Abstract
The Web search engines offer significant facilities for locating information on line but the provided indexing schemas and querying services seems inadequate. The large amount of information items and the lack of adaptive search engines that adjust to the evolving users needs and interests results into the 'lost in space' situation. To cope with this problem we developed a system for classification of structured HTML documents into user's categories of interests. The user defines his/her categories of interests by describing them with simple keywords or, key-phrases. The system processes the user profile and a set of representative documents for each category of interest elaborating on a classification schema for other HTML documents. Classification is carried out via a simple instance-based learning method realized by the formation of Class Relevant Weighted Vectors, and respective similarity matching operations. The system is capable to reserve several classification schemas corresponding to different user profiles and different representative documents. Though, it is able to adapt to the evolving interests of users. Experimental results on Web located documents indicate the reliability, efficiency and flexibility of the approach.

1 Introduction
The explosive growth of the World Wide Web (WWW or, Web) causes a significant and continuous increase to the amount of information accessed through the Internet. Locating the information that is mostly relative with user preferences has become a difficult and complex procedure due to the large amount of available reference documents. To locate and retrieve documents, or other types of information, several search engines have been developed such as Yahoo, AltaVista, and others. The search engines create and store indexes for information published in the WWW, and supply the users with the capability to locate specific documents in an easy way, using keyword-based queries. The search techniques, employed in most of the WWW search engines, usually locate a large amount of documents, which are of low relevance with the actual domain of user's interests. Furthermore, even experienced users, which may know clearly the domains of their interests, usually submit queries composed of keywords not capable to capture and describe accurately their interests. Therefore, users need to put more effort in order to locate the documents that present real relevance to their interests.

There are several different approaches to manage the proliferation of the amount of information published in the WWW. An important part of the work has been devoted to the development of intelligent agents used to locate and retrieve information in the Web. For example Balabanovic, [Balabanovic, 1995], describes an adaptive agent that can bring back Web pages of interest, daily. Other approaches include the well known browsing assistants, InfoFinder, [Krulwic, 1996], and Amalthea, [Moukas, 1996]. Moreover, systems to filter information coming from the Web have been developed, SIFT, [Yan, 1995], and Pefna, [Kilander, 1996], are examples of such systems.

Utilization of Machine Learning (ML) approaches to identify and recall, relevant to users' interests documents
from the Web have also been considered. The most known systems are WebWatcher, [Armstrong, 1995], and Syskill & Webert, [Pazzani and Billsus, 1997]. Text categorization have also been tackled with ML techniques, [Cohen, 1995], and combined ML and natural language processing approaches, for extracting text-based information from the Web, [Soderland, 1997]. It is evident that ML techniques compose a natural framework to tackle with information filtering tasks.

In this paper we present our work on a system for classification of structured HTML documents into user's interests. The user may define a category of interest by providing a respective list of keywords or, key-phrases. The system utilizes information retrieval techniques and metrics [Salton and McGill, 1983], in order to create a Class Relevant Weighted Vector (CRWV) for each of the categories. The formation of these vectors is based on a set of representative and indicative documents for each of the categories. Then, it resembles an instance based learning operation, [Aha et. al., 1991], in order to classify new (and unseen) HTML documents. Furthermore, a classified HTML document can be characterized as low or, high relevant with respect to its category class. So, easy and fast identification of the most relative and interesting documents is supported.

The system may be used in cooperation with a Web search engine to automatically filter, and classify the identified documents into user's categories of interest. In addition, each classified document is attached with a similarity index helping users to infer its relevance. It is on the responsibility of the user to supply the system with the set of indicative HTML documents. In its current implementation, the system requires that the HTML documents to be classified should be first selected by one or more search engines and stored locally. However our system can be easily adapted as a module to other applications which need to classify HTML documents.

The presented HTML classification system was implemented on a Sun Sparc Station, Solaris 2.5 UNIX system. In addition, all the modules used in the training and classification procedures was implemented using the Perl programming language [Wall and Randal, 1991; PERL]. The module used for the HTML parsing may be found at [HTML Parser]. The module used for the stemming is taken from [Stem]. The user interface was developed as a Visual Basic [VB] application. To demonstrate and evaluate the system, we developed a simple graphical user interface for Microsoft Windows© environments.

