International Journal of Computer Science and Software Engineering (IJCSSE), Volume 8, Issue 11, November 2019
ISSN (Online): 2409-4285 www.IJCSSE.org Page: 266-275

Text Mining Preprocessing In Times of Python vs MVCS

Jhonathan Quillo-Espino[1], Rosa María Romero-González[2] and Francisco Javier Paulin-Martinez[3]

[1, 2, 3] Faculty of Informatics, Autonomous University of Querétaro, Santiago de Querétaro, Querétaro 76230, México

[1]estudiantejquillo@gmail.com, [2]rossyrg04@yahoo.com.mx, [3]fpaulinm@gmail.com

ABSTRACT

This investigation illustrates in a clear and understandable manner the preprocessing execution of text mining, through two programming languages: Microsoft Visual C Sharp (MVCS) and Python. It pretends to demonstrate the usefulness of each analyzed method, mentioning which one is more efficient. The main purpose is to show the advantages and disadvantages of natural language preprocessing tools in Python, versus a personalized programming, specifically for text mining preprocesses in Microsoft Visual C Sharp.

Keywords: *PNLTK, Preprocesses, Python, Text-Mining, Visual- C-Sharp.*

1. INTRODUCTION

By nature, the human being constantly looks for continuous innovation; he pretends to build methods that allow him to improve his environment and to achieve an easier and more pleasant life. Information Technologies are only one of the many areas where advances are clearly noticeable. The invention of the computer and artificial intelligence are clear examples of technological progress in its full development. Currently, researchers have focused on trying to make computer systems understand and interpret natural human language.

Nowadays, private and public technology companies, as well as educational, financial and many other companies store and generate huge amounts of information; either tangibly on paper or digital in text documents. Here is where Text Mining (TM) plays an important role in textual analysis. [1] also define TM as the combination of different techniques including natural language processing (NLP), to automatically discover patterns and subtract information derived from large volumes of textual databases. For their part, [2] assure that TM is a useful tool for a new knowledgeable generation and it can be applied in any research area where the analysis of data texts is required. TM is useful to analyze unstructured texts and to try to structure them. At the same time [3] proposed the main TM elements. Fig. 1. Shows the general TM process.

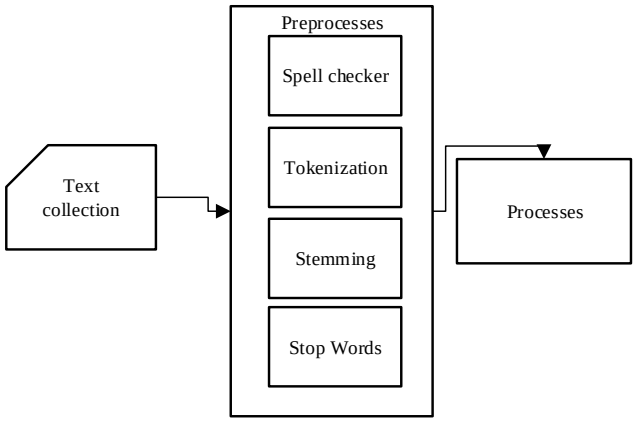


Fig. 1. General TM process. [3]

The preprocesses are crucial part for the TM development, since they are in charge of making the first contact with disorganized information and perform the process of cleaning it in such way that it can move onto the next TM stage. It is quite simple because the data and texts contain elements that are none functional such as numbers, symbols, etcetera and they must be eliminated for the TM process. The essence that allows textual database analysis in an efficient way is the information processing through preprocesses, since they perform basic activities of the NLP such as the elimination of inconsistencies and unnecessary or irrelevant information for the analysis.

When the databases have been filtered, formats, characters, spelling symbols, and others are eliminated contributing to the use of the information in an efficient and direct way.

1.1 Spelling Corrector (SC).

It is the first element of the preprocesses in TM proposed by [3], which consists in eliminating existing spelling mistakes in a text. Table 1 shows an example of spell check and demonstrates its importance.

Table 1: SC example

Without spelling correction	With spelling correction
Te colo of the hose is yellou and grin	The color of the house is yellow and green.

It is observed on the left side of Table1 a sentence without spelling corrections, on the right side there is a sentence with the spelling amendment applied. As a result, the stemming (ST) process is done promptly.

