An Effective Fuzzy Clustering Algorithm for Web Document Classification: A Case Study in Cultural Content Mining

Damianos Gavalas and George E. Tsekouras

Department of Cultural Technology & Communication,
University of the Aegean,
Mytilene, Greece

dgavalas@aegean.gr, gtsek@ct.aegean.gr

Abstract

This article presents a novel crawling and clustering method for extracting and processing cultural data from the web in a fully automated fashion. Our architecture relies upon a ‘focused’ web crawler to download web documents relevant to culture. The term ‘focused crawler’ refers to web crawlers that search and process only those web pages that are relevant to a particular topic. After downloading the pages, we extract from each document a number of words for each thematic cultural area, filtering the documents with non-cultural content; we then create multidimensional document vectors comprising the most frequent cultural term occurrences. We calculate the similarity (‘weighted Hamming distances’) between the cultural-related document vectors and for each cultural theme, we use cluster analysis to partition the documents into a number of clusters. Our approach is validated via a proof-of-concept application which analyzes hundreds of web pages spanning different cultural thematic areas.

Keywords: web crawling, xHTML parser, document vector, cluster analysis, weighted Hamming distance, similarity measure, filtering.

1. Introduction

The web has become the major source of information and constitutes a huge database in constant growth, which disseminates news and documents at incredible speeds. Web crawlers (also called web spiders or robots) are programs used to discover and download documents from the web. Simple crawlers can be used by individuals to copy an entire web site to their hard drive for local viewing [3]. Crawlers typically perform a simulated browsing of the web by extracting links from pages, downloading the pointed web resources and repeating the process ad infinitum.
This process requires enormous amounts of hardware and network resources, ending up with a large fraction of the visible web on the crawler’s storage array. When information about a predefined topic is desired though, a specialization of the aforementioned process called “focused crawling” is used [3],[6],[14]. When searching for further relevant web pages, the focused crawler starts from the given pages and recursively explores the linked web pages [9],[26]. While the crawlers, used for refreshing the indices of the web search engines, perform a breadth-first search of the whole web; focused crawlers explore only a small portion of the web using a best-first search guided by the user interest and based on similarity estimations [14],[26],[27]. The generic architecture of a focused web crawler is illustrated in Figure 1.

![Figure 1. The generic architecture of a focused web crawler.](image)

To maintain a fast information retrieval process, a focused web crawler has to perform web document classification on the basis of certain similar characteristics. This task is accomplished by the similarity estimator processor shown in Figure 1. The classification of web documents is carried out by using already known document classes in combination with specially designed algorithms to extract words and phrases, thereby forming a number of collections [1], [10]. These collections comprise sets of documents strongly related with a specific class [15], [16]. Clustering has been proposed as an efficient web document classification method. Clustering refers to the assignment of a set of observations (or patterns) into subsets (called clusters) so that patterns in the same cluster are similar in some sense.

These patterns can be categorized into two types [1],[10],[15],[16]: (a) web documents presented in specific document formats like xHTML (or XML) containing control strings and text, and (b) web server log files containing access sequences of web pages visited by specific users. In both cases, relations between patterns can be obtained by using the well-known Hamming distance or the Levenshtein distance [8].
This article introduces a novel focused crawler that extracts and processes cultural data from the web. The pages' processing incorporates two phases. In the first phase the crawler surfs the web using a recursive procedure, downloads web pages and removes those not including cultural content using a simple statistical method. The main innovation of our method lies in the second phase wherein web pages are logically separated in different clusters depending on their thematic content (e.g. music pages and theatre pages will be assigned to separate clusters). Our method involves: (a) creation a multidimensional document vector comprising the most frequent word occurrences; (b) calculation of the ‘weighted Hamming distances’ between the cultural-related documents; (c) partitioning of the documents into a number of clusters.

