Document Summarization using a Structural Metrics Based Representation [1]

Augusto Villa-Monte [a,*], Laura Lanzarini [a], Julieta Corvi [a] and Aurelio F. Bariviera [b,**]
[a] *Instituto de Investigación en Informática LIDI (Centro CICPBA), Facultad de Informática, Universidad Nacional de La Plata, 50 y 120 S/N 1900 La Plata, Buenos Aires, Argentina*
E-mails: {avillamonte, laural}@lidi.info.unlp.edu.ar, julieta.corvi@gmail.com
[b] *Universitat Rovira i Virgili, Department of Business, Av. Universitat 1 43204 Reus, Spain*
E-mail: aurelio.fernandez@urv.cat

Abstract: Currently, each person produces 1.7MB of information every second in different formats. However, the vast majority of information is text. This has increased the interest to study techniques to automate the identification of the relevant portions of text documents in order to offer as a result an automatic summary. This article presents a technique to extract the most representative sentences of a document taking into account by the user's criteria. These criteria are learned using a neural network, from a minimum set of documents whose sentences have been rated by the user in terms of importance. To verify the performance of the proposed methodology, we used 220 scientific articles from the PLOS Medicine journal published between 2004 and 2016. The results obtained have been very satisfactory.

Keywords: Text Summarization, Extractive Summaries, Sentence Scoring, Feature Selection, Neural Networks

1. Introduction

At present, data generation has become a natural output of human activity. Due to technological progress many usual actions are recorded. For this reason, the efficient access and use of such data have become fundamentally necessary in all areas, in order to construct pieces of information.

Today, there are many areas interested in extracting knowledge from stored information, and even more so in the case of unstructured information. Although data is continuously generated in many formats, most of the digital information is stored in text format. For example, each email sent, each search made on the Internet or each publication generates, to a greater or lesser extent, textual data.

In general, text is stored in the form of unstructured digital documents, using a very different organization from that of traditional databases. Scientific literature is characterized by producing an enormous amount of text documents that cover all the thematic areas of human knowledge.

The use of automatic tools to summarize text is essential, since it could facilitate the access to pertinent and available information. When the volume of information is immense, separating manually what is essential is a difficult and time consuming task. The development of computational solutions to summarize text constitutes one of the current lines of research aiming at reducing the problems generated by the excessive growth of textual information.

Obtaining computer summaries reduces the enormous volume of unstructured information to its core content, in order to facilitate its manipulation. By reading an automatic summary, the user could get access to the main content of a document in less time than it would had taken after reading the original document.

Although the definition of summary may vary from one author to another, we could fairly agree to define the summary as a reduced, clear and objective repre-

[*] Post-Doctoral Fellow at National University of La Plata.
[**] Corresponding author. E-mail: aurelio.fernandez@urv.cat

sentation of the core and essential contents of a document.

There are different approaches to automatically summarizing text documents. The extractive approach is one of the most used in literature. This type of summary is formed by certain parts of the original text (single words or whole paragraphs), which are extracted verbatim. This extraction is usually done by means of scoring metrics that allow ranking the parts of a document and selecting properly the ones that will constitute the final summary.

This work aims to learn the criteria used by a person when summarizing a text. This will be achieved through the selection of the subset of sentence scoring methods that best approximate human summarization. Once this criterion has been obtained, it can be applied to other documents in order to obtain, as a result, a summary similar to that which the user would have made manually.

The remaining of the paper is organized as follows. Section 2 contains an overview of the current state of the art. Section 3 develops the proposed methodology and describes the document representation used in this paper. Section 4 presents the results. Finally, Section 5 draws the main conclusions and future research lines.

2. Related works

Automatic abstract generation is a subject that began in 1958 with the seminal work by Luhn [1] and research continues at present [2]. Throughout all these years, different approaches have emerged for the automatic generation of document abstracts. There are two main approaches to this topic: extractive and abstractive automatic summaries

Extractive summary studies are predominant in literature [3]. This approach selects portions of a document without requiring a complex semantic analysis, as opposed to the one required by the abstractive approach. On the contrary, abstractive summaries are based on the "ideas" developed in the document and do not use exact phrases from the original document. This type of summary involves generating new text, providing a higher level of generalization. Taking into account the required linguistic knowledge (such as ontologies, thesauri and dictionaries) required by abstractive summaries, the cost of the extractive approach is lower. This is the reason why it is more frequently investigated [4]. In [5] the differences between both approaches are clearly specified.

Since an extractive summary is formed by a set of portions of text (from single words to whole paragraphs) literally copied from the source text, it is generally considered a classification problem of two classes [6]. Each part of the document is labeled either as "correct " (if it is part of the summary), or "incorrect " (if it is not included in it) [7].

In order to perform such classification, a score is usually assigned to each part of the document [8]. This score allows ranking the parts of a document [9]. Such score is achieved using a set of metrics. Each metric analyzes a particular characteristic of the document. We will review the most frequently used metrics.

