Preface

The World Wide Web is a rich source of information about human behavior. It contains large amount of data organized via interconnected web pages, traces of information search, user feedback on items of interest, etc. One of the important characteristics of the Web in addition to large data volumes is its dynamics, where content, structure and usage is changing over time. This shows up in the rise of related research areas like communities of practice, knowledge management, web communities, and peer to peer. In particular the notion of collaborative work and thus the need of its systematic analysis become more and more important. For instance, to develop effective web applications, it is essential to analyze patterns hidden in the usage of web resources, their contents and their interconnections. Machine Learning and Data mining methods have been used extensively to find patterns in usage of the network by exploiting both contents and link structures.

This ECML/PKDD 2006 workshop continues a series of workshops on these topics, which have been held at International Conferences in the last years (KDD 2003-2005, IJCAI 2003-2005, AAAI-2005) and at the past ECML/PKDD conferences (Semantic Web Mining 2001-2002, EWMF 2003-2005).

The paper "A version space algorithm for inducing Wrappers in XPath" by Tobias Anton, describes a wrapper induction algorithm for extracting information from tree-structured documents (HTML or XML). XPath-compatible extraction rules are induced from annotated documents. The method learns a generalized tree traversal pattern (augmented with conditions), that covers all positive examples and no negative ones. A variant of the method is also presented and the robustness of the learned rules, as well as their capability of expressing knowledge of the target concept, are discussed.

Understanding the dynamic of the relationship between topics and users in blogs with the aim of developing more effective recommender systems is the main subject of the paper "An Analysis of Bloggers and Topics for a Blog Recommender System" by Hayes et al. Since the relationship between topics and users is fluid, the authors suggest that effective recommendation strategies should recalculate regularly neighbours (or clusters) in order to track topic drift and emergence. The paper proposes a set of measures for computing user and topic drift, and shows how these measures can be used to explain user behaviour. Finally, the authors demonstrate how frequently used tags can be exploited as meta labels for clusters.

Recommender systems is also the main topic of the paper "Collaborative Filtering: Fallacies and Insights in Measuring Similarity" by Symeonidis et al. Specifically, the authors conducted a thorough study of nearest-neighbour collaborative filtering (CF) algorithms and identified existing fallacies in the computation of the most popular similarity measures, namely Pearson correlation and cosine similarity, which represents one of the crucial issues for the effectiveness of the resulting recommendations. The contribution proposes a novel approach, called UNION, for measuring similarity in nearest-neighbour CF algorithms. In case of sparse data, UNION exploits more information than in the classical similarity measures (that use only co-rated items), and provides more accurate recommendations. Finally, the authors propose an evaluation procedure that provides new insights on existing approaches.

The paper "Discovering User Profiles from Papers by Using Word Sense Disambiguation" by Semeraro et al., also describes a recommender system that learns semantic user profiles from documents represented using WordNet synsets. The hypothesis is that replacing words with synsets in the indexing phase helps learning algorithms to infer more accurate (semantic) user profiles. An approach combining different algorithms to disambiguate distinct parts of speech (nouns, verbs, adjectives and adverbs) is presented and evaluated. Semantic user profiles are exploited by the "conference participant advisor" service, which can be profitably integrated in a scientific congress workbench to recommend papers to be read and talks to be attended by a conference participant.

The contribution "Semi-Supervised Learning to Extract Attribute-Value Pairs from Product Descriptions on the Web" by Probst et al., tackles an important issue in several e-commerce scenarios (product recommendations, comparison of products/offerings, demand forecasting): The automated extraction of attribute-value pairs from product descriptions on the Web. The problem is cast as a classification task and the adopted technique combines Naive Bayes with co-EM, a multi-view semi-supervised algorithm. Indeed, from unlabeled data this technique extracts an initial seed list, which serves as training set for the classification algorithm. Furthermore, co-EM uses the unlabeled data to extract product attribute-value pairs. The authors show promising results on an interesting and challenging domain, namely Web product descriptions of sporting goods, in which attributes cannot be detected in a straightforward way.

A comprehensive empirical study on conceptual web log generation and XML mining over conceptual logs is presented in the paper "Web Usage Mining and XML Mining: a real case study" by Facca. Conceptual web logs are XML logs enriched with information about structure and content of the web site. The paper shows how these logs can be automatically generated starting from a proper logging application and a conceptual application model, and how this richer log representation allows both to perform the data mining process at different levels of abstraction and to analyze more easily the results of the mining process.

Rey et al., in the paper "Mining Associations from Web Query Logs", focus on mining web search session logs to determine intent-associated queries (for example, a web searcher successfully querying for "skis" might also benefit from the results for "ski gloves", since these items are associated with the same task). The contribution describes an algorithm which derives user intent associations from search query session logs. An interesting experimental session has been carried out on web search query logs from a real world dataset (web search query logs from 2005 shopping data of Yahoo!).

Finally, the organizers seize the opportunity to thank sincerely the PC members and additional reviewers for their useful comments on the submitted papers, the authors for inspiring papers, the audience for the interest in this workshop, the local organizers from the ECML/PKDD 2006 team, and the Workshop Chair.

August 2006

Bettina Berendt
Andreas Hotho
Dunja Mladenic
Giovanni Semeraro
Workshop Chairs
WebMine 2006

Organization

WebMine 2006 was organized as part of the 17th European Conference on Machine Learning (ECML) and the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD).

Workshop Chairs

Bettina Berendt	Institute Information Systems Humboldt University Berlin, Germany http://www.wiwi.hu-berlin.de/%7Eberendt/
Andreas Hotho	Group of Knowledge and Data Engineering University of Kassel, Germany http://www.kde.cs.uni-kassel.de/hotho
Dunja Mladenic	J. Stefan Institute, Ljubljana, Slovenia http://kt.ijs.si/Dunja
Giovanni Semeraro	Department of Informatics University of Bari, Italy http://lacam.di.uniba.it:8000/people/semeraro.htm

Program Committee

Sarabjot Anand	University of Warwick, UK
Mathias Bauer	DFKI, Germany
Janez Brank	J. Stefan Institute, Slovenia
Michelangelo Ceci	University of Bari, Italy
Marco Degemmis	University of Bari, Italy
Miha Grcar	J. Stefan Institute, Slovenia
Marko Grobelnik	J. Stefan Institute, Slovenia
Pasquale Lops	University of Bari, Italy
Bamshad Mobasher	DePaul University, USA
Ion Muslea	Language Weaver, Inc., USA
Myra Spiliopoulou	Otto-von-Guericke-Univ. Magdeburg, Germany
Gerd Stumme	University of Kassel, Germany
Maarten van Someren	University of Amsterdam, Netherlands

Additional Reviewers

Table of Contents

A version space algorithm for inducing Wrappers in XPath

Tobias Anton

Technical University of Darmstadt, Germany
tobias.anton@web.de

Abstract. We introduce a wrapper induction system for extracting information from tree-structured documents like HTML or XML. From a set of annotated documents, tree-structured extraction patterns are derived that are consistent with the annotations and that can be converted into statements in theXPath language. Minimally generalized extraction patterns are generated by applying a refinement operator on the DOM tree structures around the positive data. These patterns are then generalized by selecting a subset of conditions that (a) preserve consistency with negative training data, (b) lead to a minimal overall document coverage, and (c) refer to nodes preceding the targets rather than following them. Finally, we discuss why this selection strategy leads to robust extraction rules.

Keywords: Wrapper Induction, Rule Learning, Information Extraction, XPath.

Introduction

The internet is a huge source for almost any kind of information. Many web applications provide a view to a database by encapsulating records of a single table in formatting HTML code. Our goal is to make the data behind such pages accessible for machine processing. For a machine, however, the underlying structure is not obvious, mostly due to ambiguities in the usage of tags for different entities as well as lack of semantics in HTML tags. Even though great efforts are taken to enrich the web with semantics [Berners-Lee, et al., 2001], it is still necessary today to hand-craft wrappers, which essentially consists of a series of typical tasks: (1) Define the target schema to be extracted and annotate portions of example documents as belonging to the schema, (2) find patterns in or around the annotated examples, (3) create a parser that recognizes these patterns and (4) write a program to extract the contents identified by them.

If the schema underlying the presented data is simple enough and if the source is sufficiently regular, these tasks can be transferred in part to a computer. If only task (1) is left to the user, the automated part is referred to as “wrapper induction”.

Related Work

Information extraction is recognized as an application of standard machine learning techniques to the problem of classifying document fragments based on features derived from their context [Finn & Kushmerick 2004]. As such, Wrapper induction exists in supervised and unsupervised flavours; even semi-supervised variants have been presented [Scheffer, et al., 2001].

Many wrapper induction systems use a high-level representation for the wrapping task, so that the code generation in tasks (3) and (4) reduces to interpreting the set of patterns output by task (2) with a generic wrapper engine. Depending on the flexibility of the wrapper engine, more or less complex wrappers can be induced by the learning component of the system. The complexity of rules that can be learned by an algorithm is referred to as "expressiveness".

Wrapper induction systems can also be divided up into string-based, token-based and HTML-aware ones. String-based and token-based wrapper induction systems regard the document as a sequence of characters or tokens, respectively. The algorithms behind these systems usually search for delimiters, delimiter patterns or regular delimiter languages that suit the training data well. HTML-aware systems are either designed like their token-based counterparts, but use a tokenizer that is adapted to HTML, or they try to exploit the tree-structure of HTML explicitly, with the expectation that the tree-view on the document exhibits structure that would remain hidden otherwise.

WIEN [Kushmerick, et al., 1998] by Nicholas Kushmerick was the first wrapper induction system to our knowledge. It is a string-based supervised learning procedure that creates wrappers by finding the shortest prefix and suffix that delimit all training examples in the example documents and nothing else. Under not too pessimistic assumptions, if such a pattern exists, the HLRT variant of the algorithm will discover it in quadratic time with respect to the document size [Kushmerick, 2000].

That algorithm and the derived classes HLRT, OCLR and HOCLRT are somewhat limited because they only consider regular delimiter patterns without variables. To bypass this limitation, BWI [Freitag & Kushmerick, 2000] was developed on top of these algorithms. It learns an set of LR-wrappers using AdaBoost, combining rules by weighted voting.

The **STALKER**-algorithm [Muslea, et al., 2000] from the ARIADNE-Project [Ambite, et al., 1998] is a token-based approach that also creates extraction patterns from examples. It is more expressive than WIEN, because the patterns it learns may contain wildcards and even disjunctive rules. The learning component treats the documents as a sequence of tokens, but it allows a hierarchical structuring of the learning task: Multiple rules can be nested to allow for recognition of complex structures such as lists of tuples or lists of lists. The extraction result of one wrapper is used as the input for the wrappers on the deeper level. This nesting model also allows to represent parent-child relationships analogous to the HTML structure of the source, but the nesting structure must be defined manually.

SoftMealy [Hsu & Dung, 1999] is a token-oriented, statistical approach that learns a markov model. In the model, some states are associated with data fields, while others are only used to skip document fragments. For extracting data, the model is matched to a new document and for each token, the most likely state defines whether to skip it and otherwise into which field to extract it.

RoadRunner [Crescenzi, et al., 2001] is an unsupervised, token-based learning system that learns target schema based on the contents of two or more un-annotated example documents. A wrapper is represented as a sequence of tokens with wildcards, repetitions and optional elements. Patterns can be learned incrementally. Whenever a mismatch occurs, a generalization step is done: Depending on the type of mismatch, the conflicting tokens are replaced for a wildcard, a repetition or an optional sequence of tokens.

XWrap Elite [Han, et al., 2001] discovers tree-patterns in an unsupervised manner. The user presents a page and chooses a heuristic. Based on the result, which is a list of extracted elements, she may impose further constraints on the number of results or choose another heuristics. After some refinement, a piece of java source code is generated that represents the learned concept. However, since no annotated documents are accepted as examples, no consistency constraints can be imposed to the system.

A few interactive programming approaches were found that explicitly exploit the tree structure of HTML:

W4F [Sahuguet & Azavant, 1999] requires the user to write a valid extraction rule in a language called "HEL", which can be seen as a hybrid between XPath and SQL. It assists the user only in discovering the "tree path" that leads to a target node. The tree path is the sequence of tag names of all ancestor tags from the document root to the target node, each decorated with a serial number. The user can generalize this path manually by substituting serial numbers for variables and defining relationships between these variables in an SQL-style "WHERE"-clause.

Lixto [Baumgartner, et al., 2001], in contrast, has a convenient user interface for selecting one starting node. Based on the selection, a path is generated, but in contrast to W4F, the path is a sequence of tag names or wildcards derived from the sequence of nodes preceding the selection. Further constraints can then be defined interactively by the user.

ANDES [Myllymaki & Jackson, 2002] is an information extraction system from IBM. It is completely based on open standards. It consists of a web crawler whose configuration files are stored in XML and a wrapper shell that uses XSLT scripts as extraction patterns. These scripts require XPath expressions to identify target elements that must be hand-crafted by the user. Even though [Myllymaki & Jackson, 2002] describe in detail how robust extraction rules should be written, no learning component is presented.

The purpose of this paper is to propose a learning algorithm for information extraction based on XSLT. Due to the close relation to the approach presented here, we will outline the concepts from [Myllymaki & Jackson, 2002] in brief, but before, we will provide a short introduction to XPath in the next section.

DOM and XPath

XPath [W3C, 1999] is a language for traversing DOM trees, which is a tree representation for well-formed XML documents. For short, a DOM tree is an ordered tree with every node being assigned a name and a value. Nodes can be either ELEMENT nodes or TEXT nodes. TEXT nodes cannot have children. Every node bears a node name and a node value. The name of text nodes is always "#TEXT". The names of element nodes are variable, their value is the concatenation of the values of its text node descendants in document order. Additionally, every element node has associated a set of attribute-value pairs.

```
<BODY>
 <IMG SRC="filename.gif"/>
 <H1>Playing tonight:</H1>
 <P>  Star Wars <HR/>
 </P>
</BODY>
```

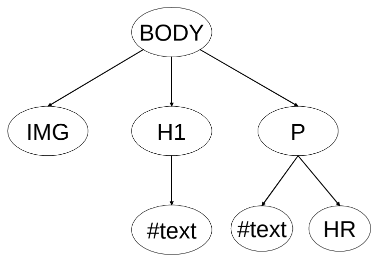


Figure 1 – a sample DOM tree

Figure 1 shows a small XML document along with DOM tree that results from parsing the following XML document:
For simplicity, only node names are shown, while node values as well as the attribute-value pairs are omitted.

An XPath statement (or path, synonymously) defines a traversal through a DOM tree. We will only provide a quick overview, while for the details, we refer to [W3C, 1999]. A path is a sequence of steps separated by slashes. Each step consists of three parts: A search direction (also called "axis"), followed by "::", a node filter and a sequence of predicates.

The axis describes the direction of document traversal in a step, e.g. to the parent, to the child, to following or preceding neighbours. Node filters restrict the set of applicable nodes along the search direction. Finally, predicates are XPath expressions in square brackets that are used to restrict the set of results further. Examples for XPath statements are:

```
/descendant-or-self::H1
/child::BODY/child::H1
/child::BODY/child::H1[following-sibling::P[child::HR]]
/descendant-or-self::H1[preceding-sibling::IMG
[@src='filename.gif']]/child::text()
```

Each step is evaluated on a set of DOM nodes and yields a set of DOM nodes as their result. Paths consisting of multiple steps are evaluated step by step, feeding the results forward.

The node filter matches nodes along the search axis whose name corresponds to the value of the node filter. The wildcard "*" is supported, matching all element

nodes, but no text nodes ; the node filter "`text()`" matches all text nodes, but no element nodes, and finally "`node()`" matches both.
If the node filter is a tag name, the axis specification and the double colon are optional, in which case the axis is is assumed to be "`child`".

Expressions inside predicates are evaluated in a boolean context. Particularly, path expressions are true for non-empty result sets, arithmetic expressions and functions are true if they have a nonzero value. A number constant C is a shortcut to select the C'th node in Result_N, e.g. the predicate "`[2]`" would filter out all but the second element of Result_N. Finally, expressions starting with "@" are interpreted as accesses to attributes.

Traversal graphs

In this section, we will introduce traversal graphs as a formal representation of tree traversal patterns. As we will see later, every traversal graph can be expressed as a path in the XPath language.

[Myllymaki & Jackson, 2002] propose traversal sequences to represent extraction rules. A traversal sequence is a linear directed graph whose nodes are called *anchors*, which are connected by *hops*. Hops that specify the descendant axis as their search direction are called *search*, all others are *path*s. On every anchor, at most one condition is evaluated and all nodes that don't comply are removed from the list of targets.

The ANDES authors propose three different types of anchors: Depending on whether the anchor imposes a constraint on text content, on attribute content or on the existence of related elements with a certain node name, it is called *content*, *attribute* or *structure* anchor.

Such traversal sequences can obviously be expressed as XPath statements. This is done by converting each hop into one XPath step, appending the condition of the following anchor as a predicate and concatenating all steps by delimiting slashes.

Extending traversal graphs

The machine learning algorithms presented here will use a similar, but more expressive pattern representation:

- An anchor may impose *multiple constraints* on the previous hop's result set
- *search*-hops may be also be directed to the "following-sibling" or "preceding-sibling" axis.
- The directed traversal graph may contain branchings at each level.

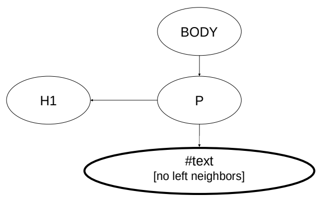


Figure 2:

An extended traversal graph with a target, one branch and one anchor condition.

The last extension introduces a conceptual problem: The extraction result of a linear traversal sequence is the result set of the last hop. When branchings are allowed, the top of the tree is no longer unique. We will solve this by designating a special leaf to be the *target node*. We call the path from the root to the target node the *stem*. Note that the root may be chosen arbitrarily, thus the path between the root and the target node need not contain any hops along the `child`- or `descendant`-axis. However, throughout this paper, we will always select an ancestor of the target node as the root.

To represent branchings from a hop of the stem in XPath, one sub-path is inserted as a predicate into the XPath expression. Since paths are evaluated in a Boolean context, such a predicate will match all nodes that are in fact surrounded by DOM structures matching the branches of the extended traversal pattern.

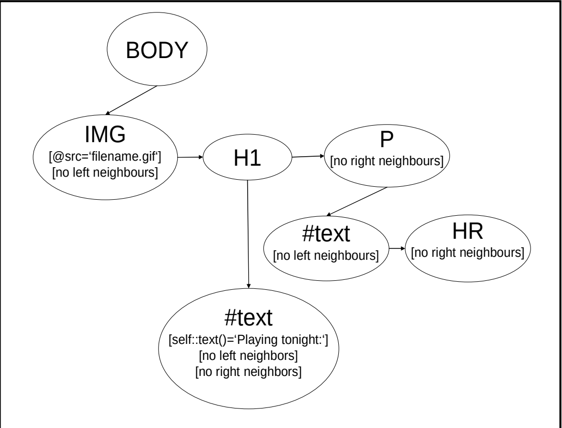


Figure 3: sequential node test for the DOM tree from Figure 2

The System

In the next section, we will introduce the wrapper induction algorithms and the setting in which they work.
Documents are considered collections of DOM nodes, connected through parent-child and following-preceding-sibling relations. For each slot to be filled, a set of nodes chosen by the user is considered the positive training data. Another set of nodes, disjoint to the positive training data, is considered the negative training data. All documents that contain any positive or negative training example are called the training documents, In our implementation, we used all but the selected nodes the training documents as negative training data. Another (possibly empty) set of documents containing neither positive nor negative examples is referred to as the set of test documents. Positive examples are provided by the user in an interaction sequence: After annotating one or more DOM nodes as positive training examples, the user may run the algorithm on the set of training documents and/or test documents. Depending on the result, she may choose to annotate another example from the training or test documents or accept the generated pattern.

The learning task

The learning task will be formulated as a search problem for a generalized tree traversal pattern that, when evaluated on all training documents, will have a result set that contains all positive examples and no negative ones.

The hypothesis language

We introduce augmented traversal graphs as a representation for the extraction rules generated by the learning algorithms. An augmented traversal graph, or ETG for short, is a rooted, acyclic, undirected, edge-attributed and node-attributed graph. To avoid confusion with nodes in DOM trees, we call its nodes "anchors" and its edges "hops".

Definition 1: augmented traversal graph

An augmented traversal graph ATG is a tuple (A, H, t, D), where A is a set (of anchors), H is a set (of hops), t is an element of A, D is a partial mapping from $A \times A \rightarrow X$, and N is a partial mapping from $A \times A \rightarrow \mathbb{N}$. An anchor is a tuple (NF, P), where NF can take any value the name of a DOM node can take and P is a vector of XPath predicates. A hop is a 2-elemented set (F, T) where F and T are elements from A. t is called the "target" of ATG. D is called the traversal direction function, where X is an XPath axis. For every $h=(A_{from},A_{to}) \in H$, both $D(A_{from},A_{to})$ and $D(A_{to},A_{from})$ exist and are opposed to each other. N is called the max-skip function and its domain equals that of the traversal direction function. Its default value is 0. A sequence A_0, H_1, A_1, H_2, … A_{n-1},H_n with $A_i \neq A_j$ for all $i \neq j$, $(A_i,A_{i+1}) \in H$ for all $0 \leq i<n$ and $H_i \in A$ for all $0<i<n$ is called a path of ATG. For every pair of anchors of A, there is exactly one path to every other anchor of A. A hop in the context of a path is said to be an X-hop iff $D(A_{i-1},A_i)=X$. Additionally, it is said to be an immediate X-hop iff $N(A_{i-1},A_i)=0$.

Semantics of augmented traversal graphs

Augmented traversal graphs can certainly be applied to DOM trees. In our implementation, we actually used grammar primitives of the XPath parser to represent nodes and edges of augmented traversal graphs. Thus, the semantics of an augmented traversal graph is defined by its corresponding XPath. Formally, an ATG T is said to extract a node N from a DOM tree D iff the XPath corresponding to T evaluated on D has a result set containing N.

Deriving STTs

For every DOM tree, one can construct an augmented traversal graph that extracts exactly the root node of exactly that DOM tree. We call such an extended traversal graph a "sequential tree test", or STT for short. Constructing an STT is done as follows: Start at the root of the subtree in question. In the example from Figure 2, this would be the "BODY"-element. Construct an anchor A_0. Let its N be the name of the current node, "BODY" in our example. If the node is an element node, add one predicate to **P** for each attribute. Then, if the current node is a leaf, create a predicate that defines the leaf node content: For elements, add to **P** the expression "`[count(child::node())=0]`", and for text nodes, add "`[self::text()='value']`", where *value* is the current node's node value. Otherwise, do for all children C_i of the current node: Construct a new anchor A_i, let its NF be the name of the C_i and execute this procedure recursively for C_i. Remove the

target node from the set of anchors and replace all references to t in H, D, and N for A_i. If i=1, insert into H and D a “descendant”-hop from A_{i-1} to A_i, otherwise insert a “following-sibling” hop. When the first or last child nodes are processed, add to the current anchor the predicate “`[count(preceding-sibling::node())=0]`” or “`[count(following-sibling::node())=0]`”, respectively.
As an example, Figure 2 shows the resulting augmented traversal graph of applying this procedure to the DOM tree from
Figure 1. The equivalent XPath to that is:

```
/body[child::node()[1][self::IMG[@src='filename.gif']
/following-sibling::node()[1][self::H1[child::node()[1]
[self::text()='Playing tonight:'][count(following-sibling::node())
=0]]/following-sibling::node()[1][self::P[child::node()[1]
[self::text()='Star Wars']/following-sibling::node()[1]
[self::HR][count(following-sibling::node()=0]]]
```

Creating CCPs from the flipped tree

Even though STTs can readily be used to identify nodes within documents by means of XPath statements, their expressive power does not exceed that of a sequential token-matching algorithm, because DOM trees are still traversed strictly in document order. In order to produce traversal graphs that reflect and exploit the tree structure of a document, we aim for traversal graphs whose paths between root and target node consist exclusively of child- and descendant-hops. We call such a traversal graph a “constrained child path”:

Definition 2: constrained child path

A constrained child path is a tuple (A, H, t, D, r), where A, H, t, and D are defined as in Definition 1. r is called the “root” and is an element of A. The path from r to t is called the stem. The stem consists solely of “descendant”-hops. Every maximal tree of anchors and hops that does not contain any stem anchors is called a branch and every stem node to which at least one branch is connected, is called a forking anchor. Every path that has no stem nodes except for a forking anchor starts with a series of one or more “preceding-sibling”-hops or with a series of “following-sibling”-hops. Eventually, these hops may be followed by a series of “descendant”-hops.

Constructing a CCP can be done by changing the algorithm slightly: Informally speaking, we create the stem whilst traversing the DOM tree from the target node up to the root and connect one branch for the left and right neighbours on each level. During the traversal, a cursor is used to keep track of the current node. Thus, we define a cursor to be a reference to a DOM node.

This is the exact algorithm:

```
Let cursor C be the target node.
Let A be an anchor.
If C is an element node, create a STT for C and replace its target node for A. Otherwise, insert a predicate into A that describes C's content.
Let A's NF be the name of C. If C is an element node, insert one predicate into A for each attribute of C.
Set CL = C and AL=A.
Repeatedly, let CL be the next left neighbour of CL and do:
create an anchor ALnew and connect it to AL by a "preceding-sibling"-hop.
repeat steps 3 and 4 for CL and AL.
set AL = ALnew
Insert into AL the predicate "[count(preceding-sibling::node() = 0]".
Process the left neighbours analogously.
If C is the document root, return P
Let C be the parent node of C
Let Anew be an anchor. Connect A to Anew with a "parent"-hop.
go to (4).
```

Applying this procedure to the text child of the H1 element in the DOM tree of Figure 1 would yield the XPath:

```
/body/child::node()[self::H1] [preceding-sibling::node()
[self::IMG][@src='filename.gif'] [count(preceding-
sibling::node())=0]] [following-sibling::node()[self::P]
[child::node()[1][self::text()='Star Wars'] /following-
sibling::node()[1] [self::HR]] [count(following-sibling::node())=0]]
/child::node() [self::text()='Playing tonight:'] [count(preceding-
sibling::node())=0] [count(following-sibling::node())=0]
```

Note that, although this expression is equivalent to the sequential tree pattern from the previous section when evaluated under a Boolean context, their structure is quite different. Figure shows the order of document traversal of the STT compared to the CCP. The difference between these variants is the connection from a node to its child: In the sequential tree test, a parent is always connected its first child. The constrained child path, on the other hand, defines a straight traversal from the root node to the target and imposes conditions to the left and right neighbour chains along its stem.

