Personalized Classification of Web Documents

George Potamias\textsuperscript{1,2}, Vasilis Raxenidis\textsuperscript{2} and Alexandros Papadakis\textsuperscript{2}

\textsuperscript{1}Institute of Computer Science, FORTH, Vassilika Vouton, P.O.Box 1385, GR-711 10 Heraklion, Crete, GREECE
\textsuperscript{2}Dept. of Computer Science, University of Crete, P.O. Box 1470, GR-714 09, Heraklion, Crete

Abstract. State-of-the-art Web search engines are not capable to adapt their operations to the evolving needs, interests and preferences of users. In most of the cases the users are ‘lost in the information cyberspace’. To cope with this problem we developed a system for classification of HTML (or, XML) documents into user specified categories of interests, the user-profile. The system processes the user profile and a set of representative documents for each category of interest and produces a classification schema, presented as a set of representative category vectors. The classification schema is then utilized in order to classify new incoming Web documents. The user may modify and enrich his/her profile depending on his/her current search needs and interests. In this respect the personalized delivery of Web-based information is achieved. Experimental results on indicative sets of Web pages show the reliability and efficiency of the approach.

1 Introduction

Text classification or, categorization research aims towards the automated assignment of ‘natural’ text references (i.e., texts written in natural language) to a set of two or more predefined categories based on their content. It differs from the traditional text-retrieval operations which aims to sort texts into two classes: those that the user would like to retrieve and see (the relevant documents), and those they would like not to see (the non-relevant documents) [8]. Furthermore, text-categorization is fundamental in text-mining operations where, the basic task is to extract useful and comprehensive knowledge from huge collections of texts [3], [14].

Since the Birth of the World Wide Web (Web), the amount of information available on the Web has been increasing at an exponential rate. It is estimated that a total of 350.000.000 Web pages are accessible from your workstation [3]. On one hand, this explosive growth has put huge amounts of information at the disposal of anyone with access to the Internet. Current state-of-the-art Web search engines, even if they offer effective ways to access and retrieve this huge amount of information, seems inadequate to locate the ‘right’ information- that is, the information that matches at a high degree of relevance the user’s interests and preferences. In other words, the reliability of Web search-engines is harmed. Furthermore, none of the widely used search-engines are capable to capture and model the evolving needs and the specific preferences of users’ in order to adapt their search and retrieval processes. In most of the cases the users are ‘lost in the information cyberspace’. In fact, all currently available search tools suffer either from poor precision (i.e., too many irrelevant documents) or, from poor recall (i.e., too little of the Web is covered by well-categorized directories). Providing an efficient and user friendly search interface for the Web is still one of the most important challenges in making it accessible to
the general public [2]. Therefore, the need to put some intelligence into search-engines, capable to integrate and adapt the available information and present it in a coherent and cohesive way seems inevitable [12].

There are several different approaches to manage the proliferation of the amount of information published in the Web. To cope with this problem, an important amount of research and development work is devoted in the development of intelligent agents used to locate and retrieve information in the Web. For example in [1], an adaptive agent is presented that can bring back Web pages of user’s interest daily. Another approach is systems known as browsing assistants. Examples of such systems are InfoFinder [7] and Amalthea [10]. Moreover, systems to filter information coming from the Web have been developed- SIFT [15] and Pefna [6] are examples of such systems. The CMU World Wide Knowledge Base project, presents another serious effort towards classification of Web documents that utilizes a mixture of information retrieval and machine learning techniques [3].

The automated classification of Web pages requires the solution of two problems, which are typical of the machine learning field [4]:
1. The definition of a representative language for HTML/ XML pages, and
2. The construction of a classifier that is able to categorize new Web pages on the basis of their representation.

In this paper we present an approach towards adaptive filtering and personalized classification of Web documents. In order to cope with the aforementioned needs and problems, we developed a system for the automatic classification of Web-documents (i.e., semi-structured HTML or XML documents) according to users’ personal interests and preferences. The system utilizes techniques from the Information Retrieval [13] and Machine Learning [9], disciplines.