The first algorithm for extractive summaries was developed in 1958. It obtained a simple summary by weighing the sentences from the frequency distribution of words [1]. At the same time, the position of the sentences was used [10]. A few years later, based on the work by Luhn, it was proposed to use the presence of certain words and the coincidences with the words in the titles [11]. Many years later, the frequency of the terms and the *Inverse Sentence Frequency* (adaptation of the TF-IDF measure used in *Information Retrieval*) [12] were used. The use of *Latent Semantic Analysis* (LSA) was also proposed to summarize the text, without using lexical resources such as *WordNet* [13]. Additionally, graph-based models were introduced for the extraction of sentences [14]. Furthermore, another method was based on the proportion of key words present in the sentences and in the document [15]. Meanwhile the idea of using the average word frequency for the selection of sentences was also put into practice [16]. These are just some of the most frequently used metrics in literature. A detailed overview of these and other metrics and methods can be consulted in [17].

The literature related to extractive abstracts is very extensive. In most works, a set of metrics are used to characterize a document and build its intermediate representation [18]. Each metric analyzes a given characteristic of the document and applies certain classification criteria to its content. These metrics are employed after the preprocessing of the document, which implies carrying out the following tasks: splitting the document into sentences, tokenizing each of them, discarding stop words, applying stemming, etc.

Recent works consider the task of producing extractive summaries as an optimization problem, where one or more objective functions are formulated in order to select the "best" sentences in the document that will be part of the summary [19]. However, documents in

those works are characterized by a set of *a priori* defined metrics, and their selection is not part of the optimization process, as is the case of, e.g. [20].

In [21], the use of an optimization technique was proposed as a solution to this problem. In that work a strategy was developed to assist the user in the task of summarizing, offering as a result an extractive summary formed by the most representative sentences according to a previously learned criterion. The learning of said criterion was based on a short summary submited by the user. The results obtained showed that the proposed strategy was effective. However, all metrics were considered in the construction of the summary.

In this work, it is proposed to train a neural network to select the most representative metrics to use and determinate its level of participation in the construction of the summary. The final score assigned to each part of the document will be as accurate as possible in relation to the user's preferences.

3. Proposed methods

It is important to remember that the extractive approach chosen to generate the summary does not guarantee the narrative coherence of the selected sentences. However, it is possible to reduce the size of the document leaving only the relevant content.

This approach has three advantages: (i) the size of the summary can be controlled. (ii) the content of the summary is obtained accurately. (iii) the relationship between the summary and the original text is immediate.

In general terms, the process of summarizing by extracting parts of a text consists of three major stages:(i) create an intermediate representation for the original text, (ii) assess each of the sentences through a score and, (iii) select the set of sentences that will be part of the summary.

Figure 1 shows a diagram of the proposed methodology, which will be developed below.

3.1. Text Pre-processing

Any task of *Text Mining* begins with the preprocessing stage. Within this stage, the most important task is the segmentation of the text and its subsequent representation. Segmentation begins with dividing the text into smaller portions preserving the order they appear in, and with certain criteria using one or several delimiters. Generally, these portions of the text are called "sentences" in a generic way, and they are obtained by recognizing orthographic signs such as punctuation marks (period, full stop, comma, and semicolon). In the case of the point, the sentences represent syntactical sentences.

A syntactical sentence is a unit of language that makes sense by itself and that is formed by a verb. Other signs can also be used, such as comma, semicolon, colon, and parentheses. Their interpretation requires special considerations such as the period, with its multiple uses: abbreviations, acronyms and numbers.

Once the sentences are available, they are divided into terms (*tokenization*) and stored appropriately so that the original document can be reconstructed at any time, ensuring its integrity. The latter is essential so that each of the metrics described in the section 3.2 can be correctly calculated for each sentence. At first, all the words that appear in the text are considered as possible terms. Words are defined as any string of characters that starts the sentence, which follows a blank space or ends in a blank space, among other possible delimiters.

In this stage, in order to reduce the working vocabulary, it is common to eliminate some terms with certain criteria. On the one hand, all the text is usually converted to either uppercase or lowercase since, in general, the case of the letter is not relevant for summarizing purposes. Eventually, there are some exceptions, such as proper names. On the other hand, words without semantic content, such as articles, prepositions, conjunctions, adverbs and adjectives, are usually eliminated. These are commonly known as stop words. It is also common to discard some verbs such as "to be", as well as very short words (e.g., 1 to 3 characters long words). Finally, the roots of words are extracted through a process known as *stemming*. This causes the most significant reduction of terms since the number of words derived from the same root is very high. Even though the root of a term provides greater semantic content, using it increases noise and once again generates the need to consider exceptions. Such exceptions arise not only for having terms with the same root and different meaning, but also for other situations as, for example, removing the significance of gender suffixes.

