International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Ingénierie des Systèmes d'Information
Vol. 27, No. 5, October, 2022, pp. 747-755
Journal homepage: http://iieta.org/journals/isi

# A Hybrid Approach for Web Pages Classification

Ouided Hioual[1*], Sofiane Mounine Hemam[1], Ouassila Hioual[1,2], Lyes Maif[3]

[1] Computer Science Department, Abbes Laghrour University, Khenchela 4000, Algeria
[2] Lire Laboratory, Constantine 2 University, Constantine 25000, Algeria
[3] Research Center on Semiconductor Technology for Energetic, TESE-CRTSE, 2 BD Frantz Fanon, 7 Merveilles, POB 140, Algiers, Algeria

Corresponding Author Email: hioual.ouided@univ-khenchela.dz

https://doi.org/10.18280/isi.270507

**Received:** 17 July 2022
**Accepted:** 6 October 2022

***Keywords:***
*keyword extraction, machine learning, supervised machine learning approach, TF-IDF, web page classification*

**ABSTRACT**

Currently, the internet is growing at an exponential rate and can cover just some required data. However, the immense amount of web pages makes the discovery of the target data more difficult for the user. Therefore, an efficient method to classify this huge amount of data is essential where web pages can be exploited to their full potential. In this paper, we propose an approach to classify Web pages based on their textual content. This approach is based on an unsupervised statistical technique (TF-IDF) for keyword extraction (textual content) combined with a supervised machine learning approach, namely recurrent neural networks.

## 1. INTRODUCTION

In recent years, the World Wide Web became the main source of data for human due it increased development. Indeed, the web has become an essential tool allowing easy and quick access to information for, research, learning, information and discovering new knowledge. It allows to better respond to the increasing internet user needs for information and knowledge. In addition, the internet users are increasingly overwhelmed by this volume made available to them [1]. Thus, the need to new methods and advanced tools to facilitate access and meet the needs of Internet users has prompted researchers to focus on the classification domain. This latter is used by human in his daily life when he tries to answer problems and questions about the category of objects, i.e. the assignment of objects to their class (by observing their formats, colors, sizes . . .etc.) [2].

Classification has a vital role in many web-based information management and retrieval tasks. The content page classification is essential for crawling targeted, assisted web directory development, subject-specific web link analysis, contextual analysis, advertising and analysis of the web's thematic structure. Classification of web pages can also improve the quality of web search [3].

The web pages classification is generally done by extracting the textual content of the page and ignoring HTML tags, CSS and JS code. So, it extracts the key data from the web page and then, it performs the classification. There are several web page classification approaches (supervised and unsupervised) and several automatic keyword extraction methods (supervised and unsupervised) [4]. Unsupervised methods are emerging methods with the particularity of abstracting from the specificity of the data processed [5], This abstraction is explained by approaches based on observations about what a keyword is in the general meaning: semantic importance, degree of information, syntactic structure [6].

In contrast to unsupervised methods, supervised methods do not use properties defined from statistical and linguistic features, but they use decision models learned from these features, calculated on the keywords of a learning corpus.

The use of a learning corpus implies that the learned models are specific to the disciplinary domain and language. This specificity can be advantageous when the domain and the language that represents the corpus are the same for the documents that are then analyzed. Otherwise, the results of the extraction can suffer.

In this paper we propose an approach for web pages classification based on supervised machine learning approach, namely recurrent neural networks (RNN)for classification and unsupervised statistical technique named Term Frequency-Inverse Document Frequency (TF-IDF) for the extraction of keywords from pages.

The remainder of this paper is structured as follows: related works about the problem of the web pages classification are presented in section 2. Section 3 describes the proposed, architecture, and its functionality. Then in Section 4, we present some experimental results. Finally, our conclusion and direction for future works are summarized at the end of this paper.

## 2. RELATED WORKS

Nowadays, the classification of web pages has become a very important issue. Indeed, many studies have been developed for this purpose in the literature. A Web page has different types of features. According to these features, Hashemi [7] has divided the research works in the web pages classification field into three kinds: Text-based, image-based, and combined usage of text-based and image-based features. However, Aydos et al. [8] divided web pages classification' related works into four main groups: (1) Textual classifications: URL address, text content, title, HTML description HTML code, etc. (2) Visual classification: images, design, videos, etc. (3) Graph-based classifications: hyperlink structures, neighbor web sites. (4) And other information: user behaviors, web directories, semantic web, raw data of domain (IP address, owner, hosting server, hosting country) [8]. Since, our paper lies in text-based classification category, we will focus, in this section, on text-based related works.

Among works text-based web pages classification, we can cite the following ones.

Alamelu and Santhosh [9] suggested a method to classify, automatically, Web pages without using the entire information of these latter. Indeed, they used only a minimum number of representative features that they extracted from a web page. In addition, the authors had modeled machine learning classifiers using the selected features. The experimental results proved that there was good improvement in classification accuracy.

Su et al. [10] have built a prediction system for web page ranking. Indeed, they proposed a new method of classification where they researched keyword density and keyword position in the web page content. They also defined their impacts on rankings by using the notion of optimal keyword frequencies.

Hashemi [7] presented methods of web page classification and usage scenarios in which a machine learning algorithm was used to classify web pages in predefined categories based on features extracted from text and HTML tags. As results, the author asserted that HTML tags play an important role in methods that use keyword frequencies to identify web page categories.

Li et al. [11], for classifying web spams, used the deep belief networks (DBN) for the first time. Then, it was combined with the Synthetic Minority Over-Sampling Technique (SMOTE) and De-Noising Auto-Encoder (DAE) algorithm after the multi-aspect research and consideration. According to authors, the proposed method had improved the classification performance of web spam, and the results showed that the classification method proposed in their paper improves the classification performance to a certain extent, which provides a good direction for the future classification of web spam.

Duari and Bhatnagar [12] proposed a supervised framework for automatic keyword extraction from single document. For this, they had modeled the text as complex network. Then, they had constructed the feature set by extracting selected node properties from it. The authors exploited several node properties, by using unsupervised graph-based keyword extraction methods, to discriminate keywords from non-keywords. In addition, they had exploited the complex interplay of node properties to design a supervised keyword extraction method. The study of the results has shown that the proposed method performs better in most cases.

