Text pre-processing tool to increase the exactness of experimental results in summarization solutions

Augusto Villa Monte[1], Julieta Corvi[2], Laura Lanzarini[1], Cristina Puente[3], Alfredo Simón Cuevas[4], and José A. Olivas[5]

[1] Institute of Research in Computer Science LIDI
[2] Faculty of Computer Science, National University of La Plata
La Plata, Buenos Aires, Argentina
{avillamonte,laural}@lidi.info.unlp.edu.ar, julieta.corvi@gmail.com
[3] Advanced Technical Faculty of Engineering ICAI
Comillas Pontifical University, Madrid, Spain
cristina.puente@icai.comillas.edu
[4] Faculty of Computer Engineering, Technological University of Havana
"José Antonio Echeverría", La Habana, Cuba
asimon@ceis.cujae.edu.cu
[5] Department of Information Technologies and Systems
University of Castilla-La Mancha, Ciudad Real, Spain
joseangel.olivas@uclm.es

Abstract. For years, and nowadays even more because of the ease of access to information, countless scientific documents that cover all branches of human knowledge are generated. These documents, consisting mostly of text, are stored in digital libraries that are increasingly consenting access and manipulation. This has allowed these repositories of documents to be used for research work of great interest, particularly those related to evaluation of automatic summaries through experimentation. In this area of computer science, the experimental results of many of the published works are obtained using document collections, some known and others not so much, but without specifying all the special considerations to achieve said results. This produces an unfair competition in the realization of experiments when comparing results and does not allow to be objective in the obtained conclusions. This paper presents a text document manipulation tool to increase the exactness of results when obtaining, evaluating and comparing automatic summaries from different corpora. This work has been motivated by the need to have a tool that allows to process documents, split their content properly and make sure that each text snippet does not lose its contextual information. Applying the model proposed to a set of free-access scientific papers has been successful.

Keywords: Automatic Summarization, Extractive Approaches, Web Scraping, Document Representation, Summaries Evaluation.

1 Introduction

For years, technological advances have allowed recording all types of information and easily storing it. As the use of the Internet becomes popular and storage costs are reduced, news, articles, books, tweets, e-mails, audios, images, videos, among others, are created, duplicated, stored, shared and accessed constantly. All this is possible thanks to textual data. To give an example, both the e-mails that are sent daily and the Web search queries that are made continually are nearly all text-based. Even a video uploaded to YouTube would remain largely inaccessible were it not for the title, description, and other metadata in text format. It is clear that not everything is text but text is everywhere [1]. Textual data surrounds our lives and has been growing continuously in recent years. Today, text represents one of the most valuable resources and the use of automated tools for its processing is essential.

In the scientific field, a myriad of articles is produced, published and hosted on repositories that can be easily accessed. These documents encompass all topic areas and represent the human knowledge. Separating what is essential from what is not, has proven to be a difficult task to be performed manually when the volume of information is immense. In this context and when it comes to text, obtaining summaries by computer reduces this huge volume of non-structured information to its most important content to facilitate its manipulation in an automatic way. This allows obtaining the core contents of a document in less time than of what it would take to do it manually. This is known as automatic text summarization.

Among the existing approaches, the summary by extraction is easy to develop since the program does not have to generate a new text that provides a degree of generalization to construct sentential meanings. The extractive approach is a relatively low-cost solution in relation to the additional linguistic knowledge resources specifically required (such as ontologies and dictionaries) to construct a summary by abstraction [2]. An extractive summary is a set of text portions (from single words to whole paragraphs) literally copied from the input that constitute the summary [3]. The extractive approach selects parts of a document without requiring complex semantic analysis [4] . However, to this end it is indispensable to assign a score to each part. This scoring allows to order all parts of a document from highest to lowest in a ranking list where the first positions are more relevant [5].

