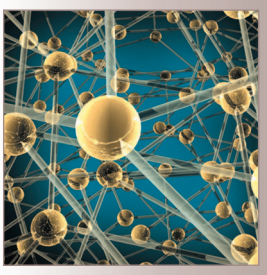

# Toward Recommendation Based on Ontology-Powered Web-Usage Mining

Content adaptation on the Web reduces available information to a subset that matches a user's anticipated needs. Recommender systems rely on relevance scores for individual content items; in particular, pattern-based recommendation exploits co-occurrences of items in user sessions to ground any guesses about relevancy. To enhance the discovered patterns' quality, the authors propose using metadata about the content that they assume is stored in a domain ontology. Their approach comprises a dedicated pattern space built on top of the ontology, navigation primitives, mining methods, and recommendation techniques.

Today, Web users are submerged in all kinds of available information, yet only a tiny part of it is usually relevant to their preferences. *Content adaptation*[1] aims to guide users toward items of interest while reducing the proposed content's quantity and variety. Recommender systems perform active adaptation by matching content objects against a user's anticipated needs based on relevance. Such a system approximates the unknown object-to-user relevance from information about similar users or objects, whereas pattern mining relies on an analysis of past experiences registered in user logs to predict the best "next" item.

Our association-based recommendation approach uses click-stream analysis[2] powered by an explicit representation of domain knowledge. We hypothesize that such knowledge should help increase both the relevance and interpretability of discovered patterns. Thus, we assume a domain ontology — in this article, e-tourism — that describes content objects as instances of generic categories (such as cities, museums, beaches, and so on) with various relations (such as location, proximity, and so on). Consequently, instead of raw click-streams, we process content-object sequences enhanced with semantic links. The discovered patterns are themselves sequences of domain con-

**Mehdi Adda and Petko Valtchev**
*University of Montreal*

**Rokia Missaoui**
*University of Quebec in Outaouais*

**Chabane Djeraba**
*University of Lille, France*


1089-7801/07/$25.00 © 2007 IEEE

cepts; the system uses them in the recommendation process as plausible scenarios to match against ongoing user sessions.

The resulting mining problem combines aspects of *structured* and *generalized* pattern discovery, which, to the best of our knowledge, haven't been studied before in this context. The problem's key difficulty is the patterns' structural complexity, which reflects both the ontology's *is-a* hierarchy and its network of properties. A previous work[3] introduced a plausible solution's key elements — namely, a pattern space and an algorithm that mines it. This article describes a complete recommendation framework and provides empirical evidence about the mining method's robustness.

## Pattern-Based Recommendation

Pattern-based recommendation uses the patterns identified by mining previous user session logs. By tracing these patterns, the system can guess, with higher probability, a user's "next step" and hence suggest relevant content. To that end, it matches the sequence of content objects already visited by the user against the pattern set and finds all patterns having a prefix corresponding to that sequence. The remainder of such a pattern is a hint as to the way the session might unfold.

### Pattern Mining

Given a universe of items $O$, a database $\mathcal{D}$ of records that combine items from $O$, and a frequency threshold $\sigma$, pattern mining amounts to extracting the family $\mathcal{F}_\sigma$ of the patterns present in at least $\sigma$ records. Two languages (the pattern language $\Gamma$ and the data language $\Delta$) and two binary relations underlie any instance of the *frequent pattern* mining problem: the *generality* between patterns $\sqsubseteq_\Gamma$ and the *instantiation* between a data record and a pattern $\lhd$. Generality follows instantiation: given a pattern $f \in \Gamma$ and a super-pattern thereof $\overline{f}$ ($f \sqsubseteq_\Gamma \overline{f}$), each record $d \in \Delta$ instantiating $f$ ($d \lhd f$) instantiates $\overline{f}$ as well.

In the simplest settings, both data records and patterns are sets of items (*itemsets*) — that is, $\Delta = \Gamma = 2^O$ — whereas $\sqsubseteq_\Gamma$ and $\lhd$ boil down to set-theoretic inclusion. Hence, the mining goal amounts to finding all the frequent subsets of a family of sets $\mathcal{D} \subseteq \Delta$.

Researchers have studied more elaborate record structures,[4] including sequences ($\Delta = \Gamma = \omega^O$) and graphs[5] with equally more complex generality and instantiation (subsequence and subgraph relations, respectively). A somewhat orthogonal research axis looks for generalized pattern languages[6] built on top of a set of concepts $C$ abstracting objects from $O$ (that is, $\Gamma$ derived from $2^C$ instead of $2^O$). Concepts form a taxonomy, $H = \langle C, \leq \rangle$, where $\leq$ is the *is-a* relationship (for example, the hierarchical catalogue of products sold through an e-commerce Web site). A taxonomy is, in fact, a simple domain ontology.

