Acceleration the Web pages classifier using the HITS algorithm

Meadi Mohamed Nadjib
Computer Science Departement, LESIA Laboratory
University of Biskra, Algeria, mmnadjib@yahoo.fr

Babahenini Mohamed Chaouki
Computer Science Departement, LESIA Laboratory
University of Biskra, Algeria, Chaouki.Babahenini@gmail.com

Abstract— Nowadays, The increasing demand for information contained in web pages requires the use of automatic classifiers. These classifiers are facing the problem of large-scale size of the input data because they must handle millions of Web pages, tens of thousands of terms or features and hundreds of categories. These constraints have revealed an important challenge which is the reduction of the input data without influencing the classification results.

In this paper, we present an approach for accelerating the learning and the classifications phases of web pages classifiers. The acceleration is achieved by reducing the size of the training set in both axes; web pages and features using the HITS (Hypertext induced Topic Search) algorithm. Originally, the algorithm was designed for ranking web pages according to the quality of the pages that are linked to them. A comparative study between the classical classifier and the accelerated one was conducted in order to test the efficiency of the proposed approach.

Key words—Web mining, link Analysis, Bipartite graph, HITS, SVM.

I. INTRODUCTION

The number of pages published on the World Wide Web is estimated hundreds of millions. The mining of these pages requires incredible intellectual efforts which sometimes exceed human capacities.

Web pages as a whole have almost no unifying structure; variability in the style and content creation is much greater than in traditional collections of textual documents [10]. So, it is impossible to apply databases management techniques and traditional information retrieval in the extraction of knowledge contained in Web pages. To remedy this problem, the Web mining has emerged as a solution for analyzing Web pages. Its objective is to use methods and techniques of data mining to extract the knowledge contained in the Web pages regardless their unstructured nature. More precisely, these data mining techniques and methods are used during the preprocessing and the features extraction phases.

The most commonly models used to represent the Web pages to be processed by information retrieval systems or Web pages classification are: the Boolean model, the vector space model and the probabilistic model [1]. Among these models, we find that the vector model is the best known and most used. This model treats each document as a set of words or terms. In other words, a document is described by a set of distinctive terms. The term is not necessarily a natural language word in a dictionary. The vector model associates with each term a weight calculated by a given method. With this representation, a collection of documents is represented by a relational table (or matrix), where each term is an attribute, and each weight is an attribute value.

Before starting the process of building an automatic Web pages classifier, we must first solve the problem of the immense size of the input data sets. The classifier must handle millions of Web pages, tens of thousands of features and hundreds or thousands of categories. Therefore, efficient mechanisms for the features selection are extremely important.

In the literature, there are several works which propose the reduction of size of the input data or the so called dimension reduction of feature vector. In [3], for example, authors propose a fuzzy paradigm to analyze and evaluate the uncertain behavior of the threshold value to extract the features. In addition, Authors in [7] propose to develop a reporting procedure which attributes high scores to useful keywords and lower scores for less useful keywords. Keywords scores can be ranked in descending order to form a ranking list. In [14], the authors proposed to use particle systems (PSO) and multi-class SVM of type One against the Rest (1$&$ R) for the selection of features.

The remainder of this paper is organized as follows: the second section is dedicated to present the theoretical definitions. In section 3, a detailed description of the proposed approach is presented. The experimental results are given in Section 4. The paper ends with a conclusion and research perspectives.
II. THEORETICAL DEFINITIONS

A. Web mining

Web mining is the application of data mining techniques to discover consistent, schemas or models in the internet resources. Web mining aims to discover useful information or knowledge from Web hyperlinks, page contents, and usage logs (or web logs). Based on the data types used in the process of exploration, the tasks of Web mining can be classified into three main types: Web structure mining, Web content mining and Web usage mining. Exploring the structure of the Web is to discover knowledge from hyperlinks that represent the structure of the Web. Exploring Web content aims to extract useful knowledge / information by exploiting the content of Web pages. The search using the Web allows the construction of user access patterns from usage logs, which record the clicks from each user [1].