The specific values are obtained by metrics whose calculations depend on the formulation of certain equations. The metrics range from identifying certain expressions (e.g. “this article describes” or “finally”) or keywords within the text to more complex calculations. In many of the works on extractive summaries, documents are modeled as n-dimensional vectors of numerical features obtained by calculating n metrics. This is based on the classic vector space model proposed by Salton, Wong and Yang in 1975 [6]. Then, these feature vectors are used to obtain an automatic summary by applying more sophisticated algorithms to them [7]. However, in few works the way in which those features were calculated is developed in depth. Designing a program that selects representative and

significative phrases from documents automatically requires precise instructions [8].

In most cases, known collections of documents (such as DUC [9]) are used in the experiments but special considerations during the preprocessing of the documents are not always detailed. This does not allow to be objective when comparing results and obtaining conclusions. This paper presents a text document manipulation tool to increase the exactness of results when obtaining, evaluating and comparing automatic summaries from different corpora. This tool ensures the correctness of calculations and allows to recalculate metrics changing the "view" of the document without too much effort. This work has been motivated by the need to have a tool that allows to process documents, split their content properly and make sure that each text snippet does not lose its contextual information (necessary for the calculations of the metrics). With this tool, the documents are adequately stored in a database and can be reconstructed according to the needs of the experiment in question. This allows the integration of automatic summary generation systems that operate with significant differences on documents, something quite useful in this field.

The remaining of this paper is organized as follows. In Section 2 the main approaches for text summarization are detailed. Section 3 describes the proposed tool and how to use it. Finally, in Section 4, conclusions and some future lines of work are presented.

2 Related work

Both the Royal Spanish Academy and the Cambridge Dictionary agree in that a summary is a series of short, clear and precise statements that give the essential and main ideas about something. The automatic generation of text summaries is the process through which a "short version" of one or more documents is created using a computer (with no human intervention) where the information is kept.

Even though it started being researched many years ago, automatic text summarization is still a relevant topic that receives ongoing scientific contributions [10, 11]. There are two main types of automatic summaries: extractive and abstractive [12]. Extractive summaries are formed by "phrases" from the document that were appropriately selected. Abstractive summaries are formed by the "ideas" developed in the document, without using the phrases exactly as they appear in the original document. The differences between the two approaches can be clearly seen in [4].

In literature there is a vast number of related works with extractive summaries whose common main goal is to reduce document size while preserving the information of the source document [13]. Even though they have different approaches on the issue, a set of metrics is commonly used. Each metric analyzes a given characteristic of the document and allows to apply sorting criteria to its content. This order is obtained by assigning a score to each part of the document and by ordering the values from highest to lowest.

In 1958, at IBM, Hans Peter Luhn developed a simple summarization algorithm which used the distribution of word frequency to weigh sentences [14]. At the same time, Baxendale used the position of the sentences [15]. Both of them made the earliest proposals for automatic indexing. A few years later, Harold Parkins Edmundson and Roland Eugene Wyllys, based on Luhn's work, proposed using the presence of certain words and the overlapping of words and titles [16]. Many years later, Joel Larocca Neto et al. used the term frequency and the Inverse Sentence Frequency (an adaptation of the TF-IDF measure used in Information Retrieval) [17]. The following year, Yihong Gong and Xin Liu proposed the use of Latent semantic analysis (LSA) for text summarization without the use of lexical resources such as WordNet [18]. In 2004, Rada Mihalcea introduced a graph-based ranking model for sentence extraction applied to text summarization [19]. The same year, Fatma Jaoua Kallel et al. presented a summarization method based on the ratio of keywords in the sentences and in the document [20]. In 2007, Lucy Vanderwendea et al. operationalized the idea of using word frequency averaged for sentence selection [21]. These are just some of the most cited references in literature. A broader list of methods can be consulted in [7, 12].

