Journal of Computer and Communications, 2018, 6, 43-54
http://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

Advantages of Using a Spell Checker in Text Mining Pre-Processes

Jhonathan Quillo-Espino, Rosa María Romero-González, Alberto Lara-Guevara

Facultad de Informática, Universidad Autónoma de Querétaro, Querétaro, México
Email: estudiantejquillo@gmail.com

How to cite this paper: Quillo-Espino, J., Romero-González, R.M. and Lara-Guevara, A. (2018) Advantages of Using a Spell Checker in Text Mining Pre-Processes. *Journal of Computer and Communications*, **6**, 43-54.
https://doi.org/10.4236/jcc.2018.611004

Received: September 21, 2018
Accepted: November 11, 2018
Published: November 14, 2018

Abstract

The aim of this work was the behavior analysis when a spell checker was integrated as an extra pre-process during the first stage of the test mining. Different models were analyzed, choosing the most complete one considering the pre-processes as the initial part of the text mining process. Algorithms for the Spanish language were developed and adapted, as well as for the methodology testing through the analysis of 2363 words. A capable notation for removing special and unwanted characters was created. Execution times of each algorithm were analyzed to test the efficiency of the text mining pre-process with and without orthographic revision. The total time was shorter with the spellchecker than without it. The key difference of this work among the existing related studies is the first time that the spell checker is used in the text mining preprocesses.

Keywords

Spell Checker, Text Mining, Stemming, Tokenization, Porter Algorithm, Snowball Algorithm

1. Introduction

Language, since the beginning of time, has been the instrument for excellence used by human beings to communicate one with each other with ideas, feelings, and knowledge. Man has looked different ways to transmit his legacy, for example, paper. The information is expressed in letters, words, dictionaries, books, etc. The continuous change and evolution of the computational technology has allowed the human being to create new methods for the evaluation, performance, and knowledge developing through the language. Thanks to researcher efforts, it has been possible to achieve capable algorithms to perform automatic and semi-automatic processing tasks using methodologies derived from data

mining (DM), as well as text mining (TM). TM stages and pre-processes were defined to find patters and discover new knowledge [1]. An internal evolution algorithm of patterns was designed to reduce noise effects in the text pattern selection to obtain more reliable results [2]. The automated process starting of artificial intelligence (AI) was proposed when the principles of natural language processing (NLP) were deduced only as a perception [3]. NLP definition and characteristics were described in detail, denoting differences among other approaches based on bivalent logic. The NLP use was highlighted as a computational framework for the meaning accuracy. NLP can be defined as an interaction and interpretation process between human and computer languages [4]. It was suggested that NLP rules and levels function as a set of dynamically related elements for language recognition, concluding that the best language understanding corresponds to highest language levels in an NLP system [5].

NLP purposes, based on computational techniques, are to learn, create, and understand human language. There are different NLP applications, such as text translation, voice recognition, and information retrieval (IR), among others. The main NLP challenge is to understand the user needs; as each person has particular concerns, the resolution for each user will be different. The question formulation will determine the use of different solution processes. IR, NLP, and DM are part of the TM application areas, since TM is considered as a multidisciplinary field. The main TM challenge is the language complexity due to several meanings for the same word. The TM function to find text patterns, requires an analysis of a large text collection to identify the most outstanding keywords. TM is a tool to obtain new knowledge from text collections; it is a process that requires other sub processes to operate correctly [6].

None of the consulted references used the spell checker in the TM pre-processes. The aim of this work is to demonstrate the time reduction obtained by using a spell checker in the TM pre-processes.

A deep TM analysis is introduced in this paper. The rest of the paper is organized as follows: Sections 2 and 3 contain a general TM review. Section 4 exhibits an analysis of TM pre-processes. The proposed methodology is exposed in Section 5. Section 6 shows the results, and finally, conclusions are in Section 7.