The rest of the paper is organized as follows: section 2 presents the representation of user profile and HTML documents; sections 3 and 4 present the specifics of the classification schema; section 5 presents the conducted experiments and respective results regarding classification accuracy; in section 6 we outline other work; finally, in section 7, we conclude and point to future research tasks and work.

2 User profile and HTML documents

A user profile is composed by a set of categories of interests where, each category is described by a set of representative keywords and/or, key-phrases. For the categories defined in the user profile, the system computes a classification schema through a training procedure which is based on the user defined categories and on a set of representative or, training documents for each of the categories.

A keyword is a single word string. A key-phrase is a string of more than one word separated by spaces. Keywords and key-phrases compose the terms of the user profile. Moreover a list of synonyms may be provided for each term. A general sample of a user profile is shown in Figure 1 below.

Figure 1: The format of a user profile file

In Appendix A, a sample user profile used for the experiments presented in section 5, is given.

Parsing: Each training HTML document is parsed. The parser identifies the terms (and their synonyms) defined in the user profile. By performing an exact matching of the terms with the terms of an HTML document we can not be sure that every concept in the user profile will be identified in a document, even if it exists. This may happen because a concept may be expressed in a document by a term that is derived by the same lexicographic root as the term used to express the same concept in the user profile. However, these terms may differ. For example the term 'databases' can not be identified in a document, if the user profile contains the term 'database' and an exact matching is performed.

To cope with this problem we perform a stemming on the words comprising the terms found in the user profile. The stemming is performed using the Porter's algorithm described in [Porter, 1980]. After stemming, a pattern is constructed for each of the terms. Then, we try to match the patterns instead of the terms themselves. If a pattern matches then, the corresponding 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. All of them have no relation with the
DOS operating system. To avoid such situations we permit users to input terms in the profile which will be tested for exact matching. Quoting the terms can do this. In general, four pattern generation rules were devised and implemented, based on respective regular expression.

1. Term 'keyword', produces the pattern \$keyword\s. For example, term 'DOS' produces the pattern, \$dos\s. So, the pattern matches the phrase, "DOS is an operating system" but not the phrase, "The dossier is green".

2. Term 'keyword_1 keyword_2 ... keyword_n', produces pattern \$keyword_1\s 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\s engineer' produces the pattern, \$software\s engineer\s, which matches the phrase, "Tom is a software engineer".

3. Term keyword, produces the pattern \$keyword-stem\s 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, \$program\s, which matches the phrase, "the perl programming language".

4. Term keyword_1\s keyword_2 ... keyword_n\s, produces the pattern, \$keyword_1-stem\w + \$keyword_2\s stem\w + \w...\w + \$keyword_n-stem\s. 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\s languages produces the pattern, \$program\w + \$language\s. This pattern matches the phrase, "The perl programming language is very simple", but it does not matches the phrase, "programming interface language".

**Term Binary Vectors:** The output of the parsing procedure is a Term Binary Vector (TBV). Suppose that in the user profile there are \(C\) categories, with \(KC\) number of terms used to define each of them. Then, \(TBV\) is a vector of \(m = C\times KC\) places. Furthermore, TBVs are ordered vectors in the sense that, the entry places of the vector will keep the order of the categories and their terms in the user profile. Each entry in a TBV will be 1, if the corresponding term occurs (identified) in the document or, 0 if the corresponding term does not occur (is not identified) in the document. For each TBV an additional place is reserved, to hold the name of the category into which the document is assigned.

### 3 Class Relevant Weighted Vectors

The training procedure produces a vector called Class Relevant Weighted Vector (CRWV) for each of the categories of interest declared in the user profile. For a specific category of interest, the respective CRWV encodes the importance (weight) of each term for this category.