3.2. Document representation

There are many ways to characterize the sentences of a document. In this article, we select 21 metrics used previously in literature, in order to grade each sentence

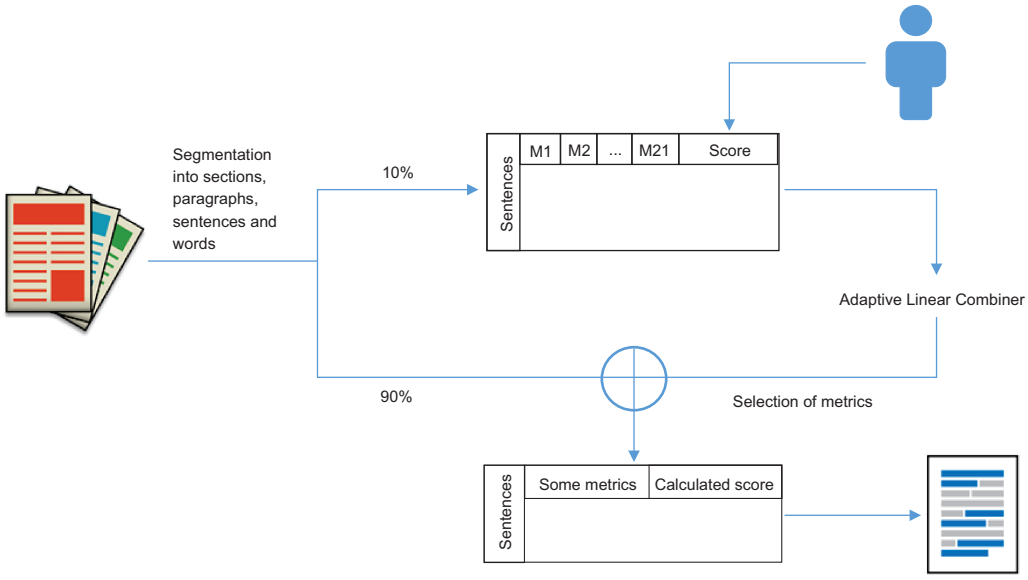

Figure 1. Scheme of the methodology proposed in this article for the generation of document summaries.

and to represent each document as a set of numerical vectors. These values are based on quantities calculated from different aspects that the sentences have in the text. In this section we give full details of the metrics considered in our work. In this way, each document will be represented by a matrix of as many rows as sentences in the document and as many columns as metrics (in this case 21).

Position. These metrics measure the closeness of the sentence S_i to the beginning, end and edges respectively, being n the total number of sentences of D and i a number between 1 and n assigned to each sentence sequentially according to its appearance within the document from start to finish [10].

$$\begin{aligned} pos\_l(S_i) &= i \\ pos\_f(S_i) &= i^{-1} \\ pos\_b(S_i) &= \max\left(i^{-1}, (S-i+1)^{-1}\right) \end{aligned}$$

Length. These metrics measure the length of the sentence S_i in terms of the number of words and characters it has [22]. $|.|$ indicates the cardinality of the set.

$$\begin{aligned} len\_w_{(S_i)} &= |words(S_i) \\ len\_ch_{(S_i)} &= |characters(S_i)| \end{aligned}$$

Frequency. In $luhn$ c_i represents the largest sequence of consecutive words that begins and ends with words considered as keywords (according to some criteria) [1]. key adds the frequencies of all the keywords that sentence S_i contains, where tf_k is the frequency of the keyword k in the document [11]. cov measures the proportion of keywords in document D and contained in sentence S_i [15]. TF calculates the average frequency of the words in sentence S_i [16]. On the other hand, $tfisf$ is an adaptation of the well-known metric *TF-IDF* used in *Information retrieval* where n_i is the number of sentences that contain the word i and $isf_w = 1 - \frac{log(n_i)}{log(n)}$ [12].

$$\begin{aligned} luhn_{(S_i)} &= \frac{|keywords(c_i)|^2}{|c_i|} \\ key_{(S_i)} &= \sum_{k \in keywords(S_i)} tf_k \\ cov_{(S_i)} &= \frac{|keywords(S_i)|}{|keywords(D)|} \\ tf_{(S_i)} &= \frac{\sum_{w \in words(S_i)} tf_w}{|words(S_i)|} \\ tfisf_{(S_i)} &= \sum_{w \in words(S_i)} tf_w \times isf_w \end{aligned}$$

Title. These metrics measure the similarity of the sentence S_i with the title T_i using three measures: overlap, Jaccard and cosine respectively. In all cases the similarity is defined through the common words that have the sentence and the corresponding title [11]. However, to calculate cosine, a two-row frequency matrix (one

for the title T_i and another for the sentence S_i), is needed. The number of columns in this matrix depends on the different words title and sentences have.