Balim and Özkan [13], a deep learning based model was proposed for functional classification of web pages, regardless of language. The authors used Transfer Learning to reduce the cost during the feature extraction process from recorded web page images. In addition, they presented results of two different experiments to show the effectiveness of their method.

Salminen et al. [14] used machine learning algorithms to predict web page rank for pages in the e-commerce gift industry. They researched 30 blogs in the selected industry that occupied first-page Google ranking. Two machine learning models (LightGBM and XGBoost) had been tested and conducted feature analysis. This led to the conclusion that the features that had most impact are links, domain security, and H3 headings. However, other keyword-related frequencies were not shown as significant.

Lee et al. [15] proposed a novel simplified swarm optimization (SSO) to learn the best weights for every feature in the training dataset and adopted the best weights to classify the new web pages in the testing dataset. They had applied a Taguchi method to determine the parameter settings because these latter has an important role in the update mechanism of SSO. In order to demonstrate the effectiveness of their algorithm, they compared its performance with that of the genetic algorithm (GA), Bayesian classifier, and K-nearest neighbor (KNN) classifiers according to four datasets. According to authors, the experimental results indicate that the SSO yields better performance than the other three approaches.

El-Hajj and Hajj [16] addressed the selection problem for classification. They had suggested a one-step method designed to select the subset of features. The authors formulated, mathematically, the selection as an optimization problem with the objective of maximizing classification accuracy while simultaneously deriving and choosing the most discriminative features. The authors proposed a statistical-based feature selection method (MFX) that considers all documents from the same category as one extended document, and chooses the most discriminative terms that are frequent and common across all documents of the same category, but rarely present in other categories. According to the authors, MFX is language independent and backed up with a mathematical formulation that finds the optimal number of features that guarantees accurate text categorization. The results indicated that MFX always performed similar to or better than other well-known feature selection methods.

Yu et al. [17] used, at first, the keyword-weight calculation method to reduce the impact of a small number of high-frequency words in the web page document on the weight calculation. It also allowed reducing the value of the low-frequency word weights so that the WPCA (Web Page Classification Algorithms) is more accurate in the calculation process. Secondly, they used Chinese web pages, calculated the similarity between the text to be classified and all the class templates, and then determines the category of all texts according to the similarity and certain classification rules. Finally, in order to improve the learning rate of DL (Deep Learning), the authors considered the use of adaptive parameters. The authors proved through this study that WPCA based on DL are more efficient, consume less system memory and faster than traditional algorithms.

## 3. PROPOSED APPROACH

With the increase of the Internet users number, the growth of websites is proportional. As a result, the ranking of web pages has become a huge topic of research in recent years. This has made an ever-increasing demand for automated classification techniques with high classification accuracy. In this context fails our contribution. This latter consists to develop a web page classification approach based on keyword extraction method and machine learning.

To reach this goal, we present in the first subsection the proposed architecture details. However, the second is devoted to the functionalities of this architecture.

### 3.1 The architecture components

In this subsection, we present the main components of our architecture. As illustrated in the Figure 1, the proposed architecture is composed of two essential parts which are: (1) the extraction part: to extract keywords from a web page and, (2) the classification part: to classify web pages using a supervised machine learning approach, more precisely the recurrent neural networks. Between these two parts, there is a hidden component. Its role is to extract the first word from the obtained result (an ordered list of key words) of the first part, and it sends it to the second part (RNN) as an input.

## 3.2 Functionality of the proposed approach

In this subsection, we present the functionality of our approach to meet the target objectives. This is illustrated through a sequence diagram as shown in Figure 2.

As mentioned above, we use a TF-IDF keyword extraction technique, and a supervised learning approach which is recurrent neural networks (RNN). We used TF-IDF technique to extract keywords from web pages. This technique allows to give as results keywords ordered, in an ascending way, according to their TF*IDF. Compared with other methods, the TF-IDF technique is easy to implement and very powerful, it can be applied in multiple languages using statistical translation and it provides the closest keywords; this is possibly because this method uses all the documents making up the corpus. In this paper, we applied this technique as it is defined in the literature.

In the next sub-sections, we will explain, in detail, each part.

### 3.2.1 Extraction part

In the literature, there are several automatic keyword extraction methods. In our work, we choose to use an unsupervised statistical technique which is TF-IDF. This latter provides keywords closest to those formulated by the web pages designers on one hand, and it uses all the documents making up the corpus on the second hand. Bellow, the TF-IDF is explained in details.

(a) TF-IDF technique

**TF-IDF (Term Frequency - Inverse Document Frequency):** is a statistical measure that evaluates the relevance of a word for a document in a collection of documents [18].