The importance of a term for a specific category of interest indicates the inclination of a document containing this term to be classified to this category. The importance of a term is represented by a real number, positive or negative. A positive importance for a term in a document, with respect to a specific category, increases the relevance of this document to the category. The appearance of a term with negative importance decreases the relevance, respectively. The set of CRWVs produced as an output of a training procedure, along with the corresponding user profile, forms a classification schema which is then used by the system to classify (test or, unseen) HTML documents into the declared categories. The system may reserve several different classification schemas as produced by different training HTML references and different user profiles. So, the users are able to form different classification schemas that capture their evolving interests and needs (e.g., by supplying a different set of training documents or, revising their profiles).

A CRWV has a number of places equal to the total number of terms defined in the user profile. In addition it is ordered exactly as a TBV is. The value held by each place of a CRWV encodes and represents the importance of the corresponding term with respect to the category the CRWV is formed for. Consider a term \(v\) and a category \(c\). The importance of a term \(v\) for category \(c\) is computed by the following formula, borrowed from the information retrieval discipline, and given in [Yu et al., 1982],

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

Where,

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

By analyzing the above formula we can conclude the following remarks about the importance of a term for a category of interest:

- The importance is proportional to the number of the training documents assigned to category \(c\) into which the term is identified; the more training documents of category \(c\) containing term \(v\) the greater the importance of this term for the category.
- Factor \((N_v - N_{v,c})\) represents the number of training documents for category \(c\) containing term \(v\) which was not identified; if the value of this factor is increased then the importance of the term is decreased.
- Factor \((N - N_v)\) represents the number of training documents of all categories into which term \(v\) was not identified.
- Subtraction factor \((N - N_v) - (N_c - N_{v,c})\) represents the number of training documents of all categories except category \(c\) into which the term \(v\) was not identified.

The importance of a term is proportional to this factor.
That is, the more training documents of all categories, except for $c$, not containing term $v$, the more important the term $v$ for category $c$.

- Factor $(N_v - N_{c,v})$ expresses the number of training documents of all other categories except category $c$ into which the term $v$ was identified. The importance of a term is reversibly proportional to this factor. That is, the more training documents of all other categories, except from $c$, containing term $v$, the less important is $v$ for this category.

- The importance of a term for a specific category is reversibly proportional to the number of training documents of the category for which the term is not identified. That is, the importance is reversibly proportional to the factor $(N_v - N_{c,v})$.

The training process, actually the formation of the respective CRWVs, is performed in two steps. In the first step the representative documents of each category of interest are parsed and the corresponding TBVs are formed. In the second step, using the TBVs, a CRWV is constructed for each category of interest declared in the user profile.

### 3.1 Importance of terms and HTML tags

Our system exploits the **structure of an HTML document** to compute more reliable importance weights for the involved terms. It is quite possible that a term identified in an HTML document will be contained within a particular HTML tag. The placement of a term within particular tags can be exploited in order to assess a more reliable and pragmatic importance weights for it.

The system allows the assignment of weights to HTML tags. Once a weight has been defined for a tag, the importance $W_{c,v}$ computed for a term is multiplied by the tag’s weight if this term is identified to be contained within this tag. It is possible that a term will be contained by more than one tag. In this case the weight assigned to the term is the greatest of all the weights of the tags containing it.

<table>
<thead>
<tr>
<th>TAG</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;H3&gt;, &lt;A&gt;, &lt;I&gt;</td>
<td>2</td>
</tr>
<tr>
<td>&lt;H2&gt;</td>
<td>3</td>
</tr>
<tr>
<td>&lt;H1&gt;, &lt;B&gt;</td>
<td>4</td>
</tr>
<tr>
<td>&lt;TITLE&gt;, &lt;META&gt;</td>
<td>5</td>
</tr>
</tbody>
</table>

**Table 1:** Weights assigned to HTML tags.

In the course of the evaluation experiments (section 5), we assigned weights for a subset of the HTML tags. These weights are shown in Table 1 above. We do not claim that these weights are the optimum ones. The assignment of weights to tags was done based on our experience acquired during the study of many HTML documents. The experiments conducted during the evaluation process indicated that a careful choice for the weights of the tags might considerably improve the performance of the system.

