user decides to not pursue any of the presented hyperlinks and instead speaks 'Accord' out-of-turn. This triggers the Accord → Honda web FD. The query is thus expanded to 'Accord Honda' and results in the webpage (right) with links representing the only two remaining car years. The dialog continues in this manner.

against all of the hyperlink labels in the hierarchy rooted at the current page to select the sequences through the site in which they appear. Out-of-turn interaction returns a hierarchy containing only these sequences to sustain the semblance of hierarchy and the progressive drill down it.

There are multiple interpretations for an out-of-turn interaction. A simple interpretation is to retain only those sequences through the site containing a hyperlink labeled with the out-of-turn input, prune all others, and shrink those remaining by removing the edges labeled with the out-of-turn input. For instance, when a user supplies 'Accord' out-of-turn to the site depicted in Fig. 1, we would return a tree with a root with only one link to the sub-site rooted at page 7, i.e., {(1, Honda, 7), (7, 2004, 15), (7, 2005, 16)}[3]. We support the user in supplying out-of-turn inputs through two interfaces: a toolbar embedded into a web browser, and not the site's pages, and a voice-user interface. We have applied out-of-turn interaction to several websites. We refer the reader interested in the details of out-of-turn interaction, including a software framework supporting it as well as these user interfaces for it, to (Narayan et al., 2004; Perugini & Ramakrishnan, 2006).

The nature of the website pruning conducted with out-of-turn interaction often results in hierarchies with several pages containing only one hyperlink, as is seen in the prior example. Using web FDs to expand the user query addresses this situation. For instance, when a user supplies 'Accord' out-of-turn to the site depicted in Fig. 1, we would use the web FD 'Accord → Honda' (satisfied by that site) to expand the query to 'Accord Honda'. This query would result in the tree rooted at page 7, which has more than one hyperlink. Fig. 2 depicts this web interaction using a voice modality to supply terms out-of-turn. Such expansions produced using web FDs ensures that the user will never be presented with a webpage containing only one hyperlink (because all others were pruned out) in response to a query. Moreover, expansions in effect provide several *shortcuts* throughout a web hierarchy. The shortcut induced through the prior interaction bypasses 2 successive hardwired hyperlinks – (1, Honda, 3) and (3, Accord, 7). Shortcuts induced by web FDs will sometimes lead the user directly to terminal information. For example, a query for 'Focus' in Fig. 1

[3] Here a tree is represented as a set of edges, where an edge is a (source, label, target) triple.



Extempore: 50mpg

Extempore: 50mpg hybrid

Figure 3: Illustration of explicit, real-time query expansion built-in into the *Extempore* toolbar (see top of each browser window) for out-of-turn interaction. The user's query is expanded from '50mpg' (left) to '50mpg hybrid' (right) because 'hybrid' is inferred by functional dependency.

retrieves page 11 in one stroke by triggering the 'Focus → {2005, Ford}' web FD. Also, notice that web FDs can be used, in the same manner, to expand a query implicitly initiated through a simple hyperlink click (i.e., the label of the hyperlink is the query; this can be called *in-turn interaction*). Since any hyperlink label contained in the hierarchy rooted at the current page is a candidate for the lhs of a web FD and available as out-of-turn input, we can use the complete set of web FDs satisfied by a site for query expansion.