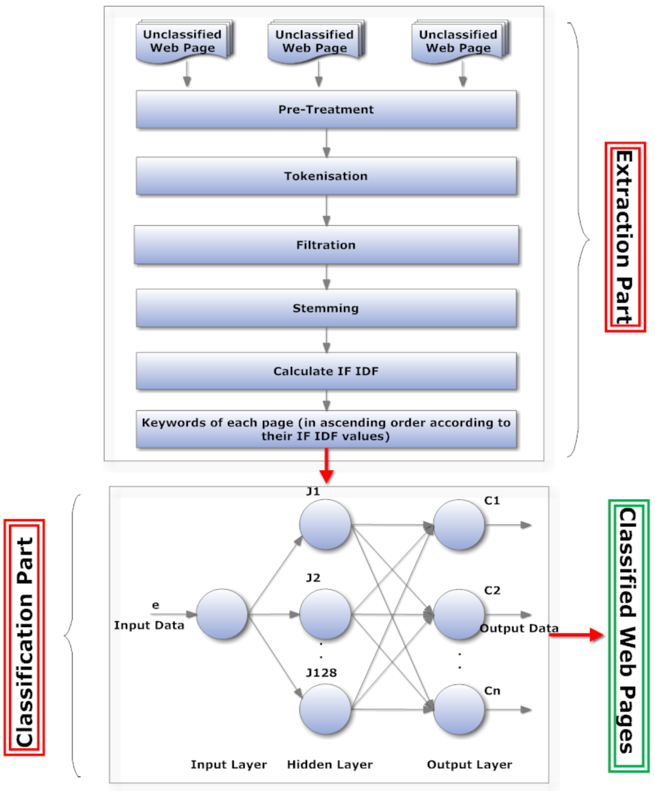


**Figure 1.** General architecture

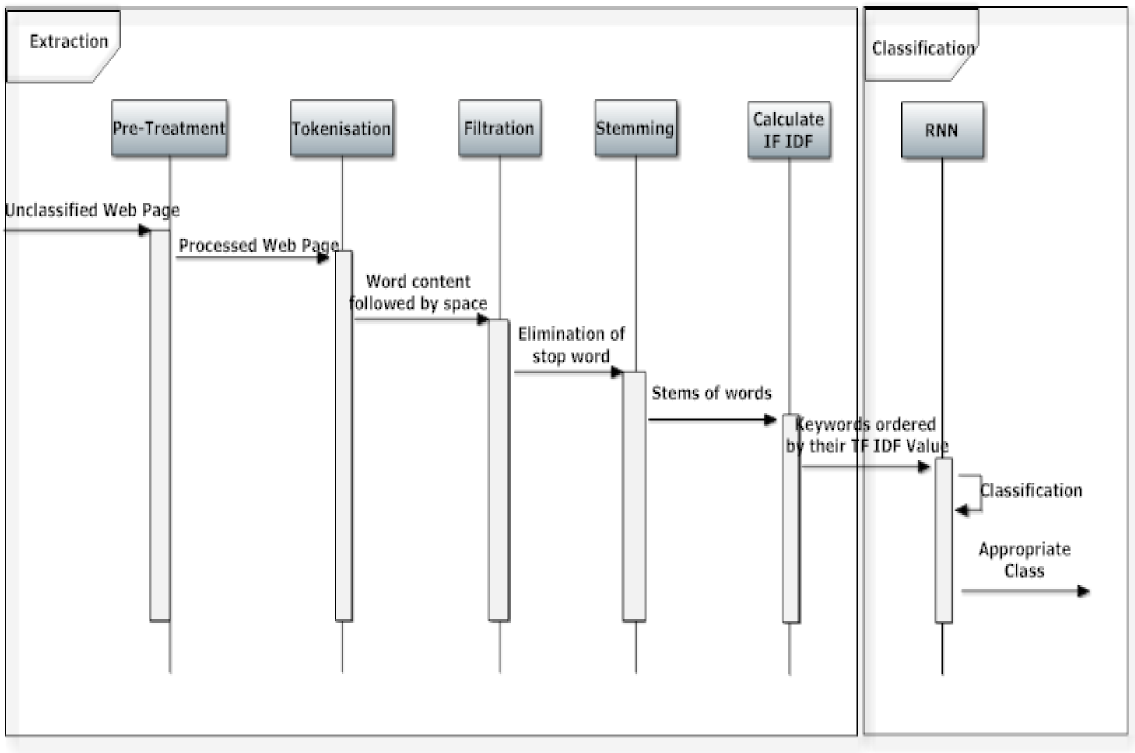


**Figure 2.** Sequence diagram of the proposed approach

**TF (Term Frequency):** The frequency of a term is the number of occurrences of this term in the considered document;

**IDF (Inverse Document Frequency):** The inverse document frequency is a measure of the importance of the term in the whole corpus;

**TF*IDF (Term Frequency - Inverse Document Frequency):** Concerns the weight of a term **T** in a document **D**. It is calculated as follow:

$$\mathbf{TF*IDF}\ (T_i, D_j)=\mathbf{TF}\ (T_i, D_j)*\mathbf{IDF}\ (T_i) \qquad (1)$$

where, **TF** $(T_i, D_j)$: is the frequency of the term $T_i$ in the document $D_j$; **IDF** $(T_i) = \log (N/ DF(T_i))$; **N:** the total number of documents in the documentary base; **DF**$(T_i)$: the number of documents containing the term $T_i$.

(b) Extraction phases

As shown in the Figure 2, the extraction part is composed of six phases, which are:

**Pretreatment phase:** In this phase the inputs are a set of web pages, each of them contains a main content and a noisy content. Pretreatment is very important phase in the classification process because the construction of the model is based on the prepared data. Indeed, web pages that are not prepared correctly can result in a non-performing model. In this phase, the web page input is an html file. Thus, this latter contains tags, CSS code, Java scripts, etc., which are considered, in our context, as the noisy content. Therefore, the pretreatment phase consists to remove all HTML tags, CSS code, java script, as well as the special character strings (example @ h12-D*65), punctuations and digits. In addition, it transforms all upper case characters into lower case. (see Figure 3).

**Tokenization phase:** This phase splits the content into words followed by space [19].

**Filtration phase:** This phase consists to eliminate all stop words. A stop word is a common word (example: I, me, my, myself, we, our, ours, ourselves, you, your, yours, yourself, yourselves, he, him, his, himself,…etc.) that there is no need to index it or to use it in a search. The stop word elimination consists of removing all the standard words (common words) in the content of the extracted web page. These words are very common and they are used in practically all texts. Their presence can degrade the performance of the classification algorithm in terms of cost and classification accuracy. To eliminate all stop words, we, first, store these latter in a list, and then we can remove them easily. NLTK (Natural Language Toolkit) [20] in python has a list of stop words stored in 16 different languages. We can find them in the nltk_data directory.

**Stemming phase:** Stemming is a process of transforming words into their stem or root. The root of a word corresponds to the part of the remaining word after removing its prefix and suffix. Since the TF-IDF technique cannot check the semantic of words, we must use the stremming process to group different forms of a particular word such as "play" and "plays" or "played" into a single word which is "play". The Stemming brings together under the same term (stem) words that have the same root. There are two main families of stemmers: algorithmic stemmers [21] and dictionary-based stemmers [22]. In this work, we used the first family that is often faster and allows to extract roots of unknown words.

At the end of this phase, we obtain as result a stemmer.

**TF-IDF calculation phase:** This phase consists to calculate the TF of each word of the stremmer, the DF and its inverse (IDF) according to their mathematical formulas. Finally, the TF-IDF is obtained according to the formula (1), and it is the product between TF and IDF.