In recent years, various metric-based techniques have been developed to extract the important contents from text documents to represent their summaries (e.g. [22] in whose work its authors assessed the performance of sentence scoring techniques individually and applying different combination strategies). The common factor in all these works is the use of metrics to model the documents as feature vectors. To do this, three steps are followed: Selecting a collection, pre-processing the documents and calculating the metrics. Subsequently, the same pre-processed documents are used to evaluate the results obtained. But it happens that several decisions made during the preprocessing of the documents have an influence on the calculation of the metrics. When using any test collections for our research works we do not have the same "view" of the documents that other authors have. It is not correct to compare summaries generated by methods whose input vectors are calculated in different ways. For example, it is not the same to calculate the metrics by sentence than by paragraph or by taking into account only nouns instead of all the words.

This work presents a tool that aims to solve these problems. The proposal consists of a set of routines to correctly preprocess the documents and a database to store them properly. In short, it is intended to increase the exactness of results when obtaining, evaluating and comparing automatic summaries obtained from different corpora using varied methods. The implementation of the proposed system will be detailed in the following section.

3 Description of Proposal

3.1 Database Model

In this paper, we propose a database design to store text documents in a structured manner allowing to record all the content of a document and

its details. This design was conceived for scientific documents taking into account the needs of extractive summarization. These types of documents use a predetermined structure that typically starts with the title of the article, the authors and other information (institutions, e-mails), followed by a summary, a set of keywords and each of the sections that compose the document.

Documents are fundamentally a sequence of characters in a string encoded with a chosen character set. Whereas plain text files only contain the characters of a text, other formats, such as XML, can express the content using a markup strategy. It consists of a tag-based structure that identifies specific parts within a document. The Center for Digital Research in the Humanities defines the XML format as an encoding standard that assists in the creation, retrieval and storage of documents. This allows processing the document and successfully storing it to the proposed database.

Normally, in these types of documents, the first sections provide general information in the introduction and then go on to detail more specific aspects in relation to the topic being discussed. Finally, they discuss the results obtained and the conclusions drawn and list bibliographic references. All this information must be stored. Also, the document has a title, a summary and a list of keywords. Then, its content is organized into sections, sub-sections, sub-sub-sections, etc., each of these with its own title. The content of each section consists of a set of paragraphs formed by sentences. Each sentence contains a sequence of words. Each word must be registered along with its location within the document. With the data design proposed here, the original document can be re-built at any time using all information stored in the database. Figure 1 shows the database design proposed for storing these types of documents.

The database consists of fifteen tables that store all the information needed to locate each word within a document. The model takes into account the journal, the issue and the articles published in each issue ("journal", "edition," and "document" tables). The text in the document is broken down into the tables called "section", "paragraph," and "sentence." The table called "section" is recursive, which allows recording section embedding, typical organization in scientific documents. The rest of the information (all titles, the keywords list and the text of sections) is related to the table called "word." Each word in the text is reduced to its stem by applying a stemming algorithm and it is stored in the table "stem" relating to the corresponding word in the table "word." This is the most significant term reduction task in Text Mining, since the number of words derived from the same stem is very high.

3.2 Assessment and Data Use

In this work, we used a subset of free access articles published in a journal on biomedical, environmental, social and political health issues from Public Library of Science (PLOS). These documents meet all requirements and can also be downloaded for free in XML format through the Internet. Having the documents in this format facilitates the identification of each of the parts of the document.