Explicit Expansion

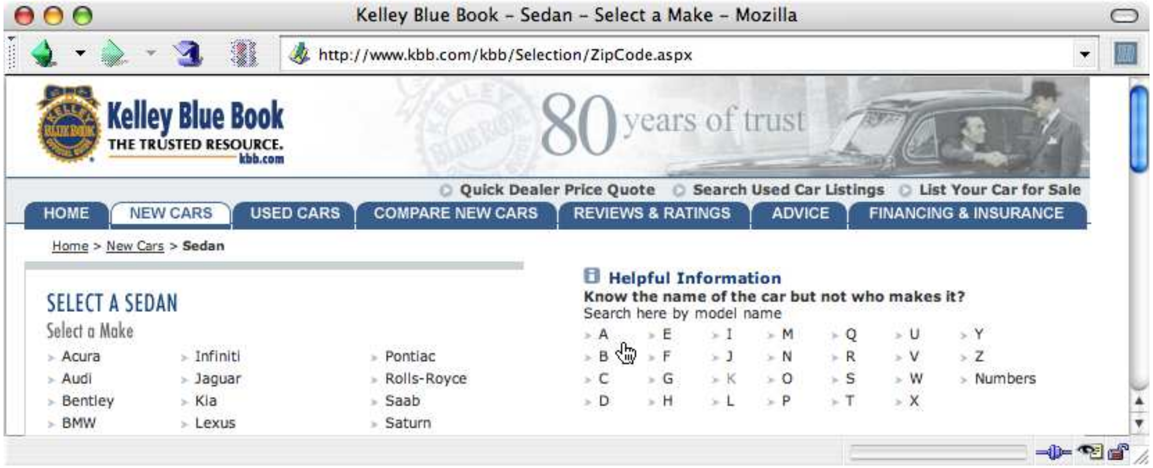
Notice that the query expansion discussed above is internal to the system and only apparent to the attentive user who notices that after supplying 'Accord', e.g., out-of-turn, the system never solicits for automobile make (because it was inferred by functional dependency). Thus, the user is only able to discern that expansion has taken place *after* their query has been processed. Alternatively, we can expand the query *in situ*. For instance, in the example given above, when the user enters the final 'd' in the specification of the term 'Accord', in real-time the system would expand that query to 'Accord Honda' prior to submission and this expansion would be visible in, e.g., the textbox in which the original query was entered by now containing the fully-expanded query 'Accord Honda'.

Such expansion built directly into the interface provides an additional layer of information exploration and discovery independent of that provided by the underlying (site pruning, transformation) system. For instance, consider the following information-seeking goal:

> "I am only interested in cars whose fuel efficiency is greater than 50 miles per gallon. However, I don't want a hybrid engine."

A user pursuing this goal, upon entering the final 'g' in '50mpg' into the out-of-turn toolbar (see Fig. 3, left), which we call *Extempore*, observes that the query expands to '50mpg hybrid' (see Fig. 3, right) and thus realizes that the site contains no cars meeting the current specification. The revelation of this constraint at this point may compel the user not to submit the query.

Readers may be familiar with the use of similar *in situ* expansion in the popular *auto-completion* feature now standard in many e-mail clients and web browsers. The presence of this feature in an e-mail application means that while a user is entering an e-mail address in the 'To:' field, the system searches the user's personal history cache, containing the address of individuals to whom the user has previously sent mail, for an e-mail with a prefix matching the partially-completed address. When a match is found, the partial address



SELECT A SEDAN

Select a Make

- Acura
- Audi
- Bentley
- BMW
- Infiniti
- Jaguar
- Kia
- Lexus
- Pontiac
- Rolls-Royce
- Saab
- Saturn

Helpful Information

Know the name of the car but not who makes it?

Search here by model name

A	E	I	M	Q	U	Y
B	F	J	N	R	V	Z
C	G	K	O	S	W	Numbers
D	H	L	P	T	X	

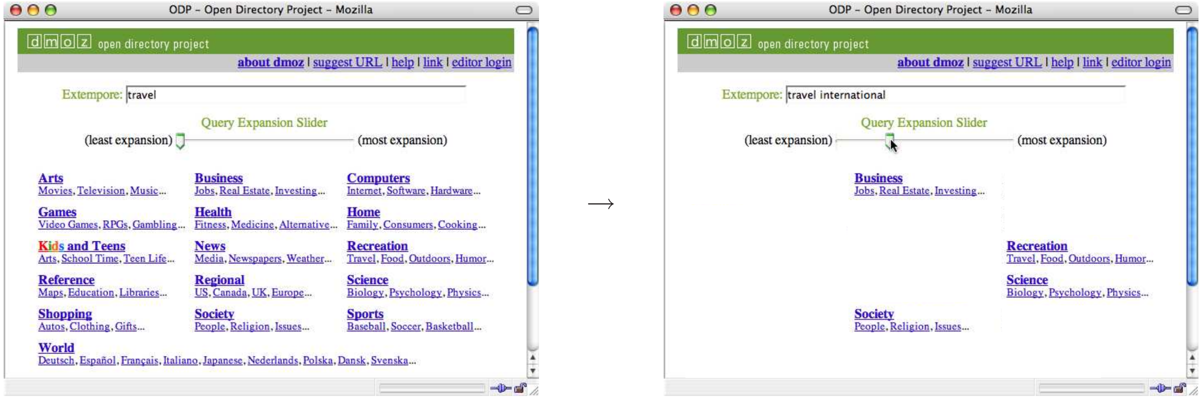
Figure 4: Automobile-make lookup service provided by Kelley Blue Book online. Here a user searches for an automobile make based on the model. This service exploits FDs of the form $make \rightarrow model$.