Pattern-mining methods search for $\mathcal{F}_\sigma$ through the pattern space $\langle \Gamma, \sqsubseteq_\Gamma \rangle$ by exploring the monotony in frequency with regard to $\sqsubseteq_\Gamma$. `Apriori` is the prototypical pattern miner that performs a level-wise top-down traversal of $\langle \Gamma, \sqsubseteq_\Gamma \rangle$.[7] On itemsets, it examines patterns at level $k$ — that is, of size $k$ — on two points: the method computes frequency of candidate patterns by matching them against the records in $\mathcal{D}$, whereas it generates $k + 1$ candidates by combining pairs of frequent $k$-patterns. Because scanning the entire database is expensive, `Apriori` exploits frequency scores for a priori invalidation of infrequent candidates. Independent studies have successfully adapted the level-wise mining approach to both structured and generalized patterns, but our work is the first attempt we know of to port it to ontology-based patterns.

### Pattern-Based Recommendation

Patterns support the determination of the items suggested as the next step in a user session. More specifically, the recommender system matches the members of $\mathcal{F}_\sigma$ to the sequence $g$ of already accessed objects to isolate the applicable patterns. The system further analyzes the selected patterns to yield plausible continuations of the session.

Pattern-based recommenders are either item- or concept-based, relying on individual and generalized patterns, respectively.[8] The former yield very topical recommendations because they suggest all yet-unvisited objects strongly associated with $g$ but experiences difficulties when users access "rare" or new objects. The latter systems work on different granularity levels and hence offer more flexible recommendations. Indeed, instead of objects from $g$, their concepts are matched against $\mathcal{F}_\sigma$ in the search for associations, so the suggestion encompasses the extensions of all associated concepts. Such a method limits the impact of specific object rareness but tends to miss the existing strong object associations.

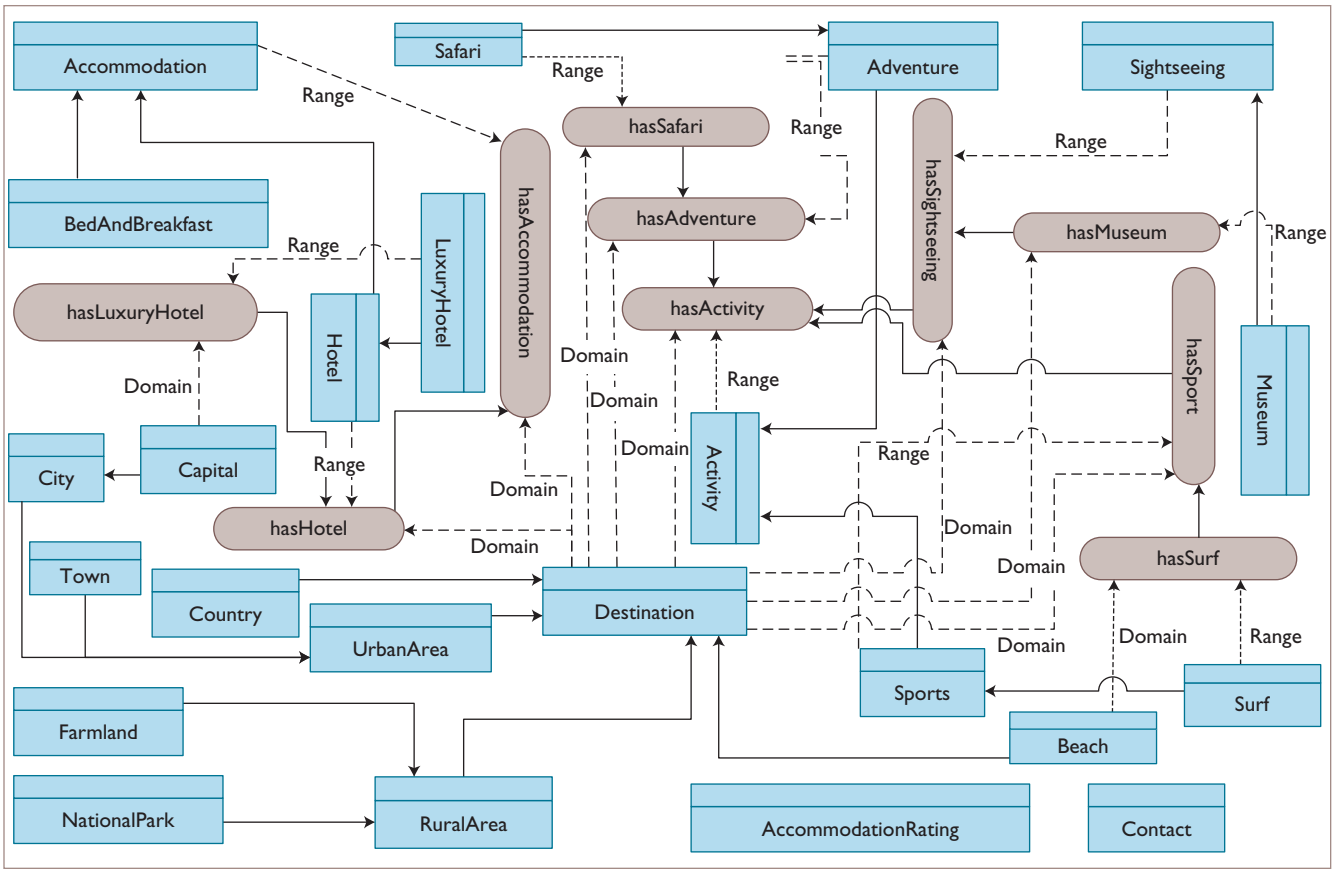


*Figure 1. Partial view of the* Travel *ontology. The rectangular nodes represent classes (domain concepts), and the shapes with rounded angles represent concept properties (roles). Dashed arrows represent domain and range links between concepts and properties, and simple arrows represent the binary relationship* is-a *between both concepts and properties.*