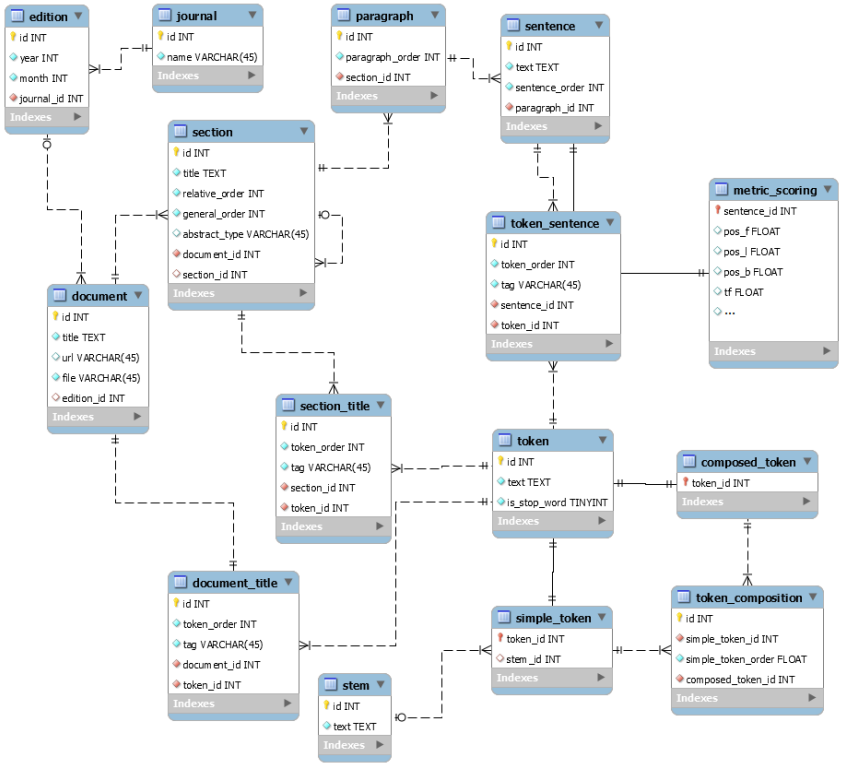

Fig. 1. Database model proposed for storing text content from scientific articles.

Files in this format can be processed with special libraries that create a tree with all the elements of the document. Each node in a tree is an object that represents a part of the document. This can be used to extract specific information from the document to store it in the database. In addition, these objects can be manipulated by programming and any change made to them will be reflected in the later visualization of the document. This can be interesting if, when identifying the relevant parts of a document, they were highlighted automatically to differentiate them from the rest of the document just like the human being does with a marker.

After retrieving the XML files, they were processed as required to upload them to the database, following these steps for each article:

- The title was extracted, together with the summary created by the author(s).
- Each of the sections was processed, and their titles identified. Certain full sections, such as Bibliography and Acknowledgments, were disregarded, as well as all figures, tables and equations included in the article.

– To obtain the sentences, the paragraphs in each section were segmented using the full stop as delimiter, except when it was used as decimal separator and as part of an abbreviation, among other uses.
– The keyword list, the titles, and the sentences were tokenized using white spaces and punctuation marks. Each token was added to the "token_sentence" table after verifying that the token existed.
– When a token does not exist previously, on the one hand it is added to the "token" table, marking it as a stopword if it is, and on the other hand, in "simple_token" or "composed_token", as appropriate. Every simple token has its corresponding root stored in the "stem" table.

The process of loading all this information to the database from the XML file was carefully carried out so as to avoid altering the order of the sections and their structure. The Data Base Management System used was MySQL. Python programming language and Natural Language Toolkit (NLTK) were used for automatic file download from the Internet and their processing [23]. In order to do this a script was developed using urllib and BeautifulSoup libraries. The Porter stemming algorithm and english stopword list of NLTK were used. Porter stemmer is one of the best known algorithms to perform this task [24]. The stoplist is a list of words with empty meaning called stopwords, which contains pronouns, prepositions, articles and a few verbs. In addition to providing stemming and stoplists, the NLTK package provides tokenization, tagging, parsing, and very useful semantic reasoning functions. All of the text pre-processing tasks that are mentioned in this paragraph are common in this area of research.

During this step, a document is transformed and several decisions have to be taken: Which words to consider stopwords? Which algorithm of stemming to use? What is considered a token? What are the delimiters to segment the text? And many more questions. The final result is a view of the document from which, for example, a series of metrics will be calculated. For this reason, it is essential to have control over the impact of the pre-processing performed on the documents.

3.3 Usefulness of the Proposed Model

From a set of documents, each of the scoring metrics used for automatic summary generation must be calculated for each sentence. This requires, among other things, processing variables of text type several times using the string split method. Even if before calculating metrics we tried different preprocessing actions, it would take much longer.