The remainder of the article is organized as follows: Section 2 describes the crawling and documents parsing and analysis procedure. Section 3 analyzes the clustering procedure. Section 4 discusses the results of the experimental validation of our algorithm and Section 5 concludes our work and presents directions for future work.

2. Related Work

Lin et al. [9] proposed a method for automating the extraction of specialized information from the web. Their focus has been on extracting ‘sentiments’ of the Chinese Stock Market resulting from a daily scan of thousands of web pages that are then interpreted by means of self-learning techniques with simple statistical processing. Tan et al. [19] proposed a cluster-based web crawling policy that addressed the problem of predicting the change behavior of web pages so that clusters of pages with frequent change patterns would be crawled more often.

Clustering appears to be a reliable solution for web document classification [10]. The steps to classify web documents involve the utilization of already classified documents in combination with specially designed algorithms to extract words and phrases, typically termed items [1]. These items and their synonyms form collections, wherein indices are used to indicate which item is related to a specific class. Moreover, the collections along with the respective indices carry information about how strongly each item is associated with a specific class [16]. The task of assigning a new document to a class is accomplished through the definition of appropriate similarity or dissimilarity measure. One of the most efficient approaches to classify web documents is to use cluster analysis. Clustering is an unsupervised learning method that partitions a set of patterns into groups (clusters), where elements (patterns) that belong to the same group are as similar as possible, while elements belonging to different groups are as dissimilar as possible. We distinguish two main categories of clustering algorithms. The first category is called hierarchical clustering and it produces nested partitions generated by sequential agglomerative of divisive algorithms, which are based on
distance measures between clusters (such as single link, average link, complete link, etc) [17]. A major drawback of sequential algorithms is their strong dependence on the order in which the patterns are elaborated. The second category is referred to the so-called partitional clustering algorithms [2]. Their implementation is based on the alternating optimization of a certain objective function. Many clustering algorithms assume that the patterns are real vectors, called numerical patterns. However, in the web we usually consider non-numerical patterns. These patterns can be categorized into two types: (a) web documents presented in a specific document formats like xHTML containing control strings and text [12], and (b) web server log files containing access sequences of web pages visited by specific users [15]. Relations between non-numerical patterns can be obtained by using the Hamming distance or the Levenshtein distance [8].

3. Retrieval of Web Documents and Calculation of Documents Distance Matrix

Prior to performing clustering of web documents, our algorithm involves the documents retrieval (crawling) and parsing and also the calculation of their distance vector and distance matrix. The high-level process of crawling, parsing, filtering and clustering of the downloaded web pages is illustrated in Figure 2. Our algorithm is described in detail in the following subsections.

Figure 2. The high-level process of crawling, parsing, filtering and clustering of the downloaded web pages.
3.1. Crawling Procedure

For the retrieval of web pages we utilize a simple recursive procedure which enables breadth-first searches through the links in web pages across the Internet. The application downloads the first (seed) document, retrieves the web links included within the page and then recursively downloads the pages where these links point to, until the requested number of documents has been downloaded.

In order to shorten the search space, our focused web crawler predicts the probability that a link to a particular page is relevant to culture before actually downloading the page. Our predictor utilizes an approach inspired by [26] wherein only links with anchor text containing or surrounded by cultural terms are assumed to point to culture-related documents. The respective pseudo-code implementation is given in Figure 3. We used standard HTTP connections for retrieving the code of xHTML documents (java.net.URLConnection class).

```
Initialize:
UrlsDone = ; UrlsTodo = {‘firstsite_seed.htm’, ‘secondsite_seed.htm’..}  
Repeat:
    url = UrlsTodo.getNext()
    ip = DNSlookup( url.getHostname() )
    html = DownloadPage( ip , url.getPath() )
    UrlsDone.insert( url )
    newUrls = parseForLinks( html )
    For each newUrl
        If not UrlsDone.contains( newUrl )
            and probabilityCulturalPage (newUrl ) > P
                then UrlsTodo.insert( newUrl )
```

Figure 3. The web crawling algorithm.

