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

George E. Tsekouras and Damianos Gavalas*

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

gtsek@ct.aegean.gr, dgavalas@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 focused crawler is a web crawler that searches and processes 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 dissimilarity 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 dissimilarity, similarity measure.

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 high speed. 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

* Corresponding Author

[1]. 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 [12, 24]. When searching for further relevant web pages, the focused crawler starts from the given pages and recursively explores the linked web pages [17, 38]. 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 [24, 38, 40].

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. 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, 4, 9, 18, 20, 29]. These collections comprise sets of documents strongly related with a specific class [14, 16, 23, 25, 26]. 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, 18, 22, 25]: (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.

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) a multidimensional document vector comprising the most frequent word occurrences; (b) calculation of the weighted Hamming dissimilarity measures 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 presents related work in the field. Section 3 describes the crawling and documents parsing and analysis procedure. Section 4 analyzes the clustering procedure. Section 5 discusses the results of the experimental validation of our algorithm and Section 6 concludes our work and presents directions for future work.

2. Related Work

Lin et *al*. [17] 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*. [30] 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. In [2], Chakrabarti et *al*. showed that there is a great deal of usable information on a 'href' source page about the relevance of the target page. This information, encoded suitably, can be exploited by a supervised apprentice, which takes online lessons from a traditional focused crawler by observing a carefully designed set of features and events associated with the crawler. Huang and Ye [12] developed a focused web crawler called wHunter that implements incremental and multi-strategy learning by taking the advantages of both support vector machines and naïve Bayes theory. Tao et *al*. [31] pointed out that since a driven crawler chooses the best URLs and relevant pages to pursue during Web crawling, it is difficult to deal with irrelevant pages. In order to deal with this problem they introduced a focused crawler, which used specialized metrics and an evaluation function for ranking the pages' relevance. A different framework to take into account the relevancy of documents was proposed in [38]. Specifically, the crawler retrieved documents by speculating the relevancy of the document based on the keywords in the link and the surrounding text of the link. Then, the relevancy of the documents is reckoned measuring the semantic similarity between the keywords in the link and the taxonomy hierarchy of the specific domain. Jalilian and Khotanlou [13], developed a method to weigh the concepts related to topic to be used as a main component in the architecture of semantic crawler to compute relevance of web page with the topic. The concepts related to the topic are retrieved by ontology graph and their weights are computed by a proposed fuzzy inference system. In [35], Xu and Zuo proposed a first-order focused web crawler to handle the predicates that are used explicitly to represent the relevance clues of the unvisited pages in the crawl frontier and then, to perform a classification of these pages using rules induced in terms of a subgroup discovery technique. Ye et *al*. [36] developed the iSurfer focused crawler that uses an incremental method to learn a page classification model and a link prediction model. It employs an online sample detector to incrementally distill new samples from crawled pages for online updating of the model learned. The incremental learning strategy starts from a few positive samples and gains more integrated knowledge about the target topics over time. In [40] was discussed the problem of embedding reinforcement learning in a focused web crawler. The result was an algorithmic scheme according to which the reinforcement learning was combined with naive Bayes classifiers and fuzzy center-averaged clustering to online estimate the immediate action reward and classification of the newly crawled web pages. Hati et *al*. [11] employed a vector space model to quantify the relevancy score. Then, they calculated the un-

visited URL score based on its anchor text relevancy, its description in Google search engine and calculated the similarity score of description with topic keywords, cohesive text similarity with topic keywords and relevancy score of its parent pages. To this end, they came up with an effective adaptive focused crawler. In [8], Dong and Hussain proposed a conceptual framework for implementing semantic focused crawling processes for automatically discovering, annotating, and classifying the service information of digital ecosystems with the semantic web technologies. In [19], Mali and Meshram proposed a trhee-stages learning process, which focus on page selection policy and page revisit policy. Dey et *al*. [7] developed a focused crawler for crawling of country-based financial data. Hati and Kumar [10] quantified the relevancy between seed page and child page by vector space model in terms of a URL distance model. Yu et *al*. [37] developed an algorithm based on similarity computation, which is able to handle pages of multimedia and text content. Martin and Khelif [22] proposed a source selection process to filter out pages that are not relevant to the target one, and used a state space model to measure the relevance of the pages.