Once the documents are stored in the proposed database each of the metrics can be calculated through a single SQL query for all sentences in the document. But if it is done in this way the query will be complex and not easy to prepare. A better solution is to combine SQL queries with Python code. The query will retrieve the corresponding identifiers of the tokens associated with each sentence of the document, and with Python the frequencies necessary to calculate each

metric will be easily computed. The interesting thing is that, like their stems, the tokens can be retrieved, and even the stopwords or certain types of words (such as nouns, verbs, etc.) can be filtered. In addition, it is possible to recalculate the metrics by paragraphs without requiring many modifications.

Once the metrics are calculated, each document will be represented by a matrix of as many rows as sentences the document has, and as many columns as metrics are calculated. These matrices will be stored in the table "metric_scoring" shown in Figure 1 and can be obtained by filtering the tuples of each document.

As has already been mentioned, creating extractive summaries requires selecting certain parts of the document that contribute to the desired summary. Just like humans mark on paper the text that they consider important, an automatic system should assign each sentence in the original document to one of two possible classes: "Summary," if the sentence belongs to the summary, or "Non-Summary," if it does not. Instead of drastically deciding if a sentence belongs to the summary or not, the list of sentences ordered by metric from highest to lowest can be split into two groups using a threshold value: the class "Summary" can be assigned to the best positioned N, and the class "Non-Summary" to all others. The value of N is a parameter specified by the user that allows to select the N most relevant sentences and to control the size of the resulting summary.

Once the sentences that will form the automatic summary have been selected, they should be evaluated by comparing them with an expected summary. For this task, the identifiers of the tokens of the generated and expected summary can be obtained through an SQL query. Then, a small program in Python calculates the intersection between both sets to get the exact value of Rouge. ROUGE, developed by Chin-Yew Lin, provides different measurements frequently used in literature to the assessment of automatic summaries quality [25].

Since the specific size of the summary is not known a priori, summaries are usually evaluated by varying the value of N and averaging the results. Also, since not all the words in the expected summary are used in the document, the rouge values obtained between documents for the different sizes are not comparable to each other. For this reason, before averaging it is very important to divide each value by the maximum value of rouge obtained with the full document as summary.

Another useful aspect of this model is the flexibility to reconstruct a document according to the input format of other automatic systems. Sometimes, experiments are carried out using online summarization systems, whose input must be built in a certain way. This tool was used in two previous research papers related to automatic summaries [26, 27].

4 Conclusions and future works

Automatically obtaining summaries from text documents continues to be a subject of study that constantly receives scientific contributions. The extractive summarization approach is one of the most commonly used methods in literature,

since it can create a summary by setting aside semantic aspects that the abstractive approach usually requires.

In this article, a data model for storing scientific documents with a predefined structure was presented. By using it, a set of metrics can be calculated for each document using SQL queries and Python code. Thus, the sentences in a document are sorted following specific criteria, which allows generating various extractive summaries of a controlled size. It should be noted that open source tools were used for every aspect of this work.

This proposal is a practical contribution to summarization area that allows to easily calculate metrics, diminishing developing time and some possible ambiguity of interpretation. It saves calculation time, clearly expresses how each metric is calculated, and facilitates experimentation in this research area. Also, the document can be easily reconstructed according to the needs of the experiment to be carried out, allowing the integration of automatic summary generation systems that operate over the documents with significant differences.

In the future, we intend to introduce changes in the model to equip it with semantic. It is expected that incorporating a semantic analysis, the calculation of the metrics based on frequency or intersection of words can be improved. Additionally, an application will be developed to allow users to view the document with the parts that were selected by each of the summaries.