$$title\_o_{(S_i)} = \frac{|words(S_i) \cap words(T_i)|}{\min\left(|S_i|, |T_i|\right)}$$
$$title\_j_{(S_i)} = \frac{|words(S_i) \cap words(T_i)|}{|words(S_i) \cup words(T_i)|}$$
$$title\_c_{(S_i)} = \frac{\vec{S_i} \times \vec{T_i}}{|\vec{S_i}| \times |\vec{T_i}|}$$

Coverage. These metrics also use the three aforementioned measures of similarity between the sentence S_i and the remaining sentences of the document($D - S_i$) [18].

$$d\_cov\_o_{(S_i)} = \frac{|words(S_i) \cap words(D - S_i)|}{\min\left(|S_i|, |D - S_i|\right)}$$
$$d\_cov\_j_{(S_i)} = \frac{|words(S_i) \cap words(D - S_i)|}{|words(S_i) \cup words(D - S_i)|}$$
$$d\_cov\_c_{(S_i)} = \frac{\vec{S_i} \times \vec{D - S_i}}{|\vec{S_i}| \times |\vec{D - S_i}|}$$

Graphs. In graph-based metrics, the nodes of the graph represent the sentences of the document. The greater the number of links that a node has, the more relevant will be the sentence that it represents. In $graph\_s$ the weights of all the links of a node are added to conform their degree. Each weight is determined by the number of common words between two sentences. Both $luhn\_deg$, key_deg and cov_deg are extensions of the metrics $luhn$, key and cov but use the degree of the node to adapt their calculation.

3.3. Learning the user's criteria

The user's preference for a sentence is given by the assigned score. It is a positive integer value, and proportional to the estimated degree of importance. Those sentences that receive 0 as a score will be interpreted as irrelevant, while those that receive the highest values will be the most significant.

On the other hand, since there are several metrics calculated for each statement of the document, it is expected that a linear combination of them represents the user's criteria. Therefore, the problem to solve consists in finding the coefficients $c_1, c_2, \ldots, c_k$ such that applied to the metrics of the sentence S_i, $m_{i1}, m_{i2}, \ldots, m_{ik}$, allow to approximate the score indicated by the user.

To solve this, a *Adaptive Linear Combiner* was used: a neural network formed by a single neuron with as many inputs as metrics are used to represent each sentence, and a single output whose value is given by what is indicated in equation 1. The coefficient c_0 works as an independent term and the entry m_{i0} is always 1.

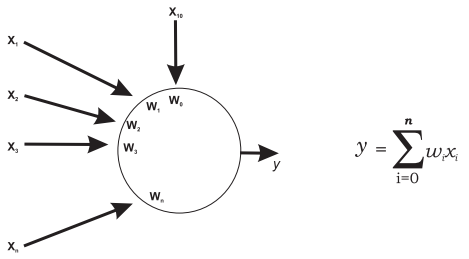

Figure 2. *Adaptive Linear Combiner.*

$$score_i = \sum_{j=0}^{k} (c_j * m_{ij}) \tag{1}$$

The weights of the arcs that join each metric with the neuron act as scale factors and regulate the importance or participation of each metric in the score. Figure 2 illustrates the architecture of a *Adaptive Linear Combinator* of n inputs.

The learning process of the neuron consists of establishing the coefficients associated with each metric represented by the weights of the arcs that join it with the input information. To achieve this, the algorithm must show a set of labeled examples; in this case, a set of sentences for which the expected score is indicated.

The method used to adapt weights is the *Least Mean Square* (LMS) defined by Widrow and Hoff in [23] and known as Delta Rule [24]. This algorithm minimizes the sum of the squares of the linear errors incurred on the set of training sentences. It starts with random coefficients $c_0, c_1, \ldots, c_k$, and iterates over the examples. For each entered example, the algorithm calculates the corresponding output. In this case, the examples are sentences represented by the metrics, so the output obtained in each case corresponds to the prediction of the corresponding score, calculated according to equation 1. Then, the error in the answer is estimated using equation 2, where P_i corresponds to the score assigned by the user to the the sentence under prediction.

$$Error_i = (P_i - score_i)^2 \tag{2}$$

Changes in the coefficients are done following the direction of the negative gradient according to equation 3, where α is a value between 0 and 1 that controls the size of the modifications to be made, also called

factor or *learning speed*.

$$c_j = c_j + \alpha * Error_i * m_{ij} \quad \forall j, 0 \leq j \leq k \quad (3)$$

This process is repeated by repeatedly entering the examples of the training set, one at a time, and modifying the coefficients after each error calculation. The end criterion is given by a threshold in the difference of two consecutive average errors or by a maximum number of iterations, whichever happens first.

Once the Linear Combiner is already trained, it is possible to identify the metrics that have the greatest participation in the prediction of sentence scores. To do this, the proportion of the metric value must be taken into account (weighted by the corresponding coefficient) with respect to the pre-stablished score for the sentence.