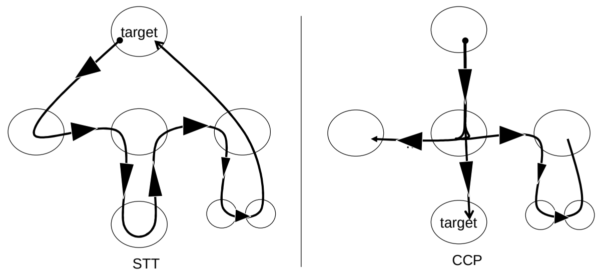


Figure 4: Structure of a sequential tree test (STT) and a constrained child-path (CCP). The CCP has a stem that leads from the root to the target node, while the STT, just traverses the DOM tree in document order.

Generalizing constrained child-paths on multiple target nodes

In this section, we will introduce the first of two extraction algorithms as a modification to the procedure described above that we call "F-GCP", which stands for Full Generalized Child-path Pattern. F-GCP is essentially identical to the algorithm that derives a CCP, but works on a set of target nodes instead on a single one. It creates a path that will match all at least all target nodes. Generalization is done by ignoring content and attributes that differ among the cursor nodes, as well as by skipping non-matching nodes and adapting the traversal pattern accordingly. These are the modifications compared to the algorithm for finding a CCP:

- The cursor C is enhanced to contain a set of nodes instead of merely a single one. Thus, we define the generalized cursor C as a set of DOM nodes with equal name.
- A heuristic is introduced to find a generalized description of the common structures under a cursor position to replace step 3. Specifically, instead of creating an STT in step 3, all possible child-paths (i.e. paths that consist of a series of immediate descendant-hops) that match at least one DOM node from every element of C are found and appended to A.
- Step 4 is modified to insert only predicates for attributes that are satisfied for every element of C.
- The cursor movements of step 6 and 10 are replaced for a search that yields the nearest matching cursor position in a given search direction. The operation takes a cursor C and yields another cursor C_{new} that may be empty if no matching neighbours were found.
- During the steps 6a and 11, we additionally set the value of the max-skip function for $N(A,A_{new})$ or $N(A_L,A_{Lnew})$ to n-1, where n is the last value assigned to n during the search procedure above.
- Step 7 is skipped if the search failed with n>1.

With these modifications, we essentially have a greedy strategy for finding an approximation to the least general generalization of a set of completely constrained paths. Intuitively speaking, we try to match the trees around the positive training examples to each other and compute an intersection of all those subtrees, augmented with information about gaps between the hops that are reflected by the values of the max-skip function and by predicates indicating the absence of structure in a direction.

When the F-GCP is converted into an XPath statement, this information is reflected by three kinds of expressions:

- The first predicate on a "preceding-sibling"- or "following-sibling"-step is "`[1]`" (i.e. "consider only the first resulting element"), if there is no gap between the two adjacent anchors.
- Vertical path steps on the stem of the child-path pattern use the "child"-axis instead of the "descendant"-axis if the max-skip function of the corresponding hop is zero.
- The last predicate on a "preceding-sibling" or on a "following-sibling"-step is "`[count(preceding-sibling::node() = 0]`" or "`[count(following-sibling::node() = 0]`", if the search for the corresponding cursor position failed with n=1.

Limitations

Since every additional example in the positive training set may shrink the generalized pattern, the number of available conditions decreases as the number of training examples increases. If the positive examples are too diverse, over-simplification will occur. This tendency to over-simplify is a severe limitation of the learning algorithm: If a *negative* example matches all conditions of the pattern, the concept can no longer be learned.

In practice, however, we found the extraction rules generated by this software to have 100% precision and recall on the training data. In fact, when applying the system to automatically generated pages, we encountered huge generalized child-path patterns with more than 50 conditions. Consequently, in the one-per-document experiments, we did not encounter any problems with over-simplification. The lack of robustness, however, is a greater problem: Frequently, conditions in extraction rules are only found because the training examples are not drawn independently. For example, some rules contained a test for the following condition:

```
/HTML/BODY[preceding-sibling::HEAD [child::TITLE="Meldung vom
27.08.2004"]]
```

Obviously, this condition will produce a wrapper that fails after at most one day. Many other examples like this could be found. This problem gets even worse if beside the contents, also the structure of an HTML template changes slightly over time.

Incorporating negative training data

For generating robust wrappers, it is thus desirable to keep the number of features used in the rule as small as possible. However, the F-GCP learner selects all conditions it possibly can, from which only a small fraction contributes to the precision of the resulting extraction rule.

According to [Myllymaki & Jackson, 2002], robustness can be achieved by "relying less on page structure and more on content." In other words, by preferring content and attribute search patterns over structure search patterns. We think that in addition to this, the total number of hops should be minimized because every additional hop increases the risk of failure. However, minimizing the number of anchors is essentially equivalent to the proposal of [Myllymaki & Jackson, 2002], because every additional hop represents a constraint on page *structure*.

In order to create more robust wrappers, we will now introduce the second algorithm "M-GCP", which is based on the F-GCP algorithm, but it additionally uses negative examples to minimize the number of conditions and hops in the generalized child-path pattern. Therefore, we considered all nodes of all training documents as negative examples unless they were labelled positive.

For explaining M-GCP, let us recall that F-GCP incorporates all conditions into the augmented traversal pattern that match all positive training examples. This can be thought of as "growing" the tree from the target anchor. M-GCP, in contrast, checks whether a particular condition has any discriminative power before adding it to the pattern. This is done by calculating the false positive rate of the traversal graph before and after adding another predicate. If the false positive rate is not influenced, the predicate is rejected.

Provided that the GCP is large enough to separate all positive from all negative examples, this feature selection strategy will reject every new candidate condition once a consistent set of conditions has been found. But still, candidate conditions added later might imply the truth value of any candidate condition added earlier. For that reason, irrelevant conditions are pruned away in a second pass.

Discussion of M-GCP's selection strategy

The order of creation of hops and anchors, as well as the matching strategy for cursor elements defines the search bias of M-GCP towards certain hypotheses. Since M-GCP traverses the documents the same way as F-GCP does, conditions whose containing anchor's path towards the target anchor contains fewer descendant-hops are always preferred over those with more. Among those conditions with an equal number of descendant-hops, conditions preceding the stem are preferred to those following it. Among each of those, anchors that are nearer, i.e. whose path to the stem has fewer hops in total, are preferred. Among those, again, tests for text content is preferred over tests for attribute content, and those are preferred over tests for attribute existence. But every condition that is not completely irrelevant is picked up in the rule. This is a major difference to most other rule learning systems that usually maximize the discriminative power of conditions. M-GCP, however, focuses on another quality of the available conditions, namely their document coverage. Since minimizing document coverage minimizes the structural risk, we think that this system is ideally biased for creating robust extraction rules in the XPath language.

Lessons learnt

From a series of tests not presented here, we conclude that M-GCP can successfully be used to create precise extraction rules on the level of DOM nodes. For wrapping pages auto-generated from the same source or at least generated by hand from the same template page, disjunctive expressions (as for example induced by the STALKER algorithm) that increase the structural risk anyway are often expendable. On the other hand, when leveraging a conjunctive hypothesis space for learning extraction rules supposed to be run continuously, the usefulness generalizing steps for covering certain difficult examples is questionable and must be assessed individually. We have encountered two non-trivial cases where robustness is being traded off for coverage: When few examples are used, the resulting wrapper may not be general enough. On the other hand, when more examples are added, the expression tends to span a larger portion of the document, and thus loses robustness with respect to changes in page layout.

In most cases, there is enough common structure around the training examples to derive a precise rule consistent with the training data. Sometimes, the use of multiple conditions and the introduction of structural constraints of the type "`count(SOME_AXIS::node()) = 0`" is necessary. Thus, the sequential traversal graph as pointed out by the ANDES project is not expressive enough as a hypothesis space for a machine learning algorithm. Even though one can argue that every augmented traversal graph can be converted into a traversal sequence, preserving all conditions from the augmented traversal graph and still being consistent with the training data, the concept of having multiple branches to match different document parts that reside in different relative positions to the target anchor requires a tree representation rather than by a sequence.

References

[Ambite, et al., 1998] Ambite, J.-L.; Ashish, N.; Barish, G.; Knoblock, C. A.; Minton, S.; Modi, P. J.; Muslea, I.; Philpot, A.; and Tejada, S. Ariadne: A system for constructing mediators for internet sources. *Proc. ACM SIGMOD International Conference on Management of Data*, ACM Press, 1998.

[Baumgartner, et al., 2001] Robert Baumgartner, Sergio Flesca, Georg Gottlob. Visual Web Information Extraction with Lixto. *Proc. 27th International Conference on Very Large Data Bases*: 119 – 128, 2001

[Berners-Lee, et al., 2001] T. Berners-Lee, J. Hendler, O.Lassila, *The Semantic Web*, Scientific American, May 2001

[Crescenzi, et al., 2001] Valter Crescenzi, Giansalvatore Mecca, Paolo Merialdo. RoadRunner: Towards Automatic Data Extraction from Large Web Sites. *Proc. 27th Intl. Conf. on Very Large Data Bases*, 2001.

[Finn & Kushmerick, 2004] Aidan Finn and Nicholas Kushmerick. Information Extraction by Convergent Boundary Classification. *Proc. Workshop Adaptive Text Extraction and Mining*, AAAI 2004.

[Freitag & Kushmerick, 2000] Dayne Freitag and Nicholas Kushmerick. Boosted wrapper induction. *Proc. American Nat. Conf. Artificial Intelligence*, AAAI 2000.

[Han, et al., 2001] Wei Han and David Buttler and Calton Pu. Wrapping web data into XML. *Special section on advanced XML data processing*, 30(2): 33-38, ACM Press, 2001

[Hsu & Dung, 1998] Chun-Nan Hsu and Ming-Tzung Dung. Generating Finite-State Transducers for Semi-Structured Data Extraction from the Web. *Information Systems*, 23(8): 521-538, 1998.

[Kushmerick et al., 1998] Nicholas Kushmerick. Wrapper Induction for information extraction. *Proc. 15th Intl. Conference on Artifical Intelligence*, ICAI 1998.

[Kushmerick, 2000] Nicholas Kushmerick. Wrapper Induction: Efficiency and Expressiveness. *Artificial Intelligence,* 118(1-2):15-68, 2000

[Muslea, et al., 2000] Ian Muslea, Steve Minton, Craig Knoblock: Stalker: Learning extraction rules for semistructured, web-based information sources, *Proc. AAAI-98: Workshop on AI and Information Integration*. AAAI Press, 1998

[Myllymaki & Jackson, 2002] J. Myllymaki, J. Jackson. Robust Web Data Extraction with XML Path Expressions. *IBM Research Report*, May 2002.

[Sahuguet & Azavant, 1999] A. Sahuguet, F. Azavant. Building light-weight wrappers for legacy Web data-sources using W4F. *Proc. 25th Intl. Conf. on Very Large Data Bases*, ACM Press, 1999.

[Scheffer et al. 2001] Tobias Scheffer, Christian Decomain, Stefan Wrobel. Active Hidden Markov Models for Information Extraction. *Proc. Intl. Symposium on Intelligent Data Analysis*, IDA 2001.

[W3C, 1999] World Wide Web Consortium. XML Path Language (XPath) Recommendation. http://www.w3c.org/TR/xpath/, 1999.

An Analysis of Bloggers and Topics for a Blog Recommender System

Conor Hayes Paolo Avesani Sriharsha Veeramachaneni

ITC-IRST,
Via Sommarive 18 - Loc. Pantè, I-38050 Povo, Trento, Italy
{*hayes,avesani,veeramachaneni*}*@itc.it*

Abstract. Over the past few years the web has experienced an exponential growth in the use of weblogs or *blogs*, web sites containing journal-style entries presented in reverse chronological order. In this paper we provide an analysis of the type of recommendation strategy suitable for this domain. We introduce measures to characterise the blogosphere in terms of blogger and topic drift and we demonstrate how these measures can be used to construct a plausible explanation for blogger behaviour. We show that the blog domain is characterised by bloggers moving frequently from topic to topic and that blogger activity closely tracks events in the real world. We suggest that a blog recommendation strategy must be able to quickly detect and model topics and determine the potential relevance of each topic to each blogger. Finally, we demonstrate how we can use blog tags as topic meta-labels.

1 Introduction

A weblog (blog) is a website containing journal-style entries presented in reverse chronological order and generally written by a single user. Over the past few years, there has been an exponential growth in the number of blogs [19] due to the ease with which blog software enables users to publish to the web, free of technical or editorial constraints. This paper presents an analysis of the blog domain in terms of the relationship between topics and bloggers in order to prepare the ground for the development of suitable recommendation and organising strategies.

The decentralised and independent nature of blogging has meant that tools for organising and categorising the blog space are lacking. In contrast to a forum such as Usenet, which is logically organised into topics and threads of conversation, there is no standard way to link blogs that are similar to each other. Although the popular collective term *blogosphere* originally implied a dynamic, cross-linked social network, recent research suggests that less-connected or unconnected blogs are in the majority on the Web [8]. Researchers on the Semantic Web project have proposed frameworks in which blogs are marked up using machine-readable meta-data, written in a language such as RDF, which would facilitate cross-blog indexing [4, 9]. In contrast, the popular 'grassroots' approach is to use frequently used *tags* to link blogs by theme [14]. Tags are short informal descriptions, often one or two words long, used to describe a set of entries in a blog [14, 3]. There is no globally agreed list of tags the blogger can choose from, nor is there an agreed best practice for tagging. Bloggers use tags for organisational purposes, subdividing their blogs into topics or themes. For example, the web site Technorati[1], which provides centralised access to millions of blogs, uses tags to link blog entries. Recent research would suggest, however, that tags are poor at retrieving documents with similar content [3, 7] and that only a fraction of tags are used repeatedly, thus excluding a large portion of the blog space [7]. At a local level, bloggers themselves often maintain a list of links to blogs they have found interesting, known as a *blog roll*. Our original idea for a blog recommender system involved using the blog roll to recommend posts similar to the active blogger's current interests [1]. This technique assumed a static relationship over time between a community of bloggers and the topics they share. This paper will make clear why that assumption is problematic.

In general, recommender systems address the problem of finding relevant information by making personalised suggestions based on previous examples of the user's interests [15]. Where there is data from multiple users, such as in the blog domain, a typical recommendation technique is to firstly aggregate user data using a similarity metric and then to select and rank items each user has not viewed before. This is the basis of automated collaborative filtering [11, 2] and recommendation strategies where clustering is used to aggregate user data [18, 12, 10]. In this paper, we examine the potential for organising the blogosphere using similar techniques.

A key decision is how often the neighbourhood set or clustering is calculated [17]. In a domain where the same users remain similar over time, neighbour selection can be carried out infrequently and recommendations can be made using a selection and ranking policy on new data from the static neighbour set. On the other hand, if similar users at time t are no longer similar at time $t{+}1$, models derived from data at time t may become obsolete very quickly.

In this analysis, each blogger is represented by the posts indexed under his/her most frequently used tag. We examine the relationship between bloggers over time, based on shared topics of interest. We suggest a set of measures to track topic and user drift and we provide an explanation of topic evolution with reference to independently observed news events during the clustering period. We show that the attachment of the blogger to a global topic, defined by a cluster of 'similar' users, tends to be shortlived. This suggests the need to devise recommendation strategies capable of tracking topics and determining their relevance to bloggers.

In Section 2 we describe our data sets. Section 3 introduces our clustering method and the criterion we use for assessing cluster quality. In Section 4 we describe our experiments for tracking the relationship of users to topics as clustering is carried out on 6 data sets, each representing a week's worth of blog data. In Section 5, we describe topic drift. We suggest a set of measures to track topic and user drift and using these measures we provide an explanation of topic

[1] http://www.technorati.com

evolution in a cluster with reference to independently observed news events. We present our conclusions in Section 6.

2 Data

Our blog data set is based on data collected from 13,518 blogs during the 6-week period between midnight January 15 and midnight February 26, 2006[2]. All blogs were written in English and used tags. We found that blogging activity obeys a power law, with 88% of bloggers posting between 1 and 50 times during the period and 5% posting very frequently (from 100 to 2655 posts). On inspection, many of these prolific bloggers were either automated spammers ('sploggers') or community blogs. We selected data from 7209 bloggers who had posted from 6 to 48 times during the evaluation period. The median for this sample is 16 posts. On average, each user posted at least once per week during the 6-week period.

For each blog we selected the posts from the most frequently used tag during the 6-week period. This allowed us to associate a single topic (as defined by the blogger's tag) with each of the 7209 blogs. We chose to examine one topic per blog because blog topics from a single blog are often similar as the blogger may use multiple tags for each post. Thus each of the 7209 blog 'documents' constitutes a single topic from a single blogger from the 6-week period.

The data was divided up into 6 data sets, each representing post data from a single week. As all 7209 bloggers do not post every week, the data sets have different sizes and overlap in terms of the blog instances they contain (see Table 1). Each instance in a data set is a 'bag of words' made up of the posts indexed under the most frequently used tag from a single blog during that week, *plus* the posts made in the previous 2 weeks (using the same tag). As the posts in a single week are often quite short and take the form of updates to previous posts, we include the previous 2 weeks to capture the context of the current week's updates. For example, if a blog is updated in week 3, the instance representing that blog in the dataset for week 3 is based on the posts in weeks 3, 2 & 1. If the blog is not updated in week 4, the instance representing the blog is excluded from the data set for week 4. As shown in Table 1, on average, 71% of the blogs present in the data set win_t will also be present in the data set win_{t+1}.

We processed each data set independently, removing stop words and stemming the remaining words in each document. We then removed low-frequency words appearing in less than 0.2% of the documents, and high-frequency words occuring in more than 15% of the documents. Documents with less than 15 tokens were not condsidered at this point. Each word was weighted according to the standard TF/IDF weighting scheme and the document vector normalised by the L^2 norm. This created a feature set of approximately 3,500 words for each data set. Table 1 gives the window period, size and overlap with the subsequent window.

[2] The blog URLs were kindly supplied by Natalie Glance of www.blogpulse.com

data set	**Dates (2006)**	**Size**	**Overlap win_{t+1}**	**%**
win_0	Jan 16 to Jan 23	4163	3121	75
win_1	Jan 23 to Jan 30	4427	3234	73
win_2	Jan 30 to Feb 6	4463	3190	71
win_3	Feb 6 to Feb 13	4451	3156	71
win_4	Feb 13 to Feb 20	4283	2717	63
win_5	Feb 20 to Feb 27	3730	-	-
mean	-	**4253**	**3084**	**71**

Table 1. The periods used for the windowed blog data set. Each period is from midnight to midnight exclusive. User overlap refers to the overlap with the same users in the data set for the next window.

3 Clustering

The blog domain contains many millions of documents, constantly being updated. A reasonable goal would be to try to organise these documents by topic or type. Researchers working on the Semantic Web have proposed the construction of formal ontologies as a solution to this problem [4, 9]. On the other hand, advocates of the tagging approach propose a bottom-up, knowledge-light approach where frequently used tags are deployed as meta-data [14]. The first proposal has the disadvantage of being too knowledge intensive, and risks being ignored by web users. Although tagging is widely used by blog users, its effectiveness as an accurate organising mechanism has yet to be demonstrated [3, 7].

Document clustering is a well established technique for organising unlabelled document collections [20]. Clustering has two goals: to uncover latent structures that accurately reflect the topics present in a document collection and to provide a means of summarising and labelling these structures so that they can be interpreted easily by humans. Clustering has been used for improving precision/recall scores for document retrieval systems [16], browsing large document collections [5], organising search engine return sets [13] and grouping similar user profiles in recommender systems [18, 12, 10].

As our objective was to analyse user behaviour using a clustering solution, we implemented the *spherical k-means* algorithm, a well understood variation of the k-means clustering algorithm that scales well to large document collections and produces interpretable cluster summaries [6]. Spherical k-means produces k disjoint clusters, the centroid of each being a concept vector normalized to have unit Euclidean norm.

3.1 Clustering Quality

Given a set of data points, the goal of a clustering algorithm is to partition them into a set of clusters so that points in the same cluster are close together, while points in different clusters are far apart. Typically, the quality of a clustering solution is measured using criterion functions based on intra- and intercluster distance. Following [21], the quality of cluster r is given as the *ratio* of intra- to intercluster similarity, $\mathcal{H}_r$. Given S_r, the set of instances from cluster r, intracluster similarity, $\mathcal{I}_r$, is the average cosine distance between each instance, $d_i \in S_r$ and the cluster centroid, C_r. Intercluster similarity, $\mathcal{E}_r$, is the cosine distance of the cluster centroid to the centroid of the entire data set, C (see Equation 1).

In previous work, we have confirmed that clusters with high $\mathcal{H}_r$ scores tend to be clusters with large proportions of documents of a single class [7]

$$\mathcal{H}_r = \frac{\mathcal{I}_r}{\mathcal{E}_r} = \frac{\frac{1}{|S_r|} \sum\limits_{d_i \in S_r} \cos(d_i, C_r)}{\cos(C_r, C)} \tag{1}$$

4 Tracking Users

In these experiments we do not address the issue of selecting an optimal value of k and, as such, we cluster the data at several values of k. For each value of k, a random seed is chosen after which k-1 seeds are incrementally selected by choosing the seed with the greatest distance to the mean of the seeds already selected. In order to track user and topic drift from week to week, the seeds for the clusters in week t are based on the final centroids of the clusters produced in week t-1, except in the case of the first week where the seeds are chosen to maximise interseed distance.

In order to cluster data using the seeds based on the centroids from the previous week we map the feature set from the previous week's data to the feature set of the current week. In each pair of adjacent windows, the feature set overlap between windows is greater than 95%. The feature values for each seed are the feature weights from the corresponding centroid in the previous week.

In order to compare clustering in adjacent windows we define the following measures: *user entropy* per cluster, $\mathcal{U}_r$, and *interwindow similarity* per cluster, $\mathcal{W}_r$. User entropy, $\mathcal{U}_r$, for a cluster is a measure of the dispersion of the users in one cluster throughout the clusters of the next window. For a fixed value of k, if many of the users in a single cluster in win_t are also in a single cluster in win_{t+1}, then entropy will approach zero. Conversely, if the neighbourhood of users at win_t is spread equally among many clusters at win_{t+1}, entropy will tend toward a value of 1.

$$\mathcal{U}_r = -\frac{1}{\log q} \sum_{i=1}^{q} \frac{n_r^i}{n_r} \log \frac{n_r^i}{n_r} \tag{2}$$

$c_{r,t}$ is cluster r at win_t; $c_{i,t+1}$ is a cluster i at win_{t+1} which contains users from $c_{r,t}$. S_{t+1} are all the instances in win_{t+1}. q is the number of $c_{i,t+1}$ (the number of clusters at win_{t+1} containing users from cluster $c_{r,t}$). $n_r = |c_{r,t} \cap S_{t+1}|$. n_r^i is $|c_{r,t} \cap c_{i,t+1}|$, the number of users from cluster $c_{r,t}$ contained in $c_{i,t+1}$.

The interwindow score, $\mathcal{W}_r^{t+1}$, for a cluster r in window win_t is the similarity between the centroid of cluster r and the centroid of the corresponding cluster r in window win_{t+1}. Likewise, $\mathcal{W}_r^{t-1}$ is the similarity between the centroids of cluster r at windows win and win_{t-1}. Intuitively, $\mathcal{W}_r^{t+1}$ is a measure of the drift of the centroid concept, C_r, at win_t, where C_r is also the seed for cluster r at win_{t+1}.

$$\mathcal{W}_r^{t+1} = \cos(C_{r,t}, C_{r,t+1}) \tag{3}$$

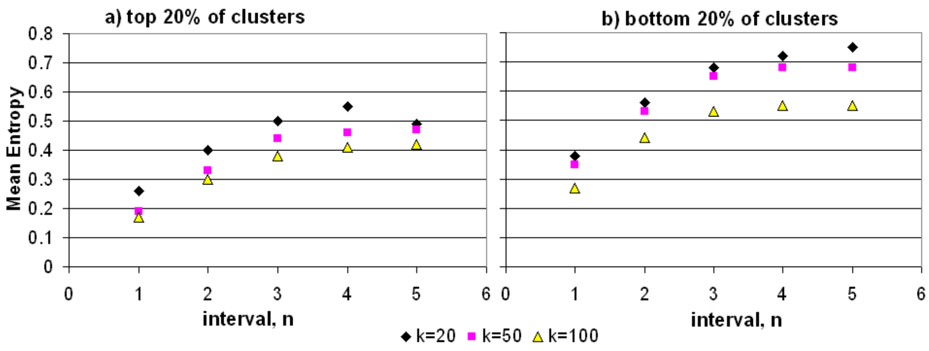


Fig. 1. Mean user entropy recorded where the intervals between windows varies from 1 to 5. The diagram on the left gives the entropy recorded for the top 20% of clusters according to $\mathcal{H}_r$. The diagram on the right gives the user entropy for the bottom 20% of clusters.

4.1 User Drift

In this section we examine whether users stay together as the data is clustered window by window. We demonstrate the degree of user drift by increasing the interval over which we calculate user entropy. We cluster the data in each window at k =20, 50 and 100, as described in the previous section. For each clustering we calculate user entropy for each cluster in win_t in relation to the clusters in window win_{t+n} where the interval n is defined as $1 \leq n \leq 5$.