1.2 Tokenization

Is the process that allows separating or decomposing a text in independent elements. [4], [5] suggested that the main purpose of tokenization is to explore the words in a sentence. The result of tokenizing a sentence is to aid each element called token to be analyzed separately and at the same time to be eliminated fast. Table 2 shows a basic example of tokenization.

Table 2: Tokenization example

Without tokenization	Tokenization
The children play in the corner park	The', 'children', 'play', 'in', 'the', 'corner', 'park'

It is observed on the left side of Table 2 a sentence without tokenization and on the right side the tokens are delimited through single quotes and a comma becoming independent elements.

1.4 Stemming (ST)

For their part, [2], [6] explained that the main idea is to take any existing word to the textual database to its root. Subsequently, [7] emphasized that two types of stemming exist, one is flexible and the other one derivational, being the last one highly used by the Porter [8] and Snowball algorithm [9] (for the Spanish language). Table 3 shows a clear example of ST taking words to their roots through the ST preprocess using Snowball algorithm.

Table 3: ST example

Word without ST	Word with ST
Presentar presenta presente	present present present

It is observed on the left side of Table 3 a set of 3 words without the ST process, on the right side the words are already transformed onto their root by the ST.

1.3 Stop Words (SW)

As [10], [11] assured, SW consists in the elimination of common terms without semantics that do not add relevant information for MT. Therefore, [12] propose that the removal of the SW word list helps the efficiency of preprocess without altering the words. The word list is clearly defined. Table 4 shows an example of a list of words contained in SW. A set of words in Spanish that do not provide significant value for the text analysis are shown.

Table 4: Example list of ST

['de', 'la', 'que', 'el', 'en', 'y', 'a', 'los', 'del', 'se', 'las', 'por', 'un', 'para', 'con', 'no', 'una', 'su', 'al', 'lo', 'como', 'más', 'pero', 'sus', 'le', 'ya', 'o', 'este', 'sí', 'porque', 'esta', 'entre', 'cuando', 'muy', 'sin', 'sobre', 'también', 'me', 'hasta', 'hay', 'donde', 'quien', 'desde'

1.4 Python with natural language tool kit (PNLTK)

Python with natural language package tools. Python is a recent powerful programming language that allows complex tasks in a simple way. [13] It can use preloaded libraries to perform tasks automatically and quickly. According to [14] Python is a language for learning real world programming. In addition, [15] affirm that Python is the most appropriate language for learning programming; since it has powerful tools that reflect the way people think, therefore; they will be able to implement their code.

1.4.1 Advantages

Code transcription is so simple that facilitates its learning in a dynamic manner. It is a powerful tool for processing data and linguistics. The implementation is simple, even without having any previous programming knowledge. There is a diversity of libraries than can be implemented or rewritten to perform any activity. There are advanced functions and methods for development when one has prior Python knowledge. Furthermore, Python has libraries developed natively for the PLN. As a result, it makes the implementation easier. It is a language program that has multiple supports of thousands of programmers working together to improve the libraries. For instance, in [16], where projects are shared and questions can be asked, the community providers offer assistance to help you solve different problems. [17] Python is an open source tool, besides it is a multiplatform including, Windows, Mac Os and Linus

among others. As [18] assured Python is easy to use, it offers a better structure and a stronger support for heavier programs. It also has fewer errors than in C, making it a convenient and efficient program. It is an interpreter that saves you time during its execution since it does not require linkage or compilation. It is easy to install and it is an object oriented interpreter. It allows to type programs in a compact way, subsequently; it is shorter compared to developments in C, it stands out from other programs because it allows to add modules to the interpreter.

1.4.2 Disadvantages

No mistakes should be made when calling libraries, otherwise they will not work. It can be somewhat difficult to install additional complements that grant the library function. Libraries can take up a significant amount of space on the network, therefore; it is necessary to download and install them locally in order for it to work efficiently. Some programmers impose their own programing style, for instance; programming syntax of another language, for example C++, which could complicate the union with another program if it is not developed correctly.
In such way, [13] conclude that in spite of the PLN advances, one of the NTLK limitations is not to be able to make common sense reasoning and it is necessary to wait until Technology Science can solve such problems in the future.

1.5 Microsoft Visual C Sharp (MVCS)

MVCS is an object oriented language programming developed by Microsoft in 1998. Its syntax comes from the C and C++, using the object model of the NET platform.