Considering the sentence S_i, it can be said that the total contribution provided by all the metrics to the value $score_i$, $total_i$, is given by the sum of their absolute weighted values, as indicated in the equation 4. Therefore, the proportion $degree\_part_{ij}$, indicated in the equation 5 can be used to estimate the participation of the j-th metric of the i sentence in this sum.

$$total_i = \sum_{j=1}^{k} abs(c_j * m_{ij}) \quad (4)$$

$$degree\_part_{ij} = \frac{abs(c_j * m_{ij})}{total_i} \quad (5)$$

Repeating this for each sentence of the training set and averaging the values obtained for each metric it will be obtained a vector $P = (p_1, p_2, \ldots, p_k)$ with the values of the degrees of participation of each metric in the estimation of the scores. Equation 6 indicates how the average for a set of cardinality sentences M should be performed.

$$p_j = \frac{\sum_{i=1}^{M} degree\_part_{ij}}{M} \quad (6)$$

Then, those metrics whose degree of participation exceeds the average value $\sum_j p_j/k$ will be considered as relevant for the estimation of the score.

4. Experimental results

To verify the performance of the proposed methodology, we used scientific articles extracted from the journal *PLOS Medicine*[1] published between January 2004 and December 2016. The choice of the journal was conditioned not only by its importance from the medical point of view, but also because it offers the possibility of downloading the files in different formats. In this case, documents labeled in XML format were obtained. This aspect is essential to automate the extraction of information for later upload to the database.

We extracted 223 scientific articles, each of which consisted of approximately 30 paragraphs of 4 or 5 sentences each and each sentence had an average of 20 words. A database was designed to be able to persist the documents in a suitable way for its later processing. This design can be found in [25]. Before storing them in the structure, they were preprocessed using Python's NLTK (*Natural Language Toolkit*) library which allows manipulating the natural language and simplifies tokenization, stop words removal and stemming processes, among other things.

It is important to remember at this point that the objective of this paper is to learn the criteria used by the user to select the sentences that give rise to the extractive summary. In this way, counting on a reduced number of summary documents, it will be possible to reproduce the used criteria in the remaining documents. However, in order to measure the performance of the proposed linear combiner, it is necessary to know the score of all sentences in the corpus. Manual labeling is a very tedious task for the user. For this reason it was decided to solve it automatically, using a web application. After analyzing several applications that are used for carrying out online summaries from a text, it was decided to use Online summarize tool[2] since, from those available on the Internet, it was the only one that fulfilled all the requirements: (1) each sentence corresponds to a syntactic sentence from the text of the document, (2) it allows to rank all the sentences of the document, (3) it establishes the ranking of sentences assigning a score to each one of them, and (4) it has a Web interface that could be integrated to the development carried out.

Once the whole corpus was labeled, the Linear Combiner described in the section 3.3 was trained

[1] https://journals.plos.org/plosmedicine/
[2] https://www.tools4noobs.com/summarize/

through a cross-validation of 10 parts, using in each case 10 % of the documents to learn the criteria and the 90 % remaining to test if the criterion found allows to obtain the expected summary.

Since the value of the coefficients obtained through the linear combiner depends on the random values with which they are initialized, 40 independent runs were made of each case and the results obtained were averaged.

Before starting each one of the trainings, the values of the metrics were linearly scaled between 0 and 1

Table 1 summarizes the results obtained after carrying out a cross-validation of 10 parts. The values correspond to the average of 40 independent runs for each threshold cut. Four variants of the linear combiner were taken into account:

- **CL-All**: The linear combiner was trained using the 21 metrics. This neuron is the one that received the most information.
- **CL-Corr**: The Linear Combiner was trained using only the 14 metrics that presented a correlation lower than 0.85. This threshold was established empirically.
- **CL-Pruned**: For each partition the metric selection criteria was applied according to the degree of participation indicated in section 3.3. With this information, the entries corresponding to the unselected metrics of the combiner used in CL-All were removed and applied to the set of corresponding test sentences. Note that the evaluation process of the linear combiner is made up of 40 independent runs of 10 parts each. In other words, this selection process was carried out 400 times.
- **CL-Sel + Train**: This variant uses the same selection as in CL-Pruned, but instead of using the CL-All weights associated with the selected metrics, a new Linear Combiner was re-trained with those inputs. The maximum number of iterations used in the training was 200.
- **CL-Selecc**: The degrees of participation of all the sentences of the corpus in each execution were averaged, and a single selection of metrics was made. With this information, a single linear combiner was trained. Then it was applied to all partitions. This is the best solution. However, it is feasible only in case of having the summary of all the corpus. Therefore, it was only evaluated as a benchmark.