Clustering appears to be a reliable solution for web document classification [18]. 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 [26]. 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) [27, 28, 34, 39]. 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 [3, 6, 15, 32]. 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 [21], and (b) web server log files containing access sequences of web pages visited by specific users [25]. Lee and On [15], in order to avoid certain problems related to the use of partitional clustering, developed an agglomerative clus-

tering bisection and merging process. Although this method appears to be effective, it constitutes a highly computationally complex process. In [3], Chen generated candidate clusters based on connective semantic relations among the clusters. The final algorithmic scheme can be applied to multilingual web documents in an unsupervised learning fashion. Tan et *al*. [30] applied a clustering based sampling approach, according to which all local web pages are assigned into different clusters such that each cluster contains web pages with similar change pattern. Then, they sampled web pages from each cluster to estimate the change frequency of all the web pages in that cluster and let the crawler to re-visit the cluster containing web pages with higher change frequency with a higher probability.

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 1. Our algorithm is described in detail in the following subsections.

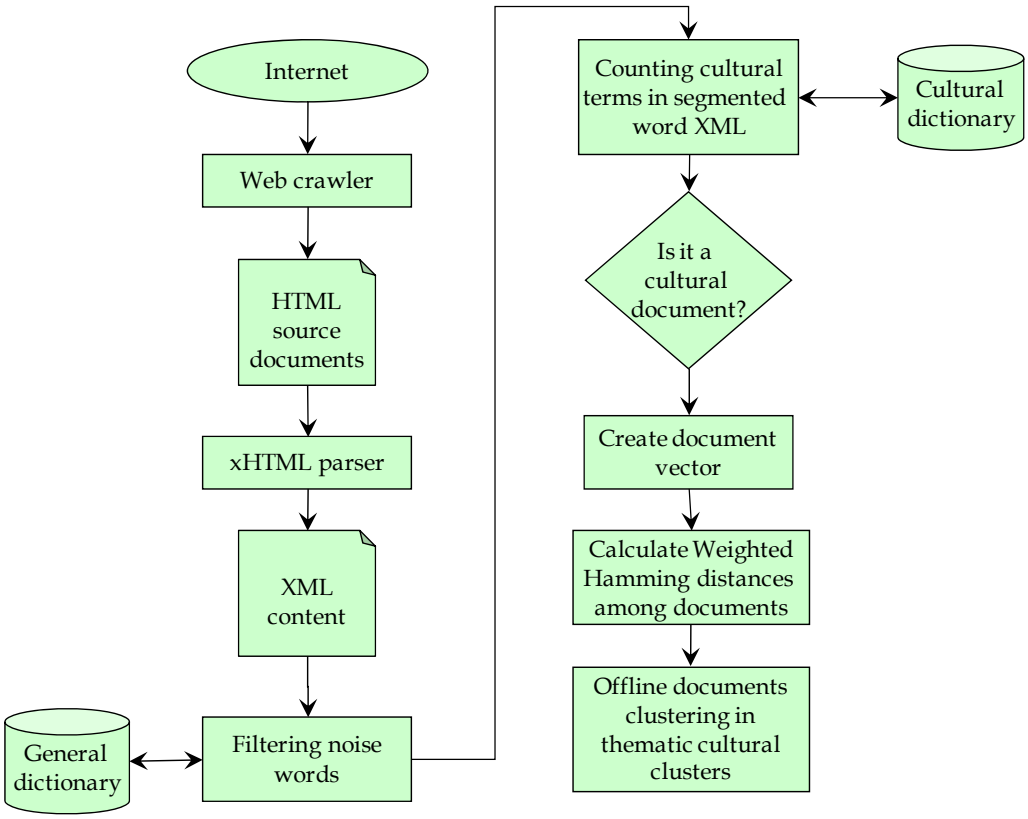


Figure 1. The high-level process of crawling, parsing, filtering and clustering of the downloaded web pages.

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 [38] wherein only links with anchor text containing or surrounded by cultural terms are assumed to point to culture-related documents. We used standard HTTP connections for retrieving the code of xHTML documents (java.net.URLConnection class).

3.2. Parsing of xHTML Documents

Each retrieved document is parsed using an open-source Java-based xHTML parser [33]. 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 DV_i includes the words $a, b, c,$ and d with frequencies 3, 2, 8 and 6, respectively, then its document vector will be: $DV_i = [3a, 2b, 8c, 6d]$. The dimension (i.e the length) of DV_i is designated as $|DV_i|$ and is equal to 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_is are of equal dimension (i.e of equal length). 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 document vectors. Each scalar element DM_{ij} of this matrix equals the weighted Hamming dissimilarity between DV_i and DV_j: $WH(DV_i, DV_j)$, which is based on 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 document vectors DV_i and DV_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.