2. Development

The textual content (TC) analysis is an interesting and new topic for researchers. Studies showed that organizations still have 80% of the information contained in textual documents [7]. That 80% content is unstructured textual data. Therefore, scientists are encouraged to find motivation to conduct research in the TC field [8]. TM intends to extract meaningful information from the language. It is a huge challenge, since it tries to extract explicit knowledge and the semantic relationship that exists in the natural language, but with the great distinction of being unstructured texts. The main difference is that DM contains structured data, usually stored in databases, also an index is added helping to perform quick

searches. Unstructured texts can be notes, paper, records or any textual document that contains relevant information of the organization. In addition, there may be semantic ambiguities that cause complication during the analysis time [9]. TM could also work with HTML documents. This is an opportunity for companies that are interested in analyzing their own websites or those of the competition for the information extracting. As a consequence, it will be possible for better results. The TM method can be used in any system that attempts to analyze large text amounts for the interpretation of the natural language [10].

Different approach proposals coincided in the structure, development, and in the process set corresponding to the TM. They also agreed on the need to perform the TM pre-processes, which are necessary to increase efficiency, reduce time and eliminate useless words. Every day, thousands of information sheets are generated in organizations, creating large textual databases with significant information implicit in them. The main objective of the TM, in addition to the information extraction (IE), is to convert a set of unstructured texts to structured texts through its different areas such as IR, NLP, etc. Generally, TM is the most prudent common process to extract information from text [1] [10] [11]. There are other proposals where the representation of extracting knowledge through the text refinement is exposed. The text refinement transforms the free structure into an understandable form, while the knowledge distillation is deduced and applied to the pattern reduction. This is an interesting proposal; however, it has not been very successful [7]. For a text set to be analyzed correctly and quickly, must apply automatic learning techniques, which will help to categorize the texts. In this way, searching efficiency is improved, besides a reduction of the disorganization problem that is generated by having sets of unstructured texts [12].

3. Keys in the TM Development

TM is usually composed by three initial processes called: pre-processes (tokenization, stemming, stop removal words). TM processes are IE, IR, NLP, categorization, clustering, and as a result, discovered knowledge [13]. **Figure 1** shows a diagram of TM pre-processes and processes.

4. Text Mining Pre-Processes

To perform the TM process, it is necessary to have the collection of textual data. It is the first TM contact, where the outstanding and functional information is chosen. The pre-processing consists of transforming textual data into clear elements, eliminating inconsistencies for its future interpretation. In general, paper or digital text documents can be included, also notes, web pages, and any document that contains information to be analyzed. However, if the text collection set is in paper format, it will be necessary to digitize using optical character recognition (OCR) techniques. The computer pretends to have the ability to recognize words and convert them to data by digitizing written characters [14] [15]. It

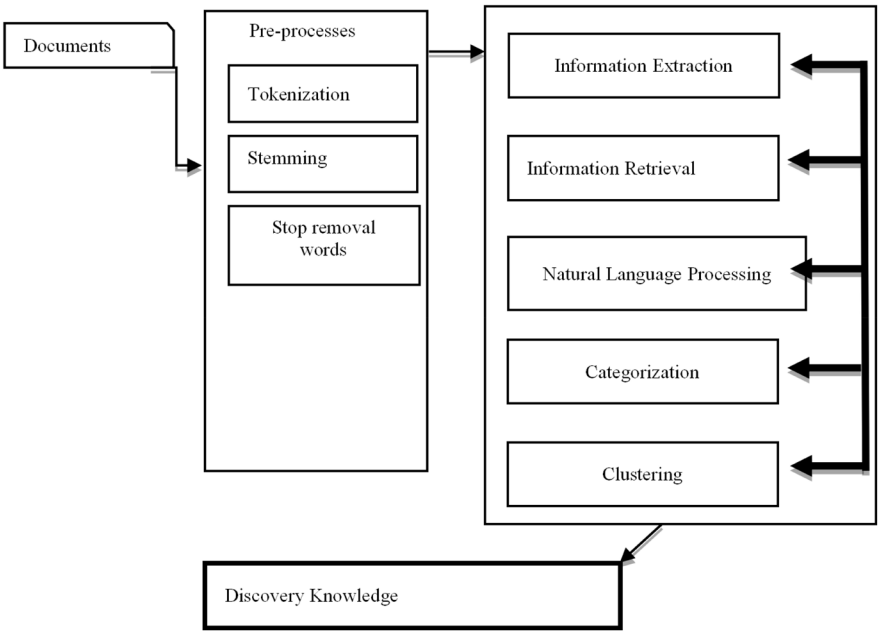

Figure 1. Text mining pre-processes and processes.

must be considered that there may be errors, but there are different ways to correct them as proposed [16] and mentioned [17]. When there is a moderate error in noise, there will be little impact on the IR effectiveness. The pre-processes are those in which most of the time is taken into the TM process of extracting knowledge. Obviously, the less error exists in the character recognition process.