3.2. Parsing of xHTML Documents

Each retrieved document is parsed using an open-source Java-based xHTML parser [22]. The parser maintains the title and the ‘clear’ text content of the document (the series of words included within its body section) and the URL addresses where the document links point to. The title and the document’s body content are then translated to a convenient, easy to process XML format. The ‘noise’ words such as articles, punctuation marks, etc, are filtered out. The documents not including sufficient cultural content (in our prototype, ‘valid’ cultural terms are included in a ‘cultural dictionary’), i.e. those wherein the total number of cultural terms fall below a predefined threshold are deleted. The documents corresponding to the document’s URL addresses are retrieved and
parsed on the next algorithm’s execution round, unless they have been already appended in the ‘UrlsDone’ list or they are unlikely to point to non culture-related pages.

3.3. Calculation of the Document Vectors

For each parsed document we calculate the respective document vector denoted as \( DV \). The role of the \( DV \) is to indicate the descriptive and most representative words that are included within the document and the frequency of appearance (i.e. number of occurrences) of each word. For instance, if a document \( D_i \) includes the words \( a, b, c \) and \( d \) with frequency 3, 2, 8 and 6, respectively, then its document vector will be: \( DV_i = [3a, 2b, 8c, 6d] \). The dimension of \( DV_i \) equals the number of individual words it includes (for the previous example it is \( DV_i = 4 \)) and varies for each document.

Next, we re-order (using the well-known ‘quick sort’ algorithm) each \( DV_i \) in descending order of words frequencies, that is we move the words with higher frequencies to the beginning of the \( DV_i \), so the vector of the previous example becomes: \( DV_i = [8c, 6d, 3a, 2b] \).

Finally, we filter the \( DV_i \), maintaining only a specific number of \( T \) words, so that all \( DV_i \)'s are of equal dimension. Thus, for \( T=2 \), the vector of the previous example becomes: \( DV_i = [8c, 6d] \). The filtering excludes some information, since we have no knowledge of which words with small frequencies are included in each document. Yet, it is necessary to improve the performance of the next step of our algorithm. If no filtering is applied, then the worse case scenario for a dictionary of \( W \) words is a \( W \)-dimensions \( DV_i \), where each word of the dictionary appears only once in a document.

3.4. Calculation of the Document Vectors Distance Matrix

We calculate the distance matrix \( DM \) of the parsed web documents. Each scalar element \( DM_{ij} \) of this matrix equals the weighted Hamming distance between \( DV_i \) and \( DV_j \): \( WH(DV_i, DV_j) \), which is an alternative of the well-known Hamming distance (defined in the Section 3.4.1) and represents the dissimilarity of the words included within the corresponding documents, i.e. the more different the words (and their frequencies) of documents \( D_i \) and \( D_j \), the higher the \( WH \) value. Hence, for \( N \) documents, the distance matrix is depicted in Table 1.

Table 1. An example of the distance matrix for \( N \) document vectors.

<table>
<thead>
<tr>
<th>( WH(DV_1, DV_1) )</th>
<th>( WH(DV_1, DV_2) )</th>
<th>( WH(DV_1, DV_3) )</th>
<th>( \ldots )</th>
<th>( WH(DV_1, DV_N) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( WH(DV_2, DV_1) )</td>
<td>( WH(DV_2, DV_2) )</td>
<td>( WH(DV_2, DV_3) )</td>
<td>( \ldots )</td>
<td>( WH(DV_2, DV_N) )</td>
</tr>
</tbody>
</table>
The Weighted Hamming Distance

The Hamming distance is a distance between two strings $S_1 = \{x_1x_2...x_n\}$ and $S_2 = \{y_1y_2...y_n\}$ of equal length $n$, defined as [8]:

$$H = \sum_{i=1}^{n} |x_i - y_i|$$

(1)