$WH(DV_1, DV_1)$	$WH(DV_1, DV_2)$	$WH(DV_1, DV_3)$	.....	$WH(DV_1, DV_N)$
$WH(DV_2, DV_1)$	$WH(DV_2, DV_2)$	$WH(DV_2, DV_3)$	……	$WH(DV_2, DV_N)$
.....	.....	.....	.....	.....
$WH(DV_N, DV_1)$	$WH(DV_N, DV_2)$	$WH(DV_N, DV_3)$	.....	$WH(DV_N, DV_N)$

3.4.1. The Weighted Hamming Dissimilarity Measure

The Hamming distance is a distance between two document vectors $DV_1 = [x_1, x_2, ..., x_n]$ and $DV_2 = [y_1, y_2, ..., y_n]$ of equal length *n*, defined as:

$$H(DV_1, DV_2) = \sum_{i=1}^{n} |x_i - y_i| \quad (1)$$

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 [20, 28, 36]. 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 dissimilarity measure (WH), defined as follows:

$$WH(DV_1, DV_2) = \gamma \sum_{i=1}^{n} w_i |x_i - y_i| \tag{2}$$

where $n = \min\{|DV_1|, |DV_2|\}$ if $|DV_1| \neq |DV_2|$ else $n = |DV_1| = |DV_2|$, γ is defined below in eq. (4), while the weighted variable w_i is a constant used so as the words that exclusively appear either to the document vector DV_1 or DV_2 to contribute more to the actual distance between them (likewise, words that exist in both document vectors, even with different frequencies, are assigned smaller weight values). The weighted variable is given by:

$$w_i = \begin{cases} c, x_i \in DV_2 \text{ or } y_i \in DV_1 \\ 1, x_i \notin DV_2 \text{ and } y_i \notin DV_1 \end{cases} \tag{3}$$

where c is a constant such that $c = [0, 1)$. If, for instance $c = 0.5$ then for the abovementioned example document vectors (for $\gamma = 1$), the $WH(DV_1, DV_2) = 3$ while $WH(DV_3, DV_4) = 6$. It is noted that the weighted Hamming dissimilarity measure has been used in scientific and several application domains (e.g. in [34]), under diverse definitions.

Special attention should be paid to ensure a fair treatment of web documents with different text lengths. Otherwise, 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 γ, appearing in (2), is given by:

$$\gamma = \begin{cases} 1 \quad , \ if \ |DV_1| = |DV_2| \\ \left| |DV_1| - |DV_2| \right|, \ if \ |DV_1| \neq |DV_2| \end{cases} \tag{4}$$

Thus, the WH value among two documents is proportional to the difference among their respective length. Notice that, although the equations (2)-(4) can be applied in the general case where the document vectors are of different lengths, in our case the parameter γ is equal to 1 since all vectors appear to have the same 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. Sec-

ondly, 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 length (i.e. dimension). The potential of the i-th document vector is defined as follows,

$$Z_i = \sum_{j=1}^{N} S_{ji} \tag{5}$$

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

$$S_{ji} = \exp\left\{-\alpha \cdot WH\left(DV_j, DV_i\right)\right\}, \qquad \alpha \in (0,1) \tag{6}$$

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 $\beta \in (0,1)$. Initialy, set the number of clusters equal to L=0.

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

Step 2. Set L=L+1.

Step 3. Calculate the maximum potential value:

$$Z_{\max} = \max_{1 \le i \le N}\{Z_i\} \tag{7}$$

Select the document vector $DV_{i_{\max}}$ that corresponds to $Z_{\max}$ as the center element of the L^{th} cluster: $C_L = DV_{i_{\max}}$.

Step 4. Remove from the set X all the document vectors having similarity (given in eq. (6)) with $DV_{i_{\max}}$ greater than β and assign them to the L^{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 α and β. 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 α constant and modified β. Therefore, to simplify the approach, we set $\alpha = 0.5$ and we calculate the optimal value of β by using the validity index described next.

An efficient cluster validity index is synthesized by two basic components [32]: (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:

$$COMP = \sum_{k=1}^{L} Z_k^C \tag{8}$$

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:

$$SEP = g\left(\min_{i,j}\left\{ WH(C_i, C_j)\right\}\right) \tag{9}$$

where *g* is a strictly increasing function of the form:

$$g(x) = x^q,\ q \in (1, \infty) \tag{10}$$

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:

$$V = \frac{COMP}{SEP} \tag{11}$$

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

5. Experimental Evaluation

To evaluate the applicability and the 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.