Next section presents a brief introduction to the fundamental operations of our system. Section 2 presents the specifics of the user profiling operations. In section 3 we present the representation issues for both the user-profile and for the Web-documents. In section 4 the specifics of the training process are presented. Section 5 describes the pattern matching metrics and functions. In section 6 we present experimental results on an indicative set of pre-classified Web-documents. In the last section we conclude and point to future research and development aspects.

2 Outline of the approach
The fundamental operations of the system are based on the following representation models, metrics and operations.

a. Representation. We rely on the well-known vector-space representation model [13]. Each Web-document is represented by a text binary vector (TBV), where bin ‘1’ is used to represent the occurrence of an index-term in the document, and ‘0’ its absence. The entries in the vector keep the pre-specified appearance of the index-terms in the declared user-profile; in that sense it is an ordered vector.

b. User profiling. The user profile is composed by a set of user-declared categories of interest each one described by a set of user-specified key-words or, key-phrases (i.e., the index-terms). The users may adjust their profiles according to their evolving needs, interests and preferences, via a set of respective tools.

c. Training and representative category-vectors. Utilizing techniques from information retrieval, a weight for each index-term is computed. The weight represents the
importance or, the representative-power of an index-term in describing the categories. The weights are computed according to a pre-selected set of training Web documents. The final outcome of the training process is a set of class relevant weighted vectors, each one presenting a representative vector for each category.

d. **Classification and prioritization.** Each binary-vector is matched against each of the class relevant vectors. The matching process is realized by the introduction of specially devised similarity-metrics. The final outcome is the classification of new incoming Web-documents to the category with which they exhibit the highest similarity. Furthermore, for each of the classified documents a priority index is computed to indicate the degree with which each document belongs to each category. Moreover, the Web-documents that exhibit similarity-indices lower than a user-specified threshold are considered as of no-relevance to the specified Web inquiry.

e. **Exploiting tagging information.** In the course of similarity matching we exploit the structure of the Web-document (i.e., its tagging information) by assigning a special weight to the index-terms that occur within pre-specified tags. The users may adjust the type of tags they want to focus-on and the weights to assign. In this way, the system is capable to capture and model the dynamically changing interests and preferences of the users and so, to achieve personalized delivery of related Web-based information.

### 3 User Profile and Text Binary Vectors

**User Profile.** User profiles are modeled by a set of declared categories of interest each one described by a set of pre-specified index-terms (representative keywords or, key-phrases). Each index-term is accompanied by a set of synonyms. A useful utilization of synonyms is for multi-lingual text collections where, the basic reference term in one language is declared, and its potential translations follow as synonyms. In the current version of the system (and in the presented experiments as well) the index-terms are declared and introduced directly by the user. We have also developed a tool for the automatic extraction of index-terms from user-provided text references (titles, pages etc). With this tool, our system may be expanded and enhanced to serve automatic user-profiling operations [5]. The format of a user profile is shown in figure 1 below.

```xml
<CATEGORY – 1 – name>
term-1-1-name-string [synonym-111-string, synonym-112-string, …]
term-1-2-name-string [synonym-121-string, synonym-122-string, …]
............... other categories
...............<CATEGORY – C – name>
term-C-1-name-string [synonym-C11-string, synonym-C12-string, …]
term-C-3-name-string [synonym-C31-string, synonym-C32-string, …]
```

**Fig. 1.** The Format of a User Profile file

The reason that we rely only on the user-specified sets of index-terms, i.e., on a controlled-vocabulary, and not on all the terms occurring in the collection of documents have to do with the fact that: the pre-selected training HTML documents may refer to some terms that are not descriptive of the incoming Web-documents to be classified. So, it is quite natural to expect that with the interference of ‘irrelevant’ terms the classification performance will
decline. Moreover, we want to keep the classification outcome as more transparent as possible. With the introduction of terms, for which the user may be potentially unaware, the transparency is lost.

_Parsing_. Each Web-document, training or testing (i.e., new incoming document) is parsed. The parser identifies the index-terms (and their synonyms) defined in the user profile. The output of the parsing procedure is a _Text Binary Vector_ - TBV.