4.1. Tokenization

According to the TM model, tokenization is the first pre-process which consists of dividing a text sequence and reducing words, symbols, and phrases to significant elements known as tokens [13]. Tokenization involves the pre-processing of certain documents to generate the respective tokens. Everything indicates that tokens are regularly the product of the text normalization [18]. The question to answer is: Why to tokenize? The main tokenization objective is to explore words of a sentence to identify the keywords. When the user has the need to analyze multiple documents, a common problem is to identify the relevant information among large amounts of textual data, because of the lack of an index. As there are unstructured textual documents, the extraction process in not efficient enough; an intense analysis is necessary, that is an arduous task [19]. Another reason because tokenization is carried out, is the analysis to organize, count, divide, and index textual documents. It should be considered that the pre-process complexity depends on the document's origin, which is addressed if necessary, to carry out the pre-processing in documents. It is inferred that tokenization affects the paragraph identification, as well as the quality of extracted grammatical rules [20].

On the other hand, indexing is the important TM process to manage the token collection, since it generates functionality allowing a simplified character

process. Indexing knows the entire token collection, but it only represents the keyword set of the elements. The main indexing function is the logical structure creation for a word database. The character sequence must be divided into significant units, but they must be done before any other text processing. The great challenge derives from the fact that the statements are not usually found orthographically well written, punctuated or abbreviated. The tokenization sub-process is directly accompanied by the elimination of unwanted elements as symbols, letters, and special characters such as /, \, ´, “, ”, {,}, [,], [,], 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, etc. Main tokenization functions are the segmentation and analysis of the word lexicon, identifying those that are distinctive or contain a greater value. Depending on the type of analysis, words must be counted for the database creation to be indexed later. Although tokenization is an arduous pre-process, the text transformation is sought, helping to reduce the time response for indexing and operations such as frequency calculation. It should be mentioned that tokenization can be applied in different areas such as banking, web, etc. Tokenization commonly supports the TM development to optimize the lexical analysis. Frequently it is applied at a basic level, directly in the word selection to form the text that has been analyzed. Tokenization has become a fundamental IR part, because it seeks to discover implicit tokens in the textual documents. The main tokenization objective is the meaningful keyword identification. Its application will be functional as long as there are delimiters in the text. To facilitate the word identification, a simple way is to add blank spaces before and after each word [21]. **Table 1** shows a practical example of the tokenization appliance. The word frequency <allows = 2> is 2 since it appears twice. This tokenization type can be applied to any textual document, although it is simple, but its operation is practical.

The operation can be affected by the language typographical structure, such as the isolation. Words are not divided into small units. In the Mandarin Chinese language words can be divided into small units by adhesion. The set of remaining words of the tokenization product is called the dictionary collection. Obviously, the tokenization complexity depends on different factors, such as: 1st if the analyzed text has been written properly; 2nd if the abbreviation has been written correctly, point delimiters should be considered to detect the beginning and the end of a sentence; 3rd the language, because the existence of a wide language vocabulary becomes difficult, 4th the textual volume “large vs small”. 5th The existence of available tools by language, which generally help to modernize the tokenization pre-process. However, everything obeys the researcher’s need of improving the response time. If the desired result were not found for the first instance, it would be necessary to perform the processing again. A compound word divided in two should be separate every token [21].

Analyzing the TM process with pre-processes, a spell checker is proposed to introduce before the tokenization. The purpose is to correct the spelling automatically, according to characteristics of the used language, obtaining two main

benefits: 1st words would be correctly written since the source text, 2nd without orthographic errors the stemming execution would be effective for the text transformation [1]. **Figure 2** shows the spell checker addition to the TM pre-processes.