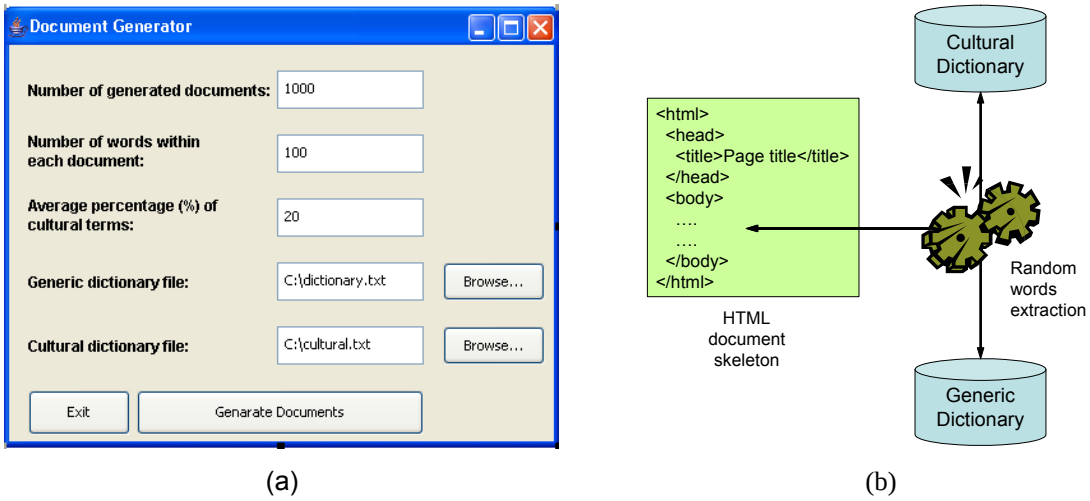


Figure 2. (a) User Interface and (b) functioning of the xHTML document generator.

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 2a, the user sets: (i) the number of documents to be generated; (ii) the number of words (excluding xHTML tags) per document; (iii) 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); and (iv) 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 2b).

In our experiments, we created 30,000 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.

Table 2. The web page categories for cultural information.

1	*Cultural conservation*	*9*	*Folklore*
2	*Cultural heritage*	*10*	*Music*
3	*Painting*	*11*	*Theatre*
4	*Sculpture*	*12*	*Cultural Events*
5	*Dancing*	*13*	*Audiovisual Arts*
6	*Cinematography*	*14*	*Graphics Design*
7	*Architecture Museum*	*15*	*Art History*
8	*Archaeology*		

As regards the filtering of the document vectors, we have used a common threshold value T=30. Notice that this quantity defines the dimensionality of the feature space. We can keep the feature space dimension large, but in this case the computational cost will increase, while the creditability of the results will remain questionable.

Target topics were defined and the page samples were obtained through meta-searching the Yahoo search engine. Table 2 depicts the 15 web page categories related to cultural information.

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 [1, 32]:

$$fr_{IDF} = \frac{f_w}{f_{w\_\max}} \log\left(\frac{P}{P_w}\right) \tag{12}$$

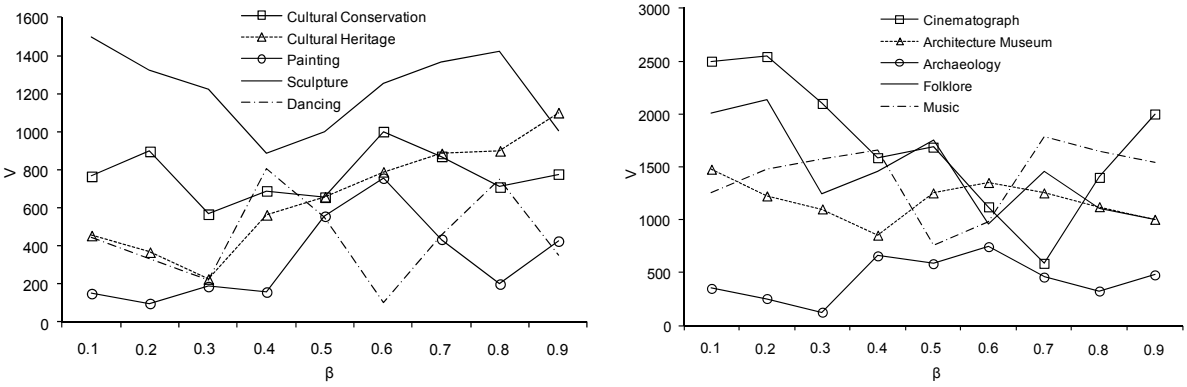
where f_w is the frequency of occurrence of the word in the category's document collection, $f_{w\_\max}$ is the maximum frequency of occurrence of any word in the category's collection, P is the number of documents of the whole collection, and P_w is the number of documents that include this word. Finally, the stop words are removed, while for each document vector we preselect the most frequently reported words in the corresponding document. In the next subsections we analytically describe the conducted experiments.