### 4 Classification of HTML documents

Once the training process has produced a classification schema, i.e., CRWVs are computed and formed, it can be used to classify HTML documents. An HTML document to be classified is represented by a document vector. A document vector is similar to a TBV apart from the extra place holding the name of the assigned category. This place is missing in the document vector. A document vector is also ordered exactly as a TBV is and each place holds a value of $1$ or $0$ depending on the occurrence or, not of the corresponding term in the respective document, respectively.

An HTML document is classified to the most similar category of interest. The **similarity** between an HTML document and a category of interest is computed as the *inner product* of the document vector and the respective CRWV of the category.

Consider a document $t$ and its corresponding vector, $vt = \langle v_t1, v_t2, ..., v_tk, v_i \rangle$, $v_i \in \{0,1\}$. Consider also the class relevant weighted vector of a category $c$, $crw(v) = \langle w_{c,v1}, w_{c,v2}, ..., w_{c,vk} \rangle$ where, $w_{c,v}$ 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 following formula:

$$similarity(t,c) = \sum_{i=1}^{k} v_i * w_{c,v_i}$$

Factor $v_i$ takes values from set $\{0,1\}$; it is $I$ if $v_i$ is present in document $t$ and $0$ otherwise.

If a tag specific weight, $tw(v_i)$, has been computed for a term $v_i$, due to its HTML formatting then, the above similarity formula forms as follows,

$$similarity(t,c) = \sum_{i=1}^{k} v_i * w_{c,v_i} * tw(v_i)$$

Prioritizing documents: Once a set of HTML documents has been classified we may compute the mean similarity for each category of interest. Consider a number of $n$ documents, $t_1, t_2, ..., t_n$, classified to category $c$. The mean similarity for category $c$ is computed by the following formula:

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

A classified HTML document is considered of high relevance for its assigned category, if its similarity with respect to this category is greater than the mean similarity of the category. Otherwise it is considered of low relevance.

### 5 Experimental evaluation

We evaluated the system through a series of ten experiments, following a V-fold, $V=10$, cross-validation procedure, [Breiman et al., 1984]. A respective user profile was constructed, given in Appendix A, and a set of 208 HTML was extracted from the Web. The terms defined in the user profile for each of the declared categories were used to form respective queries to a specific Web search engine. The located documents where candi-
dates for assignment to this category. From the candidates located for each category only those that were considered as most representative where finally selected. During the selection procedure, we identified more terms that enhance the description of each of the categories in the user profile, and these terms were actually inserted. The final set of selected HTML documents were assigned to five (5) different categories of interest, with a varying number of documents for each of then:

"software" (Software): 39
"infosys" (Information Systems): 52
"HCI" (Human Computer Interaction): 27
"network" (Communication Networks): 49
"sport" (Sports):

The results of the 10-fold cross-validation procedure are shown in Table 2 below. Furthermore, a respective learning curve was formed, shown in Figure 2.

5.1 Discussion

Regarding the results in Table 2, and the learning curve of Figure 2, the following observations could be made.

- The performance of the system ranges between ~85% and ~97%. An overall accuracy of 92.2% was achieved. This is due to the fact that the training documents used for the experiments were chosen to contain many of the terms defined to describe the category they were assigned to. Also the selected terms were distinctive for each category. In addition the use of the Porter’s stemming algorithm and the use of regular expressions for the search of the terms in the documents contributed to the increased performance of the system.

- However, we claim that the high performance is mainly accomplished due to the use of weights for the terms contained by HTML tags. Experiments conducted without the use of weights for the HTML tags, in comparison with those conducted by using weights, produced poor results. The results of the experiments indicated that the performance of the system is improved by assigning a high weight to the <META> HTML tag. This is due to the special usage of this tag - it usually contains index terms for the document, and sometimes it may contain a short abstract for it.

Learning curve: The expected intuitive form of the learning curve would be different from the one presented in Figure 2. A normal learning curve would increase to a turning point and then, more or less, stabilize. The curve in Figure 2 although it increased at the beginning then fluctuated almost randomly. Mainly, these fluctuations are caused because some of the training documents either, do not contain enough category descriptive terms or, contain many terms describing other categories, besides the assigned one.