B. Link analysis

Web pages are connected by hyperlinks, which carry important information. Some hyperlinks are used to organize a large amount of information on the same website, and thus only point to pages on the same site. Other links point to pages in other Web sites. These links often provide an implicit means of transportation of the authority to the pages pointed. Therefore, these pages that are pointed by many other pages are likely to contain reliable information. These links should obviously be used in evaluating the page ranking in search engines.

During the period 1997-1998, Kleinberg and Brin and Page have introduced two search algorithms, based on the analysis of hyperlinks, which are PageRank and HITS algorithms. PageRank is the algorithm that powers the Google search engine. PageRank and HITS exploit the hyperlink structure of the web to rank web pages according to their levels of authority [1][2].

C. HITS

HITS is a link analysis algorithm which helps the rating of web pages. Unlike PageRank which is a static ranking algorithm, HITS depends entirely to the search query. When the user issues a search query, HITS first expands the list of relevant pages returned by a search engine, and produces two rankings of the expanded pages set; the ranking authority and the ranking Hub.

An authority is a page with a lot of inbound links (Fig.1). The idea is that a given page may have good content on a particular topic; so many pages can express their confidence by sending a link to it. A Hub is a page with many outgoing links (Fig.1). The Hub page serves as an organizer of information on a topic and points to many good authority pages on the particular subject. When a user comes to a Hub page, it will find many useful links that are at the best pages on the subject [1][5].

D. SVM

Support vector machines (SVM) have been developed based on the principle of structural risk minimization theory of statistical learning [15][16]. The main idea of SVMs is to construct a hyperplane which maximize the separation margin between positive and negative examples:

\[ \{ (x_1, y_1), ..., (x_i, y_i) \}, x_i \in \mathbb{R}^n, y_i \in \{+1, -1\} \]  

Consider the problem of separating two classes represented by \( n \) examples: We need to find a linear function to separate the two classes (Fig. 2):

\[ f(x) = y = w \cdot x + b \]  

i.e we must find the widest margin between the two classes, which is to minimize \( \frac{1}{2} \| w \|^2 \).

SVM Learning led to quadratic optimization problem as follows:

\[
\begin{align*}
\minimize: & \quad L_0 = \sum_{i=1}^n a_i - \frac{1}{2} \sum_{j=1}^n y_i y_j a_i a_j <x_i, x_j> \\
\text{With:} & \quad \sum_{i=1}^n y_i a_i = 0 \quad \text{and} \quad a_i \geq 0, i=1,2,..., n
\end{align*}
\]  

(3)

In the case where the training data are not linearly separable, we can relax the margin constraints by introducing slack variables \( \xi_i \geq 0 \) relative to the boundaries of the separation margin with a penalty parameter \( C \) and the problem (3) becomes a convex quadratic programming problem:

\[
\begin{align*}
\minimize: & \quad \frac{1}{2} \| w \|^2 + C \sum_{i=1}^n \xi_i \\
\text{With:} & \quad y_i (w \cdot x_i + b) \geq 1 - \xi_i \text{and} \quad \xi_i \geq 0, i=1,2,..., n
\end{align*}
\]  

(4)

For many data sets, the decision boundaries are not linear. The basic idea is to transform the data in the input space \( X \) into a feature space \( F \) via a nonlinear function \( \phi \). This function is called the kernel, which is a symmetric function that satisfies the conditions of Mercer [1]:

\[ \phi: X \rightarrow F \]  

(5)

After the transformation of the problem (4) the optimization will be:
Its dual form will be:

\[
\text{Minimize: } L_0 = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \left< \phi(x_i); \phi(x_j) \right>
\]

\[
\text{With: } \sum_{i=1}^{n} y_i \alpha_i = 0 \quad \text{and} \quad \alpha_i \geq 0, i = 1, 2, \ldots, n
\]

Therefore, the decision rule for classification is:

\[
f(x) = \text{sign}(\sum_{i=1}^{n} y_i \alpha_i \left< \phi(x_i); \phi(x) \right> + b)
\]

### III. PROPOSED APPROACH

Our approach, schematized in Figure (Fig. 3), aims to reducing the size of the training set using a link analysis algorithm which is HITS. This algorithm was originally used for classify the Web pages according to the quality of Web pages link to it.