5.1. Study of the Effect of the Parameter β

In this experiment, we quantify the effect of the parameter β upon the learning performance of the proposed crawler. As it can be easily seen, this parameter is highly important for the overall behavior of the algorithm.

To perform the experiment we employed the crawler on each of the above categories. Thus, the algorithm elaborated 1000 documents with respect to each category. To obtain strong quantitative and qualitative results, we monitored the value of the validity index V given in eq. (11) as a function of the parameter β. To carry out these simulations we used for each topic nine values for β,

namely: 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9. The results are depicted in Figure 3. Since the objective is to minimize the index V, the selected values of the parameter β are those that correspond to minimal values of V. The resulting number of clusters along with the optimal values of β are given in Table 3.



(a) (b)

Figure 3. The validity index V as a function of the parameter β for several cultural information categories.

Table 3. Number of clusters obtained by the algorithm for each category and the respective optimal value for the parameter β.

Category	Number of Clusters	Optimal Value of β	Category	Number of Clusters	Optimal Value of β
1	31	0.3	9	36	0.6
2	46	0.3	10	55	0.5
3	34	0.2	11	53	0.5
4	26	0.4	12	31	0.2
5	28	0.6	13	34	0.3
6	53	0.7	14	29	0.4
7	27	0.4	15	27	0.2
8	38	0.3			

5.2. Classification Capabilities Study

To test the method we used 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 Rate [2, 12, 24, 35] to validate the classification results.

Table 4. Comparative classification results for different crawlers with respect to the Harvest Rate.

Category	BFS	AFWC	FOC	wHunter	CBWC	iSurfer	AdFWC	Proposed
1	48.7%	65.3%	68.9%	69.1%	63.2%	60.3%	70.7%	69.6%
2	52.1%	72.4%	73.7%	75.0%	72.2%	82.6%	70.9%	77.2%
3	70.6%	72.5%	76.4%	74.2%	75.8%	70.0%	82.0%	77.1%
4	52.0%	67.2%	71.6%	69.8%	69.2%	59.0%	85.0%	72.1%
5	66.8%	73.8%	88.8%	88.5%	80.3%	89.1%	88.4%	90.6%
6	67.4%	84.7%	90.2%	91.6%	78.6%	88.1%	80.2%	85.4%
7	55.4%	50.5%	78.3%	76.2%	56.2%	70.2%	79.5%	72.1%
8	59.7%	59.8%	80.0%	77.4%	79.2%	69.4%	90.5%	84.9%
9	60.8%	64.9%	82.5%	72.6%	83.3%	73.0%	89.5%	84.0%
10	65.2%	85.4%	93.0%	80.5%	90.2%	70.9%	91.6%	93.4%
11	71.5%	87.2%	91.5%	87.7%	95.9%	93.0%	90.8%	92.0%
12	58.8%	74.0%	90.6%	91.6%	92.0%	79.9%	90.3%	87.1%
13	63.3%	68.4%	82.8%	88.0%	82.8%	74.7%	82.9%	84.7%
14	68.8%	69.0%	78.6%	83.5%	78.8%	84.4%	91.7%	82.9%
15	48.7%	59.6%	60.9%	61.9%	54.0%	57.4%	56.7%	59.0%

In order to extract safe conclusions about the classification accuracy of the crawler, we have also designed and applied the following crawlers: (a) Best-First Search (BFS), (b) Accelerated Focused Web Crawler (AFWC) [2], (c) First-Order Crawler (FOC) [35], (d) wHunter [12], (e) Clustering-Based Web Crawler (CBWC) [30], (f) iSurfer [36], and (g) Adaptive Focused Web Crawler (AdFWC) [11]. Table 4 depicts the obtained results, which are highly convincing since in most of the cases the proposed crawler outperformed the others, maintaining a highly effective performance.