## Feeding a Full-Scale Ontology into the Process

To achieve a better trade-off between recommendation flexibility and precision, our approach feeds the mining process with knowledge about semantic links between objects. Our basic assumption is that co-occurrences between objects often reflect the existence of a link between them (clicks on a Paris page, for example, are usually followed by clicks on pages of Parisian hotels). Hence, manipulating links explicitly can increase the focus of concept-based recommendation while preserving its flexibility.

Because ontologies constitute the standard way of representing domain concepts and relations, we assume that an ontology powers the target recommendation system. In our settings, the ontology has two *is-a* taxonomies — concepts and relations — whose elements are connected by an *attribution* relation. Figure 1 provides a partial view of the *Travel* ontology (see http://protege.cim3.net/file/pub/ontologies/travel/travel.owl). Formally, an ontology is a four-tuple, $\Omega = \langle C, \mathcal{R}, \sqsubseteq_\Omega, \rho \rangle$, where $C$ is the set of concepts, or *classes*, and $\mathcal{R}$ is the set of domain relations, or *properties*. Figure 1 depicts these as two-section rectangles and ovals, respectively (for example, *LuxuryHotel* is a class and *hasAccommodation* is a property). The generality order $\sqsubseteq_\Omega$ holds both for concepts and properties ($\sqsubseteq_\Omega \subseteq (C \times C \cup \mathcal{R} \times \mathcal{R})$); plain arrows depict the generality links (such as *LuxuryHotel* $\sqsubseteq_\Omega$ *Accommodation* and *hasLuxuryHotel* $\sqsubseteq_\Omega$ *hasAccommodation*). Attribution is a ternary relation $\rho \subseteq C \times \mathcal{R} \times C$, which connects a property $r$ to classes $c$ and $\overline{c}$, whose instances are, respectively, origin and destination of the links of type $r$. In Figure 1, attribution is expressed through *domain*- and *range*-labeled dashed arrows pointed to a property. We define the precedence relation $\prec_\Omega$ on top of $\Omega$ as a transitive reduction of $\sqsubseteq_\Omega$ (for example, *LuxuryHotel* $\prec_\Omega$ *Hotel* and *hasLuxuryHotel* $\prec_\Omega$ *hasHotel*).

We assume that descriptions of class and property instances are organized into a separate knowledge base. Formally, let $K = \langle O, \not\perp, \rho_O \rangle$ be such a

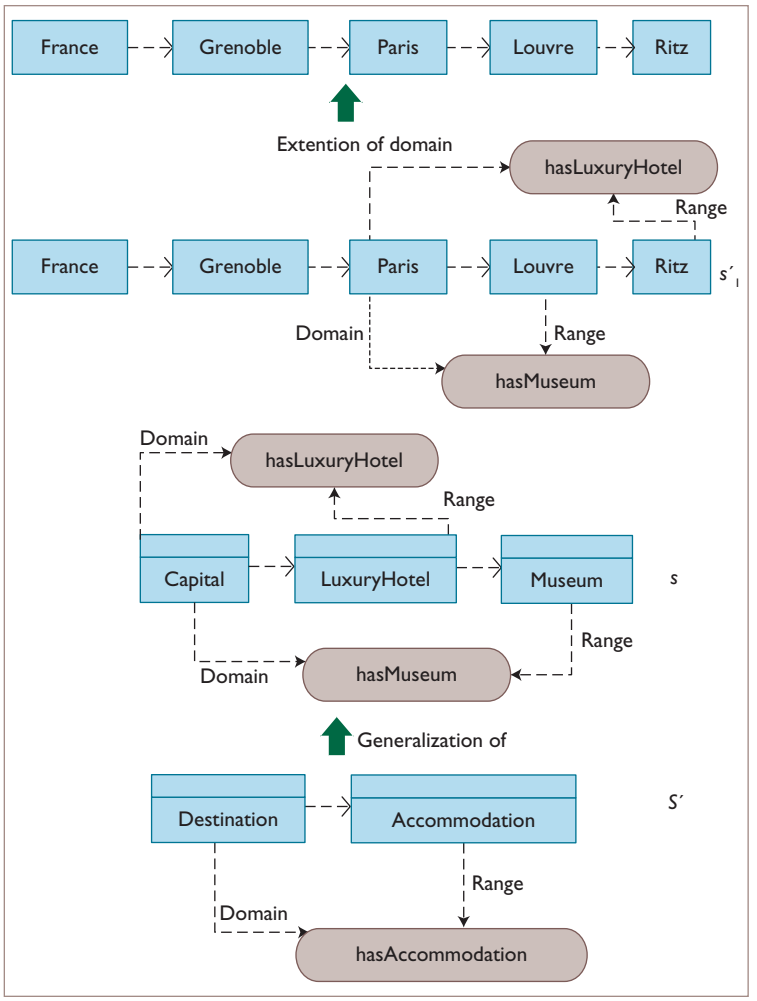


*Figure 2. Sample sequences of ontology entities. On top, we see a simple sequence together with its extended version (obtained by incorporating all existing links from the knowledge base). Below, the upper pattern is more specific than the lower one.*