However, rather than averaging the cluster entropy scores between windows over all clusters for a particular value of k, we examine 'strong' clusters (high $\mathcal{H}_r$) against weak clusters (low $\mathcal{H}_r$). Our hypothesis is that users associated with a strong cluster at window win_t will also tend to be together at window win_{t+1}. Conversely, we would expect greater user drift from clusters with low $\mathcal{H}_r$ scores. For the clustering produced at k in each window win_t we rank the clusters in descending order according to $\mathcal{H}_r$. For each pair of windows, win_t and win_{t+n}, we calculate $\hat{u}$, the average entropy of the top 20% of the ranked clusters in win_t, and $\breve{u}$, the average entropy of the bottom 20% of the ranked clusters in win_t. We also calculate $\hat{f}$ and $\breve{f}$, the respective fractions of the data set represented by the top and bottom 20% of the ranked clusters in win_t. Table 2 shows the $\hat{u}$ and $\breve{u}$ scores for each pair of windows where $n = 1$. Figure 1 demonstrates the $\hat{u}$ and $\breve{u}$ scores for $1 \leq n \leq 5$.

n=1	k = 20				k = 50				k = 100			
	top 4		bottom 4		top 10		bottom 10		top 20		bottom 20	
Interval	$\hat{f}$	$\hat{u}$	$\check{f}$	$\check{u}$	$\hat{f}$	$\hat{u}$	$\check{f}$	$\check{u}$	$\hat{f}$	$\hat{u}$	$\check{f}$	$\check{u}$
0 - 1	0.1	0.34	0.35	0.48	0.16	0.24	0.29	0.42	0.19	0.2	0.27	0.28
1 - 2	0.1	0.28	0.39	0.37	0.13	0.21	0.35	0.38	0.15	0.17	0.32	0.27
2 - 3	0.1	0.26	0.41	0.37	0.12	0.2	0.37	0.37	0.13	0.19	0.39	0.3
3 - 4	0.11	0.24	0.40	0.37	0.12	0.15	0.40	0.3	0.1	0.15	0.38	0.22
4 - 5	0.11	0.19	0.43	0.32	0.08	0.15	0.39	0.27	0.11	0.14	0.44	0.25
Mean	0.1	0.26	0.4	0.38	0.12	0.19	0.36	0.35	0.14	0.17	0.36	0.26

Table 2. The mean user entropies between adjacent window periods calculated over the top and bottom 20% of clusters ranked according to $\mathcal{H}_r$.

As a baseline, we observe that if the users in the clusters of win_t were randomly dispersed among the clusters in win_{t+n}, both $\hat{u}$ and $\check{u}$ would have values close to 1. On the other hand, if the users in each cluster of win_t were again clustered together in win_{t+n}, then both $\hat{u}$ and $\check{u}$ would have a value of 0. Table 2 shows that, even where n is low, the values for $\hat{u}$ indicate user dispersion from window to window. As the interval increases with n, $\hat{u}$ and $\check{u}$ also increase. This would suggest that the relationship between users, based on a shared topic, is short-lived rather long term. Comparing the values for $\hat{u}$ and $\check{u}$, it is clear that user drift is more pronounced for the clusters with low $\mathcal{H}_r$. However, Table 2 also shows that $\hat{f}$, the fraction of the data set contributing to the $\hat{u}$ score, is smaller than $\check{f}$, the fraction of the data set contributing to $\check{u}$, by at least a factor of 2. This means that, in clusters where user drift is shown to be relatively low, the proportion of the total users involved is actually quite low.

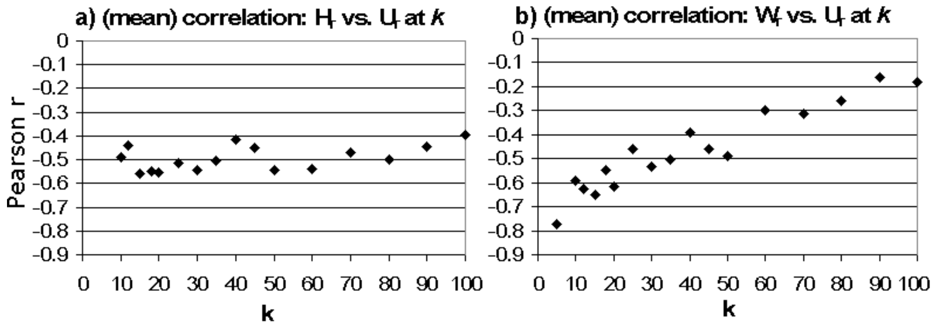


Fig. 2. a) Mean correlation between $\mathcal{H}_r$ and $\mathcal{U}_r$ as measured between pairs of windows at varying values of k. b) Mean correlation between $\mathcal{H}_r$ and $\mathcal{W}_r$ at k

By correlating $\mathcal{H}_r$ against $\mathcal{U}_r$ we can confirm the relationship between the two scores. For each pair of adjacent windows (win_t, win_{t+1}) we calculate the correlation of $\mathcal{H}_r$ against $\mathcal{U}_r$ at values of k from 5 to 100. We find a negative correlation for each of the 5 window pairs at every value of k. For each value of k we average the correlation scores produced from the 5 window pairs. Figure 2(a) graphs the mean correlation against k. The consistent negative correlation supports our observation that clusters with well defined topics (high $\mathcal{H}_r$ scores) are more likely to have less user drift (i.e. low $\mathcal{U}_r$) than clusters with low $\mathcal{H}_r$ scores. However, we should recall that clusters with high $\mathcal{H}_r$ consistently make up a small proportion of the overall data set.

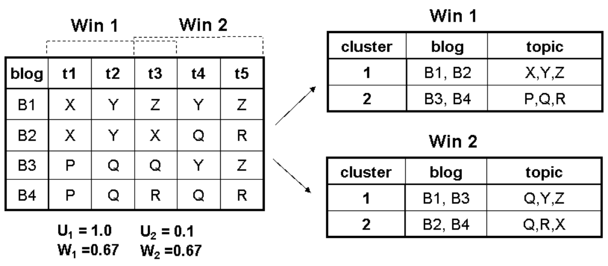
5 Topic Drift

Topics do not remain stable over time. They emerge and decay or become transformed as lowly weighted features in one window are boosted in another window. Clearly, during this period of transition the relationship between users and clusters will be fluid. In order to demonstrate this we firstly examine the correlation between clusters in two adjacent windows in terms of their user entropy scores and their interwindow scores. Figure 2(b) demonstrates the mean correlation at k for $\mathcal{W}_r$ against $\mathcal{U}_r$ calculated between the clusters from the 5 pairs of adjacent windows. The strong negative correlation, particularly evident for $k < 50$, suggests that user drift is strongly related to concept drift.

5.1 Analysis

As bloggers add new posts they modify the topic description of the posts currently indexed under their tag. They also collectively modify the global topics that will be detected in the next clustering iteration. If users assigned to a cluster r in win_t post new material in win_{t+1} *dissimilar* to the cluster centroid of r, then it is most likely that these users will not be associated with the same topic in win_{t+1}.

This type of behaviour is illustrated in the simple example in Figure 3 in which there are 4 blogs {B1,B2,B3,B4}, each with five posts from the set P = {P,Q,R,X,Y,Z}. Each blogger has posted once at each time increment (t1 to t5). For the sake of simplicity we assume that the similarity between blogs is based on the proportion of overlap of elements from the set P. We cluster the blogs at time $t3$ and $t5$, using k =2. For each clustering, each blog is represented by the elements from P that fall within the period *win 1* and *win 2* respectively. In window *win 1*, the clusters produced are {B1,B2} and {B3,B4} and the respective cluster topic descriptions are {X,Y,Z} and {P,Q,R}. During window *win 2*, the bloggers B2 and B3 change topics, each selecting posts not associated with their cluster assignment from window 1, while bloggers B1 and B4 choose posts consistent with their cluster assignment. Clustering the data in window *win 2* produces the assignments shown in the bottom right of Figure 3. The clusters are {B1,B3} and {B2,B4}. This causes the user entropy $\mathcal{U}_r$ for each cluster in *win 1* to go to 1. We can also see that the topic descriptions in the clusters from *win 2* have been modified. The $\mathcal{W}_r$ score for each cluster is 0.67 where $\mathcal{W}_r$, in this case, is based on the proportion of overlapping elements. So in this example we can see topic drift between clusters is caused by bloggers moving away from the clusters they were assigned to in the first period. While this causes a large entropy score, we should also observe that the overall topic descriptions are changed but still have a degree of similarity with the topic descriptions produced in *win 1*. As such we suggest that although $\mathcal{U}_r$ and $\mathcal{W}_r$ are clearly related, the rate of topic drift may be considerably slower than the rate of user drift. We can see this from Table 3 below where, after win_2, topic drift is extremely low ($\mathcal{W}_r \geq 0.93$), while user drift is low but not negligible ($\mathcal{U}_r \geq 0.19$).



blog	t1	t2	t3	t4	t5
B1	X	Y	Z	Y	Z
B2	X	Y	X	Q	R
B3	P	Q	Q	Y	Z
B4	P	Q	R	Q	R

$U_1 = 1.0$ $U_2 = 0.1$
$W_1 = 0.67$ $W_2 = 0.67$

Win 1

cluster	blog	topic
1	B1, B2	X,Y,Z
2	B3, B4	P,Q,R

Win 2

cluster	blog	topic
1	B1, B3	Q,Y,Z
2	B2, B4	Q,R,X

Fig. 3. A simplified example of how user and topic drift occurs.

5.2 A Real World Example

In this section we provide an analysis of the relationship between user and topic drift based on independent empirical observation of news events.

win_t	$\mathcal{W}_r^{t-1}$	$\mathcal{H}_r$	$\mathcal{U}_r$	**centroid key words**	**freq. tags**	nyt	wp	tg
win_0	-	0.41	-	sharon, bbc, mr, pilot, ariel	current, affair, politics, bsg, culture	0	0	0
win_1	0.78	0.44	0.39	sharon, israeli, palestinian, hamas, lee	current, affair, bsg, israel, politics	0	0	1
win_2	**0.28**	**0.87**	**0.66**	**muslim, cartoon, islam, danish, prophet**	**politics, religion, current, affair, war**	**1**	**9**	**55**
win_3	0.93	0.93	0.21	muslim, cartoon, islam, danish, prophet	politics, current, affair, war, society	13	14	42
win_4	0.96	0.96	0.19	cartoon, muslim, islam, danish,prophet	politics, current, affair, religion, culture	7	8	15
win_5	0.94	0.87	0.25	cartoon, muslim, islam, danish, prophet	politics, current, affair, religion, islam	7	5	4

Table 3. The change in $\mathcal{W}_r$, $\mathcal{H}_r$ and $\mathcal{U}_r$ scores between windows as the 'Danish Newspaper Muhammad Cartoon Controversy' topic emerges. In each row the $\mathcal{W}_r$ and $\mathcal{U}_r$ scores refer to the drift since the *previous* window, win_{t-1}.

In Table 3 we show the $\mathcal{W}_r$, $\mathcal{H}_r$ and high $\mathcal{U}_r$ scores for cluster 24 (from k =50) in each window. The cluster was chosen because it clearly illustrates a transition from a weak to strong topic where the values $\mathcal{W}_r$, $\mathcal{H}_r$ and $\mathcal{U}_r$ can be explained with reference to independent evidence.

In win_0 cluster 24 has a low $\mathcal{H}_r$ score and the most highly weighted terms from the cluster centroid suggest a cluster that may be mixing several topics. As the values of $\mathcal{W}_r$ and $\mathcal{U}_r$ refer to the difference between the previous window and the current window we do not have these values for win_0. The topic descriptions in win_1 suggest that the topic has become more coherent, concentrating on Israeli/Palestinian affairs and the surprise win by the Hamas party in the Palestinian elections during win_1. The cluster has moderate similarity (0.78) to the previous week and a moderate level of entropy, suggesting that many users from the previous week have drifted away.

Win_2 brings a very large change. This is the week that the 'Danish Newspaper Muhammad Cartoon Controversy' began its month-long run in the world media[3]. By win_2, bloggers in our data set have begun to reference this issue and the topic immediately begins to dominate cluster 24. The cartoon controversy topic emerges from a weak cluster in win_1 ($\mathcal{H}_r = 0.44$), describing events in the middle east, to become a 'strong' topic ($\mathcal{H}_r$=0.87) in win_2. The rapid growth in $\mathcal{H}_r$ is accompanied by an equally rapid drop in $\mathcal{W}_r$ from 0.78 to 0.28, suggesting that the increase in $\mathcal{H}_r$ is due to the introduction of a stronger topic into the cluster. Furthermore, the $\mathcal{U}_r$ score at win_2 undergoes a large increase, suggesting that a large proportion of the users in cluster 24 at win_1 are no longer together in win_2. From win_2 to win_5, the cluster enters a stable period, with high $\mathcal{H}_r$ and $\mathcal{W}_r$ scores and lower $\mathcal{U}_r$ scores than before.

We can synchronise this behaviour with the real events. We suggested earlier that posts about the controversial election of Hamas in the Palestinian elections during win_1 had contributed to the increase in coherence of cluster 24. However, with regard to the cartoon controversy we can be more precise. The columns marked *nyt*, *wp* and *tg* in Table 3 refer to the *New York Times*, *Washington Post* and *The Guardian* newspapers respectively. The numbers in the columns refer to the number of articles, commentaries and features carried by each newspaper about the controversy. To get these numbers we queried the archive sections of these newspapers using a query term extracted from the 5 most highly weighted terms in win_2 : 'muslim, cartoon, islam, danish, prophet'. From these figures, we can see that the emergence of this story in the international press is synchronised by its emergence in the blogosphere. Furthermore, we can construct a plausible explanation for user behaviour using the measures we defined.

5.3 Tag Meta-Labels

An interesting by-product of the clustering is that it allows us to produce good meta-descriptions of the clusters using the most frequently used tags in each cluster. We process each tag by removing stop words and stemming each word.

[3] http://en.wikipedia.org/wiki/Jyllands-Posten_Muhammad_cartoons

We then rank the tag terms in each cluster in terms of frequency of occurrence. The majority of tags occur only once. However, most clusters have a small portion of tag terms that repeat. As we can see from Table 3, these terms are useful meta-descriptors for the cluster concepts, offering more abstract summaries than those created by the cluster centroid. Furthermore, these tags often furnish information not apparent in the centroid descriptions. In this case, the term 'bsg' explains the poor $\mathcal{H}_r$ scores for the first 2 windows in Table 3. 'bsg' is an acronym for the cult science fiction TV show 'Battle Star Galactica', which has a central character called Sharon, a fighter pilot. Therefore, we can see that the centroid keywords for windows 0 and 1 refer to a set of documents concerning former Israeli prime minister Ariel Sharon, the Israeli-Palestinian conflict and a fictional account of war in space. The tag 'bsg' disappears in window 2 at the same time as user entropy increases dramatically, suggesting that the 'bsg' fans have moved from this cluster.

6 Conclusions

We suggest that if one wishes to design a user-based recommendation strategy for a subset of the blogosphere (either users or topics), an analysis similar to the one presented here will provide insights into its expected efficacy. In this analysis, each blogger was represented by a single 'topic', extracted from his/her most frequently used tag. We created 6 data sets, each representing a week's worth of data from the user's tag. We proposed a set of measurements for measuring user and topic drift and we demonstrated how they can be used to construct a plausible explanation for user behaviour. We found that the blog domain is characterised by bloggers moving frequently from topic to topic. Little evidence was found to suggest that neighbourhoods of users remain consistent for any length of time. Although strong clusters tend to have lower user entropy, these clusters form a small proportion of the overall data set. These observations would suggest that the majority of bloggers tend to write in a 'shallow' way about a variety of different subjects. We demonstrate the fluid relationship between bloggers and topics using a real world example of bloggers quickly reacting to an important breaking news story. Rather than relying upon a set of nearest neighbours, we would suggest that a blog recommendation strategy must be able to quickly detect and model topics and to determine the potential relevance of each topic to each blogger. Furthermore, by identifying user associations using information other than content, such as links or trust scores, it may be possible to determine blog sub-communities that behave with greater consistency than the 'neighbourhoods' we have observed.

References

1. P. Avesani, M. Cova, C. Hayes, and P. Massa. Learning contextualised weblog topics. In E. Adar, N. Glance, and M. Hurst, editors, *WWW 2005, 2nd Annual Workshop on the Weblogging Ecosystem*, Chiba, Japan, May 2005.

2. M. Balabanovich and Y. Shoham. Fab: content-based, collaborative recommendation. *Commun. ACM*, 40(3):66–72, 1997.
3. C. H. Brooks and N. Montanez. An analysis of the effectiveness of tagging in blogs. In *2005 AAAI Spring Symposium on Computational Approaches to Analyzing Weblogs*. AAAI, March 2005.
4. S. Cayzer. Semantic blogging and decentralized knowledge management. *Commun. ACM*, 47(12):47–52, 2004.
5. D. R. Cutting, D. R. Karger, J. O. Pedersen, and J. W. Tukey. Scatter/gather: a cluster-based approach to browsing large document collections. In *15th international ACM SIGIR conference on Research and development in information retrieval*, pages 318–329, New York, USA, 1992. ACM Press.
6. I. Dhillon, J. Fan, and Y. Guan. Efficient clustering of very large document collections. In R. Grossman and a. R. N. G. Kamath, editors, *Data Mining for Scientific and Engineering Applications*. Kluwer, 2001.
7. C. Hayes, P. Avesani, and S. Veeramachaneni. An analysis of the use of tags in a blog recommender system. In *ITC-IRST Technical Report*. http://sra.itc.it/people/hayes/pubs/06/hayes-ijcai07-tech-report.pdf, June 2006.
8. S. Herring, I. Kouper, J. Paolillo, and L. Scheidt. Conversations in the blogosphere: An analysis "from the bottom up". In *Proceedings of HICSS-38*, Los Alamitos, 2005. IEEE Press.
9. D. R. Karger and D. Quan. What would it mean to blog on the semantic web. *Journal of Web Semantics*, 3(2):147–157, 2005.
10. J. Kelleher and D. Bridge. An accurate and scalable collaborative recommender. *Artificial Intelligence Review*, 21(3 - 4):193 – 213, June 2004.
11. J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl. Grouplens: applying collaborative filtering to usenet news. *Commun. ACM*, 40(3):77–87, 1997.
12. M. O'Connor and J. Herlocker. Clustering items for collaborative filtering. In *ACM SIGIR Workshop on Recommender Systems*, Berkeley, CA, 1999.
13. O. E. O. Oren Zamir. Grouper: A dynamic clustering interface to web search results. In *8th International WWW Conference*, Toronto, Canada, May 1999.
14. E. Quintarelli. Folksonomies: power to the people. *paper presented at ISKO Italy-UniMIB Meeting, Mi*, June 2005.
15. P. Resnick and H. R. Varian. Recommender systems. *Commun. ACM*, 40(3):56–58, 1997.
16. C. J. V. Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, Newton, MA, USA, 1979.
17. B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *10th international conference on WWW*, pages 285–295, New York, NY, USA, 2001. ACM Press.
18. B. M. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering. In *5th International Conference on Computer and Information Technology*, 2002.
19. D. Sifry. State of the blogosphere, april 2006 part 1: On blogosphere growth. *http://technorati.com/weblog/2006/04/96.html*, 2006.
20. M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *6th ACM SIGKDD, World Text Mining Conference*, Boston, MA., 2000.
21. Y. Zhao and G. Karypis. Empirical and theoretical comparisons of selected criterion functions for document clustering. *Machine Learning*, Volume 55(3):311 – 331, Jun 2004.

Web Usage Mining and XML Mining: a real case study

Federico Michele Facca

Dipartimento di Elettronica e Informazione,
Politecnico di Milano, Italy
facca@elet.polimi.it

Abstract. In this paper we report our first extended experiments on Conceptual Web log generation and XML Mining over generated Conceptual logs. Conceptual logs are rich XML Web log containing rich information about the Web site structure and content. Furthermore they can be automatically generated starting from a proper logging facility and a conceptual application model. This allows an easier analysis of the results of the mining process, thanks to the rich information provided and allows to perform the data mining process at different levels of abstraction. In this work we used WebML as conceptual model, and XMINE as mining tool.

Key words: Web Usage Mining, XML Mining, XML Conceptual log, WebML, XMINE

1 Introduction

Web Usage Mining, often referred as Web Log Mining, aims to extract knowledge from Web servers logging facilities. Many research papers have been published on it and many commercial tools have recently reached maturity. At the same time XML is becoming widely used on the Web; nevertheless the research in the area of XML Mining is still at the first steps and few real case studies have been proposed and analyzed. In this paper we perform a data mining task over a rich XML Web log. Therefore, we adopt XML Mining techniques in the area of Web Usage Mining. In particular we collected more than 20,000 user sessions from the Web site of the Computer Science department of our university[1] and performed the XML Mining task with XMINE [1], a tool to mine rules from XML data. Various experiments and researches have been conducted in this field in the last decade; most of these researches evidenced that the most demanding task is the analysis of the results produced by the data mining process. This is mostly caused by the poor information contained in the logs about the real content browsed by users. Cooley [2] showed that, not only is the Web Usage Mining process enhanced by content and structure knowledge, but it cannot be completed without it. Hence data preprocessing becomes one of the fundamental

[1] http://www.elet.polimi.it.

task to improve the result of the Web Usage Mining task and to simplify their analysis. Only few researches on Web Usage Mining deal with the problem of enriching the information contained in the Web logs to improve the quality of the extracted knowledge. Stumme et al. [3] address the problem using Semantic Web techniques to add knowledge about the page content to the Web log; in [4] pages navigated by users are tagged with keywords extracted from themselves; Punin et all [5] enrich Web log information adding Web site's maps. A complete survey on Web Usage Mining research can be found in [6]. Here we follow a different approach: we developed a framework to automatically generate enriched Web logs from the conceptual model of the application. This approach was first introduce in [7]. This work follows the way paved by [7] conducting the first extended experiments of enriched XML log generation and enriched XML log mining. The approach is based on the Web Modeling Language (WebML) [8] and its supporting CASE tool WebRatio [9], for the design and the development of data-intensive Web applications. However, the illustrated results are of general validity and apply to any application that has been designed using a model-driven approach, provided that the conceptual schema is available and the application runtime architecture permits the collection of customized log data.

This paper is organized as follows: Section 2 summarizes the current efforts in the area of XML Data Mining and introduces the tool we used for the XML Mining task. In Section 3 we introduce the XML Conceptual logs. Section 4 presents the analyzed Web application and Section 5 reports some of the evidences we found mining the Conceptual logs. Finally, in Section 6, we address future research efforts and draw some conclusions.

2 XML data mining and XMINE

The eXtensible Markup Language (XML) has rapidly become an important standard for representing and exchanging information through its applications. With the dramatic increase of information available in XML, there is a pressing need for languages and tools to manage collections of XML documents, as well as to mine interesting information from XML document collections.

The XML data mining research can be divided in two main areas: mining frequent pattern from XML data [1,10] and classifying XML data [11,12]. The literature shows that, despite the fact that XML is more and more used and a large number of XML documents is available, most of the researches deal with classifying XML data, disregarding that data classification, can efficiently performed only starting from data pattern. Our focus is on the few researches that somehow aim at extracting frequent patterns from XML data, as these are the most used techniques in Web Usage Mining. First studies in this area used techniques derived from *Text Mining*. Text Mining is an area of data mining that focused on finding repeating patterns inside text databases, i.e. Text Mining find frequent pattern of words inside a collection of phrases. In this framework an XML document is considered as a bag of words, and patterns are extracted from such bag [13].

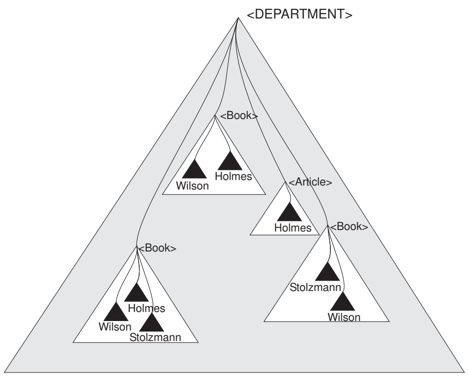


Fig. 1. A graphical representation of a XML document presenting people and publications from a computer science department. The grey triangle represents the entire document. The white triangles represent the XML fragments corresponding to the various publications that appear in the document, described by the tags `<Book>` and `<Article>`. The black triangles represent the XML *fragments* corresponding to the authors of various publications, described by the `<Author>` tag.

A second approach, *native XML data mining* is quite new to the area of data mining. As far as we know, most of the studies in this area focus on mining frequent trees inside XML files. Tree mining over XML was first proposed in [10], a similar approach is also used in [11,14]. In [1] a language to extract association rules from XML documents, `XMINE`, is proposed and extended in to mine sequential patterns in [15]. We define approaches like the one introduced by Zaki [11], that finds all the frequent tree structures repeating in a collection, *brute force* approaches, while approaches like the one in [15], that finds only the frequent patterns corresponding to a certain XML structure, *structure-based* approaches.

2.1 `XMINE`: Mining Rules from XML data

Braga et al. [1] proposes a language to mine association rules from XML data. The language is later extend in [15] to sequential rules. In this work also a formal definition of the two problems is given.

Nevertheless the complexity of its formalism, thanks to the introduction of a language to express the mining task by means of XPath and XQuery, `XMINE` can be easily used by XML experts. The language proposed is based on the assumption that information like DTD and XSD schema can simplify the data mining task reducing the problem by providing proper constraints on the structure of the expected resulting pattern.

For example, let us consider an XML document which presents various information about a department. In particular, it stores information about the available Ph.D. courses (identified by the tag `<PhDCourse>`) and about the people

in the department (`<People>`). These can be either students (`<PhDStudent>`) or professors (`<FullProfessor>`). For each of them, some personal information are stored (`<PersonalInfo>`) as well as the list of works published (`<Publications>`) such as books (`<Book>`), journal papers, or conference papers (`<Article>`). For the proposed XML document, an interesting task may be the problem of mining frequent associations among people that appear as coauthors in the publications appearing in the XML document. In practice, we are interested in finding associations of the form: "$\{Wilson\} \Rightarrow \{Holmes\}$" which states that, in the department, the papers which are authored by *Wilson* are also likely to have *Holmes* as author. Figure 1 presents a simplified depiction of the introduced XML document evidencing the XML elements relevant for the proposed task.
The `XMINE RULE` statement for the mining problem introduced will be:

```
XMINE RULE
IN document("www.cs.atlantis.edu/research.xml")
FOR ROOT IN //People/*/Publications/*
LET BODY := ROOT/Author,
    HEAD := ROOT/Author
EXTRACTING RULES WITH
 SUPPORT = 0.1 AND CONFIDENCE = 0.2
```

3 From logs to conceptual logs

Web servers can collect large amount of information in their log files and in the log files of the databases they use. These logs usually contain basic information e.g.: name and IP of the remote host, date and time of the request, the request line exactly as it came from the client, etc.