References

1. Schreibman, S., Siemens, R., Unsworth, J.: A New Companion to Digital Humanities. Blackwell Companions to Literature and Culture. Wiley (2016)
2. Mani, I.: Automatic Summarization. Natural language processing. J. Benjamins Publishing Company (2001)
3. Mani, I.: Summarization evaluation: An overview (2001)
4. U., H., I., M.: The challenges of automatic summarization. Computer **33**(11) (2000) 29–36
5. Edmundson, H.P., Wyllys, R.E.: Automatic abstracting and indexing—survey and recommendations. Commun. ACM **4**(5) (1961) 226–234
6. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Commun. ACM **18**(11) (1975) 613–620
7. Nenkova, A., McKeown, K. In: A Survey of Text Summarization Techniques. Springer US, Boston, MA (2012) 43–76
8. Cremmins, E.T.: The art of abstracting. Professional writing series. ISI Press (1982)
9. National Institute of Standards and Technology (NIST): Document Understanding Conferences (DUC). http://www-nlpir.nist.gov/projects/duc/index.html (2002)
10. Spärck Jones, K.: Automatic summarising: The state of the art. Inf. Process. Manage. **43**(6) (2007) 1449–1481
11. Gambhir, M., Gupta, V.: Recent automatic text summarization techniques: a survey. Artificial Intelligence Review **47**(1) (2017) 1–66
12. Torres Moreno, J.M.: Automatic Text Summarization. Cognitive science and knowledge management series. Wiley (2014)
13. Gupta, V., Lehal, G.S.: A survey of text summarization extractive techniques. Journal of emerging technologies in web intelligence **2**(3) (2010) 258–268

14. Luhn, H.P.: The automatic creation of literature abstracts. IBM J. Res. Dev. **2**(2) (1958) 159–165
15. Baxendale, P.B.: Machine-made index for technical literature: An experiment. IBM J. Res. Dev. **2**(4) (1958) 354–361
16. Edmundson, H.P.: New methods in automatic extracting. J. ACM **16**(2) (1969) 264–285
17. Larocca Neto, J., Santos, A.D., Kaestner, C.A., Freitas, A.A. In: Generating Text Summaries through the Relative Importance of Topics. Springer Berlin Heidelberg, Berlin, Heidelberg (2000) 300–309
18. Gong, Y., Liu, X.: 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, New York, NY, USA, ACM (2001) 19–25
19. Mihalcea, R.: 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, Stroudsburg, PA, USA, Association for Computational Linguistics (2004)
20. Jaoua Kallel, F., Jaoua, M., Belguith Hadrich, L., Ben Hamadou, A.: Summarization at laris laboratory. In: Proceedings of the Document Understanding Conference. DUC'04 (2004)
21. Vanderwende, L., Suzuki, H., Brockett, C., Nenkova, A.: Beyond sumbasic: Task-focused summarization with sentence simplification and lexical expansion. Inf. Process. Manage. **43**(6) (2007) 1606–1618
22. Oliveira, H., Ferreira, R., Lima, R., Lins, R.D., Freitas, F., Riss, M., Simske, S.J.: Assessing shallow sentence scoring techniques and combinations for single and multi-document summarization. Expert Systems with Applications **65**(Supplement C) (2016) 68 – 86
23. Bird, S., Klein, E., Loper, E.: Natural Language Processing with Python. 1st edn. O'Reilly Media, Inc. (2009)
24. Porter, M.F.: Readings in information retrieval. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1997) 313–316
25. Lin, C.Y.: Rouge: a package for automatic evaluation of summaries. In Marie-Francine Moens, S.S., ed.: Text Summarization Branches Out: Proceedings of the ACL-04 Workshop, Barcelona, Spain, Association for Computational Linguistics (2004) 74–81
26. Villa-Monte, A., Lanzarini, L., Rojas-Flores, L., Olivas, J.A.: Document summarization using a scoring-based representation. In: 2016 XLII Latin American Computing Conference (CLEI). (2016) 1–7
27. Puente, C., Villa-Monte, A., Lanzarini, L., Sobrino, A., Olivas, J.A.: Evaluation of causal sentences in automated summaries. In: 2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE). (2017) 1–6