***Table 1. Partial extensions for some* Travel *classes.***

| Class | Instances |
|---|---|
| Country | France, Australia, Italy |
| Capital | Paris, Roma |
| City | Grenoble, Cairns, Sydney |
| Museum | Louvre, Castel_Sant'Angelo |
| LuxuryHotel | Ritz, Grand_Hotel_Plaza, Swissotel |
| Beach | Clifton_Beach, Bondi_Beach |
| Safari | Cape_York_Safari |
| Surf | Bondi_Beach_Surf |

base, where $O$ stands for objects and $\prec$ for the instantiation relation; Table 1 provides a set of $\prec$ links. In Figure 2, single-section rectangles represent objects, whereas instantiation remains implicit (for example, *Paris* instantiates *Capital*). Furthermore, $\rho_0$ expresses interobject links ($\rho_0 \subseteq O \times \mathcal{R} \times O$) in a way similar to $\rho$ (for example, *hasLuxuryHotel* links between *Paris* and *Ritz*); Table 2 provides a set of attribution links.

## Mining Ontology-Based Patterns

The knowledge encoded in $\Omega$ and $K$ induces a characteristic pattern space with a specific mining discipline.

### Descriptive Languages and Pattern Space

Our data language $\Delta_\Omega$ is derived from sequences of page URIs translated into object IDs from $K$. Table 3 gives the set of object sequences comprising our running example.

Each sequence is further extended with all the links from $\rho_0$ exclusively involving objects from the sequence. The resulting structures are labeled digraphs, in which vertices are objects and edges either represent semantic links or sequential order.

Formally, $\Delta_\Omega$ is made of pairs $s = (\varsigma, \theta)$, where $\varsigma \in \omega_0$ is an object sequence and $\theta \in 2^{\rho_0}$ is a set of links. For conciseness, we simplify the triplets in each $s.\theta$ from $(o_d, r, o_r)$ to $r(l, m)$, where $l$ and $m$ are natural numbers denoting the ranks of $o_d$ and $o_r$, respectively, within $s.\theta$. Table 4 illustrates the $\theta$ completions of the sequences in Table 3. Currently, $s = (\varsigma, \theta)$ requires that all links in $s.\theta$ be colinear with the sequence $s.\varsigma$ – that is, $r(l, m) \in s.\theta, l \leq m$.

The language $\Gamma_\Omega$ has a structure identical to that of $\Delta_\Omega$: a pattern $S \in \Gamma_\Omega$ is a pair of a sequence and a triplet set, $(\varsigma, \theta)$. However, here $\varsigma \in \omega^C$ and $\theta \in 2^\rho$ follow the ontology rather than the knowledge base. We also impose the following constraints:

1. no class appears at two neighbor positions in $S.\varsigma(\forall i \in [2, |S.\varsigma|], S.\varsigma[i] \neq S.\varsigma[i-1])$, and
2. no redundancy in links $(\forall r(l, m), r'(l, m) \in S.\theta, r' \not\sqsubseteq_\Omega r)$.

Instantiation between data records and patterns relies on graph morphism – that is, a mapping between two graphs that preserves edges. Intuitively, $s \in \Delta_\Omega$ instantiates $S \in \Gamma_\Omega$, denoted $s \prec_{\Gamma_\Omega} S$, whenever a substructure of $s$ exists that could be homomorphically mapped onto $S$. Formally, $s \prec_{\Gamma_\Omega} S$ if a partial integer map $\iota:[1, |s.\varsigma|] \rightarrow [1, |S.\varsigma|]$ exists that satisfies these constraints:

- $\iota$ is an order-preserving and surjective function,

- $\forall i \in [1, |s.\varsigma|]$, if $\iota(i)$ is defined, then $s.\varsigma(\iota(i)) \not\lhd S.\varsigma(i)$, and
- $\forall r(l, m) \in S.\theta$, $r'(l', m') \in s.\theta$, such that $r' \sqsubseteq_\Omega$ and $\iota(l') = l$, $\iota(m') = m$.

The generality order $\sqsubseteq_{\Gamma_\Omega}$ on $\Gamma_\Omega$ follows $\lhd_{\Gamma_\Omega}$, so a pattern $S_1$ is a superpattern of another one, $S_2$, if every data sequence that instantiates $S_2$ instantiates $S_1$. Given the *Travel* ontology and the knowledge base composed of Tables 1 and 2, for example, we could check that each sequence instantiating $S$ from Figure 2 also instantiates $S'$. However, the inverse isn't true, so $S'$ is strictly more general than $S$.