The system’s parser utilizes the _stemming_ techniques of the Porter algorithm [11], with the inclusion of specially defined grammar. After stemming, a _pattern_ is constructed for each of the terms. Then, the patterns are matched instead of the terms themselves. If a pattern matches then, the corresponding index-term is supposed to be identified. However the use of stemming may cause additional problems. For example, consider the term ‘DOS’ (i.e., the operating system). A casual stemming would produce the stem ‘DO’ for this term. A pattern for this term would be one that matches every word starting with ‘do’ or, ‘DO’- consider how many words start with these two sub-strings. To avoid such situations we permit users to input terms in the profile that will be tested for exact matching. _Quoting_ the terms can do this.

Four pattern generation rules were devised and implemented, based on respective _regular expression_.

- Term ‘keyword’, produces the pattern \skeyword\s. For example, term ‘DOS’ produces the pattern, \sdos\s. So, the pattern matches the phrase, “DOS is an operating system” but not the phrase, “The dossier is green”.
- Term ‘keyword_1 keyword_2 ... keyword_n’, produces pattern \skeword_1 keyword_2 ... keyword_n\s. In this case, the pattern matches all occurrences of phrase, ‘keyword_1 keyword_2 ... keyword_n’. For example, the term ‘software engineer’ produces the pattern, \software engineer\s, which matches the phrase, “Tom is a software engineer”.
- Term keyword, produces the pattern \skeyword-stem where, keyword-stem is the result of stemming on term keyword. This pattern matches with all the words starting with the root of the stemmed keyword. For example, the term ‘programming’ produces the pattern, \sprogram, which matches the phrase, “the perl programming language”.
- Term keyword_1 keyword_2 ... keyword_n, produces the pattern, \skeword_1-stem\w + \skeword_2 stem\w + ... \skeword_n-stem. This pattern matches all phrases which consist of the words, keyword_1 keyword_2 ... keyword_n (with this order) or, all phrases, which consist from the roots of these words. For example, the term, _programming languages_ produces the pattern, \sprogram\w + \slanguag. This pattern matches the phrase, “The perl programming language is very simple”, but it does not matches the phrase, “programming interface language”.

_Text Binary Vectors_. For each parsed document _t_, a corresponding _Text Binary Vector_, TBVi is formed. Suppose that in the user profile there are _c_ categories, each one described with _Cc_ number of terms. Then, the _TBV_ is a vector of _m = c \times C_ places (= the total number of terms in the user profile). The _TBV_ is an _ordered_ vector in the sense that, the entries in the vector keep the order of the categories and the appearance of their respective index-terms in the user profile. Every entry in the _TBV_ will be ‘1’ if the corresponding term occurs (identified) in the document or, ‘0’ if the corresponding term does not occur (was not identified) in the document. The _TBV_ s constructed during the training procedure have one more place than the total number of the terms in the user profile. The extra place holds the name of the category into which the document is assigned.
4 Category Relevant Weighted Vectors

Each category is linked with a set of representative Web-documents, the training set. The outcome of the training process is a set of Category Relevant Weighted Vectors- CRWV each one corresponding to a respective category. The entries in each CRWV represent the importance (or, the weight) that each index-term posses in describing the respective category and discriminating between the categories.

Consider a term $v$ and a category $c$. The importance of term $v$ for category $c$ is computed by the following formula.

$$W_{c,v} = \log \left\{ \frac{(N_{c,v} + 0.5) \times \left[ (N_c - N_{c,v}) - (N_c - N_{c,v} + 0.5) \right]}{(N_c - N_{c,v} + 0.5)(N_c - N_{c,v} + 0.5)} \right\}$$

where, $N$: the number of training documents, $N_c$: the number of training documents for category $c$, $N_{c,v}$: the number of training documents for category $c$ with term $v$ present, and $N_{v}$: the number of training documents for all categories with term $v$ present (the 0.5 entries are used in order to avoid indeterminate forms of the formula).