is expanded into the matching address. Web browsers employ a similar technique to expand partial URLs into those of sites previously visited. While the concept of such real-time, *in situ* expansion is ubiquitous in these desktop applications, the real-time query expansion based on web FDs is fundamental different. The expansion in the auto-completion feature depends on the current status of the individual user's cache. In other words, two different users are not guaranteed the same expansion for the same initial input. On the other hand, real-time query expansion depends on the set of web FDs currently satisfied by the (currently instance of the) site. Therefore, we say auto-completion is *user-dependent* and real-time query expansion is *user-independent*.

We feel that the feedback provided by simply interacting with a real-time query expansion interface (without perhaps even ever submitting a query) to out-of-turn interaction can be helpful in decomposing and solving a complex constraint satisfaction problem such as planning a vacation. Moreover, we feel that due to its rapid query–expand–feedback loop and ability to naturally expose dependencies in, including those most central to, the underlying domain, it has use as an instrument to help users assimilate a new information domain in an exploratory setting. For example, in PVS, this interface makes dependencies, such as 'Democrat → ¬Republican', 'Republican → ¬Democrat', 'House → ¬Senate', 'Senate → ¬House', 'Senior seat → ¬Junior seat', and 'Junior seat → ¬Senior set', which are at the core of the structure of the US government, salient. For this reason we say the set of web FDs satisfied by a site operationalizes domain knowledge encoded its hierarchy.

Lastly, note that since the website is pruned and reduced as the user interacts with it, through techniques such as out-of-turn interaction and browsing, it is practical and efficient to compute web FDs satisfied by (the most recent version of) the site in *real-time*. For instance, the web FD 'Ohio, Senate → Republican' holds in the pristine PVS site because both Ohio Senators are Republicans[4]. After a user communicates 'Ohio' to PVS, the site contains only those (20) sequences involving Ohio congresspeople. These 20 sequences satisfy the 'Senate →Republican' web FD (not satisfied by the original site) among others. However, due to the real-time query expansion application of web FDs, we need not mine the complete set of FDs satisfied by this reduced set of sequences ever! Rather we can take a greedy approach and mine only the one web FD

[4] This was the scenario during the 2006 congressional landscape.



ODP – Open Directory Project – Mozilla
dmoz open directory project
about dmoz | suggest URL | help | link | editor login
Query Expansion Slider
(least expansion)
(most expansion)
Arts
Movies, Television, Music...
Business
Jobs, Real Estate, Investing...
Computers
Internet, Software, Hardware...
Games
Video Games, RPGs, Gambling...
Health
Fitness, Medicine, Alternative...
Home
Family, Consumers, Cooking...
Kids and Teens
Arts, School Time, Teen Life...
News
Media, Newspapers, Weather...
Recreation
Travel, Food, Outdoors, Humor...
Reference
Maps, Education, Libraries...
Regional
US, Canada, UK, Europe...
Science
Biology, Psychology, Physics...
Shopping
Autos, Clothing, Gifts...
Society
People, Religion, Issues...
Sports
Baseball, Soccer, Basketball...
World
Deutsch, Español, Français, Italiano, Japanese, Nederlands, Polska, Dansk, Svenska...
→
ODP – Open Directory Project – Mozilla
dmoz open directory project
about dmoz | suggest URL | help | link | editor login
Query Expansion Slider
(least expansion)
(most expansion)
Business
Jobs, Real Estate, Investing...
Recreation
Travel, Food, Outdoors, Humor...
Science
Biology, Psychology, Physics...
Society
People, Religion, Issues...