5.3. Quality Study of the Resulting Clusters

To conduct the experiment we compare the proposed algorithm to the following clustering-based crawlers: (a) Clustering-Based Web Crawler (CBWC) [30], (b) Fuzzy Logic Categorical Data Clustering (FLCDC) [32], and (c) Clustering based on Document Structure (CDS) [6]. The comparison is carried out in terms of the quality of the resulting clusters, which is evaluated by the *F*-measure and the Entropy [6, 30]. Within the next paragraphs we briefly report the appropriate definitions and properties of these quantities.

Let us assume that the correct classification of the set of web documents consists of the clusters $C_1, C_2, ..., C_M$ and the clusters computed by the model are denoted as $\tilde{C}_1, \tilde{C}_2, ..., \tilde{C}_{\tilde{M}}$. To this end, the concepts related to the definition of the above measures are the precision (Pr) and recall (Re), which are calculated in terms of the following equations,

$$\Pr(C_i, \tilde{C}_j) = \frac{\left|C_i \cap \tilde{C}_j\right|}{\left|\tilde{C}_j\right|} \tag{13}$$

$$\mathrm{Re}(C_i, \tilde{C}_j) = \frac{\left|C_i \cap \tilde{C}_j\right|}{\left|C_i\right|} \tag{14}$$

An effective way to calculate the quantities P_r and R_e is to use the confusion matrix, the elements of which are the counts of the documents (i.e. pages) from the cluster C_i that have been assigned to the clusters $\tilde{C}_j$. By denoting the confusion matrix as $P = [p_{ij}]$, the P_r and R_e are practically estimated as follows,

$$\Pr(C_i, \tilde{C}_j) = \frac{p_{ij}}{\sum_{l=1}^{\tilde{M}} p_{il}} \tag{15}$$

$$\mathrm{Re}(C_i, \tilde{C}_j) = \frac{p_{ij}}{\sum_{l=1}^{M} p_{lj}} \tag{16}$$

Taking into account the above results, the *F*-measure is defined as:

$$F(C_i, \tilde{C}_j) = \frac{2\Pr(C_i, \tilde{C}_j)\mathrm{Re}(C_i, \tilde{C}_j)}{\Pr(C_i, \tilde{C}_j) + \mathrm{Re}(C_i, \tilde{C}_j)} \tag{17}$$

Table 5. Comparison of different clustering algorithms with respect to the measure F_T.

Category	**CDS [6]**	**CBWC [30]**	**FLCDC [32]**	**Proposed**
1	0.895	0.821	0.847	0.916
2	0.887	0.848	0.745	0.943
3	0.957	0.893	0.769	0.786
4	0.743	0.978	0.795	0.775
5	0.923	0.932	0.856	0.923
6	0.785	0.958	0.896	0.963
7	0.869	0.963	0.956	0.976
8	0.777	0.712	0.702	0.825
9	0.912	0.705	0.743	0.886
10	0.935	0.761	0.712	0.936
11	0.958	0.802	0.895	0.945
12	0.985	0.813	0.995	0.986
13	0.975	0.693	0.974	0.978
14	0.792	0.863	0.765	0.702
15	0.902	0.896	0.736	0.939

Table 6. Comparison of different clustering algorithms with respect to the measure E_T.

Category	CDS [6]	CBWC [30]	FLCDC [32]	Proposed
1	0.0062	0.0047	0.0125	0.0025
2	0.0756	0.0489	0.0654	0.0069
3	0.0965	0.0365	0.1258	0.0584
4	0.0029	0.0425	0.0025	0.0096
5	0.0092	0.0125	0.0698	0.0015
6	0.0489	0.0089	0.0888	0.0058
7	0.0664	0.0032	0.0935	0.0061
8	0.0784	0.0045	0.0175	0.0098
9	0.0964	0.0854	0.0069	0.0074
10	0.0063	0.0087	0.0074	0.0265
11	0.0085	0.0689	0.0332	0.0044
12	0.0102	0.0574	0.0895	0.0172
13	0.0099	0.0269	0.0147	0.0111
14	0.0455	0.0125	0.0225	0.0085
15	0.0247	0.0655	0.0121	0.0065

Then, the *F*-measure for the correct class C_i is defined as:

$$F_i = \max_{1 \le j \le \tilde{M}} \left(\frac{2 \Pr(C_i, \tilde{C}_j) \operatorname{Re}(C_i, \tilde{C}_j)}{\Pr(C_i, \tilde{C}_j) + \operatorname{Re}(C_i, \tilde{C}_j)} \right) \tag{18}$$

Finally, for the whole partition the total *F*-measure is given as the weighted aggregate of all the F_i measures:

$$F_T = \sum_{i=1}^{M} \left(F_i \frac{|C_i|}{\left| \bigcup_{l=1}^{M} C_l \right|} \right) \tag{19}$$

Note that $F_T \in [0,1]$. The higher the value of F_T the better the quality of the resulting partition.