**The results display phase:** In this phase, the words are displayed in ascending order according to their TF-IDF values.

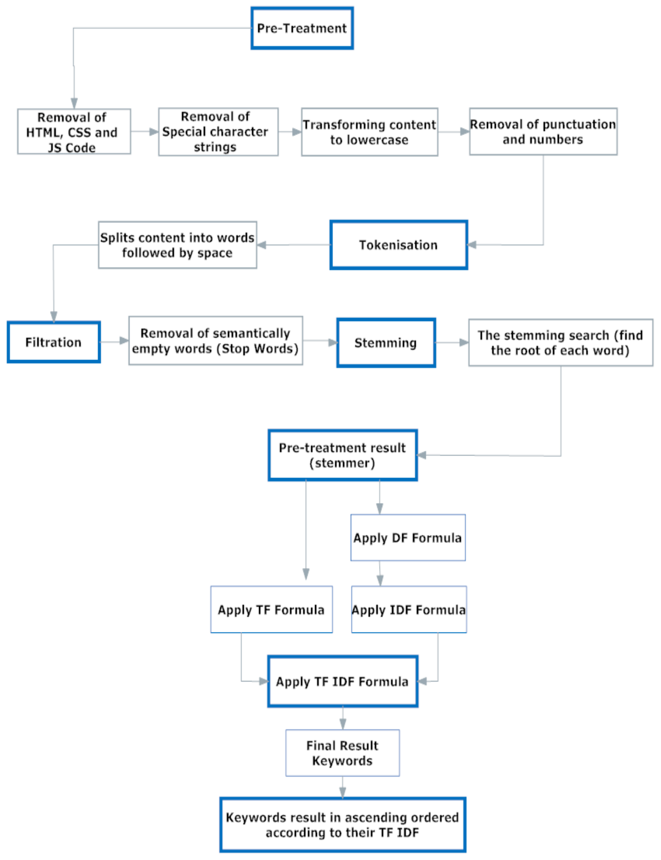

Pre-Treatment
Removal of HTML, CSS and JS Code
Removal of Special character strings
Transforming content to lowercase
Removal of punctuation and numbers
Splits content into words followed by space
Tokenisation
Filtration
Removal of semantically empty words (Stop Words)
Stemming
The stemming search (find the root of each word)
Pre-treatment result (stemmer)
Apply DF Formula
Apply TF Formula
Apply IDF Formula
Apply TF IDF Formula
Final Result Keywords
Keywords result in ascending ordered according to their TF IDF

**Figure 3.** Extraction phases

3.2.2 Classification part

Neural networks are, generally, optimized by statistical-type learning methods thanks to their capacity for paradigms allowing the generation of large functional, flexible and partially structured spaces. They also belong to the artificial intelligence methods family which they allow to take decisions relying more on the perception than on the logical reasoning. The neural network is a calculation model where the design is very schematically inspired by the operation of real classification and generalization [23].

A neural network is, generally, made up of a succession of layers. Each layer (i) is composed of Ni neurons, taking their inputs from the Ni-1 neurons of the previous layer. Each synapse is associated with a synaptic weight. So, the Ni-1 are multiplied by this weight and then added to the level i neurons, which is equivalent to multiplying the input vector by a transformation matrix. Putting one behind the other, the different layers of a neural network would amount to cascading several transformation matrices, and could be reduced to a single matrix. The product of the others, if there were not at each layer the output function which introduces nonlinearity at each step. This shows the importance of the judicious choice of a good output function, a neural network whose outputs would be linear would have no interest. There are several types of neural networks, including recurrent (looped) neural networks that are used in our work [24].

(a) Recurrent neural networks

A looped (recurrent) network, governed by one or more differential equations, results from the composition of the functions carried out by each neuron and the delays associated with these connections.

Figure 4 represents the mathematical structure of the recurrent neural network. Thus, the below mathematical formulas (2) and (3) allow to calculate $h_t$ and $y_t$ in a recurrent way [25, 26].

$$h_t = g_h(W_i * x_t + W_R * h_t - 1 + b_h) \quad (2)$$

$$y_t = g^y(W_y * h_t + b_y) \quad (3)$$

where, $w_i$ is the input weight matrix, $w_y$ is the output weight matrix, $w_R$ is the hidden layer weight matrix, $g_h$ *and* $g_y$ is the activation function, and $b_h$ *and* $b_y$ is the bias. Eqns. (2) and (3) is useful for recursively calculating the values, $h_1$, $h_2$, … and $y_1$, $y_2$, ….

Formulas (4) and (5) make it possible to calculate $h_0$ and $y_0$ respectively at time t=0.

$$h_0 = g_h(W_i * x_0 + b_h) \quad (4)$$

$$y_0 = g_y(W_y * h_0 + b_y) \quad (5)$$

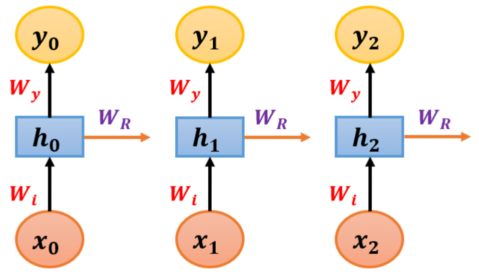


**Figure 4.** Mathematical Representation of Recurrent Neural Network [27]

Whether detailed or simplified, the representation of a recurring network is not easy, because it is difficult to show the temporal dimension on the diagram. This is particularly the case for recurring connections, which use information from the previous time. To solve this problem, we often use a representation of the network "unfolded in time", in order to make it appear explicitly. The following Figure 5 shows an example of an unfolded network [28].

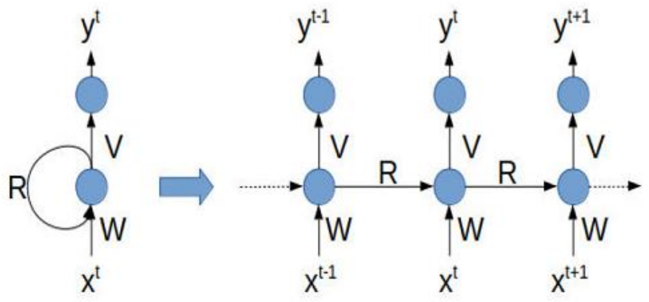


**Figure 5.** RNN: recurring version and unfolded version

This time-unfolded version, clearly, shows the input variables over time: $x^{t-1}$, $x^t$, $x^{t+1}$, etc. (same for the output), and the impact of previous outputs on the current network output. In its unfolded version, the matrices WW, RR and VV are duplicated and thus appear on the diagram as many times as the number of unfoldings of the network in the time.