Hamming distance is referred to as an appropriate distance metric for error detection and correction codes and has also been used for defining similarity measures among web pages or web links [11],[18],[25]. However, we argue that it is an inappropriate metric for measuring the similarity between web pages content as it can produce quite unreliable results. Let, for example, two document vectors $DV_1 = [3a, 4b, 2c]$ and $DV_2 = [3a, 4b, 8c]$. Their hamming distance is $|3-3|+|4-4|+|8-2|=6$, as much as the hamming distance between $DV_3 = [a, b, c]$ and $DV_4 = [d, e, f]$ vectors, which do not have any common words though. To address this issue, we introduce the ‘weighted hamming’ distance (WH) metric, defined as follows:

$$WH\{x, y\} = n \cdot \sum_{i=1}^{n} w_i |x_i - y_i|$$

(2)

where the weighted variable $w_i$ is used so as the words that exclusively appear either to the document vector $x$ or $y$ to contribute more to the actual distance of their corresponding document vector (likewise, words that exist in both document vectors, even with different frequencies, are assigned smaller weight values). The weighted variable is given by:

$$w_i = \begin{cases} c, & x_i \in S_2 \text{ or } y_i \in S_1 \\ 1, & x_i \not\in S_2 \text{ and } y_i \not\in S_1 \end{cases}$$

(3)

where $c$ is a constant: $c = [0, 1)$. If, for instance, $c = 0.5$, then for the abovementioned example document vectors (for $n=1$, where $n$ is defined below), the $WH\{DV_1, DV_2\} = 3$ while $WH\{DV_3, DV_4\} = 6$. It is noted that the term ‘weighted Hamming distance’ has been used in scientific and several application domains (e.g. in [24]), under diverse definitions.

Special attention should be paid to ensure a fair treatment of web documents with different text lengths. Otherwise, document vectors of relatively long documents are expected to exhibit larger word frequencies over those of shorter documents, inevitably resulting in large $WH$ values. Therefore, document vectors should be normalized prior to performing the clustering procedure; the normalization factor $n$, appearing in (2), is given by:

$$n = \frac{1}{|x| - |y|}$$

(4)
where $|x|$ and $|y|$ denote the length of document vectors $x$ and $y$, respectively. Namely, the $WH$ among two documents is conversely proportional to the difference among their respective text length.

4. Clustering Process

Here, we first use the concept of the potential of a numerical data point (introduced by Chiu in [5]) and modify it, to process the non-numerical document vectors and the respective distances. Secondly, to overcome certain problems related to the determination of the optimal partition, we introduce a cluster validity index.

Let $X = \{DV_1, DV_2, ..., DV_N\}$ be a set of $N$ document vectors of equal dimension. The potential of the $i$-th document vector is defined as follows,

$$Z_i = \sum_{j=1}^{N} S_{ji} \cdot \left(1 \leq i \leq N\right)$$

(5)

where $S_{ji}$ is the similarity measure between $DV_j$ and $DV_i$ given as follows,

$$S_{ji} = \exp\left[-\alpha \cdot d(DV_j, DV_i)\right] \quad \alpha \in (0, 1)$$

(6)

From eqs (5) and (6) the following useful remark can be easily observed:

A document vector that appears a high potential value is surrounded by many neighboring document vectors. Therefore, a document vector with a high potential value represents a good nominee to be a cluster center.

Based on this remark the potential-based clustering algorithm is described as follows,

**Potential-Based Clustering Algorithm**

Select values for the design parameters $\alpha \in (0, 1)$ and $\beta \in (0, 1)$. Initially, set the number of clusters equal to $n=0$.

**Step 1.** Using eqs (5) and (6) and the distance matrix calculate the potential values for all document vectors $DV_i \ (1 \leq i \leq N)$.