#### A. Preprocessing

In this step, we performed several tasks in order to get a bag of words. These tasks are the followings:

- Tags removal: delete all the HTML and scripting tags and extract the original text.
- Number deleting: remove all the number from the text, and keep only the alphanumerical words.
- Word Stemming: stem words using the Porter stemming algorithm.
- Stop words removal: clean the text from all stop words such as "for", "while", "and" …

#### B. Building of bipartite graph

In the graph theory, a graph is called bipartite if there is a partition of the set of nodes into two subsets U and V which each edge have an end in U and the other in V, (see Fig 4).

In this phase, we will represent the cleaned learning set as a bipartite graph. The departures nodes are the Web pages and the arrivals nodes are the features.

To represent this graph we have chosen the matrix form, where we used the matrix A [N, M].

- N: Represents the number of the web pages,
- M: Is the size of the feature vector.

The values in this matrix are binary values as follows:

- \( A[i, j] \) is set to 1 if the feature number j is held in the page number i,
- \( A[i, j] \) is set to 0 otherwise.

#### C. Application of HITS algorithm

Originally the HITS algorithm was applied to calculate the weight of Web pages based on the weight of the linked to Web pages. However, in this work we will use this algorithm to rank Web pages using the weight of features that they contain. Each Web page is considered as a hub and each feature is considered as an authority where hubs point to authorities.
The weights of Web pages can be described as a vector $\mathbf{h}$ of dimension $N$ (Number of documents), where $h_i$ is the weight hub of the page $p_i$. Weights of the features can be described as a vector $a$ of dimension $m$, where $a_i$ indicates the authority value of the feature $i$. So, we have two formulas to apply [2][9][17]:

$$a_i = \sum_{j \neq i} h_j$$

$$h_j = \sum_{i \neq j} a_i$$

The two previous relations (9) can be written as follows:

$$a = A^T \cdot h$$

$$h = A \cdot a$$

(10)

Where $A$ is matrix representing the bipartite graph.

The algorithm HITS can be summarized as follows [2, 9, 17]:

**Algorithm 1 HITS algorithm**

Set values of the two vectors to 1

$$h_0 = (1, 1, ..., 1) \; \; a_0 = (1, 1, ..., 1) \; \; k = 0$$

Repeat:

1. $k \leftarrow k + 1$
2. $a_k \leftarrow A^T h_{k-1}$
3. $h_k \leftarrow A a_k$
4. Normalize the two vectors;
5. Until: $\|a_k - a_{k-1}\| < \varepsilon_a$ et $\|h_k - h_{k-1}\| < \varepsilon_h$

return $a_k$ and $h_k$

This algorithm produces two vectors $a$ and $h$. The vector $a$ reflects the importance of each feature in the input corpus. Vector $h$ indicates the importance of each web page according to the HITS algorithm.

**D. Web pages reduction**

At the beginning of this phase, we sort the vector Hub in the descending order and determine a threshold $Th$. Then, we delete all web pages that have hub values less than the threshold $Th$.

The reduction of web pages from learning set can produce a situation where there are features that do not belong to any web pages from the remaining pages (orphan features). Thus, we remove them from the feature vector.

At the end of this phase, we find that we achieve two types of minimization: vertical and horizontal. This minimization certainly accelerates the following phases, essentially the learning and the testing phases.