By explicitly showing the temporal dimension, the unfolded version suggests three possible uses of a recurrent network: sequence labeling, sequence classification or sequence generation [29]. In our work, we are interested in the sequence classification.

- Sequence classification

In this mode of operation, the network traverses the input sequence of size T according to the direction of reading, and produces an output only once the input sequence is finished, as illustrated in the following Figure 6:

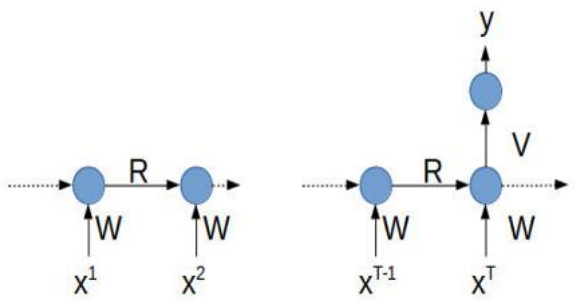


**Figure 6.** Sequence classification: the network "reads" the sequence in its entirety, and produces its output at the last time step

In this case, the output is not a sequence, but only a label. This approach also works in regression; in this case, the output is a value or a vector of values.

## 4. EXPERIMENTAL RESULTS

The experiments run on an Intel® Core™ i7-8700k CPU 3.7GHZ processor with a memory capacity of 32768 MB, under Windows 10, 64 bits with an NVIDIA graphics card. we used Anaconda browser and Spyder as development environment developed with Python.

Our model is divided into two parts. The first part is the extraction of keywords and the second part the classification of web pages based on the results of the first part.

### 4.1 Keywords extraction

The principle of the first part consists of:

-Read the number of web pages to be classified as well as their links

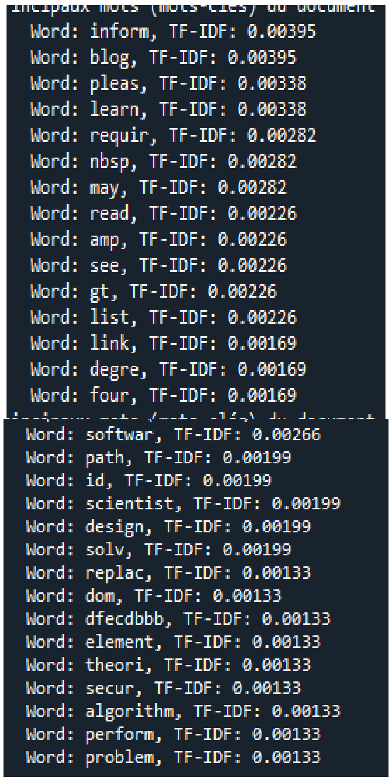
- Removal of HTML, CSS and JS tags.
- Pretreatment which is the separation of words with spaces
- Remove stop-words (empty words).
- Find the root of words (Stemmer)
- Finally the calculation of TF IDF.

In our work we will take two web pages their links are:

1: C:\Users\pc\web pages\page1.html

2: C:\Users\pc\web pages\page2.html

As a result of this first part is: The keywords of the web page displayed in ascending order according to their frequency of appearance on the page (Cf. Figure 7).in our work we take the first 15 keywords.



```
incipaux mots (mots-clés) du document
  Word: inform, TF-IDF: 0.00395
  Word: blog, TF-IDF: 0.00395
  Word: pleas, TF-IDF: 0.00338
  Word: learn, TF-IDF: 0.00338
  Word: requir, TF-IDF: 0.00282
  Word: nbsp, TF-IDF: 0.00282
  Word: may, TF-IDF: 0.00282
  Word: read, TF-IDF: 0.00226
  Word: amp, TF-IDF: 0.00226
  Word: see, TF-IDF: 0.00226
  Word: gt, TF-IDF: 0.00226
  Word: list, TF-IDF: 0.00226
  Word: link, TF-IDF: 0.00169
  Word: degre, TF-IDF: 0.00169
  Word: four, TF-IDF: 0.00169
  Word: softwar, TF-IDF: 0.00266
  Word: path, TF-IDF: 0.00199
  Word: id, TF-IDF: 0.00199
  Word: scientist, TF-IDF: 0.00199
  Word: design, TF-IDF: 0.00199
  Word: solv, TF-IDF: 0.00199
  Word: replac, TF-IDF: 0.00133
  Word: dom, TF-IDF: 0.00133
  Word: dfecdbbb, TF-IDF: 0.00133
  Word: element, TF-IDF: 0.00133
  Word: theori, TF-IDF: 0.00133
  Word: secur, TF-IDF: 0.00133
  Word: algorithm, TF-IDF: 0.00133
  Word: perform, TF-IDF: 0.00133
  Word: problem, TF-IDF: 0.00133
```

**Figure 7.** The result of TF IDF (page1 and page2)

## 4.2 Classification

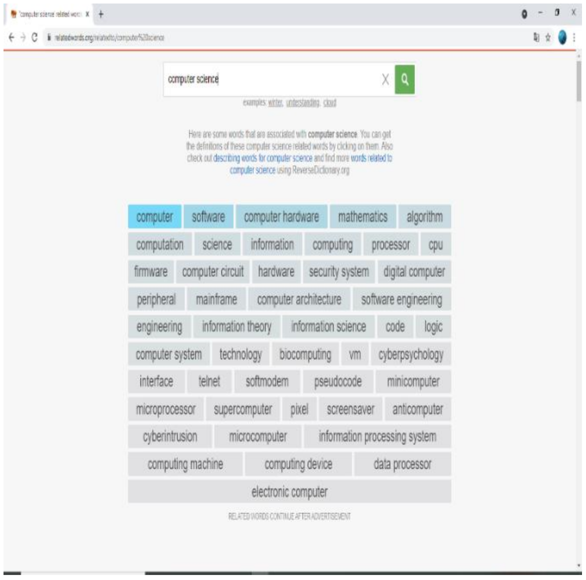
In this subsection we will describe in detail the classification part. Let's start with our dataset first.