Because instantiation-based generality tests are unpractical, we can use an alternative computation mechanism that mimics the homomorphic mapping in instantiation tests. The only difference with instantiation resides in the use of interclass specialization instead of object-to-class instantiation links. In the example from Figure 2, the function $\iota$ for $S$ and $S'$ is $\{(1, 1), (2, 2)\}$. Moreover, we can define a level in the new space as the set of patterns that have the same generality degree reflecting the generality degree of pattern components, classes, and properties within the ontology.

Thus, we introduce the *generality rank* measure (*rank*: $\Gamma_\Omega \to \mathbb{N}$), which is based on the depth of classes/properties ($h$: $C \cup \mathcal{R} \to \mathbb{N}$) – that is, the length of a maximal path from a component to a root of its respective *is-a* hierarchy (for example, $h$(*LuxuryHotel*) = 3, as in Figure 1). A pattern's rank is the sum of its component depths (in Figure 2, *rank*($S'$) = 3 and *rank*($S$) = 16). The level $k$ is thus made of all patterns of rank $k$.

Level-wise descent relies on the generation of $k$ + 1-level patterns from $k$-level ones, which amounts to the computation of the precedence $\lhd_{\Gamma_\Omega}$ on $\Gamma_\Omega$ (transitive reduction of $\sqsubseteq_{\Gamma_\Omega}$). The target operations clearly increase the pattern rank by 1, and they're easily translated in terms of component precedence. The list includes adding a new class/property tuple and specializing an existing class in the pattern to a subclass or replacing a property by one of its subproperties. For instance, we can get $S_{16}$ from $S_9$ by specializing the *Destination* class to *UrbanArea*.

Although these operations constitute an effective generation mechanism, their unrestricted use is redundancy-prone and hence would harm the efficiency of the global mining process. As an illustration, take $S_{16}$ in Table 7, which can be obtained from both $S_9$ and $S_8$ (given in Table 6) by class substitution and insertion, respectively. Although it might not be possible to completely eliminate this kind of redundancy, we can greatly reduce it by setting the position where operations apply. Thus, only canonical operations – that is, those involving the last class of $S.\varsigma$ – are allowed. In other terms, classes can be appended only at the end of $S.\varsigma$ and substituted to the last class. Similarly, triplets $r(l, m)$ involved in insertion/substitution satisfy $m = |S.\varsigma|$. In the new settings, $S_{16}$ cannot be canonically generated from $S_9$.

Canonical operations, when chained, result in stronger forms of pattern generalities – called prefix relations – that support recommendation. More precisely, $S'$ is a prefix of $S''$ whenever $S''$ is derivable from $S'$ by a series of canonical operations.

**Table 2. A subset of interobject links.**

| Property | Object pairs |
|---|---|
| hasMuseum | (Paris, Louvre), (Roma, Castel_Sant'Angelo) |
| hasLuxuryHotel | (Paris, Ritz), (Roma, Grand_Hotel_Plaza), (Sydney, Swissotel) |
| hasSafari | (Cairns, Cape_York_Safari) |
| hasSurf | (Sydney, Bondi_Beach_Surf) |

**Table 3. Initial object sequences.**

| ID | Object sequences ($\varsigma$) |
|---|---|
| $s_1$ | ⟨France, Grenoble, Paris, Louvre, Ritz⟩ |
| $s_2$ | ⟨Australia, Cairns, Clifton_Beach, Cape_York_Safari⟩ |
| $s_3$ | ⟨Italy, Roma, Castel_Sant'Angelo, Grand_Hotel_Plaza⟩ |

**Table 4. Relational completions of object sequences.**

| ID | Relation triplet sets ($\theta$) |
|---|---|
| $s_1$ | {hasLuxuryHotel(3,5), hasMuseum(3,4)} |
| $s_2$ | {hasSafari(2,4)} |
| $s_3$ | {hasLuxuryHotel(2,4), hasMuseum(2,3)} |

## Mining Method

The pattern-mining problem now looks like this: given $\Omega$, $K$, and $\mathcal{D} \subseteq \Delta_\Omega$, $\sigma$, find $\mathcal{F}\sigma \subseteq \Gamma_\Omega$.

As a first step toward a resolution, we propose the `xPMiner` method, which combines the control structure of `Apriori` with the use of canonical operations. More specifically, `xPMiner` (see Algorithm 1) performs a level-wise descent in $\langle \Gamma_\Omega, \sqsubseteq_{\Gamma_\Omega} \rangle$.

1: **Input**:
2: $\Omega$, $K$; ▷ Domain ontology and a knowledge base
3: $D$, $\sigma$; ▷ Set of extended sequences, support threshold
4: **Output**:
5: $\mathcal{F}_\sigma$; ▷ Set of frequent patterns
6:
7: $\mathcal{F}_c^1 \leftarrow \{\langle\langle c\rangle, \varnothing\rangle \mid c \in \max_{\sqsubseteq_\Omega}(C)\}$; ▷ *1* candidates
8: $\mathcal{F}_\sigma^1 \leftarrow candTest(\mathcal{F}_c^1, \mathcal{D})$; ▷ *1* patterns
9: **for** $(k = 2; \mathcal{F}_\sigma^{k-1} \neq \varnothing; k++)$**do**
10: $\mathcal{F}_c^k \leftarrow \varnothing$;
11: **for all** $S \in \mathcal{F}_\sigma^{k-1}$ **do**
12: $\mathcal{F}_c^k \leftarrow \mathcal{F}_c^k \cup AddCls(S) \cup SplCls(S) \cup AddRel(S) \cup SplRel(S)$
13 **end for**
14: $\mathcal{F}_\sigma^k \leftarrow candTest(\mathcal{F}_c^k, \mathcal{D})$;
15: **end for**
16: return $\bigcup_{i=1,k} \mathcal{F}_\sigma^i$