1.5.1 Advantages

It is a very competitive language because of its high international use and its effortless adequacy in both installation and finding implementations that can be personalized with any application that is being developed. It also counts with an efficient, solid and modern interface. It uses an interface that permits to find and analyze in real time the syntax code and identifying immediately the mistakes. It can be integrated into any Microsoft application such as Office among others. This allows users to personalize their work environment, (identification of the labor environment, IDLE). It has the capacity to develop software components that allow the use in distributed environments. Being similar to C and C++, its migration by developers will be easily done. It helps the portability of the source code.

1.7.1 Disadvantages

Some complements need to be programmed and developed to adapt to the required needs. There are some existing developments that can be similar to PNLTK, however; they are limited since the programmers have a tendency for new language programming for their variety, simplicity and even for the ease to manipulate the program development. It is not multiplatform; it only works natively with Windows and Mac Os, but with limitations.

2. DEVELOPMENT

To be able to make a comparison, the MT preprocess programming in MT was necessary, while for PNLTK it was necessary to make adjustments to the Spanish language code.

2.1 MVCS and PNLTK Preprocess

MVCS is a precise and very useful language programming that allows the operation of different types of programs. It is a perfect IDLE for any development, it also allows to load extensions faster. Besides, the code debugging is simple and fast. In order to perform the MT processes, it was necessary to use the following libraries: System, System Collection, Generic, System Component Model, System Data, System drawing, System linq, System Text, System Threadings Tasks, System Windows Forms, System Reflection, System IO, System Text Regular Expressions. Python was downloaded and installed from the official website. Subsequently, it was necessary the installation of Numpy extension in order to aid with the management of mathematic functions. In addition, the NLTK library was installed, this one includes a group of libraries in charge of the PLN, assisting in making easier the linguistic sources previously installed. Table 5 shows the technical specifications of hardware and software that have been used to carry out this investigation.

Table 5: Hardware and software specifications

Hardware specifications:	Software specifications
Laptop XPS 13" processor Intel Core 15 with 8gb of ram.	Microsoft Windows 10, Microsoft Visual Studio 2017, Microsoft visual Code Python 3.6, NLTK. 3.3.5

2.2 File reading

To be able to open a text file in MVSC it was necessary to create a form using the openfiledialog.openfile method. In such way the open and close buttons are displayed allowing you to open the explorer form to browse the required file. After selecting, the result of the file opened is sent in a resulting variable. If the Dialog result method is correct, a class streamreader result is sent. It will be the same as the variable read, and the result will be show in Richtextbox, where it is seen as the desire file text using a total of 40 code lines to be able to program.

Fig. 2 shows the process diagram used to open the text file in MVCS for further analysis. This is the normal process to open a file, so this one can be analyzed by the MT preprocesses.

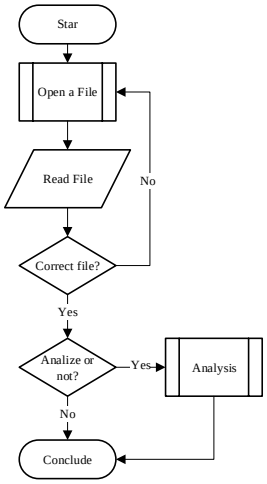


Fig. 2. Process diagram used to open the text file in MVCS Proposed beam former.

2.3 Spell Checker (SC)

[3] state that preprocesses are carried out efficiently using a spell checker as the first preprocess. The SC is a very useful tool achieving two main functions, the first one is to eliminate and correct spelling mistakes, while the second one helps to diminish the ST preprocess timing. In order for the SC to work in MVCS it was necessary to use the implicit tools in Microsoft Office. Streamreader class was used, this one allows reading the text files saving the results in a variable denominated read.

Subsequently, the SC denominated as component Check spelling was initiated, creating an app variable and saving them in a string. As long as the value is different to null, the SC will be executed. Afterwards through a conditional, words with spelling mistakes are found, they are pointed out and possible options are shown. If no spelling mistakes are found, another conditional takes you to the end of the text and asks you if you want to save changes. Finally, the results are sent to the textbox. Here a total of 75 code lines were used. In PNLTK a total of 7 code lines were used for its execution. Fig 3 shows the spell check process diagram with MVCS. And Fig 4 shows the spell check process diagram with PNTLK.