Recently, besides the de facto standards such as Common Log Format and Extended Log Format, the research of John Punin et al. proposed in [5] the specification of a new log format based on XML, the Log Markup Language (LOGML). LOGML files are obtained from standard Web logs (e.g., Common Log Format), and Web site maps expressed as XGMML, an XML language to describe graphs. LOGML generation was experimented for a simple static website.

The LOGML format, even if richer than common ones, still misses a rich description of the content of the navigated pages. Such description of content could be performed by extracting keywords or using other Web Content Mining techniques, again, augmenting the time and computational cost of generating such logs.

Intuitively, once data are available and their format can be easily processed, they can be trivially exploited to efficiently enrich Web logs. Web applications modeled and deployed using conceptual facilities can exploit the information contained in the conceptual schema of the Web application to enrich the Web logs. In particular we experimented the generation of conceptual Web logs within the WebML/WebRatio framework. WebML (Web Modeling Language) is a conceptual model for Web application design [8], which is an ingredient of a broader development methodology, supported by a CASE tool, named WebRatio [8,9].

```
<PAGE auxiliary:split-subpages="yes" graphmetadata:go="page1_go" id="page1" landmark="no"
 localize="no" name="Teacher" presentation:page-layout="BasePage(+link)" secure="no">
  <CONTENTUNITS>
    <DATAUNIT entity="ent6" graphmetadata:go="dau1_go" id="dau1" inc-links="22"
     inc-links-from-dru="2" name="Teacher Data">
      <DISPLAYATTRIBUTE attribute="Name"/>
      <DISPLAYATTRIBUTE attribute="Surname"/>
      <DISPLAYATTRIBUTE attribute="Picture"/>
      <DISPLAYATTRIBUTE attribute="Curriculum"/>
      ...
      <LINK automaticCoupling="yes" graphmetadata:go="ln807_go" id="ln807"
       name="To Publications" newWindow="no" to="dau132" type="normal"/>
      ...
    </DATAUNIT>
    <DATAUNIT entity="Section" graphmetadata:go="dau175_go" id="dau175" inc-links="1"
     inc-links-from-dru="1" name="Section">
      <SELECTOR defaultPolicy="fill">
        <SELECTORCONDITION id="sel252" name="Teacher Section"
         predicate="in" relationship="Teacher2Section" type="required"/>
      </SELECTOR>
      <DISPLAYATTRIBUTE attribute="Name"/>
    </DATAUNIT>
    <INDEXUNIT distinct="no" entity="Address" graphmetadata:go="inu87_go"
     id="inu87" name="Addresses">
      <SELECTOR defaultPolicy="fill">
        <SELECTORCONDITION id="sel124" name="Teacher Addresses" predicate="in"
         relationship="Teacher2Address" type="required"/>
      </SELECTOR>
      <DISPLAYATTRIBUTE attribute="Location"/>
      <DISPLAYATTRIBUTE attribute="Floor"/>
      <DISPLAYATTRIBUTE attribute="Office"/>
      <DISPLAYATTRIBUTE attribute="Telephone"/>
      <DISPLAYATTRIBUTE attribute="Fax"/>
    </INDEXUNIT>
    <INDEXUNIT distinct="no" entity="Email" graphmetadata:go="inu43_go"
     id="inu43" name="Email Addresses">
      <SELECTOR defaultPolicy="fill">
        <SELECTORCONDITION id="sel11" name="Teacher Email Addresses" predicate="in"
         relationship="rel33" type="Teacher2Email"/>
      </SELECTOR>
      <DISPLAYATTRIBUTE attribute="Email"/>
    </INDEXUNIT>
  </CONTENTUNITS>
</PAGE>
```

Fig. 2. A portion of XML serialization of the WebML model for the page *Teacher*, of the DEI Web site (e.g., `http://www.elet.polimi.it/people/facca`).

WebML offers a set of visual primitives for defining conceptual schema that represent the organization of the application contents and of the hypertext interface. Besides having a visual representation, WebML primitives are also provided with an XML-based representation, to specify those additional properties that would not be conveniently expressible by a graphical notation. Figure 2 reports a simplified XML specification of a hypertext page, named *Teacher*, taken from the WebML-based hypertext schema of the `http://www.elet.polimi.it` application. The page includes several content units. The first, a *data unit* publishes some attributes taken from a single instance of `Teacher`, which is an entity of the data schema. Moreover, from the *data unit* `dau1` a link originates, whose destination is a further unit (`dau132`) defined elsewhere in the application hypertext schema. A second *data unit* selects the instance to be published from the database according to a *selector condition*, specified over a relationship involving `Section`. Also, two *index unit*s are present in the page and publish lists of instances of entity `Address` and entity `Email`.

Webratio runtime environment for WebML applications logs not only the information collected normally from the Web servers but also the session identifier, e.g. `a_0YRnHNcly8`, allowing an easier reconstruction of user sessions.

The Webratio runtime provides also a *Runtime XML log* containing all the info about the processing of requested pages. The Runtime XML log includes all the events generated by the application runtime when serving a request page and populating its contents units. Each event is delimited in the XML log file by the `<event>` tag. Since each page request is managed by a specific thread, the events generated for a single page request are characterized by the same thread number. An `<event>` tag denotes either the request of an entire page, or the computation of an individual unit. It may contain further sub-tags:

- The tag `<message>` includes the event parameters. In case of content units population, it also includes the list of OIDs of the objects extracted from the data source.
- `<NDC>` stores the identifier of the conceptual element (page or unit) to which the event refers.

For example:

```
<log4j:event category="/webapps/dei/log" index="72921" timestamp="Fri, 11 June 2004 -
  02:02:01.140 AM" priority="DEBUG" thread="tcpConnection-80-4">
  <log4j:message>Continuing to serve request with id=-1503716237;
    remoteAddress=yyy.yyy.yyy.yyy; jSessionID=a_0YRnHNcly8;
    unitId=dau1; dataInstances=321;</log4j:message>
  <log4j:NDC>dau1</log4j:NDC>
</log4j:event>
```

is an event for a request to the *Teacher* page (see Figure 2). The event refers to the population of a *data unit* (`dau1`). Its `<message>` tag includes the unitID (`dau1`), the IP address and the client SessionID, and a value (`321`) representing the OID of the single database instance extracted for populating the data unit.

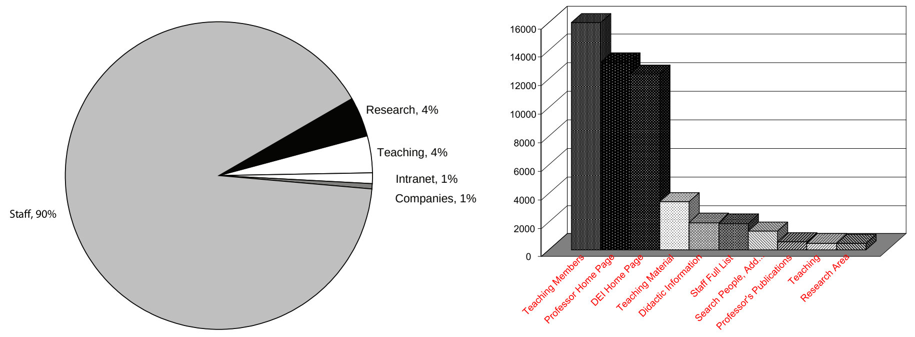
The rich informations contained in Runtime XML log, application server log, and the Web application schema are easily exploited to generate an XML Conceptual log. A fragment of a XML Conceptual log is reported in Figure 3.

4 The Case Study Web Application

First experiments on Mining XML Conceptual logs were conducted on the WebML.org Web site (`http://www.webml.org`), the reference site for the WebML community, as reported in [7]. These first experiments were not enough to stress advantages of Conceptual Web logs, as the WebML site has a quite simple conceptual model and a limited number of daily visitors. To better prove the efficacy of our methodology, experiments on a more relevant Web application are needed.

```
<ConceptualLog>
  <ConceptualSchema>
    ...
    <PAGE auxiliary:split-subpages="yes" graphmetadata:go="page1_go" id="page1" landmark="no"
     localize="no" name="Teacher" presentation:page-layout="BasePage(+link)" secure="no">
      <CONTENTUNITS>
        ...
      </CONTENTUNITS>
    </PAGE>
  </ConceptualSchema>
  <Log>
    <Session id="aFqaa3um9-1e">
      ...
    </Session>
    <Session id="a_0YRnHNcly8">
      <IPAddress>yyy.yyy.yyy.yyy</IPAddress>
      <HostName>###############</HostName>
      <Browser>Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)</Browser>
      <StartTimestamp>1089583207000</StartTimestamp>
      <EndTimeStamp>1089583257000</EndTimeStamp>
      <Duration>50000</Duration>
      <Requests>
        <Request RequestId="0">
          ...
        </Request>
        <Request RequestId="1">
          <PageName>/page1.do</PageName>
          <Page SchemaRef="page1"/>
          <Referrer SchemaRef="page8"/>
          <EntryLink SchemaRef="ln51"/>
          <RequestType>GET</RequestType>
          <RequestURI>/page1.do?dau1.oid=321&UserCtxParam=0&GroupCtxParam=0
           &ctx1=it&crc=371954722</RequestURI>
          <Bytes>69687</Bytes>
          <Status>200 - OK</Status>
          <Referer>http://www.elet.polimi.it/page8.do?link=ln51.redirect&stu34.values=it
           &src7=rossi&src6=&alt2=page12&UserCtxParam=0&GroupCtxParam=0&ctx1=it</Referer>
          <RequestTimestamp>1089583208000</RequestTimestamp>
          <RequestTime>July 11, 2004 02:02:08 AM CEST</RequestTime>
          <ElapsedTime>1000</ElapsedTime>
          <PageUnits>
            <Unit>
              <Unit_Id SchemaRef="dau1"/>
              <DataInstance>321</DataInstance>
            </Unit>
            <Unit>
              <Unit_Id SchemaRef="dau175"/>
              <DataInstance>3</DataInstance>
            </Unit>
            <Unit>
              <Unit_Id SchemaRef="inu87"/>
              <DataInstance>700</DataInstance>
              <DataInstance>707</DataInstance>
            </Unit>
            <Unit>
              <Unit_Id SchemaRef="inu43"/>
              <DataInstance>402</DataInstance>
              <DataInstance>408</DataInstance>
            </Unit>
          </PageUnits>
        </Request>
        ...
      </Requests>
    </Session>
    ...
  </Log>
</ConceptualLog>
```

Fig. 3. A fragment of the Conceptual Log for a request to the *Teacher* page illustrated in Figure 2.



(a) (b)

Fig. 4. Percentage of accesses to the different areas of the DEI Web site (a) and the top ten accessed pages (b).

Here we present the results achieved on mining Conceptual logs for the DEI Web site (`http://www.elet.polimi.it`): the Web site of the Computer Science Department of Politecnico di Milano. The DEI Web site contains more than 200 dynamic Web pages – modeled with WebML – gathering data from thousands of database's instances and more than 1,000 static Web pages belonging to the staff or to research groups. More than 2,000 people access the Web site everyday. These numbers are great enough to qualify the DEI Web application as a good field to deeply test the Conceptual Web log generation and to apply the XML Mining tasks. We generated Conceptual logs for 10 days from 11th June to 20th June 2005. The total original requests in the input Common Log Format files were more 1,500,000, while the final cleaned requests were about 350,000. The total number of users' sessions generated is 20,787 – not including robots' sessions – with an average of 17.3 requests per session. The Web application we analyzed is mainly accessed by Italian students searching for information about teachers and courses. The public part of the application modeled with WebML is divided in five areas: (i) *Research* that provides information about research areas at the DEI; (ii) *Teaching* that proves info about examinations and other teaching related topics; (iii) *Intranet* that provides informations to the DEI staff; (iv) *Companies* that provides informations to the Companies who wants to collaborate with DEI; and (v) *Staff* that provides informations about the DEI Staff. Figure 4(a) shows that the most accessed area is Staff (90%) that contains pages about Professors – the second most accessed page as shown in Figure 4(b) – and about Professors' teaching material – the forth most accessed page.

5 Results Analysis

In this section we report some of the interesting evidences we discovered analyzing the results of the different data mining tasks over the Conceptual Web logs generated for the DEI Web site.

Thanks to the richness of Conceptual logs, it is possible to extract knowledge at different levels of information abstraction. Some of the possible tasks are: (i) mining sequence over generic page sequences, i.e., considering only the sequences of page name accessed without referring to the data instances populating the page; (ii) mining sequences of data instanced pages; (iii) mining sequences of accessed entities and so on according to the selected fragments of the XML Conceptual log. The results published here regard task (i) and (ii).

5.1 Accessing Professor Home Pages

As shown in Figure 4(b), most of the user interacting with the Web site are student navigating contents published by professors. This belief is supported not only by statistics but also by mining results. In fact, most of the discovered rules with the highest confidence, regards interactions with professors' pages. In particular, we identified the most common path to access content offered by professors (Figure 5): (i) the user access the "Search Members" page where he compiles the form to search a teacher, (ii) hence, from the list of results he chooses the pertinent one and accesses the "Teacher" page. The high confidence of these rules is consistent with the fact that this path corresponds exactly to the Web application model and hence to the Web application designer objectives.

This is just one of the possible different paths that a user may navigate to access a "Teacher" page. The fact that this rule has much more higher support than rules regarding other paths to find and access professors' pages, supports the idea that this is the most efficient and effective path modeled within the Web application to satisfy such browsing goal.

Among the other frequent navigational paths performed by users to access "Teacher" pages, we notice a particular behavior for instances of teachers whose surname starts with 'A' letter. In such case we find that users prefer (i) getting the list of the whole Professors and then (ii) selecting the Professor from the first lines of the list and access his page. This is the only case where the "Staff Full List" page is used with a relevant support. This may suggest that a new list grouping professors by starting letter could be effective and useful.

Other derived rules show that students access available information in the "Teacher" page as expected. The highest support rules including the "Teacher" page as antecedent are: "$Teacher$" $\Rightarrow$ "$Available\ \ Material$" (support 0.10 and confidence 0.23); "$Teacher$" $\Rightarrow$ "$Didactic\ \ Information$" (support 0.08 and confidence 0.17) and "$Teacher$" $\Rightarrow$ "$Professor's\ \ Publications$" (support 0.02 and confidence 0.05).

A discovered rule show that users that access the "Exams and Tests Results" page, then visit the "Member Search" page with a high confidence (0.22). Probably this behavior represents the fact that the "Exams and Tests Results"

```
<SequenceRule support="0.24770289123009573" confidence="0.8326326002587322">
   <AntecedentSequence>
      <ItemSet>
         <Item>
            <PageName>
               <WebML_Page Id="page8" name="Search Members">
                  <SITEVIEW Id="sv1" name="Public">
                     <AREA Id="area5" name="Staff" landmark="yes"/>
                  </SITEVIEW>
               </WebML_Page>
            </PageName>
         </Item>
      </ItemSet>
   </AntecedentSequence>
   <ConsequentSequence>
      <ItemSet>
         <Item>
            <PageName>formpage8</PageName>
         </Item>
      </ItemSet>
      <ItemSet>
         <Item>
            <PageName>
               <WebML_Page Id="page1" name="Teacher">
                  <SITEVIEW Id="sv1" name="Public">
                     <AREA Id="area5" name="Staff" landmark="yes"/>
                  </SITEVIEW>
               </WebML_Page>
            </PageName>
         </Item>
      </ItemSet>
   </ConsequentSequence>
</SequenceRule>
```

Fig. 5. Almost the 25% of the users' session contains a user request to the "Search Members" page followed by a search for a professor and a visit to a "Teacher" page. In particular 83% of the users that access the "Search Members" performs the form submission and accesses the "Teacher" page.

page is not often used by teachers to publish exams' results, and hence, when students do not find their test results in this page they look for the teacher page by accessing the "Member Search" page. This may suggest to include within the *Staff* area a page for the exams results of each teacher, trying to improve the way teachers publish their exams results and the way students can find them. We also obtain a number of sequential rules with a quite high confidence (around 0.15) showing a relationship between teachers. These relationships are not modeled within the Web application. Analyzing the teachers involved in these relationships, we discover that these navigational relationships correspond to real relationships between teachers. We find that the involved teachers are sharing the same course or teach courses belonging to the same class year. This may suggest to improve the Web application model, allowing to include automatically links to related teachers, and adding a page that groups teachers by class year so that students can directly access all the teachers relevant to their studies.

5.2 Misleading Link Names

One of the rule with highest support and confidence, shows that users that access the "Research" page – showing info about research sectors in which DEI is involved – then access the "Search Members" page – showing form to search for the Teaching Staff Members. This rule has a quite strong support (0.01, i.e. the sequence rule is reported in about 200 sessions over $20,000$) and strong confidence (0.35, i.e. more than one third of the people that access the antecedent item of the rule then access the consequent item). To justify this navigational pattern we speculate that many users interpret the term "Research"[2] as *search for content within the DEI website*, and hence once they access the "Research" page and understand that is not what they are looking for, they move to the page that seems most promising to support their task, in this case the "Search Members" page. This may suggest to better specify the concept underlying the term "Research" using a more rich periphrasis (e.g. "Scientific Research").

Another rule shows a frequent user behavior probably caused by inappropriate link naming: users that visit the "Technical-Administrative Staff List" page then often visit the "Search Members" page. Looking in the Web log we observe that users access the latter page most of the times immediately after the first one. Furthermore, most of the requests are generated starting from the DEI home page, where links to both the two pages are present. At the time we collected the Web logs, the links to the two pages had a quite similar name. The first link was named – in Italian – "Personale Non Docente" and the latter "Personale Docente". The two links were also one belove the other, increasing the chance of confusion for users[3]. Thus we speculate that the behavior showed by the rule may be caused by the similar names of the two links and by their adjacency.

6 Conclusion and Future Work

In this paper we presented the results of the extended mining experiments we performed over the Conceptual Web log generated from our Department Web site. The experiments supported our initial hypothesis that Conceptual Web logs allow for an easier task of analyzing mining results. Furthermore they easily allowed us to mine Web logs at a different level of abstraction. The results reported evidence many problems in the current Web site and provide some directions that will be taken in account in upcoming restyling of our Department Web site. We are planning to add semantic annotation to the conceptual schema, as this should make easier the task of results analysis. This may also enable for a categorization of mined patterns according to concepts contained in the ontology used for annotation that may be exploited with clustering mining tasks.

[2] In Italian the term "Ricerca" is widely used with the meaning *search for something.*

[3] The Italian link name of the "Technical-Administrative Staff List" page has been recently update to "Personale Tecnico-Amministrativo".

Acknowledgment
The author wish to thank Pier Luca Lanzi and Maristella Matera for their precious suggestions and contributions that made this work possible.

References

1. Braga, D., Campi, A., Ceri, S., Klemettinen, M., Lanzi, P.: A tool for extracting xml association rules. In: Proceedings of ICTAI'02, 4-6 November, Crystal City, USA, IEEE Computer Society (2002)
2. Cooley, R.: The use of web structure and content to identify subjectively interesting web usage patterns. ACM Trans. Inter. Tech. **3**(2) (2003) 93–116
3. Stumme, G., Berendt, B., Hotho, A.: Usage mining for and on the semantic web. In: Next Generation Data Mining. Proc. NSF Workshop, Baltimore, Nov. 2002. (2002) 77–86
4. Heer, J., Chi, E.: Identification of web user traffic composition using multi-modal clustering and information scent. In: Proceedings of the Workshop on Web Mining, 2001 SIAM Conference on Data Mining. (2001)
5. Punin, J.R., Krishnamoorthy, M.S., Zaki, M.J.: Logml: Log markup language for web usage mining. In: WEBKDD 2001, Third International Workshop. (2001)
6. Facca, F.M., Lanzi, P.L.: Mining interesting knowledge from weblogs: a survey. Data & Knowledge Engineering **53**(3) (2005) 225–241
7. Fraternali, P., Lanzi, P.L., Matera, M., Maurino, A.: Model-driven web usage analysis for the evaluation of web application quality. J. Web Eng. **3**(2) (2004) 124–152
8. Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S., Matera, M.: Designing Data-Intensive Web Applications. Morgan Kaufmann, San Francisco, CA (2002)
9. Web Models: Webratio case tool (2006) `http://www.webratio.com`.
10. Termier, A., Rousset, M.C., Sebag, M.: Treefinder: a first step towards xml data mining. In: Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002), IEEE Computer Society (2002) 450–457
11. Zaki, M.J., Aggarwal, C.C.: Xrules: an effective structural classifier for xml data. In: KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, New York, NY, USA, ACM Press (2003) 316–325
12. Lian, W., Cheung, D.W., Mamoulis, N., Yiu, S.M.: An efficient and scalable algorithm for clustering xml documents by structure. IEEE Transactions on Knowledge and Data Engineering **16**(1) (2004) 82–96
13. Ahonen, H., Heinonen, O., Klemettinen, M., Verkamo, A.I.: Mining in the phrasal frontier. In Komorowski, H.J., Zytkow, J.M., eds.: Principles of Data Mining and Knowledge Discovery, First European Symposium, PKDD '97, Trondheim, Norway, June 24-27, 1997, Proceedings. Volume 1263 of Lecture Notes in Computer Science., Springer (1997) 343–350
14. Tan, H., Dillon, T., Feng, L., Chang, E., Hadzic, F.: X3-Miner: Mining Patterns from XML Database. In: Proceedings of the 6th International Conference on Data Mining, Text Mining and their Business Applications, Skiathos, Greece (2005)
15. Facca, F.M.: Mining patterns from xml data: a structure-based approach. Master's thesis, Politecnico di Milano, Dipartimento di Elettronica e Informatica (2004)

Semi-Supervised Learning to Extract Attribute-Value Pairs from Product Descriptions on the Web

Katharina Probst, Rayid Ghani, Marko Krema, Andy Fano
Accenture Technology Labs, Chicago, IL, USA

Yan Liu
Carnegie Mellon University, Pittsburgh, PA, USA

Abstract. We describe an approach to extract attribute-value pairs from product descriptions on the Web. The goal is to augment product databases by representing each product as a set of such attribute-value pairs. Such a representation is useful for a variety of tasks where treating the product as a set of attribute-value pairs is more useful than as an atomic entity. Examples include product recommendations, comparison of single products or complete offerings, and demand forecasting. We formulate the extraction as a classification problem and use Naïve Bayes combined with a multi-view semi-supervised algorithm (co-EM). The extraction system requires very little initial user supervision: using unlabeled data, it automatically extracts an initial seed list that serves as training data for the classification algorithm. In addition to the automatically extracted training data, the co-EM algorithm uses the unlabeled data to extract product attributes and values. Finally, the extracted attributes and values are linked to form pairs using dependency information and co-location scores. We present promising results on Web product descriptions in two categories of sporting goods products.

1 Introduction

Retailers have been collecting a growing amount of sales data that contains quite detailed information about customers and related transactions; in contrast, the information about the actual products that were sold is often sparse and limited. After discusions with many large retailers, we found that most retailers treat their products as atomic entities with very few related attributes (typically brand, size, or color). At the same time, they offer their products to customers (on a web site) that describes each product in detail, specifying the product's physical attributes, but typically in natural language, making it difficult to be used directly in many applications. The task we tackle in this paper requires a system that can process product descriptions and extract relevant attributes and values, and then form pairs by associating values with the attributes they describe. This can be accomplished by different means depending on the amount and type of information available. In this paper, we assume a scenario where a list of textual product descriptions can be scraped from a company's web site. The product descriptions are assumed to be 'unstructured' natural language text. The system described in this paper is able to extract attribute-value pairs from product descriptions with minimal human supervision. We describe the components of our system and show experimental results on a web catalog of sporting goods products.

2 Related Work

There has been a lot of research on extracting information from text documents on the Web but we are not aware of any system that addresses the same task as we are addressing in this paper. A related task that has received attention recently is that of extracting product features and their polarity from online user reviews.

Liu et al. [7] focus on extracting relevant product attributes, such as ‘focus’ in the domain of digital cameras. These attributes are extracted by use of a rule miner, and are restricted to noun phrases.The system then extracts polarized descriptors, e.g., ‘good’, ‘too small’, etc. Popescu and Etzioni [10] describe a similar approach: they first extract noun phrases as candidate attributes, and then compute the pointwise mutual information between the noun phrases and salient context patterns (such as *‘scanner has’*). Similarly to Liu et al. [7], the extraction phase is followed by an opinion word extraction and polarity detection phase. Our work is similar in that a product is expressed as a vector of attributes. The difference is that our work focuses not only on attributes, but also on extracting values, and on associating the extracted attributes with the extracted values. Also, the attributes that are extracted from user reviews are often different (and described differently) than the attributes of the products that retailers would mention. For example, a review might mention ’photo quality’ as an attribute but specifications of cameras would tend to use megapixels or the lens manufacturer in the specifications.

Information extraction with the goal of filling templates, e.g., [5, 9], is related to the approach in this paper in that we extract certain parts of the text as relevant facts. It however also differs from such tasks in several ways, notably because we do not have a definitive list of ‘template slots’ available. Recent work in bootstrapping for information extraction using semi-supervised learning has focused on the task of named entity extraction [4, 2, 3], which is related to part of the work presented here (classifying the words/phrase as attributes or values or as neither).

3 Overview of the Attribute Extraction System

Our system consists of five modules: 1) Data Collection, 2) Seed Generation, 3) Attribute-Value Entity Extraction, 4) Attribute-Value Pair Relationship Extraction, and 5) User Interaction. The modular design allows us to break the problem into smaller steps, each of which can be addressed by various approaches. In this paper, we have chosen one specific approach for each phase. We only focus on tasks 1-4 in this paper, where task 5 is largely future work that we however consider very important.

4 Data

The data required for extracting product attributes and values can come from a variety of sources such as an internal product database or from the retailer website. We crawled the web site of a sporting goods retailer (www.dickssportinggoods.com), concentrating on the domains of tennis and football. Sporting goods is an interesting and relatively challenging domain because unlike electronics, the attributes are not easy and straightforward to detect. For example, a camera has a relatively well-defined list of attributes (*resolution, zoom, memory-type*, etc.). In contrast, a baseball bat would have some typical attributes such as brand, length, material as well as others that might be harder to identify as attributes and values (*aerodynamic construction, curved hitting surface*, etc.)