*Algorithm 1. xPMiner. This level-wise traversal of the pattern space uses a generate-and-test strategy.*

**Table 5. Level-one frequent patterns and support.**

| Pattern | Support |
|---|---|
| $S_1 = \langle\langle Accommodation\rangle, \{\}\rangle$ | 2 |
| $S_2 = \langle\langle AccommodationRating\rangle, \{\}\rangle$ | 0 |
| $S_3 = \langle\langle Activity\rangle, \{\}\rangle$ | 3 |
| $S_4 = \langle\langle Contact\rangle, \{\}\rangle$ | 0 |
| $S_5 = \langle\langle Destination\rangle, \{\}\rangle$ | 3 |

**Table 6. Level-two candidate patterns and support.**

| Pattern | Support |
|---|---|
| $S_6 = \langle\langle Beach\rangle, \{\}\rangle$ | 1 |
| $S_7 = \langle\langle RuralArea\rangle, \{\}\rangle$ | 0 |
| $S_8 = \langle\langle UrbanArea\rangle, \{\}\rangle$ | 3 |
| $S_9 = \langle\langle Destination, Accommodation\rangle, \{\}\rangle$ | 2 |
| $S_{10} = \langle\langle Destination, AccommodationRating\rangle, \{\}\rangle$ | 0 |
| $S_{11} = \langle\langle Destination, Contact\rangle, \{\}\rangle$ | 0 |
| $S_{12} = \langle\langle Destination, Activity\rangle, \{\}\rangle$ | 3 |

The starting point is the computation of frequent *1* patterns whereby candidates comprise all singleton sequences of maximally general concepts (lines 7 and 8). For each of the subsequent levels $k + 1$, the method generates the candidates from the frequent $k$ patterns by applying to each pattern any possible canonic operation (line 12). Once generated, the candidates are matched by the method against the data set $\mathcal{D}$ to keep only the frequent ones (line 14).

The algorithm is provably complete. We established it by induction on the rank — that is, the length of a canonical derivation from a frequent *1* pattern, in which `xPMiner` will generate every pattern in $\mathcal{F}_\sigma$.

### Pattern-Mining Example

As an illustration, consider Tables 1 through 4 with $\sigma = 2$. First, `xPMiner` generates the candidate *1* patterns $S_1$ to $S_5$ (see Table 5) from root classes of *Travel* and their established frequencies (for example, $s_1$ and $s_3$ instantiate $S_1$, but not $s_2$). Next, it states $S_2$ and $S_4$ as infrequent (*AccommodationRating* and *Contact* have no instances in *O*). The method then keeps the frequent patterns $S_1$, $S_3$, and $S_5$ to feed the next step. For instance, various class substitutions on $S_5$ yield $S_6$, $S_7$, and $S_8$; Table 6 shows the entire set of candidates of length two and their associated support.

`xPMiner` further filters the set through instantiation tests and yields the frequent patterns of length two, as in Table 7. These, then, help generate the candidates of length three.

At this point, the first property triplets start to appear in the candidates — for example, `xPMiner` generates $S_{17}$ from $S_{12}$ by property insertion. Table 8 provides the frequent patterns of length three. The process eventually reaches step 16, which yields a unique frequent pattern $S_{30} = \langle\langle Capital, Museum, LuxuryHotel\rangle, \{hasMuseum(1, 2), hasLuxuryHotel(1, 3)\}\rangle$ (support equal to two).

## Recommendation Strategy

Our recommender exploits $\mathcal{F}_\sigma$ and the prefix-instantiation relation. Let's say our target problem is as follows:

- pattern matching (given $\mathcal{F}_\sigma$ and an extended object sequence $s$, find patterns $S$ in $\Gamma_\Omega^f$ having a prefix instantiated by $s$), and
- recommending (examine each $S$ to compute the objects to be suggested).

The prefix instantiation relation, denoted by $\lessdot_{\Gamma_\Omega}^p$, amounts to partial instantiation: $s$ matches $S$ up to a position $\mu(S, s)$ in $S.\varsigma$ with the possible exception of some tuples $r(m, \mu(S, s))$ from $S.\theta$. As an illustration, consider the sequence $s_2$ in Table 5 and the pattern $S_{18}$ in Table 8. Obviously, $s_2$ doesn't instantiate $S_{18}$, but it does prefix instantiate it

(through the prefix $\langle\langle Destination,\ Activity\rangle, \{\}\rangle$); moreover, $\mu(S_{18}, s_2) = 2$.