The above weighting metric is the so-called term-precision metric, one of the most known formulas in information and text retrieval research [16]. Its utility comes from the fact that it takes advantage of the information that concerns relevant and non-relevant documents. This is the reason why we do not utilize the well-known tf/idf (term frequency / inverse-document-frequency) term weighting formula [13], which refers just to the collection of documents with no relevance-related entries. In our case, relevant are the Web-documents that are assigned to a specific category, all others considered as not-relevant.

5 Classification of Web Documents

The formed CRWVs are utilized to classify new and unseen Web-documents. Each Web-document under classification is also represented by a TBV, which is of the same form as the ones formed to represent training documents. The only difference is the extra place holding the assigned category which is missing from the under-classification TBVs.

The similarity between a Web-document and a category of interest is computed as the inner product of the document’s TBV and the respective CRWVs. Consider a document $t$: its corresponding vector $u = <v_1, v_2, \ldots, v_k>$, $v_i \in \{0,1\}$, and a category $c$ declared in the user profile with its corresponding CRWV, crwv(c) = $<W_{c,v_1}, W_{c,v_2}, \ldots, W_{c,v_k}>$, where $W_{c,v_i}$ is the importance of term $v_i$ for category $c$. Then, the similarity between a document $t$ and a category $c$ is computed by the formula below.

$$\text{similarity} \ (t,c) = \sum_{i=1}^{k} v_i \times W_{c,v_i}$$

5.1 Exploiting HTML/ XML Structure

A novelty of our approach is the exploitation of the structure of HTML/XML documents to compute more reliable and pragmatic weights for the index-terms. It is quite possible that an HTML/XML tag will contain a term. Depending on the formatting, a term found in a Web document may be assigned an extra importance apart from the one computed by the metric introduced in section 4 above. Consider for example the term 'software' which is identified as part of the title (i.e., within the <TITLE> tag) of a training document which
is assigned to category 'science'. Consider also the term 'disk' identified as part of a paragraph of the same document. Suppose that both of these terms are defined in the user profile. Then it is natural to assume that 'software' is more important than 'disk' for category 'science', because the term appears in the title, and titles pose more descriptive power for the document they appear.

Our system permits users to define weights for the HTML/ XML tags. Once a tag weight is defined, the importance, $W_{c,v}$ of a term $v$, is multiplied by the tag’s weight if this term is contained within this tag. It is possible that a term will be contained in more than one tag. In this case the weight assigned to this term is the greatest of all the weights of the tags containing it.

Assume that $tag\text{-}weight(v_i)$ is the tag-related weight computed for term $v$ (during parsing of the document and identification of the term within a specific tags). Then, the similarity metric becomes:

$$similarity(t,c) = \nu \times crwv(c) \times wV = \sum_{i=1}^{n} v_i \times W_{c,v} \times tw(v_i)$$

**Low / High Relevance of Web-documents.** Once a set of Web documents has been classified we compute the mean similarity for each category of interest (for all documents classified to the category). Consider a number of $m$ documents say $I_1, I_2, ..., I_m$ classified to category $c$. The mean similarity for category $c$ is computed by the following formula:

$$mean\_similarity(c) = \frac{\sum_{i=1}^{m} similarity(t_i,c)}{m}$$

A classified HTML document is considered as of high-relevance to the category classified to, if its similarity with this category is greater than the mean similarity of this category. Otherwise it is considered as being of low-relevance (or, of no-relevance if its similarity index with all the categories is below a pre-specified threshold).

6 Experimental Results

6.1 Data Sets and Specifics of Experiments

We evaluated the system through a series of experiments using a manually-constructed user profile, and a set of 208 Web-pages (i.e., HTML documents). The documents used for the evaluation of the system were collected using various Web search engines (Yahoo and Altavista). The pre-classification of Web pages to specific categories relied on the respective search-engines’ classification hierarchy. A total of five- (5) categories were investigated:

(Cat.1) 27 documents in the category ‘HCI’ (Human Computer Interaction)
(Cat.2) 52 documents in the category ‘infosys’ (Information Systems)
(Cat.3) 49 documents in the category ‘network’ (Communication Networks)
(Cat.4) 39 documents in the category ‘software’ (Software)
(Cat.5) 41 documents in the category ‘sport’ (Sports)