**Step 2.** Set $n=n+1$.

**Step 3.** Calculate the maximum potential value:

$$Z_{\text{max}} = \max_{1 \leq i \leq N} \{Z_i\}$$

(7)
Select the document vector \( DV_{i_{\text{max}}} \) that corresponds to \( Z_{\text{max}} \) as the center element of the \( n \)th cluster: \( C_n = DV_{i_{\text{max}}} \).

**Step 4.** Remove from the set \( X \) all the document vectors having similarity (given in eq. (6)) with \( DV_{i_{\text{max}}} \) greater than \( \beta \) and assign them to the \( n \)-th cluster.

**Step 5.** If \( X \) is empty stop. Else turn the algorithm to step 2.

The above detailed algorithm exhibits two appealing features:

- It is one-pass through the data set and therefore it is a very fast procedure that is easy to implement.
- It does not assume any random selection of initial cluster centers but rather it selects these centers based on the structure of the data set itself.

However, a straightforward implementation of the above algorithm requires a priori knowledge of the appropriate values for \( \alpha \) and \( \beta \). A feasible way to solve this problem is to perform a number of simulations and to detect the set of values that provide the best partitioning results. However, this is a very time consuming procedure. Interestingly, a set of simulations revealed no significant difference when we kept \( \alpha \) constant and modified \( \beta \). Therefore, to simplify the approach, we set \( \alpha = 0.5 \) and we calculated the optimal value of \( \beta \) by using the validity index described next.

An efficient cluster validity index is synthesized by two basic components [21]: (a) a compactness measure \((COMP)\) that concludes whether the individual clusters are solid (i.e. compact) or not and (b) a separation measure \((SEP)\) that concludes whether the clusters are well-separated or not.

Normally, an optimal partition corresponds to a minimum compactness and a maximum separation. Therefore, the index is given as the ratio between \( COMP \) and \( SEP \). A feasible estimation for the compactness is:

\[
COMP = \sum_{k=1}^{n} Z_k^C \tag{8}
\]

where \( Z_k^C \) is the potential value for the potential value of the \( k \)th cluster center. Note that the cluster centers are document vectors from the original data set. If \( COMP \) is assigned a small value then each of the sum components in (8) are small and therefore the resulting clusters are compact (i.e. solid).

On the other hand, the separation measure is given by the next relation:

\[
SEP = g\left(\min_{i,j} d(C_i, C_j)\right) \tag{9}
\]

where \( g \) is a strictly increasing function of the form:
In the above equation, the parameter $q$ is used to normalize the separation measure so as to cancel undesired effects associated with the number of document vectors and the number of clusters, as well. Thus, the validity index is defined as:

$$V = \frac{COMP}{SEP}$$

To this end, the main objective is to select the value of the parameter $\beta$ for which $V$ is minimized.

5. Experimental Evaluation

To test the applicability and performance of our algorithm, we have created and uploaded on a web server a ‘test’ web site comprising hundreds of xHTML documents. Each document includes variable number of links to other documents, therefore creating a web of inter-connected documents.

In order to validate our proposed method, we needed to automate the creation of large numbers of web pages used. Hence, we developed a Java-based tool; through its simple user interface shown in Figure 4a, the user sets: the number of documents to be generated; the number of words (excluding xHTML tags) per document; the average percentage of cultural terms (the actual number of such terms will determine whether the document will be classified as cultural-related or not); the dictionaries where generic and cultural terms are extracted from.

The tool uses an xHTML skeleton, i.e. the start and the end tags of the xHTML document, and fills its body section with words randomly retrieved by the dictionaries and also with a link to the page generated on the next round (see Figure 4b). In our experiments, we created 1000 pages, each containing 100 words retrieved from dictionaries of 200 generic and 100 cultural terms. Certainly, some of these words may appear repeatedly within each document.
Figure 4. (a) User Interface and (b) function of the xHTML document generator.