### 4.2.1 The creation of the dataset

To create the dataset, we start by finding all words of the domain that we chose. To do this, we use an online word generator called *Related Words*. Since the result words of the TF-IDF will be in stemmer form, so the form of the dataset words will be the same.

The part below shows the different steps to build our dataset.

**Step 1:** Generate the words for each domain using the previously mentioned word generator (See Figure 8).



computer science

examples: writer, understanding, cloud

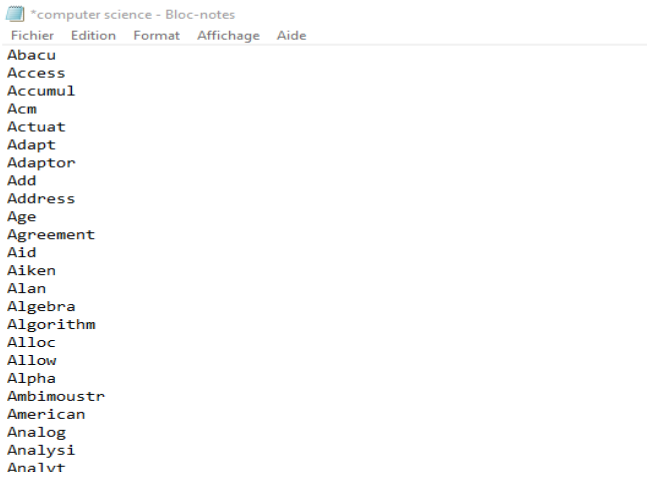
Here are some words that are associated with **computer science**. You can get the definitions of these computer science related words by clicking on them. Also check out describing words for computer science and find more words related to computer science using ReverseDictionary.org

computer | software | computer hardware | mathematics | algorithm
computation | science | information | computing | processor | cpu
firmware | computer circuit | hardware | security system | digital computer
peripheral | mainframe | computer architecture | software engineering
engineering | information theory | information science | code | logic
computer system | technology | biocomputing | vm | cyberpsychology
interface | telnet | softmodem | pseudocode | minicomputer
microprocessor | supercomputer | pixel | screensaver | anticomputer
cyberintrusion | microcomputer | information processing system
computing machine | computing device | data processor
electronic computer

RELATED WORDS CONTINUE AFTER ADVERTISEMENT

**Figure 8.** The results of words from the computer field found by the Related Words generator

**Step 2:** In this step we remove the stop words and find the root (Stemmer) of each word, classify them, put the first letter in capitals finally save them in a text file. The Figure 9 bellow shows the words of the computer science class.

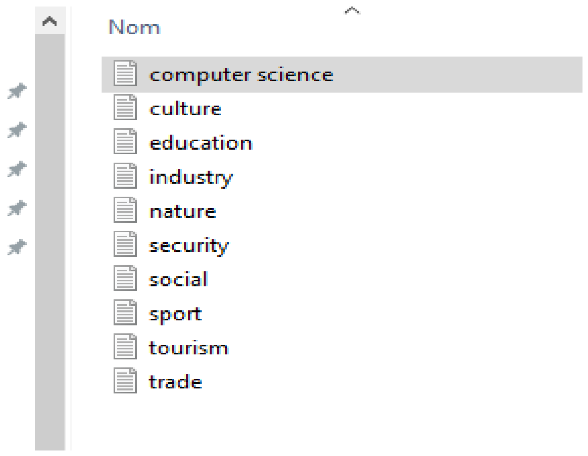


*computer science - Bloc-notes
Fichier Edition Format Affichage Aide

```
Abacu
Access
Accumul
Acm
Actuat
Adapt
Adaptor
Add
Address
Age
Agreement
Aid
Aiken
Alan
Algebra
Algorithm
Alloc
Allow
Alpha
Ambimoustr
American
Analog
Analysi
```

**Figure 9.**Text file contains the words of our computer science class

We repeat this process for all the class of our dataset. In this study we have chosen 10 class (Cf. Figure 10).



Nom

computer science
culture
education
industry
nature
security
social
sport
tourism
trade

**Figure 10.** The domain (class) of our dataset

### 4.2.2 Classification with recurrent neural network

Once our Dataset is prepared, we aim to classify the target web pages. To do this, we calculate their keyword frequencies by using the TF-IDF technique. Then we classify these web pages according to the keyword that has the highest frequency using recurrent neural networks.

We will build and train a basic RNN to rank web pages according to keywords. A character-level of RNN reads words as a series of characters, produces a "hidden state" prediction at each step and, feeding its previous hidden state into each next step. We consider the final prediction to be the output, i.e. which class the word belongs to. Specifically, we train our RNN on a few thousand words from 10 domains and predict which domain a word is from based on spelling.

The data/names directory includes 10 text files named “[Domain].txt”. Each file contains a set of words, one word per line. As result, a dictionary of lists of words per domain, {domain: [word ...]}. The generic variables "category" and "line" are used for further extensibility.

At this stage, we obtain a category_lines, a dictionary mapping each category (domain) to a list of lines (words). We also kept track of all_categories (just alist of domains) and n_categories for future reference.

*Transformation words into tensors*

Once all the words areobtained, we need to represent them into tensors to use them as an input for the RNN. So, to represent a single letter, we use a “single vector” of size <1 x n_letters>. This vector is composed of a set of 0, exceptat the index of the current letter which is equal to 1, e.g. "b" = <0 1 0 0 0 ...>. Thus the word is represented by a matrix <word_length x n_letters>.

*Creation of the network*

The input and the output of RNN are a fixed length tensors. The output length is equal to the classes number. The input is a word, and it is represented by the above cited matrix,

We divide the structure of the proposed RNN into three layers: the input layer, the hidden layer and the output layer.

- The input layer creates an input for the hidden layer. At each execution, the RNN considers as the input the word and the hidden state. The word is represented by the above-cited matrix; and the hidden state is a tensor of size <1 x n>. The maximum value of n is 128. Since the RNN takes a single tensor as input, we simply combine between the matrix and the hidden state to form the combined tensor.