Another reason that could explain those variations is that the initial manual assignment of test documents to the categories of the user profile, was not correct. For example a document actually belonging to category B, assigned (manually) to category A, and classified to category B, could be considered as a false classification.

<table>
<thead>
<tr>
<th>Train - Test %</th>
<th>Cat.</th>
<th>#Train Docs</th>
<th>#Test Docs</th>
<th>Acc. %</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 - 90 %</td>
<td>1</td>
<td>4</td>
<td>35</td>
<td>74.3</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>5</td>
<td>47</td>
<td>85.1</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>3</td>
<td>24</td>
<td>87.5</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>5</td>
<td>44</td>
<td>100.0</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>4</td>
<td>37</td>
<td>86.5</td>
</tr>
<tr>
<td>Tot.</td>
<td>21</td>
<td>187</td>
<td>36.6</td>
<td></td>
</tr>
<tr>
<td>20 - 80 %</td>
<td>1</td>
<td>1</td>
<td>8</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>10</td>
<td>10</td>
<td>92.9</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>5</td>
<td>22</td>
<td>90.9</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>10</td>
<td>39</td>
<td>92.3</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>8</td>
<td>33</td>
<td>90.9</td>
</tr>
<tr>
<td>Tot.</td>
<td>41</td>
<td>167</td>
<td>90.6</td>
<td></td>
</tr>
<tr>
<td>30 - 70 %</td>
<td>1</td>
<td>12</td>
<td>22</td>
<td>74.1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>15</td>
<td>27</td>
<td>89.4</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>8</td>
<td>19</td>
<td>100.0</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>15</td>
<td>34</td>
<td>100.0</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>12</td>
<td>27</td>
<td>89.4</td>
</tr>
<tr>
<td>Tot.</td>
<td>62</td>
<td>146</td>
<td>89.9</td>
<td></td>
</tr>
<tr>
<td>40 - 60 %</td>
<td>1</td>
<td>16</td>
<td>22</td>
<td>81.8</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>20</td>
<td>32</td>
<td>93.8</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>10</td>
<td>17</td>
<td>100.0</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>19</td>
<td>30</td>
<td>100.0</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>16</td>
<td>25</td>
<td>96.0</td>
</tr>
<tr>
<td>Tot.</td>
<td>81</td>
<td>126</td>
<td>94.3</td>
<td></td>
</tr>
<tr>
<td>50 / 50 %</td>
<td>1</td>
<td>21</td>
<td>19</td>
<td>73.7</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>21</td>
<td>28</td>
<td>89.3</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>14</td>
<td>13</td>
<td>100.0</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>24</td>
<td>25</td>
<td>100.0</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>21</td>
<td>20</td>
<td>95.0</td>
</tr>
<tr>
<td>Tot.</td>
<td>103</td>
<td>155</td>
<td>91.6</td>
<td></td>
</tr>
<tr>
<td>60 - 40 %</td>
<td>1</td>
<td>23</td>
<td>16</td>
<td>87.5</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>32</td>
<td>20</td>
<td>100.0</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>17</td>
<td>10</td>
<td>100.0</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>19</td>
<td>20</td>
<td>100.0</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>25</td>
<td>16</td>
<td>87.5</td>
</tr>
<tr>
<td>Tot.</td>
<td>126</td>
<td>82</td>
<td>95.0</td>
<td></td>
</tr>
<tr>
<td>70 - 30 %</td>
<td>1</td>
<td>27</td>
<td>12</td>
<td>100.0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>22</td>
<td>15</td>
<td>93.3</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>19</td>
<td>8</td>
<td>87.5</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>34</td>
<td>15</td>
<td>100.0</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>29</td>
<td>12</td>
<td>100.0</td>
</tr>
<tr>
<td>Tot.</td>
<td>146</td>
<td>62</td>
<td>96.1</td>
<td></td>
</tr>
<tr>
<td>80 - 20 %</td>
<td>1</td>
<td>31</td>
<td>8</td>
<td>40.0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>42</td>
<td>10</td>
<td>90.0</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>22</td>
<td>5</td>
<td>100.0</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>39</td>
<td>10</td>
<td>100.0</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>34</td>
<td>8</td>
<td>87.5</td>
</tr>
<tr>
<td>Tot.</td>
<td>167</td>
<td>41</td>
<td>95.5</td>
<td></td>
</tr>
<tr>
<td>90 - 10 %</td>
<td>1</td>
<td>35</td>
<td>4</td>
<td>75.0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>44</td>
<td>5</td>
<td>100.0</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>24</td>
<td>3</td>
<td>100.0</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>44</td>
<td>5</td>
<td>100.0</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>37</td>
<td>4</td>
<td>100.0</td>
</tr>
<tr>
<td>Tot.</td>
<td>184</td>
<td>21</td>
<td>95.0</td>
<td></td>
</tr>
<tr>
<td>100 - 100 %</td>
<td>1</td>
<td>39</td>
<td>19</td>
<td>94.9</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>52</td>
<td>52</td>
<td>96.2</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>27</td>
<td>27</td>
<td>100.0</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>49</td>
<td>49</td>
<td>100.0</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>41</td>
<td>41</td>
<td>97.2</td>
</tr>
<tr>
<td>Tot.</td>
<td>208</td>
<td>208</td>
<td>96.8</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: V-fold (V=10) cross-validation results