Extempore: travel

Extempore: travel international

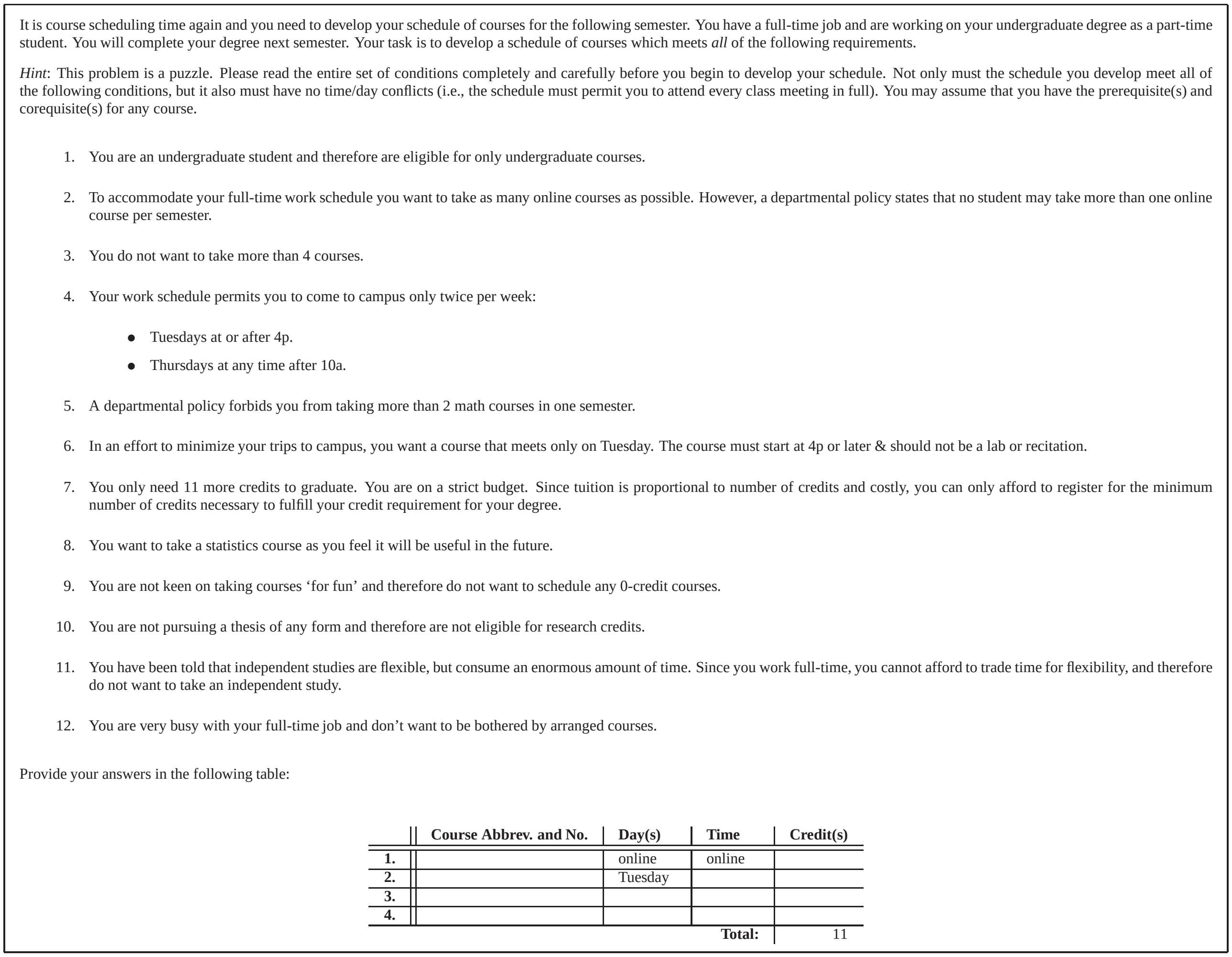
Figure 5: Slider-bar interface to approximate retrieval in web hierarchies. Here 'least expansion' means the strictest interpretation of a web FD, i.e., confidence equal to 1, which results in less website pruning. (left) User enters 'travel' and all links at the top level remain because there is a sequence beginning with each of their labels which contains the term 'travel.' (right) As the user slides the dial to the right, expansions are considered where the web FDs triggering them are computed with confidence less than 1. This cause more expansion — the term 'international' is added – and more site pruning. Only links labeled 'Business,' 'Recreation,' 'Science,' and 'Society' remain because they are the only categories of the site which contain sequences involving both terms 'travel' and 'international.'

applicable to the expansion immediately before it is required. For example, should the user enter 'Senate' next, after processing the input and retaining only those (2) sequences which contain that term, we identify terms contained in every remaining sequence, one of which is the term 'Republican.' This mining for web FDs in real-time is *linear* in the number of remaining sequences, which in this case is two. Notice further that the 'Senate→Republican' web FD is also attainable by simply removing the supplied term (Ohio) from the lhs of the '$\{$Ohio, Senate$\}$ → Republican' FD satisfied by the unaltered site.