4.2. Stemming

Stemming is used as a pre-process in the TM, as well as in NLP, whose objective is to reach the basic forms of the words. It is emphasized that the stemming phase is used to extract the root of a given word. The existing variation in the word morphology is an important challenge for the IR. Stemming is in charge of increasing the performance, besides that having the words the same root, the duplicity of words is avoided. Therefore, the dictionary is reduced and the storage size is smaller, since stemming is precisely in charge of handling this variation. Algorithms responsible for stemming are called stemmers [18]. Stemming does not always reduce the word to its basic form, what it really tries to do is an approximation to the word root [22]. It is suggested that there are two main types of errors when stemming performs. The 1st error is called over stemming, that consists of two words with different meaning both derived from the same root; it is also known as false positive. The 2nd one is under stemming when two words must be derived from the same root, but it is not true, what is known as false negative [23]. **Table 2** shows stemming techniques.

During the stemming phase, the word is forced back to its root, eliminating the suffixes, without losing the meaning. The storage size decreases dramatically because similar elements are eliminated to maximize the RI model efficiency. It can be applied directly in the user interface, allowing the user to modify the word easily. Some words cannot be forced to its root, since the morphology does not allow it, therefore, such words will not change.

Figure 3 shows an example of the stemming pre-process using the word user.

4.3. Stop Removal Words

By performing an TM process, it is necessary to filter redundant words and words without a semantic meaning. Tokenization and stemming pre-processes could produce words with a semantic value. Therefore, filtering avoids categorizing according to rules such as prepositions, adverbs, etc. It should be noted that it must be oriented to the used language. There are meaningless words that repeat in the collection dictionary set, which are generally used to join phrases or to complete sentences but without any lexical use; the reduction of these words minimizes the token volume, reducing the TM response time. Each of the analyzed texts contains words without function or utility, whose elimination will reduce the text size. As less words are generated, the text weight will be reduced in space terms. Commonly, they are words like articles, prepositions, and pronouns that do not add a value to the text, and those, therefore, should be eliminated because they are not considered keywords [24].

Table 1. Example of tokenization.

Input:	Tokenization allows identifying significant keywords for the IR, which allows counting and obtaining their frequency
Output:	Tokenization = 1 allows = 2, identifying = 1, significant = 1 keywords = 1, for = 1 the = 1, IR = 1, which = 1 counting = 1 and = 1 obtaining = 1 their = 1 frequency = 1

Table 2. Text stemming classification.

Stemming techniques	
Table column Based on rules: word variants are transformed into their root based on a predefined rule. The main advantage is an easy use due to specific rules of the language. Once created, they can be used without any additional processing.	
Brute force algorithm	A searching table is used to return to the word root.
Affix removal algorithm	Suffix and prefix of a word are focused and eliminated through variant form techniques.
Morphological algorithms	A morphological analysis is applied, that requires complex lexicons to perform the analysis.
Based on a static root: Language derivation rules are learned by training with a supervised or semi-supervised way	
Lexicon based on root analysis	A word group is analyzed to be classified graphically by lexical sets.
Root analysis	Morphological groups related by the context.
N-Gram character	Learning based on the derivation through the N-Gram frequency obtained from the words.
Hybrid algorithms: They combine former algorithms to improve the functionality; they are adapted to the used language.	

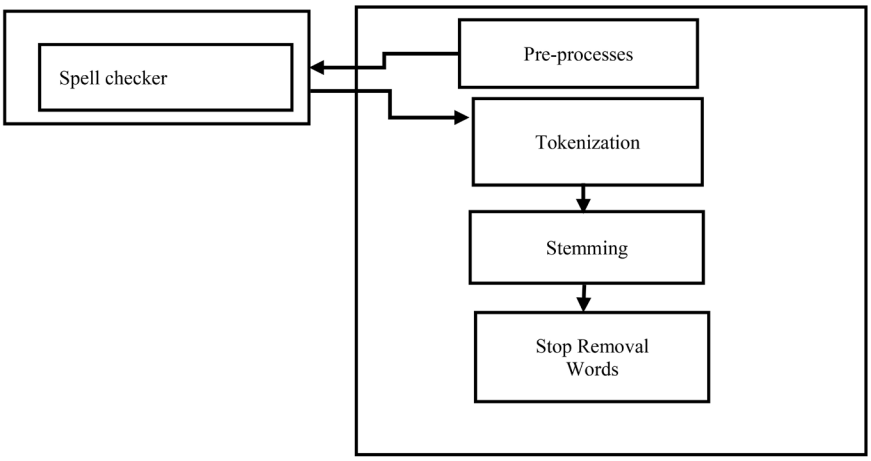

Figure 2. Spell checker addition to the pre-processes set.