The sample user profile, with all declared categories and their descriptive index-terms are presented in the Appendix, at the end of the paper.
Experiments & tag-weight/ → Tags

<table>
<thead>
<tr>
<th>Tags</th>
<th>NW (equal weights - no-weighting)</th>
<th>MT (&lt;meta&gt; and &lt;title&gt;)</th>
<th>ALL (all tags)</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;H1&gt;</td>
<td>1</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>&lt;H2&gt;</td>
<td>1</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>&lt;A&gt;</td>
<td>1</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>&lt;B&gt;</td>
<td>1</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>&lt;I&gt;</td>
<td>1</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>&lt;TITLE&gt;</td>
<td>1</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>&lt;META&gt;</td>
<td>1</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 1. Experiments and the weights assigned to tags

The evaluation was performed conducting a series of V-fold cross-validation experiments (for V=10). We conducted experiments exploiting tagging information referring to tags: <H1>, <H2>, <A>, <B>, <I>, <TITLE>, and <META>. In particular, we run the system on three different combinations of tags and weight-assignment to them; see table 1 above for the names and the specifics of each experimental run. For each of the runs a respective learning-curve was formed. All the learning-curves are shown in figure 2 below.

![Learning curves for the NW, MT, and ALL experimental runs](image)

6.3 Discussion of the Results

**ALL.** The performance of the system, when the information about all the focused tags is utilized (all tags are given the highest weight [=5]), is very-good. It starts from a predictive accuracy level of ~91% (when just 10% of the documents are provided) and reaches the highest accuracy level, 100% when all the documents are provided. The basic finding concerns the fact that the performance remains more-or-less stable, and outperforms the respective NW and MT results (see below). The results show the power of tagging-information exploitation when Web-documents classification tasks are considered, and in-a-way it proves the reliability of our approach. **NW.** The performance of the system when no-
tagging information is utilized (i.e., all weights are set equal to 1), is quite good. The predictive accuracy level starts from about 84% (on 10% of the Web-documents available) and reaches a level of 100% when all data are available. It reaches a high-accurate level of about 92-93% when about half of the data are available (~ 50-60%). The high-accurate results should be attributed to the term-precision formula used to weight the index terms (see section 4 above). The 'right' pre-classification of training and testing documents is also a factor influencing the performance. Here, it is worthwhile to notice that the right selection and 'correct' pre-classification of training documents is crucial. The system offers to the user the ability to revise/ refine his/her collections in order to devise a ‘correct’ set of representative documents. Furthermore, the addition of a relevance-feedback operation, in order to induce and acquire better term-weights, would potentially yield even better results.

**META, TITLE.** The performance of the system when the `<META>` and `<TITLE>` tag information is exploited (i.e., terms within these tags are assigned the highest weight [=5]) starts from a predictive accuracy of about 70% and reaches its highest accuracy level, -95%, when about 90% of the data are available. The performance should be considered as inferior to the NW and ALL experiments. This should be attributed to the fact that only about half of the available Web-documents in the collection have `<title>` or, `<meta>` tags in their source. So, more documents should be provided in order to reach high-levels of accuracy.

**Statistical Significance.** Applying a paired one-tail t-test statistic on the V-fold accuracy results for ALL vs. NW, a significance difference on the P>99% level was observed. The same test for ALL vs. MT, resulted in an observed significance difference on the P>95% level.

We should also note that the use of Porter’s stemming algorithm, and the use of regular expressions contributed to the performance of the system.

### 7 Conclusion and future work

The vast quantity of information that is disposed by means of the Web has turned into a difficult and some times painful process the location and acquisition of information by the users. Although the search engines of the Web offer several facilities for the location of information in the network, the techniques used for the indexing of the latter and for the expression of further queries, are proved to be too deficient to cover the evolving needs and interests of the users.