- The hidden layer performs a linear Input-Hidden transformation on the combined tensor to create an input for the output layer. So, the result of this layer is the prediction and the next hidden state.

- The result of the output layer is the class prediction. To do this, we perform the softmax on the prediction to normalize the Input-Output values between [0..1] to obtain the multi-class probabilities. For example, the value 0.01 indicates that there is a 1% probability that the word belongs to the Computer class. This output is directly compared to our target tensor, which its value is between [1..0]. This allows us to calculate the loss for each prediction.

To release the RNN, we inherit from nn.Module which is the base class for all neural networks in PyTorch. We initialize an instance specifying the input/output/hidden state sizes which help us to create the line layers and the softmax function.

*RNN Training*

To train the RNN we convert at first the training pairs to tensors and feed them into the RNN. Then, we use the optimizer, loss function and learning rate to train our RNN. Based on our input tensors and our target tensors we update the network weights at each step (i.e. backpropagation). For this, we use the loss function to calculate the gradients based on the difference between a prediction and the true value. Next, we need to specify an optimizer and a learning rate to update the network.

We train our neural network until we get the right classification result. This process takes several hours of training. below, we present the main functions and parameters used to build and train our RNN.

- Loss function: NLLLoss(x): The Negative Log Likelihood Loss function, generally named NLLLoss(x) function, allows training a classification problem with C classes. The parameter x is an optional argument, if it must be provided, it should be an 1D Tensor assigning weight to each of the classes. This is particularly useful in the case of an unbalanced training set. In our case, this argument is null.

- Optimizer function: torch.optim.SGD (rnn.par(), lr). The SGD() (SGD: Stochastic Gradient Descent) optimizer function belong to torch.optim package. This latter groups several optimizer methods. In this study, we have choose to use the SGD() function because it is most adapted to optimize the RNN. The parameters of this function are: rnn.par() and lr. The first one concerns RNN parameters to be optimized (in our case, these parameters are null). However, the second parameter concerns the learning rate, which is equal to 0.0002 (lr= 0.0002).

A learning step uses an input word and its corresponding label. Each step:

- Set model in training mode.

- Create the input tensor from the title and the target tensor from the label

- Create an initial hidden state (full of zeros)

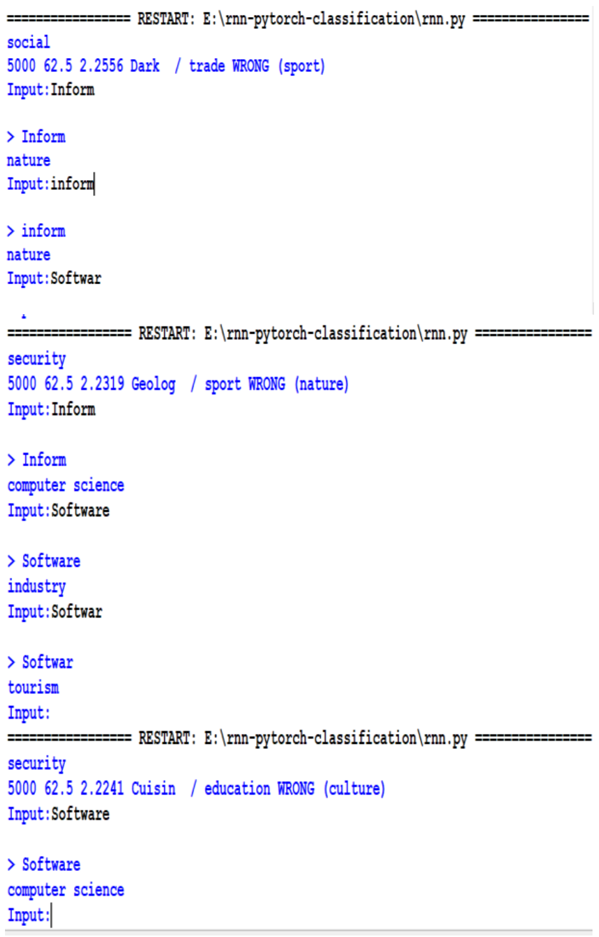
- Feed the word across the network, passing hidden states at runtime

- Calculate the loss by comparing it to the true value using the loss function

- Update network settings with optimizer

- Returns output and loss to show how the network is learning

According to obtained TF-IDF values from the extraction part (Cf. Figure 7), the RNN classify the first and the second pages using respectively the words “inform” and “software” (Cf. Figure 11), since they have the highest TF-IDF value, to the computer science class (Cf. Figure 10).



```
================ RESTART: E:\rnn-pytorch-classification\rnn.py ===============
social
5000 62.5 2.2556 Dark  / trade WRONG (sport)
Input:Inform

> Inform
nature
Input:inform

> inform
nature
Input:Softwar

================ RESTART: E:\rnn-pytorch-classification\rnn.py ===============
security
5000 62.5 2.2319 Geolog  / sport WRONG (nature)
Input:Inform

> Inform
computer science
Input:Software

> Software
industry
Input:Softwar

> Softwar
tourism
Input:
================ RESTART: E:\rnn-pytorch-classification\rnn.py ===============
security
5000 62.5 2.2241 Cuisin  / education WRONG (culture)
Input:Software

> Software
computer science
Input:
```

**Figure 11.** RNN training

## 5. CONCLUSIONS

The web pages classification is generally based on the textual content extraction and then, to classify the needed web page from the extracted words. In this context fails our proposed work. Thus, this paper is mainly based on the combination of two methods based on the supervised and unsupervised techniques. In the first part of our contribution, the automatic word extraction task is proposed. It consists to analyze a web page to extract the most representative word from this target web page using an unsupervised statistical technique, which is TF-IDF.

However, in the second part we have proposed supervised recurrent neural networks, which allows classifying web pages according to words obtained the first part. The obtained results show that the proposed approach give a good performance by classifying effectively the target web pages.

As future work, we propose to add other languages to make the system multi-language, to integrate other techniques and methods of supervised classification. In addition, TF-IDF cannot check words' semantic in documents. Therefore, it is only useful at the lexical level. It is also unable to detect words having the same root, i.e., having the same semantics. To deal with this problem, we aim to improve TF-IDF technique by a semantic verification step based on the stemming technique principle.