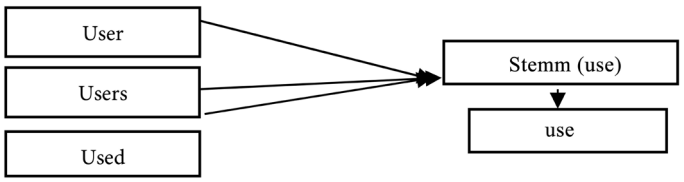

Figure 3. Stemming pre-process.

The pre-processing tasks play a vital role to facilitate a quick and efficient TM application. In addition, they simplify and accelerate the RI process with filtered information.

5. Methodology

The aim of this work was the behavior analysis when a spell checker was integrated as an extra pre-process during the first stage of the TM. Different models were analyzed. A model of the TM instances was exemplified with pre-processes as a particular element; execution importance and need were indicated; also open source tools for evaluation were summarized [1]. TM perform techniques were described; the model is completed using a web-oriented algorithm of rules and techniques [10]. A basic TM model was demonstrated with the description of different TM areas [9]. The TM time was determined through a sequential model with the objective of cleaning and identifying patterns [20]. TM areas and basic model were described and available tools were compared [6]. Grammar evaluation rules were explained to be processed by the computer during the NPL [5]. Processes were detailed with the process description approach and the pre-process model for the TM generation [11]. The process development was proposed with a non-effective pattern recognition technique [2]. Pre-process description was emphasized [25]. After studying and analyzing different TM proposals, the most complete one resulted the model [1] that considers pre-processes as the initial part of the TM. Algorithms were developed and adapted for the methodology testing. A spell checker algorithm was integrated as the first stage of the TM pre-processes. A notation was created capable of eliminating special and unwanted characters. An algorithm was developed to tokenize the analyzed word set.

Snowball algorithm was adapted from Porter stemming algorithm that was developed for the English language [26]. Snowball algorithm is a derivation of Porter stemming, only that it was developed to work with different languages such as Spanish [27]. **Figure 4** shows the adaptation of the used algorithms.

Execution times of algorithms were analyzed to test the efficiency of the TM pre-processes with two variants, as well as with and without spell checker. A total of 2363 Spanish words were evaluated. The word set resulted from interviews to UAQ students about general culture. Time values were obtained directly when algorithms were executed.

Evaluation Criteria

The evaluation was the execution time measurement of pre-processes during the first TM stage. Results are exhibited in **Table 3**.

6. Results

Table 3 shows execution time results of pre-processes with or without spell checker.