Given $\prec^{p}_{\Gamma,\Omega} S$, the recommender retrieves all the instances of the class at position $\mu(S, s)$ as well as those of the class at $\mu(S, s) + 1$ (whenever $S.\varsigma$ has such position). It then assigns the objects confidence rates: the highest rate goes to objects that, when appended to $s$, allow a strictly larger prefix of $S$ to be instantiated (for example, by matching triplets $r(m, \mu(S, s))$ from $S.\theta$ that $s$ alone couldn't match).

Our ontology-based recommendation method presents substantial advantages over recommenders that exploit more traditional pattern sorts. Assume that a new user session starts with the previously unvisited *Sydney* destination. A pure item-based recommender wouldn't be able to suggest anything, regardless of the tuning of its pattern-extraction process (even with a large-sized $\mathcal{D}$, there can still be many objects not present in any pattern due to their low frequency). If we emulate concept-based recommendation by limiting prediction to pure class patterns (no property triplets), then, for example, if we used $S_{18}$, a plausible recommendation would be the extension of *Activity*. However, because nothing indicates that activities need to be located in *Sydney*, all instances (including Paris and Rome museums) would receive the same confidence rate. Consequently, the system would recommend a large set of indiscernible activities, bringing little value to the user.

With our approach, interconcept relations help the system discriminate suggestions; for $S_{17}$ and $S_{14}$, for example, our recommender would suggest the objects *Bondi_Beach_Surf* and *Swissotel*, respectively.

## Validation

We performed a two-stage validation of our approach via a comprehensive performance study that involves synthetic user sessions. For this purpose, we integrated a Java implementation of `xPMiner` as a plug-in for the Protégé ontology manipulation platform (http://protege.stanford.edu).

Our experimental settings include the extended *Travel* ontology (36 classes and 44 properties) and a knowledge base of 99 objects. On top of these, we generated several collections of raw object sequences by using a multilevel randomization model adapted from a previous study[7] that embodies our current understanding of how user navigation unfolds. We ran our tests on a 932-Mbyte RAM Pentium 4.

**Table 7. Level-two frequent patterns and support.**

| Pattern | Support |
|---|---|
| $S_8 = \langle\langle UrbanArea\rangle, \{\}\rangle$ | 3 |
| $S_9 = \langle\langle Destination,\ Accommodation\rangle, \{\}\rangle$ | 2 |
| $S_{12} = \langle\langle Destination,\ Activity\rangle, \{\}\rangle$ | 3 |

**Table 8. Level-three frequent patterns and support.**

| Pattern | Support |
|---|---|
| $S_{13} = \langle\langle City\rangle, \{\}\rangle$ | 2 |
| $S_{14} = \langle\langle Destination,Accommodation\rangle, \{hasAccommodation\ (1, 2)\},$ | 2 |
| $S_{15} = \langle\langle UrbanArea,Activity\rangle, \{\}\rangle$ | 3 |
| $S_{16} = \langle\langle UrbanArea,Accommodation\rangle, \{\}\rangle$ | 2 |
| $S_{17} = \langle\langle Destination,Activity\rangle, \{hasActivity\ (1, 2)\}\rangle$ | 3 |
| $S_{18} = \langle\langle$Destination, Activity, Accommodation$\rangle, \{\}\rangle$ | 2 |

A first experiment tested the scale-up of `xPMiner`. Its data set comprised sequences with an average length of 8 objects, which we mined with two $\sigma$ values: 0.1 and 0.07. Figure 3a shows the CPU time's evolution with the increased data set size (from 100 to 5,000); both graphs seem to represent a roughly linear dependency.

Next, we looked at the impact of minimum support variation on `xPMiner` — with a value range of 0.05 to 0.5. As Figure 3b shows, `xPMiner`'s performance rapidly deteriorates as $\sigma$ decreases. Nevertheless, this is consistent with other studies on sequential pattern mining and reflects the sharp increase in frequent pattern numbers. With 100 sequences of average size 8 objects, for example, the decrease of the minimal support from 30 percent to 5 percent makes the pattern number grow from several dozens to several thousands (in one experiment, from 53 to 12,467).

Devising an effective mining procedure for the new pattern space is a first step, but our next goal is to optimize the mining process's performance by exploring alternative strategies (perhaps depth-first traversal without candidate generation). We also plan to examine reduced representations of the frequent pattern family, such as closed or maximal patterns.

To enlarge our ontology-based approach's scope, we could examine other aspects of the ontological knowledge, such as object attribute values. But we'll definitely be looking at ways to incorpo-