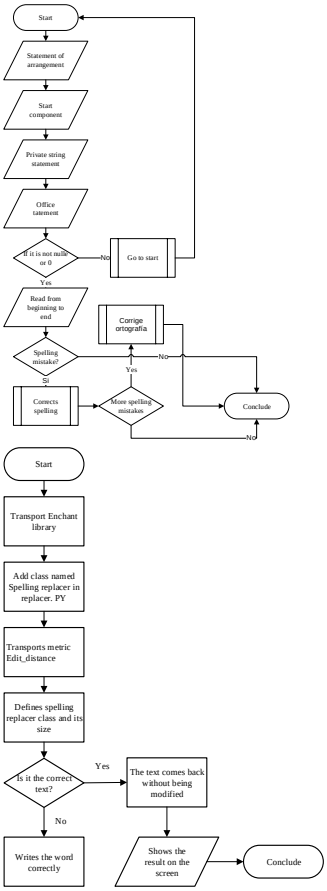


Fig. 3. Spell check process diagram MVCS. Fig. 4 Spell check process diagram PNLTK

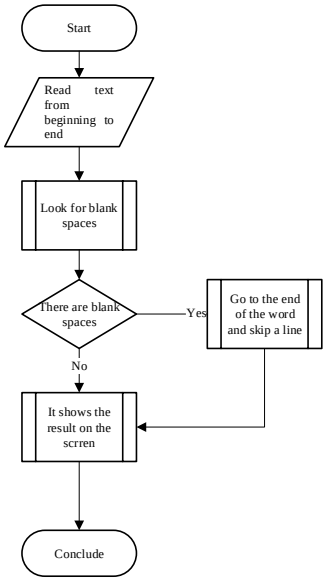
On Fig 3 we can see the process diagram correcting spelling mistakes used with MVCS provided by Microsoft Office. Fig 4 shows the process diagram to perform spell check with PNLTK. Enchanted library and Edit Distance metric are used to find the closest word in meaning to make the correction according to the previous definition with a minimum and a maximum distance until the closest word in meaning is found and suggests the correct alternative.

2.4 Tokenization

It consists in dividing all the elements into an independent one. Tokenization is based on discovering blank spaces in the text to later separate the elements

independently and transform them into tokens. To carry out tokenization in MVCS it was necessary a private class declaration called separate string. It was affirmed an arrangement called W which contains the Splitswords method, this one consists in creating a Substrins arrangement dividing the words by delimiters (blank spaces). The array resize function was used changing the number of unidimensional elements to a new specific size.

This is the easiest way to separate and convert a string into a sub-string or independent element. A total of 20 code lines for tokenization were used, while for PNLTK only 6 code lines were used. Fig 5 shows the Tokenization process diagram of a text in MVSC. Fig 6 shows the Tokenization process diagram of a text in PNLTK.



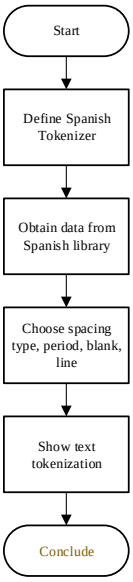
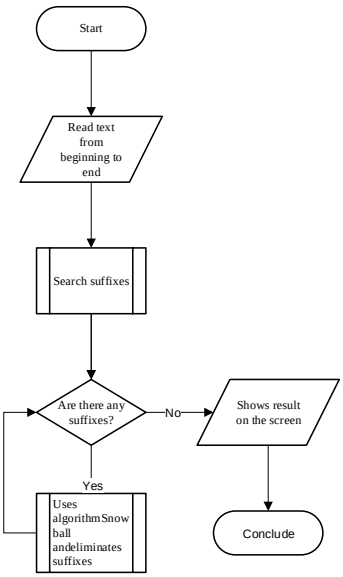


Fig. 5. Tokenization process diagram of a text MVCS. Fig. 6. Tokenization process diagram of a text PNLTK.