For the filtering of the 1000 document vectors, we have used a common threshold value $T=10$. Target topics were defined and the page samples were obtained through meta-searching the Yahoo search engine. We choose the following 15 web page categories related to cultural information:

1. Cultural conservation
2. Cultural heritage
3. Painting
4. Sculpture
5. Dancing
6. Cinematography
7. Architecture Museum
8. Archaeology
9. Folklore
10. Music
11. Theatre
12. Cultural Events
13. Audiovisual Arts
14. Graphics Design

15. Art History

Table 2. Number of clusters obtained by the algorithm for each category and the respective optimal value for the parameter $\beta$.

<table>
<thead>
<tr>
<th>Category</th>
<th>Number of Clusters</th>
<th>Optimal Value of $\beta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cultural conservation</td>
<td>31</td>
<td>0.3</td>
</tr>
<tr>
<td>Cultural heritage</td>
<td>46</td>
<td>0.3</td>
</tr>
<tr>
<td>Painting</td>
<td>34</td>
<td>0.2</td>
</tr>
<tr>
<td>Sculpture</td>
<td>26</td>
<td>0.4</td>
</tr>
<tr>
<td>Dancing</td>
<td>28</td>
<td>0.6</td>
</tr>
<tr>
<td>Cinematograph</td>
<td>53</td>
<td>0.7</td>
</tr>
<tr>
<td>Architecture Museum</td>
<td>27</td>
<td>0.4</td>
</tr>
<tr>
<td>Archaeology</td>
<td>38</td>
<td>0.3</td>
</tr>
<tr>
<td>Folklore</td>
<td>36</td>
<td>0.6</td>
</tr>
<tr>
<td>Music</td>
<td>55</td>
<td>0.5</td>
</tr>
<tr>
<td>Theatre</td>
<td>53</td>
<td>0.5</td>
</tr>
<tr>
<td>Cultural Events</td>
<td>31</td>
<td>0.2</td>
</tr>
<tr>
<td>Audiovisual Arts</td>
<td>34</td>
<td>0.3</td>
</tr>
<tr>
<td>Graphics Design</td>
<td>29</td>
<td>0.4</td>
</tr>
<tr>
<td>Art History</td>
<td>27</td>
<td>0.2</td>
</tr>
</tbody>
</table>

For each category, we downloaded 1000 web pages to train the algorithm. After generating the dictionary (refer to Section 3.2), for each category we selected the 200 most frequently reported words, using the inverse document frequency (IDF) for each word. The IDF is given by the following equation [7],

$$ \text{IDF}(w) = \log \left( \frac{P}{P_w} \right) $$

(12)

where $f_w$ is the frequency of occurrence of the word in the category’s document collection, $f_{w_{\text{max}}}$ is the maximum frequency of occurrence of any word in the category’s collection, $P$ is the number of documents of the whole collection, and $P_w$ is the number of documents that include this word.
In the next step, we defined the dimension of each document vector as $T=30$. Note that, we can keep the feature space dimension large, but in this case the computational cost will increase, while the creditability of the results remains questionable.

The optimal numbers of clusters for each category along with the respective values of the parameter $\beta$ are shown in Table 2. To carry out these simulations we used for each topic nine values for $\beta$ namely: 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9.