Table 1 and figure 3 detail the performance of the different variants of the linear combiner. It can be seen

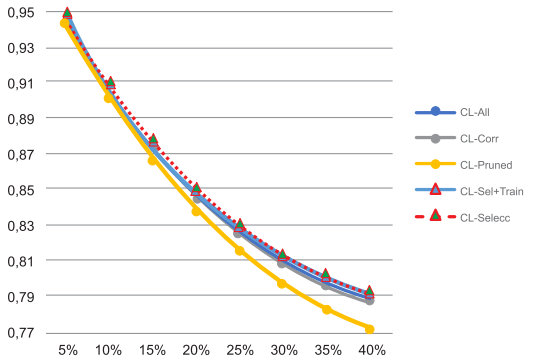

Figure 3. Average hit validation rate obtained with the proposed method during cross-validation.

that, except for the option CL-Pruned, the rest offers similar results. Figure 4 shows the confidence intervals corresponding to an ANOVA test performed with a confidence level of 95%. Although the graphs correspond to the 5 % cut, the results do not present variants in the other options.

With respect to the selection of metrics, although it is observed that it depends on the set of documents that are considered in the training, a common core formed by the metrics pos_l, pos_f, len_w, len_ch, tfisf, d_cov_j, graph_s, graph_e and luhn_deg is identified.

These metrics were present in the vast majority of the resulting sets. Figure 5 illustrates the average degree of participation of each metric, calculated according to equation 6. Figure 6 shows the number of times each metric was selected, considering each of the 10 partitions in the 40 runs. In each case, those with higher than average degree of participation were selected. The average number of metrics selected was 9, i.e. less than 50% of the total number of metrics.

Table 2 shows that there is no significant difference between the alternatives CL-All, CL-Corr, CL-Sel + Train and CL-Selecc regarding the hit rate. However, the number of metrics used in each case was 21, 14, approximately 9 and 9, respectively. The solution offered by CL-Pruned reaches the lowest performance, although its average difference with respect to CL-All is 0.0018. Considering that its use implies not having to retrain the linear combiner before using it, since it is a direct pruning of the CL-All tree, it is not a bad option.

Table 1
Average hit rate obtained with the proposed method during cross-validation.

Cut	Used metrics				
	CL-All	CL-Corr	CL-Pruned	CL-Sel+Train	CL-Selecc
10%	$0.9086_{\pm 0.0038}$	$0.9064_{\pm 0.0022}$	$0.9026_{\pm 0.0105}$	$0.9097_{\pm 0.0022}$	$0.9107_{\pm 0.0015}$
15%	$0.8751_{\pm 0.0054}$	$0.8729_{\pm 0.0033}$	$0.8669_{\pm 0.0152}$	$0.8768_{\pm 0.0028}$	$0.8784_{\pm 0.0018}$
20%	$0.8475_{\pm 0.0068}$	$0.8452_{\pm 0.0041}$	$0.8373_{\pm 0.0194}$	$0.8498_{\pm 0.0033}$	$0.8513_{\pm 0.0021}$
25%	$0.8267_{\pm 0.0079}$	$0.8249_{\pm 0.0044}$	$0.8150_{\pm 0.0226}$	$0.8290_{\pm 0.0033}$	$0.8304_{\pm 0.0022}$
30%	$0.8096_{\pm 0.0088}$	$0.8084_{\pm 0.0053}$	$0.7963_{\pm 0.0258}$	$0.8124_{\pm 0.0037}$	$0.8138_{\pm 0.0022}$
35%	$0.7978_{\pm 0.0099}$	$0.7960_{\pm 0.0057}$	$0.7829_{\pm 0.0287}$	$0.8011_{\pm 0.0042}$	$0.8030_{\pm 0.0025}$
40%	$0.7880_{\pm 0.0103}$	$0.7864_{\pm 0.0058}$	$0.7725_{\pm 0.0301}$	$0.7913_{\pm 0.0042}$	$0.7932_{\pm 0.0024}$

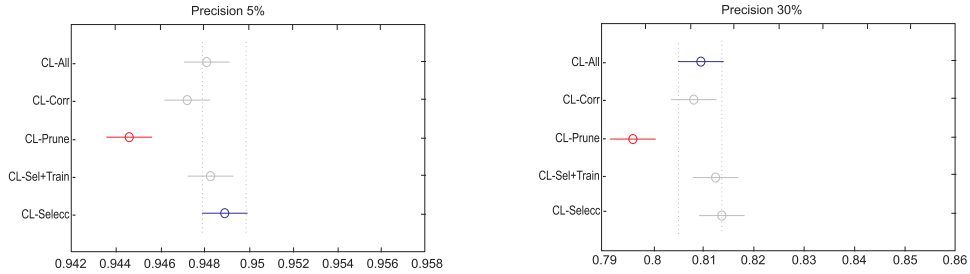

Figure 4. 95% Confidence intervals corresponding to mean equality ANOVA test.