In this paper a system was presented, which aims towards the automated classification of HTML/XML documents in users’ pre-specified categories of interest. The system combines techniques and methods from the areas of information retrieval and machine learning. Also a graphical user interface was designed for the evaluation and demonstration of the system. Apart from the basic filtering functions that have to perform, the system through its interface offers a number of management operations, which are used to adapt its functionality. Experimental results showed the reliability of our approach.

The implemented system could be quite easily re-configured and adapted to serve the need of classifying different types of semi-structured documents, other than HTML, e.g., emails, news or, XML. The user may simply call the ‘tag-manager’ of the system in order to add/ delete tags, and define/ revise their weights.

Our future research and development plans are moving towards three directions: (a) experimentation with large-scale benchmark document collections (e.g., REUTERS, TREC) in order to test the effectiveness and scalability of the system; (b) re-configuration and
adaptation of the system to other types of documents and experimentation with respective
document collections (emails, news, etc); (c) development and inclusion of modules that
offer and support automatic user-profiling and relevance-feedback services.

References

4. Esposito, F., Malerba, D., Di Pace, L., and Leo, P: WebClass: An Intermediary for the
Classification of HTML Pages, Demo paper for AI*IA ’99 (1999)Bologna, Italy.
5. Fawcett, T., and Provost, F.: Combining data mining and machine learning for effective user
7. Krulwich, B: InfoFinder Internet. *Andersen Consulting’s Center for Strategic Technology
Task. In *Proc. 15th Annual International ACM SIGIR Conference on Research and Development
Ecosystem. *Proceedings of the Conference on the Practical Application on Intelligent Agents and
papers/PAAM96/
www/wwwkb/choon-thesis.html
ftp://db.stanford.edu/pub/sift/sift.ps

Appendix A: The User Profile Used in the Experiments

<software>
software engineering {software tools, software products}
programming languages {perl, "lisp", visual basic, prolog, java, javascript}
data structures {algorithms}
object oriented programming
toolkit, visual studio]
operating systems {unix, linux, solaris, "X11", "DOS", "X windows"]
microsoft ["windows 95", "windows 98", "windows NT"]
programs [debugging, compile, run, script, applet]
files [source files, binary file, "ASCII"]
license agreement [freeware, shareware]

<information systems>
information systems
heterogeneous information sources [heterogeneous databases, heterogeneous data sources]
information retrieval [information access, information integration, text retrieval, "data mining", information gathering, Information Browsing, information processing]
databases [data bases, knowledge bases, data source, information source]
knowledge discovery [knowledge representation, information discovery]
"mediator" [warpper]
relational databases [object-oriented databases, object database]
metadata
query ["queries", thesaurus, query processing]
query languages [SQL, OQL, MySQL, JDBC]
data base vendors [sybase, oracle, ingress, gemstone,postgresql]
digital library

<HCI>
human computer interaction [human-computer interaction, "hci", user interfaces, human computer interface, human-computer interface, human machine interface, human-machine interface, user interfaces for all, UI4ALL]
computer graphics [multimedia, graphical user interface, "GUI", "UI"]
user modeling [human factors]
ergonomics [usability, usability test]
user interface software tools [Tk/Tcl, xview, motif]
user interface design [interface design and evaluation interface evaluation]

<networks>
networks [WAN, LAN, ethernet, ATM, intranet]
ISDN [BISDN]
"OSI model"
ip address [hosts, host name, hostname, domain name, "DNS", "DNS server"]
network protocols [aloha, "X.25", TCP/IP, TCP, IP, UDP, PPP, SLIP, SNMP, SMTP, OSPF, "NFS"]
browsing [ftp, internet navigation]
"FDDI"
telnet
internet providers [internet service provider, fortuthnet, "hellas on line", "computing", america on line, aol, dial up]

<sport>
sports
basketball ["NBA", "CBA", Jordan, Chicago Bulls, rebound]
football ["FIFA", World cup, football association, soccer, Panathinaikos, Pele, Olympiakos]
tennis [table tennis]
hockey ["NFL"]
volley [volleyball, beach volleyball]
baseball
softball
athletics
olympic games [olympics, gymnastics]