The second measure is the Entropy, which for the cluster $\tilde{C}_j$ is given as:

$$E(\tilde{C}_j) = -\sum_{i=1}^{M} \Pr(C_i, \tilde{C}_j) \log\left(\Pr(C_i, \tilde{C}_j) \right) \tag{20}$$

Then, the total Entropy of the overall partition is defined as follows:

$$E_T = -\sum_{j=1}^{\tilde{M}} \left(E(\tilde{C}_j) \frac{\left| \tilde{C}_j \right|}{\left| \bigcup_{i=1}^{M} C_i \right|} \right) \tag{21}$$

The smaller the value of E_T the better the quality of the resulting clusters.

Tables 5 and 6 depict the comparative results in terms of the measures F_T. As a concluding remark, we can easily see that the proposed method is very competitive to the others, obtaining a superior cluster quality in many cases.

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 corresponding 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.

Future extensions of our web crawler will enable a more intelligent focus on culture-related web content through applying sophisticated techniques such as reinforcement learning and evolutionary adaptation so as to improve its performance over longer crawls.

References

[1] I. Anagnostopoulos, C. Anagnostopoulos, V. Loumos, E. Kayafas, "Classifying web pages employing a probabilistic neural network", IEE Proceedings on Software 151 (3): 139-150, 2004.

[2] S. Chakrabarti, K. Punera, M. Subramanyam, "Accelerated focused crawling through online relevance feedback", Proceedings of the 11th International World Wide Web Conference (WWW'02): 148-159, May 2002.

[3] L.-C. Chen, "Using a new relational concept to improve the clustering performance of search engines", Information Processing and Management 47: 287–299, 2011.

[4] R.-C. Chen, C.-H. Hsieh, "Web page classification based on a support vector machine using a weighted vote schema", Expert Systems with Applications, 31(2), 427-435, 2006.

[5] S. Chiu, "Fuzzy model identification based on cluster estimation", Journal of Intelligent and Fuzzy Systems, 2(3): 267-278, 1994.

[6] V. Crescenzi, P. Merialdo, P. Missier, “Clustering web pages based on their structure”, Data & Knowledge Engineering, 54: 279-299, 2005.

[7] M.K. Dey, H.M.S. Chowdhury, D. Shamanta, K.E.U. Ahmed, “Focused web crawling: A framework for crawling of country based financial data”, Proceedings of the 2nd IEEE International Conference on Information and Financial Engineering (ICIFE’10): 409-412, 2010.

[8] H. Dong, and F.K. Hussain, “Focused crawling for automatic service discovery, annotation, and classification in industrial digital ecosystems”, IEEE Transactions on Industrial Electronics, 58(6): 2106-2116, 2011.

[9] P.-Y. Hao, J.-H. Chiang, Y.-K. Tu, “Hierarchically SVM classification based on support vector clustering method and its application to document categorization”, Expert Systems with Applications, 33(3), 627-635, 2007.

[10] D. Hati, and A. Kumar, “UDBFC: An effective focused crawling approach based on URL Distance calculation”, Proceedings of the 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT’10): 59-63, 2010.

[11] D. Hati, B. Sahoo, and Kumar A., “Adaptive focused crawling based on link analysis”, Proceedings of the 2nd International Conference on Education Technology and Computer (ICETC’10): 455-460, 2010.

[12] Y. Huang, Y.M Ye, “wHunter: A focused web crawler - A tool for digital library”, Proceedings of the 7th International Conference of Asian Digital Libraries (ICADL’2004): 519-522, 2004.

[13] O. Jalilian, H. Khotanlou, “A new fuzzy-based method to weight the related concepts in semantic focused web crawlers”, Proceedings of the 3rd International Conference on Computer Research and Development (ICCRD’11): 23-27, 2011.

[14] O.-W. Kwon, J.-H. Lee, “Text categorization based on k-nearest neighbor approach for Web site classification”, Information Processing & Management, 39(1), 25-44, 2003.

[15] I. Lee, and B.-W. On, “An effective web document clustering algorithm based on bisection and merge”, Artificial Intelligence Review: 69-85, 2011.

[16] C. S. Lim, K. J. Lee, G. C. Kim, “Multiple sets of features for automatic genre classification of web documents”, Information Processing & Management, 41(5), 1263-1276, 2005.