**E. Features reduction**

After the vertical and horizontal minimization made in the previous phase, we focus here on reducing features that have an authority value less than the threshold $Th_a$.

At the end of this task, we find some web pages with zero features i.e. web pages that have not any feature presents in the feature vector, so we must remove them from the learning set. Thus, we achieved a second minimization on both axes.

**F. Calculation of features weights**

A document in the vector space model is represented as a weight vector in which elements are weights of features that has been calculated based on the TF-IDF (Term Frequency-Inversed Document Frequency) scheme [11].

Let $N$ be the number of documents in the training set and $df_i$ is the number of documents where the term $t_i$ appears at least once. $F_i$ is the frequency of term $t_i$ in document $d_i$. Then, the normalized frequency of the term $t_i$ in $d_i$ (denoted $TF_i$) is given by:

$$TF_{ij} = \frac{f_{ij}}{\max(f_{ij}, f_{ij-1}-f_{ij})}$$

(11)

Where $|v|$ is the size of the feature vector.

The inverted frequency of documents containing the term $j$ is given by the formula:

$$IDF_j = \log(\frac{N}{|id_j|})$$

(12)

Thus, the weight of a feature (term) $i$ in document $j$ is given by:

$$wij = TF_{ij} \times IDF$$

(13)

**G. Learning**

Among the learning and data classification algorithms, we have chosen to use the SVM, because the classifiers that based on SVM have shown promising results in the classification of text and Web pages [13].

The obtained weights take different values before starting the learning phase. To obtain good results, we are obliged to pass through a normalization phase. Where the new weights have to range between -1 and 1.

The normalization value $Norm_w$ of a weight $w$ is obtained by the following formula:

$$Norm_w = 2 \times \frac{w - \min w}{\max w - \min w} - 1$$

(14)

Where:

$\max w$ and $\min w$ represent respectively the maximum and the minimum weights in the original corpus.

**IV. EXPERIMENTAL RESULTS AND DISCUSSIONS**

To test the efficiency of our approach, we conducted a comparative study between the classical web pages classifier, our approach that proposes the reduction of the learning set using the HITS algorithm.

Our experiments were run on i5 Intel PC, with 4GB main memory. The PC runs under Microsoft Windows XP.

The learning algorithm used was the SMO (Sequential Minimal Optimization) [6]. After a set of tests, we have chosen the best model of our SVM classifier. We used a model that consists of a Polynomial kernel [1] with the exponent $\rho$ and the parameter $C$ was fixed to 100.

The web pages used in the experiments are extracted from WebKB project [4]. The latter contains web pages that
collected in January 1997 from IT departments of 04 universities (Cornell, Texas, Washington and Wisconsin). These Web pages were manually classified into 7 categories: student, faculty, staff, department, course, project and others. Among these web pages, we choose a subset that has been partitioned into two classes, courses and not courses (TABLE 1). The desired class course contains Web pages of computer lessons, and not course class contains Web pages about various subjects.