## REFERENCES

[1] Qi, X., Davison, B.D. (2009). Web page classification: Features and algorithms. ACM Computing Surveys (CSUR), 41(2): 1-31. https://doi.org/10.1145/1459352.1459357

[2] Chen, R.C., Hsieh, C.H. (2006). Web page classification based on a support vector machine using a weighted vote schema. Expert Systems with Applications, 31(2): 427-435. https://doi.org/10.1016/j.eswa.2005.09.079

[3] Buber, E., Diri, B. (2019). Web page classification using RNN. Procedia Computer Science, 154: 62-72. https://doi.org/10.1016/j.procs.2019.06.011

[4] Selamat, A., Omatu, S. (2004). Web page feature selection and classification using neural networks. Information Sciences, 158: 69-88. https://doi.org/10.1016/j.ins.2003.03.003

[5] Zhang, K., Xu, H., Tang, J., Li, J. (2006). Keyword extraction using support vector machine. In International Conference on Web-Age Information Management, pp. 85-96. https://doi.org/10.1007/11775300_8

[6] Jiang, X., Hu, Y., Li, H. (2009). A ranking approach to keyphrase extraction. In Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval, pp. 756-757. https://doi.org/10.1145/1571941.1572113

[7] Hashemi, M. (2020). Web page classification: a survey of perspectives, gaps, and future directions. Multimedia Tools and Applications, 79(17): 11921-11945. https://doi.org/10.1007/s11042-019-08373-8

[8] Aydos, F., Özbayoğlu, A.M., Şirin, Y., Demirci, M.F. (2020). Web page classification with Google Image Search results. arXiv preprint arXiv:2006.00226. https://doi.org/10.48550/arXiv.2006.00226

[9] Alamelu, M., Santhosh, K. (2011). A novel approach for web page classification using optimum features. International Journal of Computer Science and Network Security, 11(5): 252-257.

[10] Su, A.J., Hu, Y.C., Kuzmanovic, A., Koh, C.K. (2014). How to improve your search engine ranking: Myths and reality. ACM Transactions on the Web (TWEB), 8(2): 1-25. https://doi.org/10.1145/2579990

[11] Li, Y., Nie, X., Huang, R. (2018). Web spam classification method based on deep belief networks. Expert Systems with Applications, 96: 261-270. https://doi.org/10.1016/j.eswa.2017.12.016

[12] Duari, S., Bhatnagar, V. (2020). Complex network based supervised keyword extractor. Expert Systems with Applications, 140: 112876. https://doi.org/10.1016/j.eswa.2019.112876

[13] Balim, C., Özkan, K. (2019). Functional classification of web pages with deep learning. In 2019 27th Signal Processing and Communications Applications Conference (SIU), Sivas, Turkey, pp. 1-4. https://doi.org/10.1109/SIU.2019.8806240

[14] Salminen, J., Corporan, J., Marttila, R., Salenius, T., Jansen, B.J. (2019). Using machine learning to predict ranking of webpages in the gift industry: Factors for search-engine optimization. In Proceedings of the 9th International Conference on Information Systems and Technologies, pp. 1-8. https://doi.org/10.1145/3361570.3361578

[15] Lee, J.H., Yeh, W.C., Chuang, M.C. (2015). Web page classification based on a simplified swarm optimization. Applied Mathematics and Computation, 270: 13-24. https://doi.org/10.1016/j.amc.2015.07.120

[16] El-Hajj, W., Hajj, H. (2022). An optimal approach for text feature selection. Computer Speech & Language, 74: 101364. https://doi.org/10.1016/j.csl.2022.101364

[17] Yu, Y. (2022). Web page classification algorithm based on deep learning. Computational Intelligence and Neuroscience, 2022: 9534918. https://doi.org/10.1155/2022/9534918

[18] Ding, Z., Zhang, Q., Huang, X.J. (2011). Keyphrase extraction from online news using binary integer programming. In Proceedings of 5th International Joint Conference on Natural Language Processing, pp. 165-173.

[19] Schütze, H., Manning, C.D., Raghavan, P. (2008). Introduction to Information Retrieval. Cambridge: Cambridge University Press.

[20] Bird, S. (2006). NLTK: the natural language toolkit. In Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions, pp. 69-72.

[21] Patel, D., Patel, M., Dangar, Y. (2015). A survey of different stemming algorithm. Int. J. Adv. Eng. Res. Dev., 2(6): 50-53.

[22] Larkey, L.S., Ballesteros, L., Connell, M.E. (2002). Improving stemming for Arabic information retrieval: light stemming and co-occurrence analysis. In Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 275-282. https://doi.org/10.1145/564376.564425

[23] Prabhu, S., Vēṅkaṭēcan̲, N. (2007). Data Mining and Warehousing. New Age International.

[24] Sutskever, I. (2013). Training Recurrent Neural Networks. University of Toronto Toronto, ON, Canada.

[25] Tang, D., Li, C., Ji, X., Chen, Z., Di, F. (2019). Power load forecasting using a refined LSTM. In Proceedings

of the 2019 11th International Conference on Machine Learning and Computing, pp. 104-108. https://doi.org/10.1145/3318299.3318353

[26] Bouktif, S., Fiaz, A., Ouni, A., Serhani, M.A. (2018). Optimal deep learning LSTM model for electric load forecasting using feature selection and genetic algorithm: Comparison with machine learning approaches. Energies, 11(7): 1636. https://doi.org/10.3390/en11071636

[27] Patel, R.B., Patel, M.R., Patel, N.A. (2020). Electrical load forecasting using machine learning methods, RNN and LSTM. Journal of Xidian University, 14(4): 1376-1386.

[28] Sutskever, I., Martens, J., Hinton, G. E. (2011). Generating text with recurrent neural networks. in ICML.

[29] Zaremba, W., Sutskever, I., Vinyals, O. (2014). Recurrent neural network regularization. arXiv Prepr. arXiv1409.2329. https://doi.org/10.48550/arXiv.1409.2329

## NOMENCLATURE

| | |
|---|---|
| RNN | recurrent neural networks |
| TF-IDF | Term Frequency - Inverse Document Frequency |