6.2 Approximate Retrieval in Web Hierarchies

By generalizing the notion of a web FD, we can make retrieval in web hierarchies approximate. Consider that the definition of a web FD — a positive-path web FD $x \rightarrow y$ indicates that *all* sequences involving x also involve y — is rigid at two levels: first, in the specific metric used to compute a score for the candidate FD (i.e., confidence), and, second, in the threshold that the score must meet to be considered a web FD (i.e., 100%). We can generalize this notion by parameterizing the definition by the metric and threshold. Thus, we say that a positive-path web FD $x \rightarrow y$ holds if $M(x, y) \geqslant T$, where M is a similarity metric and T is a threshold between 0 and 1. With this more general definition, a host of similarity metrics from IR, such as cosine or *Jaccard's* (Srehl, Ghosh, & Mooney, 2000), each with different properties (e.g., symmetry, i.e., $a \rightarrow b$ implies $b \rightarrow a$ and vice versa), and threshold values are available for use in computing a complete set of web FDs. We can experiment to test the effect of various (metric, threshold) pair on the nature of web FDs mined and, ultimately, on retrieval in web hierarchies.

Making retrieval in web hierarchies approximate is particularly helpful in voluminous web directories such as ODP. Since such sites are predominately unfaceted, supplying 1–2 terms (out-of-turn) may only reduce the sequences of the hierarchy by 5–10%. Using a threshold less than 100% when mining web FDs will increase the size of the query and reduce the size of answer and, thus, information overload. The



It is course scheduling time again and you need to develop your schedule of courses for the following semester. You have a full-time job and are working on your undergraduate degree as a part-time student. You will complete your degree next semester. Your task is to develop a schedule of courses which meets *all* of the following requirements.

Hint: This problem is a puzzle. Please read the entire set of conditions completely and carefully before you begin to develop your schedule. Not only must the schedule you develop meet all of the following conditions, but it also must have no time/day conflicts (i.e., the schedule must permit you to attend every class meeting in full). You may assume that you have the prerequisite(s) and corequisite(s) for any course.

1. You are an undergraduate student and therefore are eligible for only undergraduate courses.
2. To accommodate your full-time work schedule you want to take as many online courses as possible. However, a departmental policy states that no student may take more than one online course per semester.
3. You do not want to take more than 4 courses.
4. Your work schedule permits you to come to campus only twice per week:
   - Tuesdays at or after 4p.
   - Thursdays at any time after 10a.
5. A departmental policy forbids you from taking more than 2 math courses in one semester.
6. In an effort to minimize your trips to campus, you want a course that meets only on Tuesday. The course must start at 4p or later & should not be a lab or recitation.
7. You only need 11 more credits to graduate. You are on a strict budget. Since tuition is proportional to number of credits and costly, you can only afford to register for the minimum number of credits necessary to fulfill your credit requirement for your degree.
8. You want to take a statistics course as you feel it will be useful in the future.
9. You are not keen on taking courses ‘for fun’ and therefore do not want to schedule any 0-credit courses.
10. You are not pursuing a thesis of any form and therefore are not eligible for research credits.
11. You have been told that independent studies are flexible, but consume an enormous amount of time. Since you work full-time, you cannot afford to trade time for flexibility, and therefore do not want to take an independent study.
12. You are very busy with your full-time job and don’t want to be bothered by arranged courses.

Provide your answers in the following table:

	Course Abbrev. and No.	**Day(s)**	**Time**	**Credit(s)**
1.		online	online	
2.		Tuesday		
3.				
4.				
			Total:	11

Figure 6: An ambitious problem-solving task involving constraints and dependencies.

generalization of web FDs in our framework not only makes retrieval approximate, but also leads to novel user interface designs, especially those which give the user control over the query expansion threshold (or metric).

For example, we have designed a slider bar interface (see Fig. 5) which allows the user to control the amount of query expansion used and concomitantly observe how links on the current webpage are dynamically pruned out (as the sequences containing them are removed). When the slider is flush to the left (‘least expansion’), we use confidence of 1 to compute web FDs and, therefore, there is still some expansion. As the user shifts the slider to the right, the confidence with which we compute web FDs is reduced, thereby inducing more query expansion and, as a result, more pruning. Notice that dynamic link pruning obviates the need for a ‘Submit’ button; the query is submitted in real-time (i.e., as the user types or shift the slider). The reader will notice that positive- and negative-path web FDs are a continuum of the same concept and difference between the two only depends on where one draws the threshold.