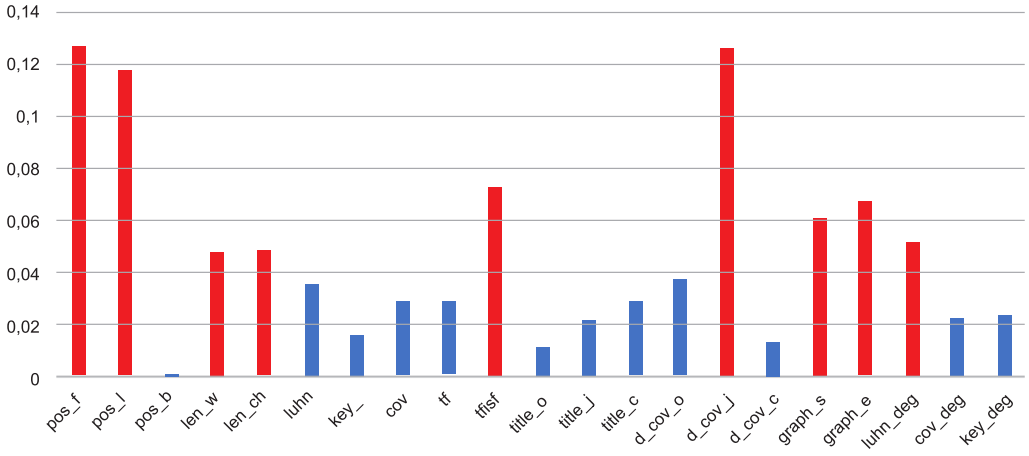

Figure 5. Degree of participation of each metric. Those with a higher than average value are selected (in this case average is 0.0476).

5. Conclusions and future research lines

We presented a technique that is able to learn the user's criteria to select the most representative sentences of a document. In other words, we showed that it is possible to make an extractive summary with the most relevant parts according to a given criterion, through a threshold.

Experimental results show that the proposed method is effective. The combination of scoring methods with a Linear Combiner allows identifying the user's most employed metrics at the time of summarizing.

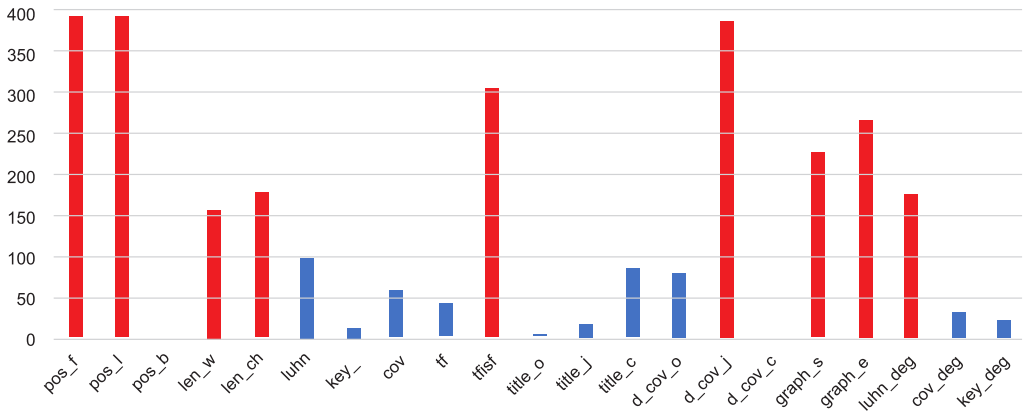

Figure 6. Total number of times each metric was selected considering each of the 10 partitions of the 40 executions. In each case, those with a higher than average participation degree were selected (red).

Table 2

Result of ANOVA test, with $p - value < 0.05$. Non significant difference between average precision of two combiners in indicated by $-$. Symbols $\triangle$ ($\triangledown$) indicate that there are significant differences, meaning that the row combiner is better (worse) that the column combiner.

	CL-Corr	CL-Pruned	CL-Sel+Train	CL-Seleccion
CL-All	$\triangle$	–	–	–
CL-Corr	–	$\triangledown$	$\triangledown$	$\triangledown$
CL-Pruned		–	–	–
CL-Sel+Train			–	–

It is important to note that the summary's quality depends on the combination of the sentence scoring methods that are used.

The performance comparison between the selected metrics and the complete set of metrics gives almost equal results in terms of identifying up to 20 % of the most significant sentences according to the user's criteria. This allows to affirm that the selection made has been correct.

In the future, it is proposed to scrutinize if there is a relationship between the selected metrics and the type of document or the theme. It could also be interesting to analyze the importance of the language in which the document is written, as well as the style used by the author. Finally, these same tests will be repeated using other markings for the documents, including not only the new ones that can be generated automatically but also other manually constructed ones.