### TABLE II. SUMMARY OF THE OBTAINED RESULTS WITHOUT REDUCING.

<table>
<thead>
<tr>
<th></th>
<th>E1</th>
<th>E2</th>
<th>E3</th>
<th>E4</th>
<th>E5</th>
<th>E6</th>
<th>E7</th>
<th>E8</th>
<th>E9</th>
<th>E10</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web pages</td>
<td>2277</td>
<td>2277</td>
<td>2277</td>
<td>2277</td>
<td>2277</td>
<td>2277</td>
<td>2277</td>
<td>2277</td>
<td>2277</td>
<td>2277</td>
<td>/</td>
</tr>
<tr>
<td>Features</td>
<td>19251</td>
<td>19084</td>
<td>19146</td>
<td>19166</td>
<td>19491</td>
<td>19358</td>
<td>19428</td>
<td>19308</td>
<td>19490</td>
<td>19186</td>
<td>/</td>
</tr>
<tr>
<td>Support</td>
<td>883</td>
<td>747</td>
<td>728</td>
<td>722</td>
<td>730</td>
<td>698</td>
<td>718</td>
<td>742</td>
<td>709</td>
<td>697</td>
<td>/</td>
</tr>
<tr>
<td>Accuracy</td>
<td>96.00%</td>
<td>99.20%</td>
<td>97.20%</td>
<td>97.60%</td>
<td>96.40%</td>
<td>99.20%</td>
<td>95.30%</td>
<td>95.70%</td>
<td>96.80%</td>
<td>95.70%</td>
<td>96.91%</td>
</tr>
<tr>
<td>F-measure</td>
<td>0.94</td>
<td>0.99</td>
<td>0.96</td>
<td>0.97</td>
<td>0.95</td>
<td>0.99</td>
<td>0.93</td>
<td>0.93</td>
<td>0.95</td>
<td>0.93</td>
<td>0.95</td>
</tr>
</tbody>
</table>

To validate our approach we used the n-cross-validation where we divided our training set into n (n=10) parts and we learn our system on (n-1) parts of observations and the model validation will be on the nth part of the observations. This operation is repeated n times.

Instead of setting the threshold value manually, we propose to calculate its value using the following formula :

\[
Th = \frac{\sum_{i=1}^{N} H_{ab1}}{N} \text{Number of Web pages}
\]  \hspace{1cm} (15)

\[
Ta = \frac{\sum_{i=1}^{M} auth_{1}}{2*M} \text{Number of features for experiment j}
\]  \hspace{1cm} (16)

For instance, the threshold Ta of the second experiment is fixed as follow:

\[
Ta_{2} = \frac{\sum_{i=1}^{M} auth_{1}}{2*M_f=2*19084} = 0.00003
\]

and the Threshold Th is fixed as follow:

\[
Th = \frac{\sum_{i=1}^{N} H_{ab1}}{N=2277} = 0.00044
\]

We propose to calculate its value using the following formula:

\[
F - measure = \frac{2\times Precision \times Recall}{Precision + Recall}
\]  \hspace{1cm} (18)

\[
Precision = \frac{true positives}{true positives + false positives}
\]  \hspace{1cm} (19)

\[
Recall = \frac{true positives}{true positives + false negatives}
\]  \hspace{1cm} (20)

Where:

- Positives: All web pages belonging to the desired class (in our case the class courses).
- Negatives: All web pages that do not belong to the desired class (in this case not courses).
- TruePositives and TrueNegatives ; mean documents were correctly classified by the classifier.
- FalsePositives and FalseNegatives: Refer to documents that were misclassified by the classifier.

Based on the Table II and Table III, we note that this approach has advantages on accelerating the learning phase with considerable minimization of the size of the training set, in both axes; vertical (number of learning examples ) and horizontal (the size of feature vector ). This approach also accelerates the classification phase due to the significant reduction of the number of support vectors.

Knowing that the runtime of the proposed phases (HITS algorithm, reducing of features and web pages) is insignificant compared to the total runtime of the system.
However, this approach has a drawback which is the dependence with the threshold value. If we made a good choice of the threshold value we can keep or may improve the accuracy of our system otherwise we will lose slightly the accuracy.

V. CONCLUSION AND PERSPECTIVES

In this paper, we have proposed a new approach for creating a web pages classifier. This approach proposes to displace the HITS algorithm from its original environment to use it for the calculation of weights of the features.

The tests that we perform on a set of Web pages extracted from the project WebKB we given an encouraging results. Our classifier reduces the learning time and the number of the support vectors and improves the accuracy of the classification in the most of the experiments.

In our future works, we plan to improve this approach for selecting the thresholds $Ta$ and $Th$ with an automatic way. We aim also to continue using the HITS algorithm but this time to calculate the weights of features and make a comparative study with the results obtained using the TF-IDF model. We also aim to extend this approach to be applicable in the multiclass classification.

REFERENCES