6.3 User Studies

We have conducted user studies of various interface designs (e.g., a non-interactive, flat HTML table; a web hierarchy; a faceted classification, and out-of-turn interaction), some of which exploit web FDs, for interacting with a variety of hierarchical-oriented websites. A user study of out-of-turn interaction applied to PVS and its results are reported in (Perugini, 2004). In this study, twenty-four users were given eight information-finding tasks about US politicians and were free to use either in-turn interaction (browsing) or out-of-turn interactions to complete these tasks. Half of these tasks were *non-oriented* (meaning they could be performed with browsing alone, if desired) and the other half were *out-of-turn-oriented* (meaning they would be cumbersome to perform via plain browsing). This experimental design generated 192 (= 24 × 8) interaction sequences (participant, task) pairs. We found that 100% of the users utilized out-of-turn interaction when presented with an out-of-turn-oriented task. Since the task type was not disclosed *a priori*, this result demonstrates that users are adept at discerning when out-of-turn interaction is desirable and actively interleaved it with browsing. However, one task of this study — find the political party of the senior senator representing the only state which has congresspeople from the Independent party — was completed successfully by only half of the participants. Notice that this task is procedural in nature. The user must first find the only state with a congressperson from the Independent party and then use that state to find the party of the senior senator from that state.

The pursuit of constraint-satisfaction problems, which are common on the web (e.g., consider comparison shopping or planning travel), tend to be procedural as well. For instance, consider the constraint-satisfaction problem given in Fig. 6 which we have used in a pilot study. In this problem, the user's task is to develop a schedule of courses which meet all of the given constraints. In response to the challenge of completing such procedural tasks, we are now developing continuation-based interfaces for them which not only expose/exploit web FDs as espoused in this paper, but also permit the user to cascade the output from one thread of information-seeking to the input of another (Perugini & Ramakrishnan, 2006).

7 Related Research

Our work lies in the area of web mining. The predominant thrust in most web mining research focuses on the discovery of patterns in site usage data, e.g., a web log, (referred to as *web usage mining*) (Eirinaki & Vazirgiannis, 2003). Our work differs from this in that we examine site structure and the relationships between terms, labeling hyperlinks, implicit along that structure, and would be considered *web structure mining*. Existing approaches based on usage mining rely on the availability of usage data, whereas our techniques are applicable more readily. The patterns resulting from web usage mining, often association rules, are typically used to induce new paths, such as shortcuts, through the sites as well as index pages. We, on the other hand, use web FDs primarily for query expansion (which under certain circumstances creates shortcuts as well) and the feedback it provides for the user. Moreover, association rules mined from web logs are *usage-dependent*; web FDs are *usage-independent*. In short, we mine different data and offer an alternate, but complementary, use. Despite these differences, the two approaches can complement each other. For example, the expansions induced by web FDs create sequences in the web log that would otherwise not exist (because they are not hardcoded into the site) and, therefore, expand the scope of usage data from which to mine. Furthermore, since the patterns resulting from both approaches are most similar to association rules, we can draw from similar algorithms. An algorithm which mines traversal patterns in web usage logs is relevant and given in (Chen et al., 1998). Algorithms for mining approximate FDs from databases are also helpful (Huhtala, Krkkinen, Porkka, & Toivonen, 1999). Others (Nambiar & Kambhampati, 2004) offer similar algorithms for mining approximate FDs and use the approximate FDs to create precise queries from imprecise web queries. In addition, algorithms for mining positive- and negative association rules (Wu, Zhang, & Zhang, 2004) are helpful for identifying positive- and negative-path web FDs.