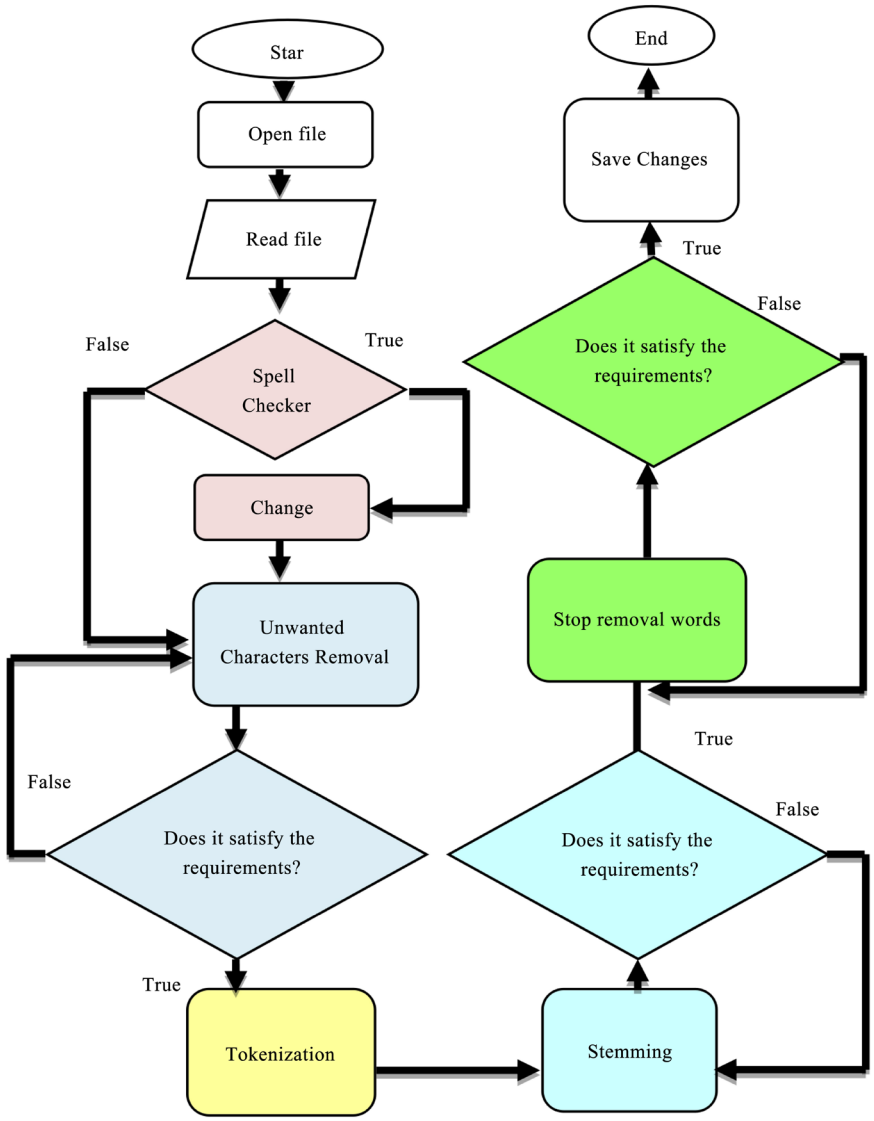

Figure 4. Algorithm adaptation flow diagram.

Table 3. Execution time results of pre-processes.

Pre-process	Time (ms)	Pre-process	Time (ms)
Tokenization without spell checker	1.90	Stemming without spell checker	1.28
Tokenization with spell checker	1.90	Stemming with spell checker	1.25
Stop removal words with spell checker	1.13	Total time with spell checker	4.28

In the tokenization pre-process was not a time difference between with and without spell checker.

The stemming pre-process with spell checker saved 0.03 ms with respect to the option without spell checker.

The stop removal word pre-process with spell checker saved 0.09 ms; that means that there is a better time benefit when words are spelled orthographically correct.

Observing total time results, there was a saving of 0.12 ms with spell checker. It is obvious that the TM pre-process time will increase with the word set size.

7. Conclusions

The key difference of this work among the existing related studies is the first time that the spell checker is used in the TM pre-processes.

The spell checker introduction as an extra pre-process during the first TM stage produces a time reduction.

The spelling correction benefits the TM stemming pre-process, but also IR and NLP processes, because those are part of the TM application areas.

The pre-process work provides support to different text formats, which will allow cleaning, analyzing, and organizing the text for processing. The purpose is to help finding patterns to extract relevant information.

Snowball algorithm is one of the main and better-known algorithms, which allow developers to be adapted to almost any language.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] G Vijayarani, S., Llamathi, J. and Nithya, Ms. (2016) Preprocessing Techniques for Text Mining. An Overview. *International Journal of Computer Science & Communication Networks*, **5**, 7-16.

[2] Zhong, N., Li, Y. and Wu, S.T. (2012) Effective Pattern Discovery for Text Mining. *IEEE Transactions on Knowledge and Data Engineering*, **24**, 30-36. https://doi.org/10.1109/TKDE.2010.211

[3] Zadeh, L.A. (2001) A New Direction in AI. Toward a Computational AI Theory of Perceptions. *AI Magazine*, **22**, 73-84.

[4] Zadeh, L.A. (2004) Precisiated Natural Language (PLN). *Studies in Computational Intelligence* (*sci*), **25**, 74-91.

[5] Reshamwala, A., Pawar, P. and Mishra, D. (2013) Review on Natural Processing. *Engineering Science and Technology*: *An International Journal* (*ESTIJ*), **3**, 113-116.