References

[1] H. P. Luhn, "The automatic creation of literature abstracts," *IBM Journal of Research and Development*, vol. 2, pp. 159–165, Apr 1958.

[2] M. Gambhir and V. Gupta, "Recent automatic text summarization techniques: a survey," *Artificial Intelligence Review*, vol. 47, pp. 1–66, Jan 2017.

[3] V. Gupta and G. Lehal, "A survey of text summarization extractive techniques," *Journal of Emerging Technologies in Web Intelligence*, vol. 2, 08 2010.

[4] I. Mani, *Automatic Summarization*. Natural language processing, J. Benjamins Publishing Company, 2001.

[5] U. Hahn and I. Mani, "The challenges of automatic summarization," *Computer*, vol. 33, pp. 29–36, Nov 2000.

[6] I. Mani, "Summarization evaluation: An overview," 06 2002.

[7] J. L. Neto, A. A. Freitas, and C. A. A. Kaestner, "Automatic text summarization using a machine learning approach," in *Advances in Artificial Intelligence* (G. Bittencourt and G. L. Ramalho, eds.), (Berlin, Heidelberg), pp. 205–215, Springer Berlin Heidelberg, 2002.

[8] A. Nenkova and K. McKeown, "A survey of text summarization techniques," in *Mining Text Data*, 2012.

[9] H. P. Edmundson and R. E. Wyllys, "Automatic abstracting and indexing—survey and recommendations," *Commun. ACM*, vol. 4, pp. 226–234, May 1961.

[10] P. B. Baxendale, "Machine-made index for technical literature: An experiment," *IBM Journal of Research and Development*, vol. 2, no. 4, pp. 354–361, 1958.

[11] H. P. Edmundson, "New methods in automatic extracting," *Journal of the ACM*, vol. 16, no. 2, pp. 264–285, 1969.

[12] J. Larocca Neto, A. D. Santos, C. A. A. Kaestner, and A. A. Freitas, "Generating text summaries through the relative importance of topics," in *Advances in Artificial Intelligence*, vol. 1952 of *Lecture Notes in Computer Science*, pp. 300–309, Springer Berlin Heidelberg, 2000.

[13] Y. Gong and X. Liu, "Generic text summarization using relevance measure and latent semantic analysis," in *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, pp. 19–25, 2001.

[14] R. Mihalcea, "Graph-based ranking algorithms for sentence extraction, applied to text summarization," in *Proceedings of the ACL 2004 on Interactive Poster and Demonstration Sessions*, ACLdemo '04, 2004.

[15] J. K. Fatma, J. Maher, B. H. Lamia, and B. H. Abdelmajid, "Summarization at LARIS Laboratory," in *Proceedings of the Document Understanding Conference*, DUC'04, 2004.

[16] L. Vanderwende, H. Suzuki, C. Brockett, and A. Nenkova, "Beyond sumbasic: Task-focused summarization with sentence simplification and lexical expansion," *Information Processing Management*, vol. 43, pp. 1606–1618, 11 2007.

[17] J. M. Torres Moreno, *Automatic Text Summarization*. Cognitive science and knowledge management series, Wiley, 2014.

[18] M. Litvak, M. Last, and M. Friedman, "A new approach to improving multilingual summarization using a genetic algorithm," in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pp. 927–936, 2010.

[19] E. Vázquez, R. Arnulfo García-Hernández, and Y. Ledeneva, "Sentence features relevance for extractive text summarization using genetic algorithms," *Journal of Intelligent & Fuzzy Systems*, vol. 35, no. 1, pp. 353–365, 2018.

[20] Y. K. Meena and D. Gopalani, "Evolutionary algorithms for extractive automatic text summarization," *Procedia Computer Science*, vol. 48, pp. 244 – 249, 2015.

[21] A. Villa Monte, L. Lanzarini, L. Rojas Flores, and J. A. Olivas, "Document summarization using a scoring-based representation," in *2016 XLII Latin American Computing Conference*, pp. 1–7, 2016.

[22] C. Nobata, S. Sekine, M. Murata, K. Uchimoto, M. Utiyama, and H. Isahara, "Sentence extraction system assembling multiple evidence," in *Proceedings of the Second NTCIR Workshop Meeting*, 2001.

[23] B. Widrow and M. E. Hoff, "Neurocomputing: Foundations of research," ch. Adaptive Switching Circuits, pp. 123–134, Cambridge, MA, USA: MIT Press, 1988.

[24] D. E. Rumelhart, J. L. McClelland, and C. PDP Research Group, eds., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*. Cambridge, MA, USA: MIT Press, 1986.

[25] A. Villa Monte, J. Corvi, L. Lanzarini, C. Puente, A. Simon Cuevas, and J. A. Olivas, "Text pre-processing tool to increase the exactness of experimental results in summarization solutions," in *Proceedings of the XXIV Argentine Congress of Computer Science*, pp. 481–490, 2018.