Others have studied term and item similarity in hierarchies. For instance, (Resnik, 1999) studies term similarity to help resolve syntactic and semantic ambiguity in natural language processing. The focus of other work has been on collaborative-filtering and item recommendation. For example, (Ganesan, García-Molina, & Widom, 2003) study the similarity of users through hierarchical representations of their item sets. We primarily use web FDs for query expansion. Query expansion is a well studied area of information retrieval research. However, most of the methods used to expand query terms are probabilistic (Carpineto, DeMori, Romano, & Bigi, 2001; Cui, Wen, Nie, & Ma, 2002) and based on the queries of prior users, again, emphasizing the role of usage logs in these approaches, or use terms from initially-retrieved, top-ranked documents to expand the query (Mano & Ogawa, 2001). Recently, term relationships in languages models have been used for query expansion (Bai, Song, Bruza, Nie, & Cao, 2005). In addition, others have used navigation as a means to expand queries in an approach called *query by navigation* (Bruza & Dennis, 1997). However, this approach still relies on logs for requirements. Some research (Ruthven, 2003) has found that interactive query expansion needs mechanisms for *users* to explore and discover relationships between terms. This work highlights the importance of interfaces such as the *in situ*, real-time query expansion interface illustrated in this article.

Other user interfaces for information search and exploration on the web are starting to incorporate similar uses of query expansion. We direct the reader to *Google Suggest* (http://www.google.com/webhp?complete=1&hl=en) and Stanford's auto-complete *Search on TAP* (http://sp06.stanford.edu) systems for two popular examples. Moreover, Kelley Blue Book online (kbb.com) provides an automobile-make lookup (by auto model) service that exploits functional dependencies of the form $model \rightarrow make$ (see Fig. 4). In summary, we distinguish our work from that of others by mining a similar type of pattern (associations), however, from a different type of data (hyperlink structure) and for alternate use (query expansion and out-of-turn interaction).

8 Discussion

We have presented several classes of web FDs and situated them for the many roles they play in enhancing information access. As discussed here, web FDs achieve two goals: they generalize schema FDs by modeling value dependencies and, when used for facilitating web interactions, also introduce a temporal aspect (i.e., arrival time of terms) as consideration for capturing dependencies. We have presented direct uses of web FDs in pruning sites based on partial input, and incrementally communicating the structure of the site to the user, as the interaction progresses.

An emerging area of research is to be able to communicate richer constraints underlying the site and, in this way, provide more complex tools for problem solving. In this vein, two promising avenues of future research arise. First, we intend to study how web functional dependencies can be used as building blocks to more expressive patterns, which we call *website axioms*. An example of an axiom in a university's online timetable of courses might be: 'it is impossible to develop a schedule of classes which meet only on Mondays and Thursdays.' Once identified, we are optimistic that such axioms will be helpful in constraint-satisfaction problems given an appropriate method of exposing them to users. Second, we plan to study how the dependencies underling a site can be harnessed to help a user decompose a complex problem-solving task into a sequence of information-seeking procedures for pursuit through a procedural-oriented interface. In other words, can web FDs be used to compose or reveal interactive workflows? Since constraint-satisfaction is ingredient central to several web interactions, we feel that this line of future research is particularly worthwhile.

References

Agrawal, R., Imielinski, T., & Swami, A. N. (1993, May). Mining Association Rules between Sets of Items in Large Databases. In P. Buneman & S. Jajodia (Eds.), *Proceedings of the ACM International Conference on Management of Data (SIGMOD)* (pp. 207–216). Washington, DC: ACM Press.

Agrawal, R., & Srikant, R. (1994). Fast Algorithms for Mining Association Rules. In *Proceedings of the International Conference on Very Large Databases (VLDB).*

Bai, J., Song, D., Bruza, P. D., Nie, J.-Y., & Cao, G. (2005, October–November). Query Expansion using Term Relationships in Language Models for Information Retrieval. In *Proceedings of the Seventh International Conference on Information Knowledge Management (CIKM)* (pp. 688–695). Bremen, Germany: ACM Press.

Bruza, P. D., & Dennis, S. (1997, June). Query Reformulation on the Internet: Empirical Data and the Hyperindex Search Engine. In *Proceedings of the RIAO'97 Conference on Computer-Assisted Information Searching on Internet* (pp. 488–499). Montreal, Canada: ACM Press.

Carpineto, C., DeMori, R., Romano, G., & Bigi, B. (2001, January). An Information-theoretic Approach to Automatic Query Expansion. *ACM Transactions on Information Systems*, *Vol. 19*(1), pages 1–27.

Chen, M.-S., Park, J. S., & Yu, P. S. (1998, March–April). Efficient Data Mining for Path Traversal Patterns. *IEEE Transactions on Knowledge and Data Engineering*, *Vol. 10*(2), pages 209–221.