2.5 Stemming (ST)

It consists in eliminating the suffixes of the words and taking them to their root. Fig 7 shows the ST diagram; its objective is to eliminate suffixes of the word to take them to their root without affecting the meaning of it using the Snowball algorithm as an external module. A class called Class 1 was added as an independent module and it was declared a public String called execution. A public class called Spanish was declared, it contains the Snowball algorithm code as an object-method that encloses the different lexicographical rules to eliminate the suffixes of the words and take them to their root. It needs to be mentioned that more than 100 algorithm Snowball code lines were used coming from Porter algorithm. Therefore, it was an adaptation for the elaboration of this investigation. 10 code lines were used for PNLTK.

Fig 8 shows the ST diagram in PNLTK. The Snowball algorithm is a class that allows eliminating the suffix of the Spanish words to transform them to their root avoiding losing their meaning.



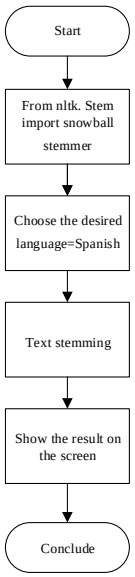
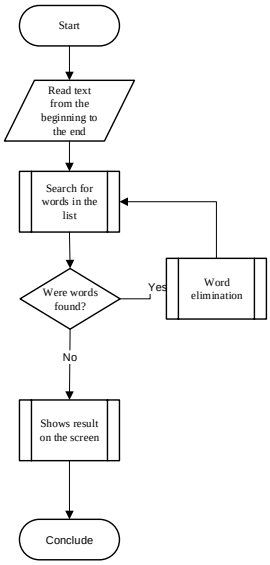


Fig. 7. ST diagram in MVCS. Fig. 8 St diagram in PNLTK.

2.7 Stop words (SW)

It consists in searching predefined words content in a list for its later elimination since they do not sum any value. It was necessary to create a static String in MVCS called Cleaninput2 with the String function. Through a conditional, symbols and words contained in a predefined list are searched, where if they are true, they are eliminated over a replace method, otherwise; it goes to the end and finishes its sequence. A total of 460 code lines are used. Fig 9 shows the SW process diagram in MVCS to eliminate words. Fig 10 shows the SW process diagram in PNLTK which contains a reader. Besides, it contains a group of preset words that do not offer any sematic value to the analysis process. Through a conditional, it searches for these words in the text, where if the result is true, they are directly eliminated and the results are displayed on the screen,16 code lines are used in NLTK.



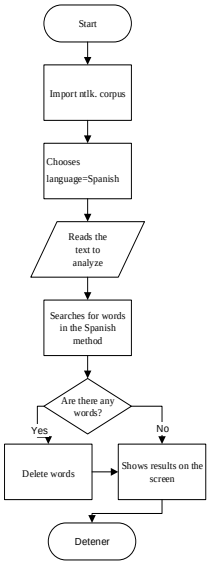


Fig. 9. SW diagram with MVCS. *Fig. 10 SW diagram in PNLTK*

2.7 Evaluation criteria among languages

The evaluation criteria allows similar characteristics to detonate among MVCS and PNLTK. Table 6 shows the comparison among MVCS and PNLTK for the MT processes.

Table 6: Evaluation criteria among languages Margin specifications

	Comparison categories	
	MVCS	PNLTK
Available documentation	Yes, online books (limited)	Yes online/forums/ books.
There is a need for the definition of data types.	Yes	No, Python automatically finds out the data types.
Compilation to execute	Yes	No, it does not require compilation be executed
Execution per modules	Yes	Yes
Portability	Yes, Windows and McOs can only be used in an operating system already developed. It can cause problems in the libraries being used if it is about migrating to another platform	Yes, multi-platform (Windows/Mac Os/Linux)
Sturdiness	Yes, when it is used in the correct environment it can be sturdy and efficient. Limited (it will depend of in the development model applied).	Yes, it allows to handle large volumes of data without problems. Yes, it allows multiple processes to be executed, depending on the hardware.
Licence type	Yes, it is free, however; it can also have significant cost in the market.	No. Tipo gratuito sin costo, software libre.
Popularity	Little. From its beginnings it was successful. However, it has been declining.	A lot Python is one of the most popular programming languages and at the same time with tendency to be one of the most used today for its simplicity, speed and easiness to learn how to use it. Being multiplatform, it helps to only add the necessary libraries to change platforms without generating any problem.