The scraping process resulted in a set of product descriptions where each product is described by a list of phrases, which we use as training data. Some examples of entries in these lists are *1 tape cutter*, *4 rolls of white athletic tape*, *Cutout midfoot*, *Extended Torsion bar* , *Synthetic leather upper*, *Audio/Video Input Jack*, *Play Dry technology offers moisture management and wicking properties*, *Vulcanized latex outsole construction is lightweight and flexible*

It can be seen from these examples that the entries are not often full sentences. This makes the extraction task more difficult, because most of the phrases contain a number of modifiers. There is often no definitive answer as to what the extracted attribute-value pair should be, even for humans inspecting the data. For instance, should the system extract *cutter* as an attribute with two separate values, *1* and *tape*, or should it rather extract *tape cutter* as an attribute and *1* as a value? To answer this question, it is important to keep in mind the goal of the system to express each product as a vector of attribute-value pairs, so as to compare between products. Therefore, it is more important that the system is consistent than which of the valid answers it gives.

5 Pre-Processsing

The product descriptions collected by the web crawler are first tagged with parts of speech (POS) using the Brill tagger and stemmed with the Porter stemmer. We also replace all numbers with the unique token *#number#* and all measures (e.g., *liter, kg*) by the unique token *#uom#*. Additionally, we compute several correlation scores (Yule's Q statistic, pointwise mutual information, and χ^2) between all pairs of words and recognized one as a phrase if all of its correlation scores exceed certain thresholds.

6 Seed Generation

Once the data is collected and processed, the next step is to provide labeled seeds for the learning algorithms to learn from.

Generic and domain-specific lists as labeled seeds. We use a very small amount of labeled data in the form of generic and domain-specific lists. The generic value lists were easily available on the web and are fairly domain-independent. We use lists of colors, materials, countries, and units of measure. In addition, we use a list of domain-specific (in our case, sports) values and attributes consisting of sports teams (such as *Pittsburgh Steelers*)

These seeds are supplemented by automatically extracted attribute-value seed pairs, as described in the following section. In other words, aside from easily replaceable generic and domain-specific lists, the system works in an unsupervised fashion.

Unsupervised Seed Generation. Our unsupervised seed generation method extracts a small number of attribute-value pairs from the unlabeled data that serve as labeled data for classification. We use correlation scores to find candidates, and make use of POS tags by excluding certain words from being candidates for extraction.

Extracting attribute-value pairs is related to the problem of phrase recognition in that both methods aim at extracting pairs of highly correlated words. There are however differences between the two problems. Consider the following two sets of phrases: *back pockets*, *front pockets*, *zip pockets* as compared to *Pittsburgh Steelers*, *Chicago Bears*. The first list contains an example of an attribute with several possible values. The second list contains phrases that are not attribute-value pairs. The biggest difference between the two lists is that attributes generally have more than one possible value, as in the above example. We exploit this observation to automatically extract high-quality seeds by defining a modified mutual information metric as follows.

We consider all bigrams $w_i w_{i+1}$ as candidates for pairs, where w_i is a candidate value, and w_{i+1} is a candidate attribute. Although the modifying value does not always occur (directly) before its attribute, this heuristic allows us to extract seeds with high precision. Suppose word w (in position $i+1$) occurs with n unique words $w_{1 \ldots n}$ in position i. We rank the words $w_{1 \ldots n}$ by their conditional probability $p(w_j | w), w_j \in w_{1 \ldots n}$, where the word w_j with the highest conditional probability is ranked highest.

The words w_j that have the highest conditional probability are candidates for values for the candidate attribute w. Clearly, however, not all words are good candidate attributes. We observed that attributes generally have more than one value and typically do not occur with a wide range of words. For example, frequent words such as *the* occur with many different words. This is indicated by their conditional probability mass being distributed over a large number of words. We are interested in cases where few words account for a high proportion of the probability mass. For example, both *Steelers* and *on* will not be good candidates for being attributes. *Steelers* only occurs after *Pittsburgh* so all of the conditional probability mass will be distributed on one value whereas *on* occurs with many words with the mass distributed over too many values. This goal can be accomplished in two phases: in the first phase, we retain enough words w_j to account for a part $z, 0 < z < 1$, of the conditional probability mass $\sum_{j=1}^{k} p(w_j | w)$. In the experiments reported here, z was set to 0.5.

In the second phase, we compute the *cumulative* modified mutual information for all candidate attribute-value pairs. We again consider the perspective of the candidate attribute. If there are a few words that together have a high mutual information with the candidate attribute, then we are likely to have found an attribute and (some of) its values. We define the cumulative modified mutual information as follows:

Let $p(w, w_{1 \ldots k}) = \sum_{j=1}^{k} p(w, w_j)$. Then

$$cmi(w_{1 \ldots k}; w) = \log \frac{p(w, w_{1 \ldots k})}{(\lambda * \sum_{j=1}^{k} p(w_j)) * ((\lambda - 1) * p(w))}$$

λ is a user-specified parameter, where $0 < \lambda < 1$. We have experimented with several values, and have found that setting λ to 1 yields robust results. Setting λ to 0 implies that a candidate pair is not penalized for the word w being frequent, as long as few words cover most of its conditional probability mass. Table 1 lists several examples of extracted attribute-value pairs.

value	**attribute**
carrying, storage	case
main, racquet	compartment
ball, welt, side-seam, key	pocket
coat, durable	steel

Table 1. Automatically extracted seed attribute-value pairs

As we can observe from the table, our unsupervised seed generation algorithm captures the intuition we described earlier and extracts high-quality seeds for training the system. We expect to refine this method in the future. Currently, not all extracted pairs are actual attribute-value pairs. One typical example of an extracted incorrect pair are first name - last name pairs, e.g., *Smith* is extracted as an attribute as it occurs as part of many phrases and fulfills our criteria (*Joe Smith*, *Mike Smith*, etc.) after many first names. Other examples of incorrectly extracted attribute-value pairs include '*more* (attribute) – *much* (value)' and '*more* (attribute) – *achieve* (value)'. However, some of the incorrectly extracted examples are rare enough that they do not have much impact on subsequent steps. The current metric accomplishes about 65% accuracy in the tennis category and about 68% accuracy in the football category. We have experimented with manually correcting the seeds by eliminating all those that were incorrect. This did not result in any improvement of the final performance of the overall system, leading us to conclude that our algorithm is robust to noise and is able to deal with noisy seeds.

7 Attribute and Value Extraction

After generating initial seeds, the next step is to use the seeds as labeled training data to extract attributes and values from the unlabeled data. We formulate the extraction as a classification problem where each word or phrase can be classified as attributes or values (or as neither). The classification algorithm is described in the sections below.

7.1 Initial labeling

The initial labeling of data items (words or phrases) is based on whether they match the labeled data. We define four classes to classify words into: *unassigned, attribute, value, or neither*. The initial label for each word defaults to *unassigned* and is changed to the label of any labeled data that it matches or to *neither* if it is a stopword.

7.2 Naïve Bayes Classification

The labeled words are then used as training data for Naïve Bayes that classifies each word or phrase in the unlabeled data as an attribute, a value, or neither. The features used for classification are the words of each unlabeled data item, plus the surrounding 8 words and their corresponding parts of speech. With this feature set, we capture not only each word, but also its context as well as the parts of speech in its context. This is similar to earlier work in extracting named entities using labeled and unlabeled data [3].

7.3 co-EM for Attribute Extraction

Since labeling attributes and values is an expensive process, we use the semi-supervised learning setting by combining small amounts of labeled data with large amounts of unlabeled data. We use the multi-view or co-training [1] setting, where each example can be described by multiple views (e.g., the word itself and the context in which it occurs). The specific algorithm we use is co-EM [8]. Co-EM with Naïve Bayes has been applied to classification, e.g., by [8], but so far as we are aware, not in the context of information extraction. The separation into feature sets we use is that of the word to be classified and the context in which it occurs. Each word is expressed in *view1* by the stemmed word itself, plus the part of speech as assigned by the Brill tagger. The *view2* for this data item is a context of window size 8, i.e. up to 4 words (plus parts of speech) before and up to 4 words (plus parts of speech) after the word or phrase in *view1*. If the context around a *view1* data item is less than 8 words long, we simply limit to the context to what is available.

co-EM Algorithm: co-EM proceeds by initializing the *view1* classifier using the labeled data only. Then this classifier is used to probabilistically label all the unlabeled data. The context (*view2*) classifier is then trained using the original labeled data plus the unlabeled data with the labels provided by the *view1* classifier. Similarly, the *view2* classifier then relabels the data for use by the *view1* classifier, and this process iterates for a number of iterations or until the classifiers converge.

Each iteration consists of collecting evidence for each data item from all the data items in the other view that it occurs with. For example, if a *view2* data item $view2_k$ occurs with (i.e., in the context of) *view1* data items $view1_{i1}$ and $view1_{i2}$, then the probability distribution for $view2_k$ is the averaged distribution of the probabilities currently assigned to $view1_{i1}$ and $view1_{i2}$, weighted by the number of times $view2_k$ appears together with $view1_{i1}$ and $view1_{i2}$, respectively, as well as by the class priors.Our goal is to label unlabeled training examples that are attributes or values, and leave the others unlabeled. Co-EM can be summarized by the following steps: 1) Initialize based on labeled data (see above). 2) Use $view1$ to label $view2$. 3) Use $view2$ to label $view1$. 4) Repeat for steps 2 and 3 *n* iterations. 5) Assign final labels to words using the predictions from both views.

Estimating class priors: When estimating class priors for labeling a view, the class priors are estimated from the respective other view's probability distributions. As each data item is associated with a set of data items from the other view with which it co-occurs, together with a count of how many times the two data items co-occurred, we could gather the class prior information by traversing through each data item and weighing the probability distributions from the aligned data elements by the co-occurrence counts.

Conceptually, however, it is easier to think of the class priors as simply obtained from the training data's current distribution in the other view. In other words, when labeling $view2$ from $view1$, the class priors for the Naïve Bayes classifier are computed only on $view1$, without reference to the $view2$ data items. The resulting probability distributions from these two approaches are the same.

The class probabilities are thus estimated as follows:

$$P(c_k) = \frac{1 + \sum_i^{n_1} cnt(view1_i) * P(c_k|view1_i)}{numclasses + \sum_i^{n_1} cnt(view1_i)}$$

Estimating word probabilities: As with class priors, word probabilities from *view1* are used as training data for *view2*. For example, if a *view1* element has a probability distribution of $p(value) = 0.5$ and $p(attribute) = 0.5$, then the data element is counted as a value example with weight 0.5, but also as an attribute example with weight 0.5.

For all words $view2_j$, estimate the new probability for each class $c_k, 1 \leq k \leq 4$, from all words $view1_i, 1 \leq i \leq n_1$. In practice, the algorithm considers only those $view2_j$ items whose cooccurrence count with $view1_i$ is greater than zero.

$$P(view2_j|c_k) = \frac{1 + \sum_{i=1}^{n_1} cooc(view1_i, view2_j) * P(c_k|view1_i)}{n_2 + \sum_{i=1}^{n_1} cooc(view1_i, view2_j)}$$

$$P(view1_i|c_k) = \frac{1 + \sum_{j=1}^{n_2} cooc(view1_i, view2_j) * P(c_k|view2_j)}{n_1 + \sum_{j=1}^{n_2} cooc(view1_i, view2_j)}$$

Labeling unlabeled examples: In each iteration, we want to use the computed class and word probabilities to label unlabeled data items in the respective other view. This can be done as follows:

$$P(c_k|view2_i) \propto P(c_k) * P(view2_i|c_k)$$

if $view2_i$ does *not* match the labeled training data.

After computing the probabilities for all classes, we must renormalize:

$$P(c_k|view2_j) = \frac{P(c_k|view2_j)}{\sum_{k=1}^{numclasses} P(c_k|view2_j)}$$

However, if $view2_i$ matches the labeled training data,

$$P(c_k|view2_i) = InitialLabeling.$$

$$P(c_k|view1_i) \propto P(c_k) * P(view1_i|c_k)$$

if $view1_i$ does *not* match the labeled training data. As in the case of $view2$, we will need to renormalize after computing the probabilities for each class. Also as above, if $view1_i$ matches the labeled training data,

$$P(c_k|view1_i) = InitialLabeling.$$

Assigning co-EM probabilities to $\langle \boldsymbol{view1_i}, \boldsymbol{view2_j} \rangle$ pairs: After co-EM is run for a pre-specified number of iterations, we assign final co-EM probability distributions to all $\langle view1_i, view2_j \rangle$ pairs as follows:

$$P(c_k|\langle view1_i, view2_j \rangle) = \frac{P(c_k|view1_i) + P(c_k|view2_j)}{2}$$

Final labels are assigned to words and phrases by averaging the predictions of each view's classifier. It should be noted that words that are tagged as attributes or values are not necessarily extracted as part of an attribute-value pair in the next phase. They will only be extracted if they form part of a pair, or if they occur frequently enough by themselves or as part of a longer phrase. The next section will describe this in greater detail.

8 Finding Attribute-Value Pairs

After the classification algorithm has assigned a (probabilistic) label to all unlabeled words, a final important step remains: using these labels to tag attributes and values in the actual product descriptions, i.e., in the original data, and finding correspondences between words or phrases tagged as attributes and values. The classification phase assigns a probability distribution over all the labels to each word (or phrase). This is not enough, because aside from n-grams that are obviously phrases, some subsequent words that are tagged with the same label should be *merged* to form an attribute or value phrase. Additionally, the system must establish *links* between attributes (or attribute phrases) and their corresponding values (or value phrases), so as to form attribute-value pairs. Some unlabeled data items contain more than one attribute and more than one value, so that it is important to find the correct associations between them. We accomplish merging and linking in an interleaved fashion, using the following steps:

– **1:** Link attributes and values if they match a seed pair.

– **2:** Merge words of the same label into phrases if their correlation scores exceed a threshold.
– **3:** Link attribute and value phrases based on directed dependencies as given by a dependency parser [6]: attribute phrases and value phrases can form a pair if there is a governor-dependent relationship between them.
– **4:** Link attribute and value phrases if they exceed a correlation score threshold: unassigned attribute phrases are linked with value phrases if their words exceed a correlation threshold.
– **5:** Link attribute and value phrases based on proximity: unassigned attribute phrases are linked with value phrases if if they are adjacent.
– **6:** Adding known, but not overt, attributes: material, country, and/or color.
– **7:** Extract binary attributes, i.e., attributes without values, if they appear frequently or if the unlabeled data item consists of only one word.

Even after all the above pair identification steps, some attribute or value phrases can remain unaffiliated. Some of them are extracted noise, and should not be output. Others are valid attributes with binary values. For instance, the data item *Imported* is a valid attribute with two possible values: *true* or *false*, where the value is simply assigned by the absence or presence of the attribute. We extract only those attributes that are single word data items and those attributes that occur frequently in the data as a phrase.

9 Evaluation

We present evaluation results for experiments performed on tennis and football categories. The tennis category contains 3194 unlabeled data items (i.e., individual phrases from the bulleted list of product descriptions), the football category 72825 items. Automated seed extraction resulted in 169 attribute-value pairs for the tennis category and 180 pairs for football. Table 2 shows a sample list of extracted attribute-value pairs (i.e., the output of the full system), and the phrases that they were extracted from. We ran

Full Example	**Attribute**	**Value**
1 1/2-inch polycotton blend tape	polycotton blend tape	1 1/2-inch
1 roll underwrap	underwrap	1 roll
1 tape cutter	tape cutter	1
Extended Torsion bar	bar	Torsion
Synthetic leather upper	#material# upper	leather
Metal ghillies	#material# ghillies	Metal
adiWear tough rubber outsole	rubber outsole	adiWear tough
Imported	Imported	#true#
Dual-density padding with Kinetofoam	padding	Dual-density
Contains 2 BIOflex concentric circle magnet	BIOflex concentric circle magnet	2
93% nylon, 7% spandex	#material#	93% nylon 7% spandex
10-second start-up time delay	start-up time delay	10-second

Table 2. Examples of extracted pairs for system run with co-EM

our system in the following three settings to gauge the effectiveness of each component: 1) only using the automatically generated seeds and the generic lists ('Seeds' in the tables), 2) with the baseline Naïve Bayes classifier ('NB'), and 3) co-EM with Naïve Bayes ('coEM'). To make the experiments comparable, we do not vary pre-processing or seed generation, and keep the pair identification steps constant as well.

The evaluation of this task is not straightforward. The main problem is that people often do not agree on what the 'correct' attribute-value pair should be. Consider the example *Audio/JPEG navigation menu*. This phrase can be expressed as an attribute-value pair in multiple ways:

Possible Attribute	**Possible Value**
navigation menu	*Audio/JPEG*
menu	*Audio/JPEG navigation*
Audio/JPEG navigation menu	*#true#*

In the last case, the entire phrase is considered a binary attribute. All three pairs are both possibly useful attribute-value pairs. The implication is that a human annotator will make one decision, while the system may make a different decision (with both of them being consistent). For this reason, we give partial credit to an automatically extracted attribute-value pair, even if it does not completely match the human annotation. In some cases, an extracted pair deserves only partial credit, while in other cases, the automatically extracted pair is an equally valid attribute-valid pair.

For each of the metrics, we report *type* and *token* performance. Type performance (at the data item level, i.e., at the level of individual product description phrases) refers to performance for unique examples (each example contributes the same regardless of frequency). The data sets contain a number of duplicates, as many attributes apply to more than one product. Token performance refers to performance including duplicates, therefore emphasizing those examples that occur more frequently than others.

9.1 Precision

To measure precision, we evaluate how many automatically extracted pairs match manual pairs completely, partially, or not at all. The percentage of pairs that are fully *or* partially correct is useful as a metric especially in the context of human post-processing: partially correct pairs are corrected faster than completely incorrect pairs. Tables 3 and 4 list results for this metric for both categories, and for both *type* and *token* evaluations.

	Seeds	**NB**	**coEM**
# corr pairs	14	20	50
# part corr pairs	54	73	132
% fully correct	**20.29**	**21.28**	**26.60**
% full or part correct	**98.56**	**98.94**	**96.81**
% incorrect	**1.44**	**1.06**	**3.19**

	Seeds	**NB**	**coEM**
# corr pairs	252	264	316
# part corr pairs	202	247	378
% fully correct	**54.90**	**51.16**	**44.44**
% full or part correct	**98.91**	**99.03**	**97.60**
% incorrect	**1.08**	**0.97**	**2.39**

Table 3. *Type* (left) and *Token* (right) Precision for *Tennis* Category

The results show that all three systems achieve very high performance for partially correct pairs. As expected, seed generation alone achieves higher accuracy than the system achieves when using unlabeled, and thus noisy, data. As we will see in the following section, however, the decrease in precision when co-EM is used is more than offset by a large increase in recall.

	Seeds	NB	coEM
# corr pairs	12	18	39
# part corr pairs	63	95	159
% fully correct	**15.38**	**14.44**	**17.65**
% full or part correct	**96.15**	**90.40**	**89.59**
% incorrect	**3.85**	**9.60**	**10.41**

	Seeds	NB	coEM
# corr pairs	4704	5055	6639
# part corr pairs	8398	10256	13435
% fully correct	**35.39**	**31.85**	**32.04**
% part or full correct	**98.56**	**96.48**	**96.88**
% incorrect	**1.44**	**3.52**	**3.12**

Table 4. *Type* (left) and *Token* (right) Precision for *Football* Category

9.2 Recall

Whenever the system extracts a partially correct pair for an example that is also given by the human annotator, the pair is considered recalled. The results for this metric can be found in tables 5 and 6. Unlike for precision, the recall differs greatly between system settings. More specifically, co-EM aids in recalling a much larger number of pairs, whereas seed generation and Naïve Bayes result in relatively poor recall performance.

	Seeds	NB	coEM
# recalled	66	87	167
% recalled	**27.62**	**36.40**	**69.87**

	Seeds	NB	coEM
# recalled	451	502	668
% recalled	**51.25**	**57.05**	**75.91**

Table 5. *Type* (left) and *Token* (right) Recall for *Tennis* Category

	Seeds	NB	coEM
# recalled	68	98	164
% recalled	**32.69**	**47.12**	**78.85**

	Seeds	NB	coEM
# recalled	12629	14617	17868
% recalled	**39.21**	**45.38**	**55.48**

Table 6. *Type* (left) and *Token* (right) Recall for *Football* Category

9.3 Word-based Label-independent Precision and Recall

Often there is partial overlap between an automatically extracted pair and a pair given by a human annotator. Sometimes both pairs are equally valid, and sometimes the automatically pair is useful even if it is not completely correct, because it can easily be corrected by a human annotator. For this reason, we also measure the word overlap between manual and automatic pairs. This gives us an idea of how well the system can predict that a word should be part of a pair, even though it may confuse whether the word should be tagged as an attribute or a value. We define the word overlap, word precision, word recall, and word F1 in the standard way. We also measure the amount of 'confusion', i.e., how often a (human-tagged) value word was automatically labeled as an attribute or vice versa. Tables 7 and 8 contain the detailed results for this metric.

As was discussed in the seed extraction section, we experimented also with correcting the automatically extracted seeds and running our system with the corrected seeds. This experiment was run only for tennis with co-EM. The result was no significant change in performance. This leads us to conclude that our algorithm is quite robust to noise. It also leads us to the conclusion that the time of a human annotator is likely better spent correcting the final output of the system rather than the input seeds. Correcting the input seeds does not necessarily lead to improved performance, whereas correcting complete output pairs is likely to do so. We will explore this issue further in the context of the active learning phase in our system.

	Seeds	**NB**	**coEM**
precision	93.84	93.35	89.53
recall	64.88	76.74	77.46
F1	73.73	82.24	81.47
confusion	10.27	22.68	26.15

	Seeds	**NB**	**coEM**
precision	96.19	95.88	93.11
recall	80.11	84.19	82.16
F1	85.29	88.13	85.84
confusion	4.76	11.14	16.64

Table 7. *Type* (left) and *Token* (right) Label-independent Word-based Results for *Tennis*

	Seeds	**NB**	**coEM**
precision	91.03	83.53	80.07
recall	61.71	69.50	69.69
F1	71.17	73.75	72.47
confusion	18.59	26.04	27.19

	Seeds	**NB**	**coEM**
precision	97.76	94.22	92.94
recall	76.10	81.35	79.70
F1	83.05	85.23	83.47
confusion	17.80	25.37	25.65

Table 8. *Type* (left) and *Token* (right) Label-independent Word-based Results for *Football*

9.4 Precision Results for Most Frequent Data Items

As the training data contains many duplicates, it is more important to extract correct pairs for the most frequent pairs than for the less frequent ones. In this section, we report precision results for the most frequently data items. This is done by sorting the training data by frequency, and then manually inspecting the pairs that the system extracted for the most frequent 300 data items. This was done only for the system run that includes co-EM classification. We report precision results for the two categories (*tennis* and *football*) in two ways: first, we do a simple evaluation of each unique data item. Then we weight the precision results by the frequency of each sentence. In order to be consistent with the results from the previous section, we define five categories that capture very similar information to the information provided above. The five categories contain *fully correct* and *incorrect*. Another category is *Flip to correct*, meaning that the extracted pair would be *fully* correct if attribute and value were flipped. *Flip to partially correct* refers to pairs that would be *partially* correct if attribute and value were flipped. Finally, we define *partially correct* as before. Table 9 shows the results.

	T nW	T W	F nW	F W
% fully correct	**51.25**	**55.89**	**51.90**	**60.01**
% flip to correct	**12.08**	**20.14**	**9.62**	**10.25**
% flip to partially correct	**2.92**	**1.75**	**0.87**	**2.14**
% partially correct	**32.92**	**21.74**	**35.27**	**25.98**

Table 9. *Non-weighted* and *Weighted* Precision Results for *Tennis* and *Football* Categories. ‘T’ stands for *tennis*, ‘F’ is *football*, ‘nW’ *non-weighted,* and ‘W’ is *weighted*

10 Discussion

The results show that we can learn product attribute-value pairs in a largely unsupervised fashion with encouraging results. One conclusion is that there is some confusion over which label an extracted word or phrase should have. This is consistent with human disagreement over the labels. Confusion levels increase when co-EM is added to the system, indicating that there were not enough seeds to train a strong classifier to differentiate between attributes and values. Future work will include user-specified lists that can serve as attribute seeds. Such labeled examples can be provided as part of an interactive step or before learning takes place, as is done currently.

The baseline Naïve Bayes algorithm also outperforms the seed only algorithm in recall. This is not surprising, as the seeds are used as labeled training data for Naïve Bayes, which in turn labels additional examples that cannot be labeled by the seeds only. It does, however, not match the recall performance of co-EM, and only outperforms co-EM slightly in terms of precision for *tennis*, but not so for *football* .

Evaluating precision on the most frequent data items yields similar results. We show that there are few incorrect pairs, and we show that especially if we weight by the frequency, the number of completely correct examples is encouragingly high. Furthermore, a fair number of examples can become completely correct if flipped. In the future, we will investigate techniques to detect pairs that should be flipped, which could lead to improved precision. Finally, we can conclude that the results are consistent for both categories, making a strong case for the scalability of the system to other domains.

11 Conclusions and Future Work

We plan to focus on adding an interactive step to the extraction algorithm that will allow users to correct extracted pairs as quickly and efficiently as possible. We are experimenting with different active learning algorithms to minimize the number of corrections required to improve the sytem. We also plan on experimenting with other categories such as office supplies. We believe that a powerful attribute extraction system can be useful in a wide variety of contexts, as it allows for the normalization of products as attribute-value vectors, which in turn enables fine-grained comparison between products and assortments and improve a variety of applications such as product reommender systems, price comparison engines, demand forecasting, assortment optimization and comparison systems.