The number of test documents is an important factor for the calculation of the learning curve. It is natural to assume a fixed set of documents (resting somewhere in the Web) which we wish to classify. So, we conducted an
The user employs a search engine to inquire relevant information by the users of the network. Although the search engines of the WWW 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 needs of users and their evolving interests. When a user employs a search engine to inquire relevant information, he is usually confronted with a vast amount of ref-

A simple paradigm of information classification is the one used by many popular WWW search engines. For example in Yahoo [Yahoo] and Alta Vista [Altavista] users may choose specific categories into which the search is performed. The meta-search engine users may know about the frequency of words inside the desired documents, and also patterns or common phrases that can be important for the searched documents.

The novelty of our system is that it uses both the user profile and a list of representative documents in order to create a classification schema. Other systems use either the user profile or the representative documents but not both of them. Moreover our system exploits the information provided by the structure and the formatting of the HTML documents in order to compute more pragmatic importance measures for the terms.

The vast quantity of information that is disposed by means of the WWW has turned into a difficult and some times painful process, the location and acquisition of information by the users of the network. Although the search engines of the WWW 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 needs of users and their evolving interests. When a user employs a search engine to inquire relevant information, he is usually confronted with a vast amount of ref-

Another adaptive system is Amalthaea [Moukas 96]. Amalthaea is a multi-agent system used to locate and filter information based on the user’s preferences. User's preferences are expressed by representative documents, which are represented by term vectors. A document to be filtered (or, classified) is represented by a similar vector. The similarity of two documents is expressed as the similarity of the corresponding vectors. Moreover the user may give relevance feedback to system by evaluating the results in order to adapt for future searches.

### 6 Related work

The large amount of information available in WWW, and the expression constraints imposed by the queries supported by WWW search engines has led researchers to implement systems to filter and classify WWW information.

A simple paradigm of information classification is the one used by many popular WWW search engines. For example in Yahoo [Yahoo] and Alta Vista [Altavista] users may choose specific categories into which the search is performed. The meta-search engine ProFusion [Gauch and Wang 96], [Gauch, Wang and Gomez 96], has the ability to analyze the queries submitted by users in order to identify the categories of information they refer to. For each category identified the most promising search engine is chosen to answer the query. These methods produces satisfying results but they do not permit users to know the descriptions for the categories neither to modify them. Consequently it is impossible for a user to adapt one of the above systems to his/her specific needs and evolving interests.

An adaptive system is presented in [Balabanovic et al. 1995]. This is an adaptive agent that can bring back Web pages of a user’s interest. The user gives relevance feedback to the agent by evaluating Web pages that were brought back. The agent then makes adjustments for future searches on relevant Web pages.