Object oriented	Yes	Yes
Connectivity with databases	Yes	Yes
Flexibility	Yes, limited to the knowledge of the programmer.	Yes, limited.
Performance	Yes, efficient if it was well programmed.	Yes, not compiling reduces runtimes.
Scalability	Yes, limited.	Yes, wide.
Learning	Yes, slow.	Yes, fast.
Consistency and ambiguity	The data and variables with which it will work should be defined without errors. If it is not defined correctly, the data size may cause errors when the code compilation is generated. For the programming of this task, the variables and types of data to be used in the customization of the algorithms in c Sharp had to be defined.	Python being a newer and different language allows the use of different variables and data and is suitable for the different sizes of data that are being used. Therefore, there could be errors when executing the code and not because of the definition or the size of the data being used. The PNLTK language tools were preset and defined to work regardless of the size or amount of data used. The PNLTK kit instructions are simple to generate text analysis.
Data types and structure	Yes. The definition of data is in a traditional way using algorithms that allow operations to be carried out to achieve the required analysis of texts, causing more time for the analysis of the development of the source code in the data structure.	Python automatically detects variable data types and structures, therefore; it causes less time consumption in setting data types.
Modularity	The algorithms were developed independently, but depending on the previous one	PNTLK performs it automatically just by giving you the instruction, so it is generated automatically and quickly.
Ease of entry and exit	The data is opened through a standard library such as UNIX which allows free access to files through basic instructions. Therefore, it is efficient and easy to perform.	Python uses a standard library to open any type of file, in this case it is a text file.
Efficiency and performanc e	The performance is efficient since it was custom development to carry out the preprocesses of the MT	The efficiency and performance are excellent since the libraries used in PNLTK were previously trained, so it will be fast to operate..

3. RESULTS

After executing the MT processes in both programming languages, the results are presented below:

3.1 Analyzed Words

In this investigation the preprocesses of TM were performed for the analysis of a set of words for each programming language. For MVCS the results obtained were a total of 14216 characters, forming a total of 2366 words. For PNLTK the results obtained from the analysis were a total of 2363 words. There was a variation of 3 characters in PNLTK because some characters were recognized as words.

3.2 Tokenization

Fig 12 shows the results of the comparison of execution time for the tokenization of a text.

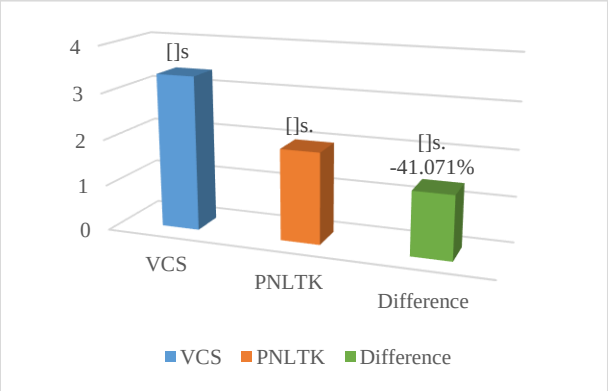


Fig. 12. Execution time for tokenization

In Fig 12 we can see that the tokenization execution time in MVCS was 3.36s, while in the PNLTK it was 1.98s, demonstrating that Python was faster at 41.071% because the code is not compiled for execution.

3.3 ST

The Fig13 shows the execution times for ST

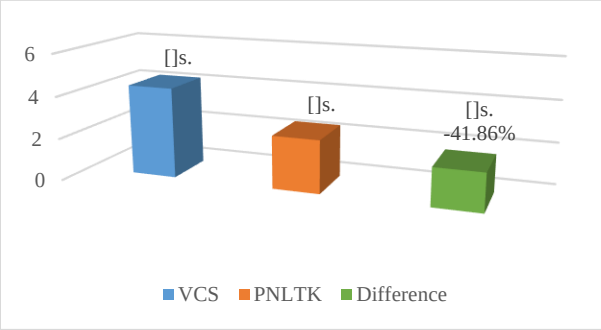


Fig. 13. Execution time for ST

It can be seen in Fig 13, that the execution time of ST in MVCS was 4.3s, while for PNLTK it was 2.5s, being less by 1.8s, equivalent to 41.86% of the total time, because there is no need to compile the code. Because the library used with the Snowball algorithm was previously trained, speed execution was achieved.

3.4 SW

The Fig 14 shows the comparison between MVCS and PNTK.