References

1. A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *COLT-98*, 1998.
2. M. Collins and Y. Singer. Unsupervised Models for Named Entity Classification. In *EMNLP/VLC*, 1999.
3. R. Ghani and R. Jones. A comparison of efficacy of bootstrapping algorithms for information extraction. In *LREC 2002 Workshop on Linguistic Knowledge Acquisition*, 2002.
4. R. Jones. Learning to extract entities from labeled and unlabeled text. Ph.D. Dissertation, 2005.
5. A. M. Kristie Seymore and R. Rosenfeld. Learning hidden markov model structure for information extraction. In *AAAI 99 Workshop on Machine Learning for Information Extraction*, 1999.
6. D. Lin. Dependency-based evaluation of MINIPAR. In *Workshop on the Evaluation of Parsing Systems*, 1998.
7. B. Liu, M. Hu, and J. Cheng. Opinion observer: Analyzing and comparing opinions on the web. In *Proceedings of WWW 2005*, 2005.
8. K. Nigam and R. Ghani. Analyzing the effectiveness and applicability of co-training. In *Proceedings of the Ninth International Conference on Information and Knowledge Management (CIKM-2000)*, 2000.
9. F. Peng and A. McCallum. Accurate information extraction from research papers using conditional random fields. In *HLT 2004*, 2004.
10. A.-M. Popescu and O. Etzioni. Extracting product features and opinions from reviews. In *Proceedings of EMNLP 2005*, 2005.

Mining Associations from Web Query Logs

Benjamin Rey, Pradhuman Jhala

Yahoo! Research,
3333 W Empire Avenue, Burbank, USA
{benjamir,pjhala }@yahoo-inc.com

Abstract. A web searcher successfully querying for "skis" might also benefit from the results for "ski gloves", since these items are associated with the same task. However, after a successful query for "skis", "snowboard" may not be useful: these two queries refer to substitutable items, and a user who has the first often has no need for the second. Algorithms for identifying related phrases and queries have typically been agnostic with respect to substitutability and associativity, and identify a mix of both. In this paper we focus on mining associated-intent queries, by distinguishing them from substitutable-intent queries. We describe an algorithm which derives user intent associations from search query session logs, based on the assumption that there are three types of relationship between queries in sessions: similar queries, associated queries and unrelated queries. Our approach is to first remove the similar relationship, to help the associative relationship surface out of the noise. To evaluate our method, we labeled query pairs coming from this algorithm, as well as coming from an algorithm which focuses on producing similar relationship. We found that our method was successful at increasing the proportion of associated query suggestions.

Keywords: Intent associations, Recommendation, Mining, Query Logs.

1 Introduction

Mining web search session queries, we can find pairs of queries such as {“*baby toys*”,” *cribs*”}. We consider such pairs as associated by their intent. Indeed, a user who has successfully performed a search about “*baby toys*” is likely to be later interested in “*cribs*”. Such pairs of queries can be really useful for applications such as commercial add-on suggestions, where once a user has completed a purchase, we suggest him an associated item.

Several recent papers have proposed related query suggestion algorithms ([1],[2],[3]), however they don’t make the distinction between pairs such as “*baby toys*” ⇔ “*infant toys*”, which we call substitutable queries (if the user is satisfied with result of one, he doesn’t need the other) and intent-associated queries. These approaches tend to produce queries which are substitutable with the initial query, together with some proportion of associated queries and some noise.

In [4], we proposed a model to bias the type of suggestions towards substitutable queries. In this work, we focus our effort in producing more suggestions of the intent-associated type, by leveraging our previous model and removing substitutable pairs from the query set we are mining.

2 Query Session Mining

We assume that web-searchers focus for a while on the main subject of their search; issue a few queries about a specific intent. Then, once they are satisfied, their next query (immediately, later in the day or even another day) will be either about an associated subject or a totally new subject, and they will start a new group of queries focused on this new intent.

If many users follow the same pattern, we can mine the query sessions and find relationships between concepts. To better understand the type of relationship we can mine, we can use an example of 3 user sessions (see Table 1.).

Table 1. Sessions for 3 users, with intent and the relationship between the different intents

	User 1	
leather sofa	A user looking for a couch sometime in the morning	Initial intent made of similar queries
leather sofa bed		
leather couch		
coffee table	Later in the afternoon, look for other pieces of furniture	Associated intent
cocktail table		
tickets los angeles lakers	And tickets to go out in the evening	New intent
tickets staple center LA lakers		
	User 2	
leather sofa	Somebody about to move in a new empty apartment, searching for all the pieces of furniture in a row	Initial intent, several similar queries
leather couch		
black leather couch		
Cocktail table		Associated intent
coffee table		
tv stand		Associated intent
television stand		
	User 3	
leather couch	A user looking for a new sofa	Initial intent
black leather couch		
history of new york city	And things about new york later in the day	new intent
books new york city		

From these 3 users, we would like to find the following relationships:

- highly substitutable queries: *"leather sofa"* ⬄ *"leather couch"* and *"coffee table"* ⬄ *"cocktail table"*
  A user who wants a *"leather sofa"* can switch to a *"leather couch"*
- associated queries: *"leather sofa"* <> *"cocktail table"*
  A user interested in furnishing his apartment will be interested in both items.

Using co-occurrence statistics in a session, the strongest links will be of the substitutable type. Thus the link between the associated queries in a user session might be too weak to be noticed. If we were able first to remove the co-occurrence counts of the query pairs with substitutable-intent, before we compute a statistical test on the co-occurrence of all query pairs, the strongest link will be about queries with associative-intent.

We have ways of predicting that two queries are substitutable. In [4] we developed an algorithm to mine consecutive queries from search logs, and showed that this way we could get highly substitutable pairs of queries. In addition, we have observed that if two queries share many words, or have a small edit distance, they are likely to be about the same intent ([1][4]). We will be using these algorithm and rules to remove the similar-intend pairs from the sessions. Then a statistical test will only focus on removing noisy associations from true associations.

3 Algorithm

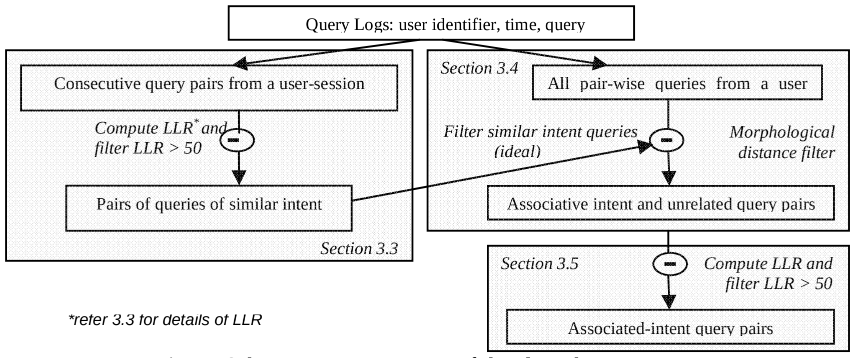

Query Logs: user identifier, time, query
Consecutive query pairs from a user-session
Compute LLR* and filter LLR > 50
Pairs of queries of similar intent
Section 3.3
Section 3.4
All pair-wise queries from a user
Filter similar intent queries (ideal)
Morphological distance filter
Associative intent and unrelated query pairs
Section 3.5
Compute LLR and filter LLR > 50
Associated-intent query pairs
*refer 3.3 for details of LLR

Fig. 1. Schematic representation of the algorithm

3.1 Overview

From the raw web query logs consisting user identifier, timestamp, and query, first, we find all the queries belongs to a user-session. After that the algorithm proceeds in two paths – one to find similar intent query pairs (section 3.3) and the other to find associative intent query pairs (section 3.4 and 3.5). Then the similar intent query pairs are filtered out from the associated intent query pairs.

3.2 What constitute a user-session

Typically a web-search session is defined by "*same user, no long elapsed time between two queries*". However, such definition is not the most appropriate for our task, as we assume that queries from such a small time span are about a unique intent. On the contrary, if we enlarge our concept of a session to a full day, or even a week, the user mind will have much more time to wander and the intents will be more diversified. For this study, we only experienced 1-day session. But will be experimenting longer period in the future.

3.3 Building substitutable query pairs

We will use method described in [4] to remove substitutable query pairs from the pool of query pairs. The basic steps are the following.
First, we compute all the consecutive pairs of queries from the logs. Then, we use the pair independence hypothesis likelihood ratio test [5] to keep only the relevant pairs. This test is based on the frequencies of each query separately in all the pairs, the frequency of a query pair, and the total number of pairs. In our empirical observation, query pairs coming from this method with log-likelihood ratio (LLR) score greater than 50 are very good substitutable query pairs.

3.4 Building pairs of queries of different intents

To generate query pairs, we take queries from a user session and generate all possible pair-wise query combinations. Then, we remove pairs which are likely to be about the substitutable intent using the following methods based on our empirical judgment (see Table 3. for an example).

- edit distance < 40%, to capture spelling changes,
- prefix, suffix overlap > 40% to capture query refinement,
- substitutable pairs with LLR>50: to capture substitutable.

Table 3. Pairs of queries kept or deleted from user 1's session

~~leahter sofa~~	~~leather sofa bed~~	~~low Edit distance~~
~~leather sofa~~	~~leather sofa bed~~	~~100% prefix overlap~~
~~leather sofa~~	~~leather couch~~	~~Substitutable~~
leather sofa	coffee table	Keep
leather sofa	cocktail table	Keep
leather sofa	tickets los angeles lakers	Keep
leather sofa	tickets stapple center	Keep
~~leather sofa bed~~	~~leather couch~~	~~Substitutable~~
…	…	
~~tickets los angeles lakers~~	~~tickets staple center los angeles lakers~~	~~word overlap~~

Unfortunately for the experiment we have carried so far, we didn't have the resources to do the full "*substitutable pairs with LLR>50*" filter at this stage and we had to do it only as a post processing stage. Instead, we used only the confidence score developed in [4], which is based on edit distance, word distance and number of substitutions

3.5 Statistical test and Post processing

To compute if a pair of query co-occurs more frequently than by chance, we use the same LLR test as for the substitutable pairs (see examples in table 4). Except that we use a normalized frequency count for the pairs. Indeed, making all possible query pair combination has the side effect of having each query occurring up to as many queries in the session even though the user issued the queries only once. We used the following normalization:

weight each pair in a session = initial #query in session / #pairs kept for this session

Table 4. examples of weight and LLR[1]

Term1	**Freq(term1)**	**Term2**	**Freq(term2)**	**Freq(pair)**	**LLR(pair)**
leather sofa	2072	coffee table	3115	51	**779.2**
leather sofa	2072	cocktail table	2301	87	**1478.2**
leather sofa	2072	tickets los angeles lakers	175	2	29.0
leather sofa	2072	tickets stapple center	351	1	11.7
leather sofa bed	97	coffee table	3115	17	**329.0**
…		…			

We now remove all the pairs which have a LLR<50 to remove the unrelated pairs. After doing this filter, we have fewer query pairs and this time we can easily remove pairs which are have a LLR>50 in the substitutable model.
The remaining query pairs are likely to be of associative intent.

4 Experiments

4.1 Data

To train our algorithm, we used web-search query logs from 2005 of Yahoo! web search query logs from shopping data (http://shopping.yahoo.com, 700M queries). For our first experiments, we restricted the source of data to the ones coming from Yahoo! shopping because it contains queries with commercial intent, and commercial

[1] The frequencies are coming from millions of other user session, not only the 3 ones we've seen earlier. For this example, we assume a total number of considered pairs of 700M.

associated intent seems easier to judge. However, we ran extra experiments on two other sources of data: Y! Reference (encyclopedia-type information), and Y! Music.

4.2 Evaluation

We sampled 100 queries from the query logs of later time (first week of March 2006) and generated all suggestions using our algorithm. For each query, we randomly picked two suggestions out of all the possible candidates. We did the same using the substitutable. Then we manually classified the 4 suggestions per query into 3 labels. And found out that we had been successful at boosting the proportion of associated intent suggestions: 42% versus 14% (see Table 5).

Table 4. Proportion of each type of relation for associated and substitutable models

	Associated	Substitutable/Closely related	unrelated
Associative model	**42%**	27%	31%
Substitutable model	14%	76%	10%

We can see some good examples in table 5. There, the associated model finds mostly associated suggestions contrarily to substitutable model finds mainly equivalent queries or close specification/generalization.

Table 6. top 5 suggestions from query association and query substitutions

Source	**Initial query**	**using association pairs**	**using substitutable**
Shopping	baby toys	stroller	baby bath toys
		car seat	baby books
		cribs	baby toysrus
		high chair	educational baby toys
		toys r us	infant toys
Music *(labeling would be much more suggestive)*	aaliya	eminem	aaliyah
		bow wow	
		50 cent	
		the used	
		system of a down	
Reference	Karate	geisha	karate gi
		self defense	karate dvd
		marshal arts	karate kid
		tai chi	karate bag
		boxing	learn karate

When the model fails, it picks suggestions which are too weakly related and in the end don't make sense. E.g.: *"pictures of home<>florida", "television<>watching".*

5 Discussions and future work

5.1 Machined learned model to rank the candidates and improve precision

Our work presented in this paper was focused on generating a pool of candidates for queries with as much associate-intent suggestions as possible. If we can build a machine learned model to predict the quality of a suggestion, then we can use this model to pick the top best suggestions for a query out of all the candidates we get.
In our previous work on query rewrite, we have used such an approach. After learning a model to rank the suggestions and keeping only the few top best ones, we were able to boost the amount of substitutable suggestions from 50% to 80%. We can hope that such an approach would be successful here again, and that we can turn our 42% associated queries into a much better number.

5.2 Using head term identification to improve coverage

As our approach is dealing only with queries which are frequent enough to have co-occurred more than by chance with some other queries, we won't be able to suggest any associated-intent term for infrequent queries. This is particularly an issue for long queries, or queries which are very specific such as product references. Even if we do have some suggestions for these queries, they have been built on little evidence, and we noticed that these are the queries for which we had the poorest suggestion quality. To address this issue, one solution would be to use a head term identification algorithm. If we are able to identify that "hp deskjet printer 932c" is about "deskjet printer", or even "printer", we will be able to suggest things like "scanner" or "digital camera", which are complementary to the head term as well as the initial query.

6 Potential applications

There are several interesting applications to intent-associated query suggestions. The first which comes to mind would be as an add-on suggestions tool in commercial sites. Another application could be for web-assisted search. Nowadays, web-search engines suggest closely related queries to help the user refine his intent. This is very helpful while the user is browsing for a specific intent. But if we could determine when the user is satisfied with his current results and is about to quit, we could suggest him an associated search term. This might retain his interests and he would continue his search. We could also use it as a way of diversifying the commercial ads displayed when a user is searching the web.

7 Conclusion

We have developed an algorithm to mine web search session and determine intent-associated queries. Our algorithm allows us to get a much higher proportion of associated suggestions than usual query related suggestion algorithm. However, there are ways to improve our algorithm by using more labeled data, and by having a machine learned algorithm to rank our suggestions.

References

1. B. M. Fonseca, P. Golgher, B. Pôssas, etc. Concept-based interactive query expansion. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management (CIKM'05)*, Bremen, Germany, 2005.
2. S. Chien, and N. Immorlica. Semantic similarity between search engine queries using temporal correlation. In *Proceedings of the 14th International Conference on World Wide Web (WWW'05)*, Chiba, Japan, 2005.
3. Shi, X. and Yang, C. C. 2006. Mining related queries from search engine query logs. In Proceedings WWW '06. ACM Press, New York, NY, 943-944.
4. R. Jones, B. Rey, O. Madani, W. Greiner: Generating Query Substitutions. WWW2006, May 22-26, 2006, Edinburgh, UK.
5. T. E. Dunning. Accurate methods for the statistics of surprise and coincidence. Computational Linguistics, 19(1):61{74, 1993}.

Collaborative Filtering: Fallacies and Insights in Measuring Similarity ⋆

Panagiotis Symeonidis, Alexandros Nanopoulos, Apostolos N. Papadopoulos, and Yannis Manolopoulos

Aristotle University of Thessaloniki, Dept. of Informatics, 56224, Greece
{symeon, alex, apostol, manolopo}@delab.csd.auth.gr

Abstract. Nearest-neighbor collaborative filtering (CF) algorithms are gaining widespread acceptance in recommender systems and e-commerce applications. These algorithms provide recommendations for products, based on suggestions of users with similar preferences. One of the most crucial factors in the effectiveness of nearest-neighbor CF algorithms is the similarity measure that is used. The most popular measures are the Pearson correlation and cosine similarity. In this paper, we identify existing fallacies in the calculation of these measures. We propose a novel approach, which addresses the problem and substantially improves the accuracy of CF results. Moreover, we propose an evaluation procedure that produces reliable conclusions about the performance of nearest-neighbor CF algorithms. Through the proposed evaluation procedure, our experimental results identify the problems of existing approaches (which could not be revealed with existing evaluation procedures) and illustrate the superiority of the proposed approach.

1 Introduction

Information Filtering has become a necessary technology to attack the "information overload" problem. In our everyday experience, while searching on a topic (e.g., products, movies, etc.), we often rely on suggestions from others, more experienced on it. In the Web, however, the plethora of available suggestions renders it difficult to detect the trustworthy ones. The solution is to shift from individual to collective suggestions. Collaborative Filtering (CF) applies information retrieval and data mining techniques to provide recommendations based on suggestions of users with similar preferences. CF is a very popular method in recommender systems and e-commerce applications. Two types of CF algorithms have been proposed: (a) *nearest-neighbor* (a.k.a. memory-based) algorithms, which rely on finding the most similar ones among the past users, and (b) *model-based* algorithms, which develop a model about user ratings. Research results and practical experience have reported that nearest-neighbor algorithms present excellent performance in terms of accuracy, for multi-value rating data [7].

Nearest-neighbors CF algorithms are influenced by several factors. The similarity measure for finding nearest-neighbors, is among the most crucial ones.

⋆ This work is conducted while the first two authors where scholars of the Greek State Scholarship Foundation.

Related research has mainly used as similarity measures the Pearson correlation and the cosine similarity.[1] One issue that impacts the accuracy of CF is the *sparsity* of past users' ratings, which emanates from the fact that each user usually rates only a very small percentage of the total items (maybe less than 0.1%). The measuring of similarity is affected by sparsity, especially due to the choice that has been followed in related work to compute the similarity between two users only with respect to the items that have been rated by *both* of them. This leads to the finding of spurious neighbors and to inability of providing correct recommendations. The reason is twofold: (a) similarity is implausibly computed based on an inadequate number of items, and (b) by ignoring the items rated by only *one* of the two users, we do not consider how much their preferences may differ. Nevertheless, the procedures used so far for the assessment of nearest-neighbor CF algorithms, are not able to identify the inefficiencies caused by the aforementioned choice.

In this paper, we first provide a thorough analysis of the factors involved in the computation of similarity measures and in the evaluation of the nearest-neighbor CF algorithms. During our examination we identify choices that have been incorrectly adopted in related work. Next, we propose a new approach, which addresses the problem and substantially improves the accuracy of CF results. Our contributions are summarized as follows:

- The revealing of existing fallacies in popular similarity measures.
- A novel method for similarity computation and an evaluation procedure that produces reliable conclusions about the performance of nearest-neighbor CF algorithms.
- Experimental results which take into account many factors and demonstrate the superiority of the proposed method (more than 40% improvements in terms of precision).

The rest of this paper is organized as follows. Section 2 summarizes the related work, whereas Section 3 contains the analysis of the examined CF factors. The proposed approach is described in Section 4. Experimental results are given in Section 5. Finally, Section 6 concludes this paper.

2 Related work

In 1992, the Tapestry system [3] introduced Collaborative Filtering (CF). In 1994, the GroupLens system [11] implemented a CF algorithm based on common users preferences. Nowadays, it is known as user-based CF algorithm, because it employs users' similarities for the formation of the neighborhood of nearest users. Since then, many improvements of user-based algorithm have been suggested, e.g., [5].

In 2001, another CF algorithm was proposed. It is based on the items' similarities for a neighborhood generation [13, 8]. Now, it is denoted as item-based or item-item CF algorithm, because it employs items' similarities for the formation of the neighborhood of nearest users.

[1] Cosine similarity is related to Pearson correlation which represents the angular separation between two normalized data vectors measured from the mean, while the cosine similarity measures the angular separation of two data vectors measured from zero.

Most recent work followed the two aforementioned directions (i.e., user-based and item-based). Herlocker et al. [6] weight similarities by the number of common ratings between users/items, when it is less than some threshold parameter γ. Deshpande and Karypis [2] apply item-based CF algorithm combined with conditional-based probability similarity and Cosine Similarity measures. Finally, Breese et al. [1] proposed the *default-voting* technique, which follows a similar direction with ours. Nevertheless, differently from [1], our focus is only on this technique, as we analyze in significant depth its motivation, we provide extensive experimental results for its evaluation and focus on its impact in order to aware researchers in CF about its usage (as the majority of works subsequent of [1] did not take this technique into account).

3 Examined factors

In this section, we provide details for the examined factors that are involved in measuring similarity and evaluating CF results.

Similarity measure: Related work [6, 9, 10, 13] has mainly used Pearson correlation and cosine similarity. In particular, user-based (UB) CF algorithms use the Pearson correlation (Equation 1)[2], which measures the similarity between two users, u and v. Item-based (IB) CF algorithms use a variation of adjusted cosine-similarity (Equation 2)[3], which measures the similarity between two items, i and j, and has been proved more accurate [9, 13], as it normalizes bias from subjective ratings.

$$\text{sim}(u,v) = \frac{\sum\limits_{\forall i \in S} (r_{u,i} - \overline{r}_u)(r_{v,i} - \overline{r}_v)}{\sqrt{\sum\limits_{\forall i \in S} (r_{u,i} - \overline{r}_u)^2} \sqrt{\sum\limits_{\forall i \in S} (r_{v,i} - \overline{r}_v)^2}}, \quad S = I_u \cap I_v. \tag{1}$$

$$\text{sim}(i,j) = \frac{\sum\limits_{\forall u \in T} (r_{u,i} - \overline{r}_u)(r_{u,j} - \overline{r}_u)}{\sqrt{\sum\limits_{\forall u \in U_i} (r_{u,i} - \overline{r}_u)^2} \sqrt{\sum\limits_{\forall u \in U_j} (r_{u,j} - \overline{r}_u)^2}}, \quad T = U_i \cap U_j. \tag{2}$$

Herlocker et al. [6] proposed a variation of the previous measures, which henceforth is denoted as Weighted Similarity (WS). If *sim* is a similarity measure (e.g., Pearson or cosine), then WS is equal to $\frac{\max(c,\gamma)}{\gamma} \cdot sim$, where c is the number of co-rated items and γ is a threshold value used by WS.

Equation 1 takes into account only the set of items, S, that are *co-rated* by both users. This, however, ignores the items rated by only one of the two users. The number of the latter items denotes how much their preferences differ. Especially for the case of sparse data, by ignoring these items we discard significant information. Analogous reasoning applies for Equation 2, which considers (in the numerator) only the set of users, T, that co-rated both the examined pair of items, and for WS, which is based on Equations 1 or 2. To address the problem, in Section 4 we examine alternative definitions for S and T.

[2] $r_{u,i}$ is the rating of u on item i. I_u is the set of items rated by u. $\overline{r}_u$, $\overline{r}_v$ are the mean ratings of u and v over their co-rated items.

[3] U_i is the set of users that rated i. Means $\overline{r}_u$, $\overline{r}_v$ are taken over all ratings of u and v.

Neighborhood size: The number, k, of nearest neighbors, used for the neighborhood formation, directly affects accuracy. Related work [5, 12] utilizes a k in the range of values between 10 and 100. The optimum k depends on the data characteristics. Therefore, CF algorithms should be evaluated against varying k. Moreover, an issue that has not been precisely clarified in related work, is whether we include in the neighborhood a user or item with negative similarity. In order to improve accuracy, we suggest keeping only the positive similarities for the neighborhood formation, even if less than the specified number k of neighbors remain. This approach is also followed in several works [10].
Positive rating threshold: Recommendation for a test user is performed by generating the top-N list of items that appear most frequently in his formed neighborhood (this method is denoted as Most-Frequent item-recommendation). Nevertheless, it is evident that recommendations should be "positive". Recommending an item that will be rated with, e.g., 1 in 1-5 scale should not contribute to the increase of accuracy. We use a rating-threshold, P_τ, to recommended items whose rating is not less than this value. If we do not use a P_τ value, then the results become misleading.
Amount of sparsity: In many real cases, users rate only a very small percentage of items, thus rating data become sparse. Due to lack of sufficient information, sparsity leads to inaccurate recommendations. For this reason, several recent works concentrate only on sparse data [6, 8, 13] (e.g., Movielens). However, there exist dense rating data sets (e.g., Jester [4]). To provide complete conclusions, we have to examine both cases.
Evaluation Metrics: Several metrics have been used for the evaluation of CF algorithms, for instance the Mean Absolute Error (MAE) or the Receiving Operating Characteristic (ROC) curve [6, 7]. MAE represents the absolute differences between the real and the predicted values and is an extensively used metric. From our experimental study (Section 5) we understood that MAE is able to characterize the accuracy of prediction, but is not indicative for the accuracy of recommendation. Since in real-world recommender systems the experience of users mainly depends on the accuracy of recommendation, MAE may not be the preferred measure. For this reason we focus on widely accepted metrics from information retrieval. For a test user that receives a top-N recommendation list, let R denote the number of *relevant recommended items* (the items of the top-N list that are rated higher than P_τ by the test user). We define the following:

- *Precision* is the ratio of R to N.
- *Recall* is the ratio of R to the total number of relevant items for the test user (all items rated higher than P_τ by him).

Notice that with the previous definitions, when an item in the top-N list is not rated at all by the test user, we consider it as *irrelevant* and it counts negatively to precision (as we divide by N). In the following we also use $\text{F}_1 = 2 \cdot \text{recall} \cdot \text{precision}/(\text{recall} + \text{precision})$. F_1 is used because it combines both the previous metrics.
Setting a baseline method: Existing experimental evaluations lack the comparison against a baseline algorithm. A baseline algorithm has to be simple and

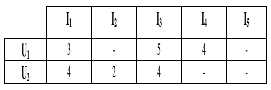
to indicate what can be attained with as little effort as possible. Through a baseline, we can see the actual improvement due to existing CF algorithms.

4 Proposed methodology

Next, we describe in more detail our proposed method. We first examine the factor of the similarity measure. Next, we elaborate on the issue of how to assign ratings to non-rated items, which is required by the proposed similarity measure. Finally, we describe the development of a baseline algorithm.