[17] L. Lin, A. Liotta, A. Hippisley, “A method for automating the extraction of specialized information from the web”, Proceedings of the 2006 International Conference on Computational Intelligence and Security (CIS’2005): 489-494, 2005.

[18] S.A. Macskassy, A. Banerjee, B.D. Davison, H. Hirsh, “Human performance on clustering web pages: a preliminary study”, Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining (KDD’1998), 1998.

[19] S. Mali, and B.B. Meshram, “Focused web crawler with revisit policy”, International Conference and Workshop on Emerging Trends in Technology (ICWET’2011): 474-479, 2011.

[20] G. S. Manku, A. Jain, A. D. Sarma, “Detecting near duplicates for web crawling”, Proceedings of the 16th International Conference on World Wide Web (WWW’2007), pp. 141-150, 2007.

[21] C.D. Manning, H. Schutze, “Foundations of statistical natural language processing”, MIT Press, Cambridge, MA, 1999.

[22] N. Martin, K. Khelif, “Focused crawling using name disambiguation on search engine results”, Proceedings of the European Intelligence and Security Informatics Conference (EISIC’11): 340-345, 2011.

[23] F. Menczer, G. Pant, P. Srinivasan, “Topical web crawlers: evaluating adaptive algorithms”, ACM Transactions on Internet Technology, 4(4): 378-419, 2004.

[24] B. Novak, “A survey of focused web crawling algorithms”, Proceedings of the 7th International MultiConference on Information Society (IS’2004), 2004.

[25] J.R. Punin, M.S. Krishnamoorthy, M.J. Zaki, “Web usage mining languages and algorithms”, Technical Report, Rensselaer Polytechnic Institute, NY, 2001.

[26] D. Qi, B. Sun, “A genetic k-means approach for automated web page classification”, Proceedings of the 2004 IEEE International Conference on Information Reuse and Integration (IRI’2004): 241-246, 2004.

[27] T.A. Runkler, J.C. Bezdek, “ Web mining with relational clustering”, International Journal of Approximate Reasoning, 32: 217-236, 2003.

[28] A. Schiffman, “Hierarchy in web page similarity link analysis”, CommerceNet Labs, CN-TR-06-02, 2006.

[29] A. Sun, Y. Liu, E.-P. Lim, “Web classification of conceptual entities using co-training”, Expert Systems with Applications, 38 (12), 14367-14375, 2011

[30] Q. Tan, P. Mitra, C.L. Giles, “Designing clustering-based web crawling policies for search engine crawlers”, Proceedings of the 16th ACM conference on Conference on Information and Knowledge Management (CIKM’2007): 535-544, 2007.

[31] P. Tao, H. Fengling and Z. Wanli, “A new framework for focused web crawling”, Wuhan University Journal of Natural Sciences, 11(5): 1394-1397, 2006.

[32] G.E. Tsekouras, C.N. Anagnostopoulos, D. Gavalas, D. Economou, “Classification of web documents using fuzzy logic categorical data clustering”, IFIP International Federation for Information Processing, 247, pp. 93-100, 2007

[33] Xerces2 Java Parser 2.11.0, http://xerces.apache.org/xerces2-j/, last visited: July 2011.

[34] Z.S. Xu, J. Chen, “An overview of distance and similarity measures of intuitionistic fuzzy sets”, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 16(4): 529-555, 2008.

[35] Q. Xu, W. Zuo, “First-order focused crawling”, Proceedings of the 16th International Conference on World Wide Web (WWW’07): 1159-1160, 2007.

[36] Y. Ye, F. Ma, Y. Lu, M. Chiu, J. Huang, “iSurfer: a focused web crawler based on incremental learning from positive samples”, Proceedings of the 6th Asia-Pacific Web Conference (APWeb’2004): 122-134, 2004.

[37] H.L. Yu, L. Bingwu, and Y. Fang, “Similarity computation of web pages of focused crawler”, Proceedings of the 2010 International Forum on Information Technology and Applications: 70-72, 2010.

[38] M. Yuvarani, N. Iyengar, A. Kannan, “LSCrawler: a framework for an enhanced focused web crawler based on link semantics”, Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI’06), 2006.

[39] H. Zhang, J. Lu, “SCTWC: An online semi-supervised clustering approach to topical web crawlers”, Applied Soft Computing, 10(2): 490-495, 2010.

[40] Q. Zhu, “An algorithm OFC for the focused web crawler”, Proceedings of the 6th International Conference on Machine Learning and Cybernetics (ICMLC’2007): 4059-4063, 2007.