[6] Sumathy, K.L. and Chidambaram, M. (2013) Text Mining: Concepts, Applications, tools and Issues an Overview. *International Journal of Computer Applications*, **80**, 29-32. https://doi.org/10.5120/13851-1685

[7] Tan, A.H. (2000) Text Mining: The State of the Art and the Challenges. Kent Ridge

Digital Labs.
https://www.researchgate.net/publication/2471634_Text_Mining_The_state_of_the_art_and_the_challenges

[8] Ergün, M. (2017) Using the Techniques of Data Mining and Text Mining in Educational Research. *Electronic Journal of Education Sciences,* **6**, 181-189.

[9] Kumar, L. and Bathia, P.K. (2013) Text Mining: Concepts, Process and Applications. *Journal of Global Research in Computer Science*, **4**, 36-39.

[10] Gupta, V. and Lehal, G. (2009) A Survey of Text Mining Techniques and Applications. *Journal of Emerging Technologies*, **1**.

[11] Sun, W., Cai, Z., Li, Y., Liu, F., Fang, S. and Wang, G. (2018) Data Processing and Text Mining Technologies on Electronic Medical Records: A Review. *Journal of Healthcare Engineering*, **2018**, Article ID: 4302425.
https://doi.org/10.1155/2018/4302425

[12] Lu, F. and Bai, Q.Y. (2010) A Refined Weighted k-Nearest Neighbors Algorithm for Text Categorization. *IEEE International Conference on Intelligent Systems and Knowledge Engineering*, Hangzhou, 15-16 November 2010, 326-330.

[13] Vijayarani, S. and Janani, R. (2016) Text Mining: Open Source Tokenization Tools, an Analysis. *Advanced Computational Intelligence: An International Journal*, **3**, 37-47.

[14] Borovikov, E. (2014) A Survey Optical Character Recognition Techniques. American Management Systems Inc.

[15] Berchmans, D. and Kumar, S.S. (2014) Optical Character Recognition: An Overview and an Insight. *International Conference on Control, Instrumentation, Communication and Computational Technologies*, Kanyakumari, 10-11 July 2014, 1361-1365.
https://doi.org/10.1109/ICCICCT.2014.6993174

[16] Taghva, K., Borsack, J. and Codit, A. (1996) Effects of OCR Errors on Ranking and Feedback Using the Vector Space Model. *Information Processing & Management*, **32**, 317-327. https://doi.org/10.1016/0306-4573(95)00058-5

[17] Lopresti, D. (2009) Optical Character Recognition Errors and Their Effects on Natural Language Processing. *International Journal on Document Analysis and Recognition*, **12**, 141-151. https://doi.org/10.1007/s10032-009-0094-8

[18] Sigh, V. and Saini, B. (2014) An Effective Tokenization Algorithm for Information Retrieval Systems.

[19] Tanu Verma, R. and Deepti, G. (2014) Tokenization and Filtering Process in RapidMiner. *International Journal of Applied Information Systems*, **7**, 16-18.
https://doi.org/10.5120/ijais14-451139

[20] Munková, D., Munk, M. and Vozár, M. (2013) Data Pre-Processing Evaluation for Text Mining: Transaction/Sequence Model. *Procedia Computer Science*, **18**, 1198-1207. https://doi.org/10.1016/j.procs.2013.05.286

[21] Kaplan, R.M. (2005) A Method for Tokenizing Text.
https://www.researchgate.net/publication/244958750_A_Method_for_Tokenizing_Text

[22] Toman, M., Tesar, R. and Jezek, K. (2006) Influence of Word Normalization on Text Classification.
https://www.researchgate.net/publication/250030718_Influence_of_Word_Normalization_on_Text_Classification

[23] Lakashimi, R.V. and Kumar, B.R. (2014) Literature Review: Stemming Algorithms for Indian and Non-Indian Languages. *International Journal of Advanced Research*

in Computer Science & Technology, **4**, 2582-2582.

[24] Saini, R.J. (2016) Stop-Word Removal Algorithm and Its Implementation for Sanskrit Language. *International Journal of Computer Applications*, **150**, 15-17. https://doi.org/10.5120/ijca2016910541

[25] Kannan, S. and Gurusamy, V. (2014) Preprocessing Techniques for Text Mining. https://www.researchgate.net/publication/273127322_Preprocessing_Techniques_for_Text_Mining

[26] Porter, M. (1980) The Porter Stemming Algorithm. https://tartarus.org/martin/PorterStemmer/index-old.html

[27] Porter, M. (2003) The Snowball Algorithm. http://snowball.tartarus.org