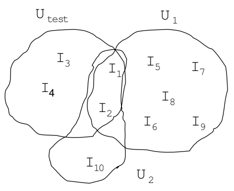
4.1 The UNION similarity measures

According to the definition of sets S and T given in Equations 1 and 2, only the items that are co-rated by both users are considered. For instance, Figure 1a depicts the ratings of two users, U_1 and U_2, over five items (dash denotes an unrated item). When only co-rated items are considered, then the similarity between U_1 and U_2 will be computed based on the ratings for I_1 and I_3.



	I_1	I_2	I_3	I_4	I_5
U_1	3	-	5	4	-
U_2	4	2	4	-	-

(a)



(b)

Fig. 1. Example of: (a) the ratings of two users over five items, (b) a test user compared against two past users.

In case of sparse data, we have a very small amount of provided ratings to compute the similarity measure. By additionally constraining S and T with co-rated items only, we reduce further the effective amount of used information. To avoid this, we consider alternative definitions for S and T, given in Equation 3:

$$S = I_u \cup I_v, \quad T = U_i \cup U_j \tag{3}$$

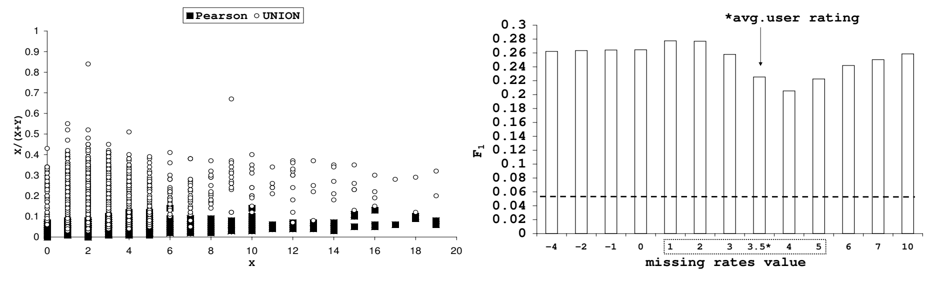
According to Equation 3, S includes items rated by at least one of the users. In the example of Figure 1a, except the ratings for I_1 and I_3, the ratings for I_2 and I_4 will be considered too (the issue of how to treat items rated by only one user, will be discussed in the following). Similar reasoning is followed for the set T, in the case of IB CF. By combining the definitions of S and T given in Equation 3 with the Pearson correlation and adjusted cosine similarity measures, we get two reformed measures: UNION Pearson correlation (for UB) and UNION adjusted cosine (for IB), respectively.[4] Notice that in case of UNION Pearson correlation, user mean ratings correspond to the average user ratings over all rated items. To further understand why should not base similarity only on co-rated items, consider the following example.

[4] Henceforth, when it is clearly understood from the context whether we discuss about UB or IB, we use only the name UNION.

Example Figure 1b depicts the items rated positively (i.e., higher than P_τ) by a test user U_{test} and two users, U_1 and U_2, belonging in the training set. U_{test} and U_1 have co-rated items I_1 and I_2. Assume that U_{test} and U_1 rated I_1 with 5 in the 1–5 scale, whereas I_2 have been rated by both of them with 4. Nevertheless, items $I_3 - I_9$ are rated only by U_{test} or U_1, and not by both. In this case, the Pearson measure of Equation 1, which is based on co-rated items only, results to the maximum possible similarity value (i.e., equal to 1) between U_{test} and U_1. However, this is based only on the 2 co-rated items and ignores the 7 items that are rated only by one of them. On the other hand, assume that U_2 rated I_1 and I_2 with 5 and 4, respectively. As previously, the Pearson measure of Equation 1 results to the maximum possible similarity between U_{test} and U_2, whereas U_{test} and U_2 differ in 3 items rated by only one of them. This example reflects the impotence of Equation 1 to capture the actual notion of similarity: despite the fact that U_{test} and U_1 differ at 7 items (which are rated by only one of them) and U_{test} and U_1 differ at 3, it assigns the same similarity value in both cases. $\square$

In the previous example, if we designate U_1 as neighbor of U_{test}, we ignore two issues: (i) Items I_3 and I_4, will not be recommended to U_{test} by U_1, as U_1 has not rated them; this fact harms recall. (ii) Items $I_5 - I_9$ will be recommended by U_1, but as they have not been rated by U_{test}, this will harm precision. It follows that a desirable property from a similarity measure is to maximize the number, x, of items that are co-rated by the test user and each of his neighbors, relatively to the number, y, of items that are rated only by one of them (in the example of Figure 1b, for U_{test} and U_1, $x = 2$ and $y = 7$). In the best case, the ratio $x/(x + y)$ has value equal to 1 and in the worst 0.

To evaluate the previously described argument, we compared Pearson correlation against UNION UB by performing the following measurement. We used the MovieLens 100K data set and for each test user we computed its k nearest neighbors (k was set to 10) from the training set. Next, we measured x and y between each test user and each of his k neighbors. Figure 2a illustrates for each x, the resulting ratio $x/(x+y)$. Figure 2a clearly presents that Pearson measure results to significantly lower ratios than UNION UB. This explains why UNION UB compares favorably to Pearson correlation in terms of precision and recall,



(a) (b)

Fig. 2. (a) Measuring the ratio $x/(x + y)$. (b) Impact of assigned value for unrated items.

as will be presented experimentally in Section 5. (Due to lack of space we do not present the analogous comparison between adjusted cosine and UNION IB.)

4.2 Assigning a value to unrated items

To calculate the UNION Pearson correlation between two users U_1 and U_2, we have to assign a rating by U_1 to an item that is rated only by U_2 (e.g., I_2 in the example of Figure 1a) and vice-versa. The same requirement holds for the UNION adjusted cosine measure in the IB case. Notice that this problem cannot be effectively solved with existing techniques for filling missing values, because the sparsity of user-item matrices severely hinders this task (more than 99.9% missing values). For this reason we assign the same rating value to all the required cases. There could be several options for this value. For instance, in a 1–5 scale, we can assign the 0 value, to reflect that user is not interested at all to rate the item. Assuming that user u did not rate item i, another option is to assign the average value of the provided ratings on other items by u, or the average of the provided ratings on i by all users.

To examine the impact of the selected value, we measured F_1 versus the assigned value, which is depicted in Figure 2b for the MovieLens 100K data set (it uses 1–5 scale). The dashed line in Figure 2b corresponds to F_1 of the Pearson correlation (it is independent from the assigned value). As shown, values between the positive threshold (in this case P_τ was set to 3) and the maximum rating of the scale, result to reduced F_1 (notice that this range also includes the user average rating value). The reason is that these values impinge the ability to distinguish the assigned values from the actual positive ratings. However, when we assign values smaller than P_τ or *outside* the rating scale, F_1 is not affected. The reason is that with such assigned values we do not miss the ability to distinguish the assigned values from the actual positive ratings (as the latter are always within the provided scale). Thus, we conclude that UNION is not significantly affected by the selection for the assigned ratings, as all values outside the rating scale or below P_τ result to about the same F_1. Even more, the values between P_τ and the upper limit of the scale result to significantly higher F_1 than Pearson measure. Henceforth, we assume that the assigned value is equal to 0.

4.3 Baseline algorithm

Considering the factors described in Section 3 regarding the evaluation procedure, we detail a baseline algorithm. We propose the one that recommends the N items that are most frequently rated positively in the entire training set. This algorithm is denoted as BL. BL is very simple and, as will be shown in our experimental results, it is quite effective. For instance, our experiments with Movielens-100K data set have shown that, with BL, when we simply propose the $N = 20$ most positively rated movies (20 most popular movies), precision reaches 40%. Therefore, the most frequently rated items are very probable to be selected by the majority of the users. For the aforementioned reasons, BL is a tool to clearly evaluate the actual improvement of existing CF algorithms. We will see in our experiments that asymptotically, as k (neighborhood size) increases, the performance of Pearson correlation and adjusted cosine tend to become equal with that of BL. In the extreme case where k is equal to the number of all users in the training set, the result of the Most-Frequent item-recommendation procedure, for the generation of the top-N list, becomes equivalent to BL.

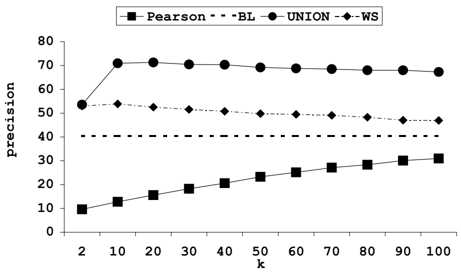
5 Performance study

In the sequel, we study the performance of the proposed approach against Pearson correlation and adjusted cosine. Both the UB and IB cases are examined. Regarding the parameters, the following default values are assumed: for the neighborhood size the default k value is 10, for the recommendation list the default N value is 20, and for the size of training set the default value is 75%. Regarding WS, the γ value was set to 5. Evaluation is performed with the precision and recall metrics (given as percentages). We also use F_1 metric and MAE.

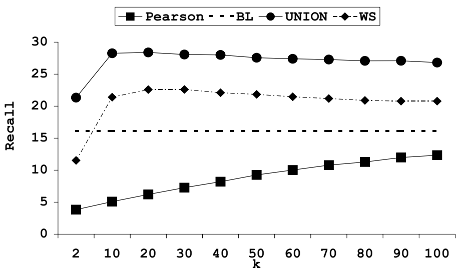
We perform experiments with three real data sets that have been used as benchmarks in prior work. In particular, we examined two MovieLens data sets: (i) the first one with 100,000 ratings assigned by 943 users on 1,682 movies, and (ii) the second one with about 1 million ratings for 3,592 movies by 6,040 users. The range of ratings is between 1(bad)-5(excellent) of the numerical scale. Moreover, we ran our experiments on the Jester data set, which contains 4.1 million ratings of 100 jokes from 73,496 users. Due to lack of space, we present results only for the first MovieLens and the Jester data sets, because they correspond to a sparse and a dense data set, respectively. The performance of the former has been verified with the results of the 1M real data set. Finally, in all data sets, we normalized the rating scale in the range 1–5, whereas P_τ is set to 3.

5.1 Results for user-based CF

First, we examine the UB CF case and compare the existing Pearson similarity and WS measures against UNION. We also include the baseline (BL) algorithm. The results for precision and recall vs. k are displayed in Figure 3a and b, respectively.



(a)

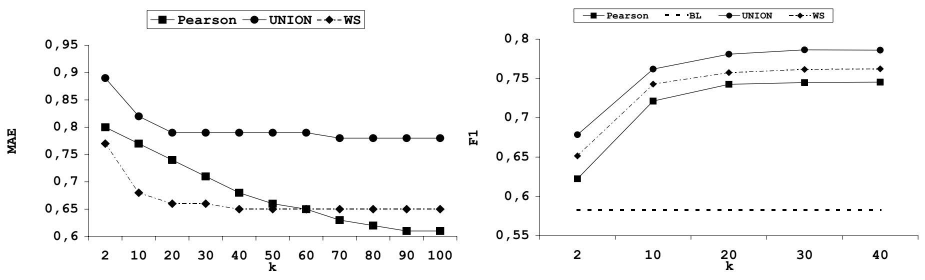


(b)

Fig. 3. Performance of user-based CF vs. k: (a) precision, (b) recall.

As shown, the existing Pearson measure, which is based on co-rated items, performs worst than BL. This result is surprising, as BL is very simple. WS improves Pearson, because the disadvantage of Pearson, due to co-rated items, is downsized by the weighting with the number of common items. UNION clearly outperforms all other measures for the reason that have been described in Section 4. Outside the examined k range (not displayed), Pearson stabilizes and never exceeds BL. As we already described, with increasing k, Pearson measure practically becomes equivalent to BL.

We now examine the MAE metric. Results are illustrated in Figure 4a (BL is only for recommendation, not prediction, thus omitted). As expected, Pearson yields the lowest MAE values, whereas WS is second best. This fact supports our explanation that MAE is indicative only for the evaluation of prediction and not of recommendation, as these measures did not attain the best performance in terms of precision and recall.



(a) (b)

Fig. 4. Performance of user-based CF vs. k: (a) MAE, (b) F_1 for dense data.

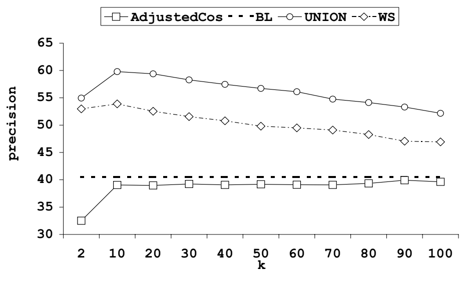
To consider the impact of density, we also examine the Jester data set. The results for the F_1 metric are depicted in Figure 4b. In this case, the relative differences are smaller than for the case of sparse data. The reason is that dense data have a sufficient amount of information, thus there is less need to exploit information in the way UNION does. Nevertheless, UNION still presents the best performance.

5.2 Results for item-based CF

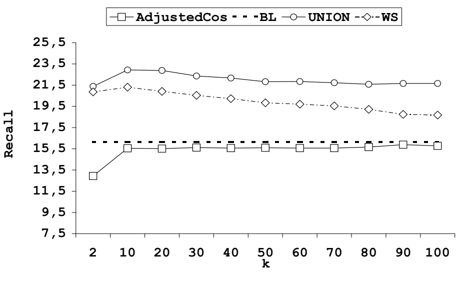
We perform similar measurements for the case of IB CF. Thus, we first examine the precision and recall for the adjusted cosine (considers co-rated items) against UNION. The results are depicted in Figure 5 and are analogous to those of the UB case. UNION clearly outperforms adjusted cosine and WS. Again, it is surprising to find that the adjusted cosine looses out by BL.

Next, we compare adjusted cosine, UNION, and WS against MAE. The results are illustrated in Figure 6a. Differently to UB, all measures have similar MAE, and for larger k values they converge to the optimum MAE. Adjusted cosine does not present better MAE, because in its denominator it considers all items and not just the co-rated ones (see Equation 2). This improves its performance for the task of recommendation and worsens the performance of prediction.[5]

[5] We have examined a variation of adjusted cosine that uses only the co-rated items in the denominator. As expected, it resulted to worse precision and recall, but to better MAE.

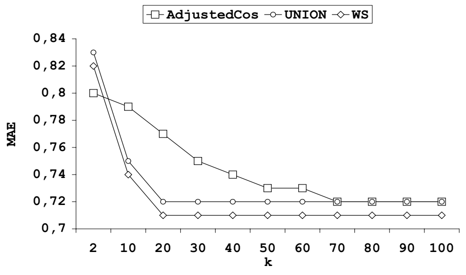


(a)

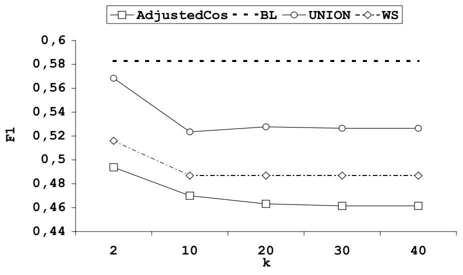


(b)

Fig. 5. Performance of item-based CF vs. k: (a) precision, (b) recall.



(a)



(b)

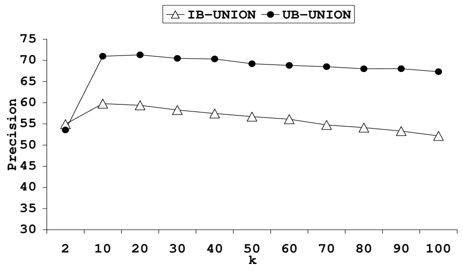
Fig. 6. Performance of item-based CF vs. k: (a) MAE, (b) F_1 for dense data.

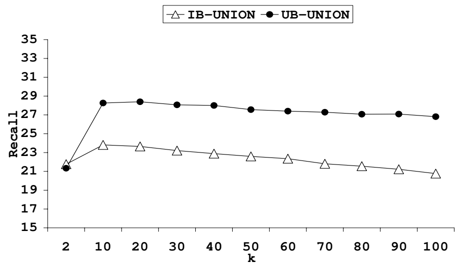
Regarding the examination of the dense data set (Jester), the results for the F_1 metric are illustrated in Figure 6b. Since IB CF has been designed to suit the needs of sparse data, we find out that for dense data all item-based algorithms are outperformed by BL. This is the case even for UNION, although it performs better than adjusted cosine. This result clarifies the need to examine CF algorithms for all the involved factors, in this case the amount of sparsity, in order to draw more complete conclusions.

5.3 Comparative results

In this section, we compare UNION for the UB and IB cases, as the corresponding UNION measures were shown to have the best performance in each case separately. The results for precision are depicted in Figure 7a, whereas those for recall are depicted in Figure 7b.

These results demonstrate that UB CF compares favorably against IB CF when UNION is used. The difference in precision is larger than 10%, whereas with respect to recall, it exceeds 5% (we refer to the optimum values resulting from the tuning of k). This conclusion contrasts the existing one, that IB is more preferable than UB, for the case of sparse data. The reason is that UB CF is more focused towards the preferences of the target user. In contrast, with IB CF, the recommended items may have been found similar by transactions of users with much different preferences than the ones of the target user. Thus, they may not directly reflect the preferences of the latter. However, this property could not be revealed with the existing similarity measures and evaluation procedures.





(a) (b)

Fig. 7. Comparison between UB and IB: (a) precision, (b) recall.

The previous conclusion is in accordance with the one resulting from the comparison for the dense data set (Jester). Due to lack of space we do not present a graph for this case. However, from Figure 4b and Figure 6b it is easy to see that UB performs much better than IB when UNION is used, as the former is better than BL and the latter is worse.

6 Conclusions

In this paper, we performed a thorough study of neighborhood-based CF, which brought out several factors that have not been examined carefully in the past. We proposed a novel approach (UNION) for measuring similarity in nearest-neighbor CF applications. UNION successfully exploits more information in case of sparse data and considers how much the ratings of two users differ in order to provide accurate recommendations. We carried out extensive experimentation which reforms several existing beliefs and provides new insights. In particular, we highlight the following conclusions from our examination:

- In contrast to what is reported in majority of related work, MAE is not indicative for the accuracy of the recommendation process. It is, though, useful to characterize the quality of the similarity measure (as reflected in the process of prediction).
- Constraining similarity measures with co-rated items, weaknesses the measure. Though it is somewhat useful to consider the number of co-rated items (as WS does), the strict constraining inside the formulae for similarity measures is not suitable.
- The proposed approach, which does not use co-rated items only, substantially improves the performance of CF in terms of precision and recall, especially for sparse data. This conclusion was also explained through the measurement of ratio $x/(x + y)$ in Section 4. This measurement demonstrated the need to minimize the number, y, of items that are rated only by one of the users, a fact that is attained by UNION and not by existing similarity measures.
- Our results showed that, user-based compares favorably to item-based CF, and that item-based CF is not appropriate for dense data.

– Finally, the proposed baseline (BL) algorithm can better characterize the performance of existing CF algorithms. Its comparison against widely-accepted algorithms has produced surprising results.

We have to notice that item-based algorithms employ off-line computation, which is an advantage over user-based algorithms in terms of execution time. For this reason, in our future work we will consider the issue of scalability and compare the two approaches for this factor as well. Moreover, we will examine new algorithms for the generation of the top-N recommendation list.

References

1. J. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proc. Conf. on Uncertainty in Artificial Intelligence*, pages 43–52, 1998.
2. M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Trans. on Information Systems*, 22(1):143–177, 2004.
3. D. Goldberg, D. Nichols, M. Brian, and D. Terry. Using collaborative filtering to weave an information tapestry. *ACM Communications*, 35(12):61–70, 1992.
4. K. Goldberg, T. Roeder, T. Gupta, and C. Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, 2001.
5. J. Herlocker, J. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proc. ACM SIGIR Conf.*, pages 230–237, 1999.
6. J. Herlocker, J. Konstan, and J. Riedl. An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. *Information Retrieval*, 5(4):287–310, 2002.
7. J. Herlocker, J. Konstan, L. Terveen, and J. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. on Information Systems*, 22(1):5–53, 2004.
8. G. Karypis. Evaluation of item-based top-n recommendation algorithms. In *Proc. ACM CIKM Conf.*, pages 247–254, 2001.
9. R. McLauglin and J. Herlocher. A collaborative filtering algorithm and evaluation metric that accurately model the user experience. In *Proc. ACM SIGIR Conf.*, pages 329–336, 2004.
10. B. Mobasher, H. Dai, T. Luo, and M. Nakagawa. Improving the effectiveness of collaborative filtering on anonymous web usage data. In *Proc. Workshop Intelligent Techniques for Web Personalization*, pages 53–60, 2001.
11. P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering on netnews. In *Proc. Conf. Computer Supported Collaborative Work*, pages 175–186, 1994.
12. B. Sarwar, G. Karypis, J. Konstan, and R. J. Analysis of recommendation algorithms for e-commerce. In *Proc. ACM Electronic Commerce Conf.*, pages 158–167, 2000.
13. B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proc. WWW Conf.*, pages 285–295, 2001.
14. G. Xue, C. Lin, and Q. e. a. Yang. Scalable collaborative filtering using cluster-based smoothing. In *Proc. ACM SIGIR Conf.*, pages 114 – 121, 2005.

Discovering User Profiles from Papers by Using Word Sense Disambiguation

G. Semeraro, P. Basile, M. Degemmis, and P. Lops

Dipartimento di Informatica
Università di Bari
Via E. Orabona, 4 - 70125 Bari - Italia
{semeraro,basilepp,degemmis,lops}@di.uniba.it

Abstract. Understanding user interests from text documents can support effectively the development of personalized information recommendation services. We present an approach based on Word Sense Disambiguation (WSD) for the extraction of user profiles from documents. This approach bases on the JIGSAW WSD algorithm, that combines three different kinds of algorithms to disambiguate nouns, verbs, adjectives and adverbs. All of them heavily rely on the linguistic knowledge in the WordNet lexical database. The proposed strategy is adopted for the semantic indexing of documents used to train a naive Bayes learner that infers user profiles as binary text classifiers (user-likes and user-dislikes). User profiles are exploited by the "conference participant advisor" service, which can be profitably integrated in a scientific congress workbench to suggest papers to be read and talks to be attended by a conference participant. An empirical evaluation has been carried out by training the system on a corpus of papers accepted to the International Semantic Web Conferences, held on 2002 and 2003, and rated by real users according to their preferences. Preliminary results show the usefulness of the "conference participant advisor" service for the Virtual Information and Knowledge Environment Framework (VIKEF)[1].
Keywords: User Profiling, Text Categorization, WordNet, Text Mining for Information Retrieval.

1 Introduction

The amount of information available on the Web and in Digital Libraries is increasing over time. In this context, the role of user modeling and personalized information access is increasing: users need a personalized support in sifting through large amounts of retrieved information according to their interests. Information filtering and retrieval systems relying on this idea adapt their behavior to individual users by learning their preferences during the interaction in order to construct a *user profile* that can be later exploited in the search process. Traditional keyword-based approaches are unable to capture the *semantics* of the user

[1] This research was partially funded by the European Commission under the 6th Framework Programme IST IP VIKEF No. 507173, Priority 2.3.1.7.

interests. They are primarily driven by a string-matching operation: If a string, or some morphological variant, is found in both the profile and the document, a match is made and the document is considered as relevant. String matching suffers from problems of *polysemy*, the presence of multiple meanings for one word, and *synonymy*, multiple words having the same meaning. The result is that, due to synonymy, relevant information can be missed if the profile does not contain the exact keywords in the documents while, due to polysemy, wrong documents could be deemed as relevant. These problems call for alternative methods able to learn more accurate profiles that capture concepts expressing users' interests from relevant documents. These *semantic* profiles will contain references to concepts defined in lexicons or, in a further step, ontologies. This paper describes an approach in which semantic user profiles are obtained by machine learning techniques integrated with a word sense disambiguation (WSD) strategy based on the WordNet lexical database [9, 3]. The paper is organized as follows: After discussing some works related to our research, we describe in Section 3 the WSD strategy proposed to represent documents by using WordNet. Section 4 presents the naïve bayes text categorization method adopted to build *WordNet-based* profiles. The method is implemented by our ITem Recommender (ITR) system. A possible application scenario for semantic profiles is given in Section 5, which presents the "Conference Participant Advisor" service, that supports participants to a conference in planning their attendance. In this context, an experimental evaluation has been carried out to evaluate our approach by comparing the performance of keyword-based profiles with that of sense-based profiles. Conclusions and future work are discussed in the last Section.

2 Related Work

Syskill & Webert [11] learns user profiles as Bayesian classifiers able to recommend web pages, but represents documents by using keywords. *LIBRA* [10] adopts a Bayesian classifier to produce content-based book recommendations by exploiting product descriptions obtained from the Web pages of the Amazon on-line digital store. Documents are represented by using keywords and are subdivided into slots, each one corresponding to a specific section of the document. Like *Syskill & Webert*, the main limitation of this work is that keywords are used to represent documents. *SiteIF* [6] exploits a *sense-based* representation to build a user profile as a semantic network whose nodes represent senses of the words in documents requested by the user. The semantic network is built by assigning to each node a score that is inversely proportional to its frequency over all the corpus, so that the score is higher for less frequent senses, avoiding that very common meanings become too prevailing in the user model. In our approach, we learn a probability distribution of the senses found in the corpus of the documents rated by the user. *OntoSeek* [4] is a system designed for content-based information retrieval from online yellow pages and product catalogs which explored the role of linguistic ontologies in knowledge-retrieval systems. The approach has shown that structured content representations coupled with linguistic ontologies can increase both recall and precision of content-based retrieval systems. According to these works, we conceived our ITR system as a text classifier able 1) to deal with a sense-based document representation obtained by exploiting a linguistic ontology and 2) to learn a bayesian profile from documents subdivided into slots. The strategy we propose to shift from a keyword-based to a sense-based document representation is *to integrate lexical knowledge in the indexing step of training documents*. Several methods have been proposed to accomplish this task. Scott and Matwin [13] proposed to include WordNet information at the feature level by expanding each word in the training set with *all* the synonyms for it in WordNet in order to avoid a WSD process. This approach has shown a decrease of effectiveness in the obtained classifier, mostly due to the word ambiguity problem, therefore it suggests that some kind of disambiguation is required. In [2] the authors experiment with various settings for mapping words to senses: No WSD, most frequent sense as provided by WordNet, WSD based on context. They found positive results on the Reuters 25178[2], the OHSUMED[3] and the FAODOC[4] corpus. None of the previous approaches for embedding WSD in classification has taken into account the fact that WordNet is a hierarchical thesaurus. In our work, we adopt a similarity measure that takes into account the hierarchical structure of WordNet.