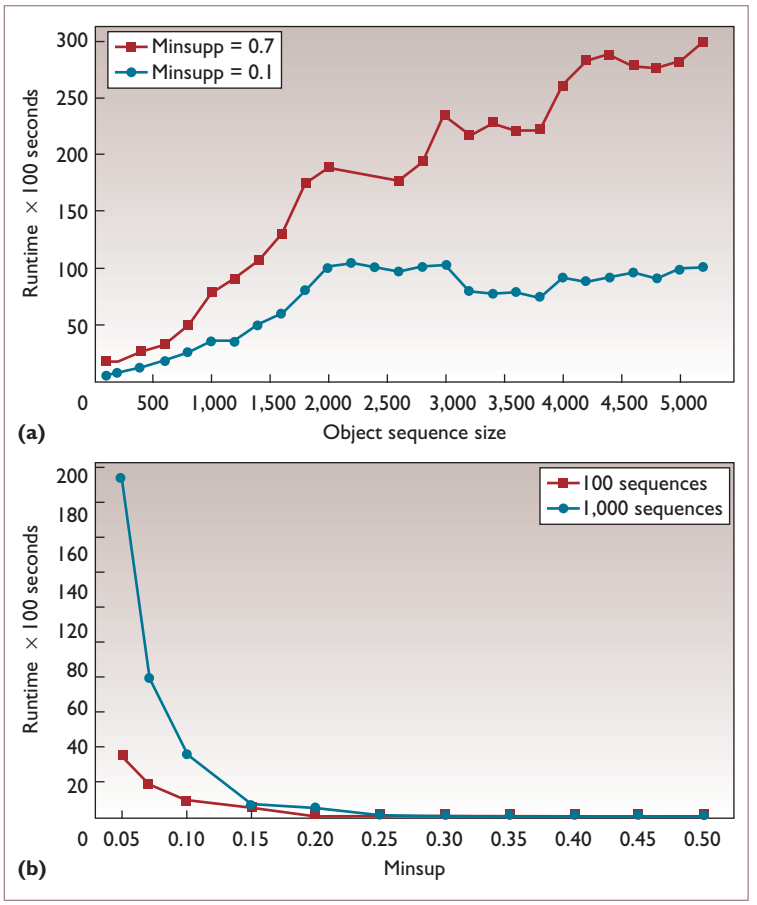


*Figure 3.* `xPMiner` *scale-up. We conducted tests on both (a) sequence size and (b) minimum support (minsup).*

rate indirect links among objects in the pattern language — that is, the links created by chains of explicit links from the knowledge base.

In the very near future, we're tuning our tools to process data from several real applications, including a virtual museum recommender and an intelligent tutoring system. We're also participating in developing a platform for personalized travel planning for the UN commerce and sustainable development body; these systems will implement our approach in a real-world setting.

## References


1. R. Kosala and H. Blockeel, "Web Mining Research: A Survey," *SIGKDD Explorations*, vol. 2, no. 1, 2000, pp. 1–15.
2. H. Dai and B. Mobasher, *Web Mining: Applications and Techniques*, IRM Press, 2005; http://maya.cs.depaul.edu/~mobasher/papers/DM04-WM-Book.pdf.
3. M. Adda et al., "Discovery of Semantically Enhanced Patterns for Content Adaptation on the Web," *Proc. Montreal Conf. e-Technologies*, 2006, pp. 177–191.
4. R. Agrawal and R. Srikant, "Mining Sequential Patterns," *Proc. 11th Int'l Conf. Data Eng.*, IEEE CS Press, 1995, pp. 3–14.
5. A. Inokuchi, T. Washio, and H. Motoda, "An A Priori-Based Algorithm for Mining Frequent Substructures from Graph Data," *Proc. 4th PKDD European Conf.*, Springer, 2000, pp. 13–23.
6. R. Srikant and R. Agrawal, "Mining Sequential Patterns: Generalizations and Performance Improvements," *Proc. 5th Int'l Conf. Extending Database Technology*, vol. 1057, Springer-Verlag, 1996, pp. 3–17.
7. R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules," *Proc. 20th Very Large Databses Int'l Conf.*, Morgan Kaufmann, 1994, pp. 487–499.
8. G. Adomavicius and A. Tuzhilin, "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions," *Trans. Knowledge & Data Eng.*, vol. 17, no. 6, 2005, pp. 734–749.



**Rokia Missaoui** is a full professor in the Department of Computer Science and Engineering at the University of Québec in Outaouais (UQO). Her research interests include knowledge discovery from databases, formal concept analysis, and the integration of data mining and data warehousing technologies. Missaoui has a PhD in computer science from Université de Montréal. Contact her at rokia.missaoui@uqo.ca.

**Petko Valtchev** is a regular professor in the Department of Computer Science at the University of Montréal, Canada (UQAM). His research interests include knowledge discovery, formal concept analysis, lattice theory, ontology engineering, and Semantic Web usage mining. Valtchev has a PhD in computer science from Joseph Fourier University. Contact him at valtchev.petko@uqam.ca.

**Chabane Djeraba** is a professor of computer science at the University of Sciences and Technologies of Lille, France. He's also a member of the Fundamental Computer Science Laboratory of Lille (LIFL), where he heads the FOX-MIIRE research team on the mining, indexing, and tracking of multimedia and complex data. His current research interests include optimization of multimedia content descriptions and user behavior analysis on multimedia and complex content. Djeraba has a PhD in computer science from Claude Bernard University. Contact him at Djeraba@lifl.fr.

**Mehdi Adda** is a PhD student at the University of Montreal. His principal research interests lie in the field of Web personalization and data mining, specifically, frequent pattern discovery, the integration of domain knowledge such as ontologies in the mining process, and the benefits of patterns in Web personalization and e-commerce recommendations. Adda has an MS in computer science from Joseph Fourier University. Contact him at adda@ieee.org.