Another adaptive system is Amalthaea [Moukas 96]. Amalthaea is a multi-agent system used to locate and filter information based on the user’s preferences. User's preferences are expressed by representative documents, which are represented by term vectors. A document to be filtered (or, classified) is represented by a similar vector. The similarity of two documents is expressed as the similarity of the corresponding vectors. Moreover the user may give relevance feedback to system by evaluating the results in order to adapt for future searches.

### 7 Conclusion and future work

The vast quantity of information that is disposed by means of the WWW has turned into a difficult and some times painful process, the location and acquisition of information by the users of the network. Although the search engines of the WWW 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 needs of users and their evolving interests. When a user employs a search engine to inquire relevant information, he is usually confronted with a vast amount of ref-
ferences and documents, which are proved to be of low relevance to what he really expects. To tackle the problem some search engines use techniques that filter the respective references and documents.

In this paper we present a system aiming to classify HTML documents into categories that captures the interests of a user. The system classifies a set of HTML documents into categories of interest and calculates the similarity of each document for each of the categories. The system could be proved to be quite useful for Internet users since it can be used to filter HTML documents that are located by a search engine and discard those with low relevance.

Also a graphical user interface was designed for the evaluation and demonstration of the system. Apart from the basic filtering functions that has to perform, the system through its interface offers a number of management operations which are used to adapt its functionality.

The operation of the system is based upon machine learning techniques. The main objective of the system is the classification of HTML documents as described above. This is achieved using a training process during which the system produces classification schemas in order to use them as the basis for the classification of those documents. For the construction of the classification schemas, the system uses the categories and the terms defined in the user profile as well as a number of indicative documents for which their class is known. Depending on the number of terms that appear in the training documents, and on the number of documents that contain specific terms, the system calculates the importance of every term for each category of interest, forming though a Class Relevant Weighted Vector for each category.

Based on these vectors, new HTML documents could be classified to their most likely category.

Although this method looks logical, in order to produce reliable results, it needs the careful selection of the terms that are used to describe each category in the user profile. Furthermore the documents that are used for the training process have to be as representative as possible for their respective category.

Although the results obtained, and which are based on the importance of the terms as defined by the formula 1, are considered satisfactory, it was realized that the performance of the system could be improved using information which is originated from the structure and the format of the documents that are represented according to the HTML standard. For example the fact that a term is located in the title of a document which has been assigned to a specific category can cause the term to become important for this category. Depending on the occurrence of a term in bold typeface might also mean that this term is particularly important for the category the document is assigned to. After some experiments that were performed, it was realized that significant improvement in performance could be achieved using the HTML formatting specifics. The tag `<META>` usually contains such information. In many documents the content of this tag comprises of terms which are representative for the document.

The experimental results with the current implementation of the system, showed that the methodology followed is right, since the performance of the system is quite high. However two further modifications were considered, which could possibly increase the effectiveness:

- Except the weighting of the HTML tags, the user could define directly importance factors for the terms inserted in its profile.
- For the calculation of the importance of terms for the categories of interest, the number of occurrences of those terms in the documents could also taken into account.

It cannot be said that the above modifications will improve or deteriorate the performance of the system. However they provide a natural direction to follow.

The aforementioned modifications concern the methodology. Further modifications could be made which relate to the use of the system. For example, the system could be modified to be able to classify some other type of structured information, apart from HTML documents, e.g., emails, news. Furthermore, the system could be directly hosted into a WWW search engine, instead of storing documents locally and then process them. Nevertheless such an extension might not be that simple since the location and acquisition of documents from the WWW is a composite and complex process and therefore should better be left to the search engines.

References


**Appendix A: The sample user profile Used in experiments**

<software>

<information systems>
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]
virtual reality

<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, forthnet, "hellas on line", "compuling", 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]
golf
hockey ["NFL"]
volley [volleyball, beach volleyball]
baseball
softball
water polo
athletics
olympic games [olympics, gymnastics]