Cui, H., Wen, J.-R., Nie, J.-Y., & Ma, W.-Y.-. (2002). Probabilistic Query Expansion Using Query Logs. In *Proceedings of the eleventh international world wide web conference (www11)* (pp. pages 325–332). Honolulu, HI: ACM Press.

Eirinaki, M., & Vazirgiannis, M. (2003). Web Mining for Web Personalization. *ACM Transactions on Internet Technology*, *3*(1), pages 1–27.

Ganesan, P., García-Molina, H., & Widom, J. (2003, January). Exploiting Hierarchical Domain Structure to Computer Similarity. *ACM Transactions on Information Systems*, *Vol. 21*(1), pages 63–93.

Garcia-Molina, H., Ullman, J. D., & Widom, J. D. (2002). *Database Systems: The Complete Book*. Upper Saddle River, NJ: Prentice Hall.

Hearst, M. A., Elliott, A., English, J., Sinha, R., Swearingen, K., & Yee, K.-P. (2002, September). Finding the Flow in Web Site Search. *Communications of the ACM*, *Vol. 45*(9), pages 42–49.

Huhtala, Y., Krkkinen, J., Porkka, P., & Toivonen, H. (1999). TANE: An Efficient Algorithm for Discovering Functional and Approximate Dependencies. *The Computer Journal*, *Vol. 42*(2), pages 100–111.

Kamdar, T., & Joshi, A. (2005). Using Incremental Web Log Mining to Create Adaptive Web Servers. *International Journal on Digital Libraries*, *Vol. 5*(2), pages 133–150.

Kolari, P., & Joshi, A. (2004, July–August). Web Mining: Research and Practice. *IEEE Computing in Science and Engineering*, *Vol. 6*(4), pages 49–53.

Mano, H., & Ogawa, Y. (2001, September). Selecting Expansion Terms in Automatic Query Expansion. In *Proceedings of the Twenty-fourth Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR)* (pp. 390–391). New Orleans, LA: ACM Press.

Nambiar, U., & Kambhampati, S. (2004, June). Mining Approximate Functional Dependencies and Concept Similarities to Answer Imprecise Queries. In *Proceedings of the Seventh International Workshop on the Web and Databases (WebDB)* (pp. 73–78). Paris, France.

Narayan, M., Williams, C., Perugini, S., & Ramakrishnan, N. (2004, May). Staging Transformations for Multimodal Web Interaction Management. In *Proceedings of the thirteenth international acm world wide web conference (www)* (pp. 212–223). New York, NY: ACM Press.

Perugini, S. (2004). *Program Transformations for Information Personalization*. Ph.D. dissertation, Department of Computer Science, Virginia Tech. (Available in the Virginia Tech ETD collection at http://scholar.lib.vt.edu/theses/available/etd-06252004-162449/. US Copyright Office Registration Number TX 6-040-012)

Perugini, S., & Ramakrishnan, N. (2003, March–April). Personalizing Web Sites with Mixed-Initiative Interaction. *IEEE IT Professional*, *Vol. 5*(2), pages 9–15.

Perugini, S., & Ramakrishnan, N. (2006, July–August). Interacting with Web Hierarchies. *IEEE IT Professional*, *Vol. 8*(4), pp. 19–28.

Resnik, P. (1999). Semantic Similarity in a Taxonomy: An Information-Based Measure and its Application to Problems of Ambiguity in Natural Language. *Journal of Artificial Intelligence Research*, *Vol. 11*, pages 95–130.

Ruthven, I. (2003, July). Re-examining the Potential Effectiveness of Interactive Query Expansion. In *Proceedings of the Twenty-sixth Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR)* (pp. 213–220). Toronto, Canada: ACM Press.

Silberschatz, A., Korth, H. F., & Sudarshan, S. (2006). *Database System Concepts* (Fifth ed.). Boston, MA: McGraw-Hill.

Srehl, A., Ghosh, J., & Mooney, R. (2000, July). Impact of Similiarity Measures on Web-page Clustering. In *Proceedings of the AAAI Worshop of Artificial Intelligence for Web Search* (pp. 58–64). Austin, TX: AAAI/MIT Press.

Wu, X., Zhang, C., & Zhang, S. (2004, July). Efficient Mining of Both Positive and Negative Association Rules. *ACM Transactions on Information Systems, Vol. 22*(3), pages 381–405.