3 Using WordNet to Represent Documents

We consider the problem of learning user profiles as a binary text categorization task: Each document has to be classified as interesting or not with respect to the user preferences. Therefore, the set of categories is $C = \{c_+, c_-\}$, where c_+ is the positive class (user-likes) and c_- the negative one (user-dislikes). There are several ways in which content can be represented in order to be used as a basis for the learning component and there exists a variety of machine learning methods that could be used for inferring user profiles. We propose a strategy to learn sense-based profiles that consists of two steps. In this section, we describe the first one, that is a WSD technique that uses the word senses in WordNet to represent documents. In the second step, described in section 4, a naïve Bayes approach learns *sense-based* user profiles as binary text classifiers (user-likes and user-dislikes) from disambiguated documents.

3.1 The JIGSAW algorithm for Word Sense Disambiguation

We extend the classical BOW model [14] to a model in which the senses corresponding to the words in the documents are considered as features. This sense-based document representation is exploited by the learning algorithm to build semantic user profiles. Here "word sense" is used as a synonym of "word meaning". The filtering phase could take advantage of the word senses to recommend

[2] http://about.reuters.com/researchandstandards/corpus/
[3] http://www.ltg.ed.ac.uk/disp/resources/
[4] http://www4.fao.org/faobib/index.html

new items (documents) with high semantic relevance with respect to the user profile. There are two crucial issues to address: First, a repository for word senses has to be identified. Second, any implementation of sense-based text classification must solve the problem that, while words occur in a document, meanings do not, since they are often hidden in the context. Therefore, a procedure is needed for assigning senses to words. This task is known as Word Sense Disambiguation and consists in determining which of the senses of an ambiguous word is invoked in a particular use of the word [7]. The goal of a WSD algorithm is to associate the appropriate meaning or sense s to a word w_i in document d, by exploiting its *(window of) context* C, a set of words that precede and follow w_i. The sense s is selected from a predefined set of possibilities, usually known as *sense inventory*. In the proposed algorithm, the sense inventory is obtained from WordNet (version 1.7.1), a large lexical database for English[5]. Each sense of the word w_i is thus represented by a unique synset s_{ik} (k-th sense of w_i). WordNet was designed to establish connections between four types of Parts of Speech (POS): Noun, verb, adjective, and adverb. The basic building block for WordNet is the SYNSET (SYNonym SET), which represents a specific meaning of a word. The specific meaning of one word under one type of POS is called a sense. Synsets are equivalent to senses, which are structures containing sets of words with synonymous meanings (words that are interchangeable in some contexts). Each synset has a gloss, a short textual description that defines the concept represented by the synset. For example, the words *night*, *nighttime* and *dark* constitute a single synset that has the following gloss: "the time after sunset and before sunrise while it is dark outside". Synsets are connected through a series of relations: Antonymy (opposites), hyponymy/hypernymy (IS-A), meronymy (PART-OF), etc. JIGSAW is a Word Sense Disambiguation algorithm based on the idea of combining three different strategies to disambiguate nouns, verbs, adjectives and adverbs. In particular, an adaptation of Lesk's dictionary-based WSD algorithm has been used to disambiguate adjectives and adverbs [1], while an adaptation of the Resnik algorithm has been used to disambiguate nouns [12]. The algorithm we developed for disambiguating verbs exploits the nouns in the context of the verb to be disambiguated and the nouns both in the glosses and in the phrases that WordNet utilizes to describe the usage of the verb. The algorithm disambiguates only words that belong to at least one synset. The motivation behind our approach is that the performances of the WSD algorithms change in accordance to the POS of the word to be disambiguated. JIGSAW algorithm takes as input a document $d = \{w_1, w_2, \ldots, w_h\}$ and will output a list of WordNet synsets $X = \{s_1, s_2, \ldots, s_k\}$ ($k \leq h$) in which each element s_i is obtained by disambiguating, when possible, the *target word* w_i based on the information about w_i and a few immediately surrounding words that can be obtained from WordNet. We define the *context* C of the target word to be a window of n words to the left and another n words to the right, for a total of $2n$ surrounding words. If w_i is near the beginning or the end of d, we add additional words from the other direction. The algorithm is based on three different procedures for nouns,

[5] `http://wordnet.princeton.edu`

verbs, adverbs and adjectives: $JIGSAW_{nouns}$, $JIGSAW_{verbs}$, $JIGSAW_{others}$, respectively. The POS of each word to be disambiguated is computed by the HMM-based tagger ACOPOST t3[6]. JIGSAW proceeds in several iterations by using the disambiguation results of the previous iteration to reduce the complexity of the next iteration. First, JIGSAW performs noun disambiguation by executing the $JIGSAW_{nouns}$ procedure, which was developed by modifying the algorithm described in [12]. Then, verbs are disambiguated by $JIGSAW_{verbs}$ by exploiting the words already disambiguated by $JIGSAW_{nouns}$. Finally, the $JIGSAW_{others}$ procedure is executed. Let's see in more detail each one of the above mentioned procedures. $JIGSAW_{nouns}$ includes in the context C for the target word w_i all the *nouns* in the window of $2n$ words surrounding w_i. The idea behind our approach is that when two polysemous words are similar, the most specific subsumer gives information about the most appropriate synset for each one of the two words. The algorithm defines a function φ that assigns to w_i the most appropriate synset s_{ih} among the sense inventory X_i for w_i. This function computes the similarity between each s_{ik} in the sense inventory and the context for w_i. The method differs from the original algorithm by Resnik in the use of the similarity measure. We adopted the Leacock-Chodorow measure [5], which is based on the length of the path between concepts in an IS-A hierarchy. The idea behind this measure is that similarity between synsets a and b is inversely proportional to the distance between them in the WordNet hierarchy. The distance is computed by counting the number of nodes in the shortest path (the path having the minimum number of nodes) joining a to b, by passing through their most specific subsumer. For example, Figure 1 shows the length of the path between *cat* (feline mammal) and *mouse* (rodent) by passing through *placental mammal* is 5. The similarity function is: `SinSim(a,b)` $= -\log(N_p/2D)$, where N_p is the number of nodes in the shortest path p from a to b, and D is maximum depth of the taxonomy ($D = 16$, in WordNet 1.7.1). The way in which the procedure works can be described by processing the sentence *"The white cat is hunting the mouse"* as an example. Let w_i="cat" be the target word. The procedure starts by defining the context C of w_i as the set of words having the same POS as w and found in the same sentence as w_i. In this case, the only other *noun* in the sentence is "mouse", then $C = \{mouse\}$. Next, the algorithm identifies both the sense inventory for w_i, that is X_{cat} = {`01789046: feline mammal, 00683044: computerized axial tomography,`...}, and the sense inventory X_j for each word w_j in C. Thus, X_{mouse}= {`01993048: small rodents, 03304722: a hand-operated electronic device that controls the coordinates of a cursor, ...` }. The sense inventory T for the whole context C is given by the union of all X_j (in this case, $T = X_{cat}$, since the only word in C is "mouse"). After this step, $JIGSAW_{nouns}$ measures the similarity of each candidate sense $s_{ik} \in X_i$ with that of each sense $s_h \in T$. The sense assigned to w_i is the one with the highest similarity score. In the example, `SinSim(01789046: feline mammal, 01993048: small rodents) = 0.806` is the highest similarity score, thus *cat* is interpreted as "feline mammal".

[6] http://acopost.sourceforge.net/

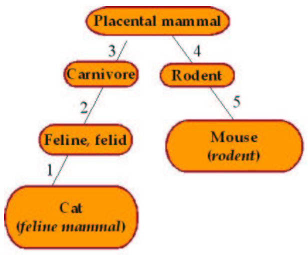


Fig. 1. The "mouse-cat" path in the WordNet hierarchy

Before describing the $JIGSAW_{verbs}$ procedure, we need to define the *description* of a synset. It is the string obtained by concatenating the gloss and the sentences that WordNet uses to explain the usage of a word. For example, the gloss for the synset corresponding to the sense n.2 of the verb *look* ($\{look, appear, seem\}$) is "give a certain impression or have a certain outward aspect". The examples of usage of the verb are: "She seems to be sleeping"; "This appears to be a very difficult problem";"This project looks fishy";"They appeared like people who had not eaten or slept for a long time". The description of the synset is "give a certain impression or have a certain outward aspect She seems to be sleeping This appears to be a very difficult problem This project looks fishy They appeared like people who had not eaten or slept for a long time". First, the $JIGSAW_{verbs}$ proceeds by including in the context C for the verb w_i all the nouns in the window of $2n$ words surrounding w_i. Let w_j a noun belonging to the context C. For each synset s_{jk} for w_j, the algorithm computes $nouns(j,k)$, that is the set of nouns in the description for s_{jk}. The procedure extends C by adding, for each w_j, $nouns(j,k)$. Then, for each synset s_{ik} for the verb w_i, the algorithm computes $nouns(i,k)$. In the above example, nouns(look, 2)={impression, aspect, problem, project, people, time}. Then, each w_j in C is compared with each noun w_l in $nouns(i,k)$ for each possible sense k for w_i, by computing the Leacock-Chodorow measure between them. In this way, we compare each possible sense k for w_i with its context. For each w_j in C, the following value is computed:

$$max_{jk} = max_{w_l \in nouns(i,k)} \{sim(w_j, w_l)\} \tag{1}$$

where $k = 1, \ldots, m$, m being the number of possible senses for w_j. In other words, max_{jk} is the highest similarity value for w_j with respect to the nouns related to the k-th sense for w_i. Finally, for each synset s_{ik} for w_i we define:

$$\varphi(i,k) = R(k) \cdot \frac{\sum_{w_j \in C} G(pos_j) \cdot max_{jk}}{\sum_h G(pos_h)} \tag{2}$$

where $R(k)$ is the ranking of s_k in WordNet (synsets are ranked according to their frequency of usage) and $G(pos_j)$ is a gaussian factor related to the position of w_j

with respect to w_i in the original text, which gives more weight to words near the target word. The synset assigned to w_i is the one with the highest φ value. The $JIGSAW_{other}$ procedure is based on the WSD algorithm proposed in [1]. The idea is to compare the glosses of each sense of the target word with the glosses of all the words in the context. For each s_{ik} in the sense inventory for the target word w_i, $JIGSAW_{other}$ computes the string $targetGloss_{ik}$ that contains the words in the gloss of s_{ik}. Then, the procedure computes the string $contextGloss_i$, which contains the words in the glosses of all the synsets s_{jk} corresponding to each w_j in the context C for w_i. Finally, the procedure computes an *overlap* between $contextGloss_i$ and $targetGloss_{ik}$ by counting the words that occur in both strings and assign to w_i the synset with the highest overlap score.

3.2 Keyword-based and Synset-based Document Representation

The WSD procedure is used to obtain a synset-based vector space representation that we called Bag-Of-Synsets (BOS). In this model, a synset vector corresponds to a document, instead of a word vector. Another key feature of the approach is that each document is represented by a set of *slots*, where each slot is a textual field corresponding to a specific feature of the document, in an attempt to take into account also the structure of documents. In our application scenario, in which documents are scientific papers, we selected three slots: *title*, *authors*, *abstract*. The text in each slot is represented according to the BOS model by counting separately the occurrences of a synset in the slots in which it appears. More formally, assume that we have a collection of N documents. Let m be the index of the slot, for $n = 1, 2, ..., N$, the n-th document is reduced to three bags of synsets, one for each slot, $d_n^m = \langle t_{n1}^m, t_{n2}^m, \ldots, t_{nD_{nm}}^m \rangle$, where t_{nk}^m is the k-th synset in slot s_m of document d_n and D_{nm} is the total number of synsets appearing in the m-th slot of document d_n. For all n, k and m, $t_{nk}^m \in V_m$, which is the vocabulary for the slot s_m (the set of all different synsets found in slot s_m). Document d_n is finally represented in the vector space by 3 synset-frequency vectors $f_n^m = \langle w_{n1}^m, w_{n2}^m, \ldots, w_{nD_{nm}}^m \rangle$, where w_{nk}^m is the weight of the synset t_{nk}^m in the slot s_m of document d_n. It is the number of times synset t_{nk}^m appears in slot s_m. Our hypothesis is that the BOS model helps to obtain profiles able to recommend documents semantically closer to the user interests. The difference with respect to keyword-based profiles is that synset unique identifiers are used instead of words. In the next session, we describe the learning algorithm adopted to build semantic user profiles, starting from the BOS document representation.

4 A Naïve Bayes Method for User Profiling

Naïve Bayes is a probabilistic approach to inductive learning. The learned probabilistic model estimates the *a posteriori* probability, $P(c_j|d_i)$, of document d_i belonging to class c_j. To classify a document d_i, the class with the highest probability is selected. As a working model for the naïve Bayes classifier, we use the multinomial event model [8]:

$$P(c_j|d_i) = P(c_j) \prod_{w \in V_{d_i}} P(t_k|c_j)^{N(d_i,t_k)} \tag{3}$$

where $N(d_i, t_k)$ is defined as the number of times word or token t_k appeared in document d_i. Notice that rather than getting the product of all distinct words in the corpus, V, we only use the subset of the vocabulary, V_{d_i}, containing the words that appear in the document d_i. Since each instance is encoded as a vector of BOS, one for each slot, Equation (3) becomes:

$$P(c_j|d_i) = \frac{P(c_j)}{P(d_i)} \prod_{m=1}^{|S|} \prod_{k=1}^{|b_{im}|} P(t_k|c_j, s_m)^{n_{kim}} \tag{4}$$

where $S = \{s_1, s_2, \ldots, s_{|S|}\}$ is the set of slots, b_{im} is the BOS in the slot s_m of the instance d_i, n_{kim} is the number of occurrences of the synset t_k in b_{im}. Our ITR profiling system implements this approach to classify documents as interesting or uninteresting for a particular user. To calculate (4), we only need to estimate $P(c_j)$ and $P(t_k|c_j, s_m)$ in the training phase of the system. The documents used to train the system belong to a collection of scientific papers accepted to the 2002-2003 editions of the International Semantic Web Conference (ISWC). Ratings on these documents, obtained from real users, were recorded on a discrete scale from 1 to 5 (see Section 5 for a detailed description of the dataset). An instance labeled with a rating r, $r = 1$ or $r = 2$ belongs to class c_- (user-dislikes); if $r = 4$ or $r = 5$ then the instance belongs to class c_+ (user-likes); rating $r = 3$ is neutral. Each rating was normalized to obtain values ranging between 0 and 1:

$$w_+^i = \frac{r-1}{MAX-1}; \qquad w_-^i = 1 - w_+^i \tag{5}$$

where MAX is the maximum rating that can be assigned to an instance. The weights in (5) are used for weighting the occurrences of a synset in a document and to estimate the probability terms from the training set TR. The prior probabilities of the classes are computed according to the following equation:

$$\hat{P}(c_j) = \frac{\sum_{i=1}^{|TR|} w_j^i + 1}{|TR| + 2} \tag{6}$$

Witten-Bell smoothing [15] has been adopted to compute $P(t_k|c_j, s_m)$, by taking into account that documents are structured into slots and that word occurrences are weighted using weights in equation (5):

$$P(t_k|c_j, s_m) = \begin{cases} \frac{N(t_k,c_j,s_m)}{V_{c_j} + \sum_i N(t_i,c_j,s_m)} & \text{if } N(t_k, c_j, s_m) \neq 0 \\ \frac{V_{c_j}}{V_{c_j} + \sum_i N(t_i,c_j,s_m)} \frac{1}{V - V_{c_j}} & \text{if } N(t_k, c_j, s_m) = 0 \end{cases} \tag{7}$$

where $N(t_k, c_j, s_m)$ is the count of the weighted occurrences of the synset t_k in the training data for class c_j in the slot s_m, V_{c_j} is the total number of unique synset in class c_j, and V is the total number of unique words across all classes. $N(t_k, c_j, s_m)$ is computed as follows:

$$N(t_k, c_j, s_m) = \sum_{i=1}^{|TR|} w_j^i n_{kim} \tag{8}$$

In (8), n_{kim} is the number of occurrences of the term t_k in the slot s_m of the i^{th} instance. The sum of all $N(t_k, c_j, s_m)$ in equation (7) denotes the total weighted length of the slot s_m in the class c_j. In other words, $\hat{P}(t_k|c_j, s_m)$ is estimated as a ratio between the weighted occurrences of the synset t_k in slot s_m of class c_j and the total weighted length of the slot. The final outcome of the learning process is a probabilistic model used to classify a new document in c_+ or c_-. The model is implemented in our ITR system to build a personal profile that includes those synsets that turn out to be most indicative of the user's preferences, according to the value of the conditional probabilities in (7).

5 Experiments in a Scientific Congress Scenario

VIKEF (Virtual Information and Knowledge Environment Framework)[7] is an application-oriented Integrated Project dedicated to advanced semantic-enabled support for ICK (Information, Content, and Knowledge) production, acquisition, processing, annotation, sharing and use by empowering information and knowledge environments for scientific and business communities. We present a service realized in the context of the applications to support scientific congress organization. The "Conference Participant Advisor" service is based on ITR and provides useful personalized support for conference participation planning. The prototype has been realized in context of the "International Semantic Web Conference 2004", by adding to the conference homepage (a copy of the official web site) a registration form to access recommendation services. The participant can register by providing a valid email address and browse the whole document repository of papers presented during 2002 and 2003 ISWC events, in order to provide ratings. Each paper can be rated and, given a sufficient number of ratings, the system builds the participant profile (at present the threshold is 20). ISWC 2004 papers are classified using the learned profile to obtain a personalized list of recommended papers and talks which is sent by email to the participant. An experimental evaluation of semantic profiles was carried out on ISWC rated papers. The goal of the evaluation was to compare the performance of keyword-based profiles with that of synset-based profiles. In this section, we present the main results of our preliminary experiments. More intensive experimental sessions are planned in the context of the VIKEF project. Experiments were carried out on a collection of 100 papers (42 papers accepted to ISWC 2002, 58 papers accepted to ISWC 2003) rated by 11 real users, that we called *ISWC dataset*. Papers are rated on a 5-point scale mapped linearly to the interval [0,1]. Tokenization, stopword elimination and stemming have been applied to obtain the

[7] `www.vikef.net`

BOW. Documents have been processed by the JIGSAW algorithm and indexed according the BOS model, obtaining a 14% feature reduction ($20,016$ words vs. $18,627$ synsets). This is mainly due to the fact that synonym words are represented by the same synset. We measured both the classification accuracy and the effectiveness of the ranking imposed by the two different kinds of profile on the documents to be recommended. Classification effectiveness was evaluated by the classical measures *precision*, *recall* [14]. We adopted the Normalized Distance-based Performance Measure (NDPM) [16] to measure the distance between the ranking imposed on papers by the user ratings and the ranking predicted by ITR, that ranks papers according to the a-posteriori probability of the class *likes*. Values range from 0 (agreement) to 1 (disagreement). In the experiments, a paper is considered as *relevant* by a user if the rating is greater or equal than 3, while ITR considers an item as relevant if $P(c_+|d_i) \geq 0.5$, computed as in equation (4). We executed one experiment for each user. Each experiment consisted in 1) selecting the ratings of the user and the papers rated by that user; 2) splitting the selected data into a training set *Tr* and a test set *Ts*; 3) using *Tr* for learning the corresponding user profile; 4) evaluating the predictive accuracy of the induced profile on *Ts*, using the aforementioned measures. The methodology adopted for obtaining *Tr* and *Ts* was the 5-fold cross validation. The results of the comparison between the profiles obtained from documents represented using the two indexing approaches, namely BOW and BOS, are reported in Table 1. We can notice an improvement both in precision (+1%) and recall (+2%). The

Table 1. Performance of the BOW - BOS profiles.

Id User	**Precision**		**Recall**		**NDPM**	
	ITR BOW	ITR BOS	ITR BOW	ITR BOS	ITR BOW	ITR BOS
1	0.57	0.55	0.47	0.50	0.60	0.56
2	0.73	0.55	0.70	0.83	0.43	0.46
3	0.60	0.57	0.35	0.35	0.55	0.59
4	0.60	0.53	0.30	0.43	0.47	0.47
5	0.58	0.67	0.65	0.53	0.39	0.59
6	0.93	0.96	0.83	0.83	0.46	0.36
7	0.55	0.90	0.60	0.60	0.45	0.48
8	0.74	0.65	0.63	0.62	0.37	0.33
9	0.60	0.54	0.63	0.73	0.31	0.27
10	0.50	0.70	0.37	0.50	0.51	0.48
11	0.55	0.45	0.83	0.70	0.38	0.33
Mean	0.63	0.64	0.58	0.60	0.45	0.45

BOS model outperforms the BOW model specifically for users 7 and 10. By the way, the main outcome is that it is difficult to reach a strong improvement both in precision and recall by using the BOS model. Even if a higher level of precision is reached (users 5 and 7), recall has not improved. Only on user 10 we observed a general improvement of both measures. NDPM has not been improved, but it

Table 2. A case in which classification is improved without improving ranking

Item	R_u	R_A	R_B
I1	6 (1)	0.65 (2)	0.65 (2)
I2	5 (2)	0.62 (3)	0.60 (3)
I3	5 (3)	0.75 (1)	0.70 (1)
I4	4 (4)	0.60 (4)	0.45 (5)
I5	4 (5)	0.43 (6)	0.42 (6)
I6	3 (6)	0.55 (5)	0.55 (4)
I7	3 (7)	0.40 (7)	0.40 (7)
I8	2 (8)	0.30 (8)	0.30 (8)
I9	1 (9)	0.25 (9)	0.25 (9)
I10	1 (10)	0.20 (10)	0.20 (10)

remains acceptable. It could be noticed from the NDPM values that the relevant/not relevant classification is improved without improving the ranking. This situation could be explained by the example in Table 2, in which each column reports the ratings of the items and the corresponding position in the ranking. Let R_u be the ranking imposed by the user u on a set of 10 items, let R_A and R_B be the ranking computed by method A and B (ratings ranging between 1 and 6 - classification scores ranging between 0 and 1). An item is considered as relevant if the rating is greater than 3 (symmetrically, the score is greater than 0.5). Method A has a better classification accuracy with respect to method B (Recall=4/5, Precision=4/5 vs. Recall=3/5, Precision=3/4). NDPM is almost the same for both methods because the two rankings are very similar. The difference is that I4 is ranked above I6 in R_A whilst I6 is ranked above I4 in R_B. The general conclusion is that method A (BOS model) has improved the classification of items whose score (and ratings) is close to the relevant/not relevant threshold, thus items for which the classification is highly uncertain. A Wilcoxon signed ranked test, requiring a significance level $p < 0.05$, has been performed in order to validate these results. We considered each user as a single trial for the test. The test confirmed that there is a statistically significant difference in favor of the BOS model only as regards recall.

6 Conclusions and Future Work

We presented a system exploiting a Bayesian learning method to induce *semantic* user profiles from documents represented using WordNet synsets obtained by the JIGSAW WSD procedure. Our hypothesis is that replacing words with synsets in the indexing phase produces document representation that could be successfully used by learning algorithms to infer more accurate user profiles. We tried this approach by using a naïve bayes learning approach to infer profiles used in the context of planning the visit to a conference. Experiments were conducted on a collection of papers in order to compare the performance of keyword-based profiles with that of WordNet-based profiles. The main outcome is that the integration of the linguistic knowledge provided by WordNet in the learning process has improved the classification of documents whose classification score is close to the likes / dislikes threshold, that are the items for which the classification is highly uncertain. As a future work, we plan to exploit also domain ontologies in order to realize a more powerful document indexing.

References

[1] S. Banerjee and T. Pedersen. An adapted lesk algorithm for word sense disambiguation using wordnet. In *CICLing '02: Proceedings of the Third International Conference on Computational Linguistics and Intelligent Text Processing*, pages 136–145, London, UK, 2002. Springer-Verlag.

[2] S. Bloedhorn and A. Hotho. Boosting for text classification with semantic features. In *Proceedings of 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Mining for and from the Semantic Web Workshop*, pages 70–87, 2004.

[3] C. Fellbaum. *WordNet: An Electronic Lexical Database.* MIT Press, 1998.

[4] N. Guarino, C. Masolo, and G. Vetere. Content-based access to the web. *IEEE Intelligent Systems*, 14(3):70–80, 1999.

[5] C. Leacock and M. Chodorow. *Combining local context and WordNet similarity for word sense identification*, pages 305–332. In C. Fellbaum (Ed.), MIT Press, 1998.

[6] B. Magnini and C. Strapparava. Improving user modelling with content-based techniques. In *Proc. 8th Int. Conf. User Modeling*, pages 74–83. Springer, 2001.

[7] C. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*, chapter 16: Text Categorization, pages 575–608. The MIT Press, Cambridge, US, 1999.

[8] A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. In *Proceedings of the AAAI/ICML-98 Workshop on Learning for Text Categorization*, pages 41–48. AAAI Press, 1998.

[9] G. Miller. Wordnet: An on-line lexical database. *International Journal of Lexicography*, 3(4), 1990. (Special Issue).

[10] R. J. Mooney and L. Roy. Content-based book recommending using learning for text categorization. In *Proceedings of the 5^{th} ACM Conference on Digital Libraries*, pages 195–204, San Antonio, US, 2000. ACM Press, New York, US.

[11] M. Pazzani and D. Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine Learning*, 27(3):313–331, 1997.

[12] P. Resnik. Disambiguating noun groupings with respect to WordNet senses. In *Proceedings of the Third Workshop on Very Large Corpora*, pages 54–68. Association for Computational Linguistics, 1995.

[13] S. Scott and S. Matwin. Text classification using wordnet hypernyms. In *COLING-ACL Workshop on usage of WordNet in NLP Systems*, pages 45–51, 1998.

[14] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1), 2002.

[15] I. Witten and T. Bell. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, 37(4), 1991.

[16] Y. Y. Yao. Measuring retrieval effectiveness based on user preference of documents. *Journal of the American Society for Information Science*, 46(2):133–145, 1995.

Author Index