Table 3. Comparative classification results with respect to the Harvest Ratio.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cultural conservation</strong></td>
<td>48.7%</td>
<td>65.3%</td>
<td>68.9%</td>
<td>69.6%</td>
</tr>
<tr>
<td><strong>Cultural heritage</strong></td>
<td>52.1%</td>
<td>72.4%</td>
<td>73.7%</td>
<td>77.2%</td>
</tr>
<tr>
<td><strong>Painting</strong></td>
<td>70.6%</td>
<td>72.5%</td>
<td>76.4%</td>
<td>77.1%</td>
</tr>
<tr>
<td><strong>Sculpture</strong></td>
<td>52.0%</td>
<td>67.2%</td>
<td>71.6%</td>
<td>72.1%</td>
</tr>
<tr>
<td><strong>Dancing</strong></td>
<td>66.8%</td>
<td>73.8%</td>
<td>88.8%</td>
<td>90.6%</td>
</tr>
<tr>
<td><strong>Cinematograph</strong></td>
<td>67.4%</td>
<td>84.7%</td>
<td>90.2%</td>
<td>85.4%</td>
</tr>
<tr>
<td><strong>Architecture Museum</strong></td>
<td>55.4%</td>
<td>50.5%</td>
<td>78.3%</td>
<td>72.1%</td>
</tr>
<tr>
<td><strong>Archaeology</strong></td>
<td>59.7%</td>
<td>59.8%</td>
<td>80.0%</td>
<td>84.9%</td>
</tr>
<tr>
<td><strong>Folklore</strong></td>
<td>60.8%</td>
<td>64.9%</td>
<td>82.5%</td>
<td>84.0%</td>
</tr>
<tr>
<td><strong>Music</strong></td>
<td>65.2%</td>
<td>85.4%</td>
<td>93.0%</td>
<td>93.4%</td>
</tr>
<tr>
<td><strong>Theatre</strong></td>
<td>71.5%</td>
<td>87.2%</td>
<td>91.5%</td>
<td>92.0%</td>
</tr>
<tr>
<td><strong>Cultural Events</strong></td>
<td>58.8%</td>
<td>74.0%</td>
<td>90.6%</td>
<td>87.1%</td>
</tr>
<tr>
<td><strong>Audiovisual Arts</strong></td>
<td>63.3%</td>
<td>68.4%</td>
<td>82.8%</td>
<td>84.7%</td>
</tr>
<tr>
<td><strong>Graphics Design</strong></td>
<td>68.8%</td>
<td>69.0%</td>
<td>78.6%</td>
<td>82.9%</td>
</tr>
<tr>
<td><strong>Art History</strong></td>
<td>48.7%</td>
<td>59.6%</td>
<td>60.9%</td>
<td>59.0%</td>
</tr>
</tbody>
</table>

To test the method we can use the extracted cluster centers (i.e. document vectors) to automatically classify unknown web documents. More specifically, for each incoming unknown document we determine the minimum distance from all the clusters in a specific category, obtaining in this way 15 distances. Then, the document is assigned to the category that appears the smallest of the above 15 distances. Thus, we downloaded another 1000 pages for each category and we utilized the well-known Harvest Ratio [6],[4], [14], [23] to validate the classification results.

For comparative purposes we also trained and tested three more methods found in the literature namely: (a) the best-first algorithm, (b) the accelerated focused crawler [4], and (c) the first-order focused crawler [23]. The results of the above simulations are presented in Table 3. From this ta-
ble, we clearly verify that except the categories “Cinematography”, “Architecture Museum”, “Cultural Events” and “Art History” (that is in 4 from the 15 categories) our method outperformed all the alternative methods.

6. Conclusions

We have shown how cluster analysis can be effectively and efficiently incorporated into a focused web crawler. The basic idea of the approach is to create multidimensional document vectors each of which corresponds to a specific web page. The dissimilarity between two distinct document vectors is measured using a variation of the well-known Hamming distance. Then, we use a clustering algorithm to classify the set of all document vectors into a number of clusters, where the respective cluster centers are objects from the original data set that satisfy specific conditions. The classification of unknown web pages is accomplished by using the minimum weighted Hamming distance. Several experimental simulations took place, which verified the efficiency of the proposed method.

As a further step of the approach, we intend to modify the filtering of document vectors, so that the threshold value $T$ will not be predefined but dynamically determined, based on the documents’ word frequencies variance, which implies that the threshold value will vary among separate documents.

Finally, future extensions of our web crawler will enable a more intelligent focus of our crawler on culture-related web content through applying sophisticated techniques such as reinforcement learning and evolutionary adaptation (as suggested in [13]) so as to improve its performance over longer crawls.

References