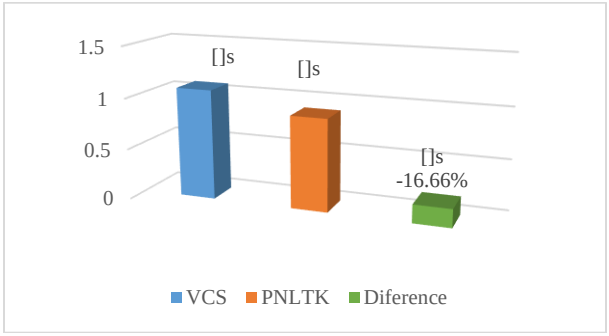


Fig. 14. Execution time SW

It is observed in Fig 14 that MVCS took 1.08s execution time, while for PNLTK 0.09s, equivalent to 16.66% less runtime in PNLTK because the library was previously trained.

3.5 Totals

The Fig 15 shows the total seconds of execution time of the preprocesses.

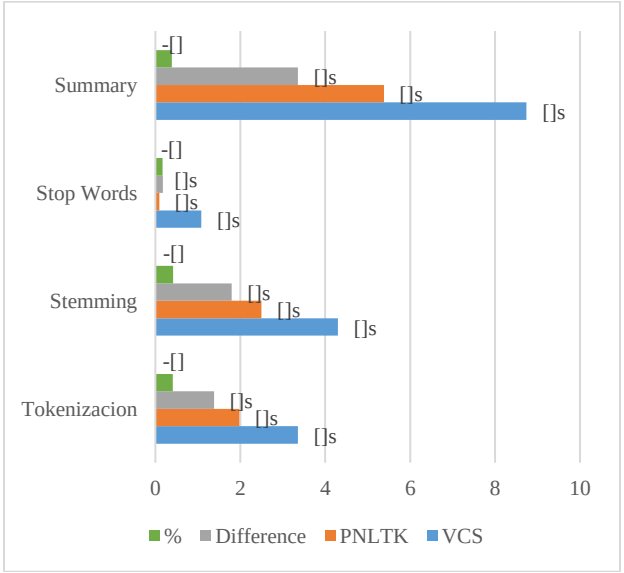


Fig. 15. Summary of preprocess execution

The total execution time in MVCS was 8.74s, while for PNLTK it was 5.38s, equivalent to -38.44%. PNLTK was 38.44% more efficient in the execution preprocesses.

4. CONCLUSIONS

MVCS allows you to use more tools belonging to Microsoft Office. In addition, it is in the Spanish language and in many other languages. We can take advantage of the fact that the tools are installed in Windows operating system, therefore; you can take more advantage of them. The spell check uses dictionaries that have already been improved in most existing Microsoft Office languages and a word comparison is made. The tokenization can be done by skipping lines, by blank spaces, or by commas, it depends on how it is needed according to the requirements.

All preprocesses were able to be programmed without any problem since MVCS is a very solid program and allows to create and use important libraries that help in code development. It is important to observe that MVCS allows to program or adapt any necessary code.

Most PNLTK language tools were made for the English language, however; some of them allow to be adapted to the Spanish language. It allows to make the correction of each of the words using the enchant library. Besides performing the combination of edit distance and apply the replacement method to change the necessary word or letter so the correct combination of letters forms a word. PNLTK tools allow tokenization in a quick way, being able to choose how to delimit the tokens either by comma, skipping lines, etc. ST and SW are previously entered libraries that make the execution take place immediately, almost instantly. In addition, Python is a friendly language that makes development efficient and simple. The benefits of using Python tools are an important advantage that was developed in order to meet the objective of analyzing large amounts of data volumes. In addition, they were previously trained which brings the benefit of speed execution in each of the processes. Other tools are included such as word classification and statistical tools that allow you to evaluate large volumes of data in an effective way.

Without a doubt PNLTK proves to be superior for its benefits of use and speed for the execution of the code. The programming instructions are simple, it is not necessary to be a professional programmer to execute them and consequently, the learning methods are simple.

REFERENCES

[1] A. K. Choudhary., P. I. Oluikpe., J.A. Harding, & P. M. Carrillo. The needs and benefits of text mining applications on post-project reviews. (2009). Computers in industry. (60)9, pp. 728-740. Obtenido el 24 de febrero de 2019 desde https://doi.org/10.1016/j.compind.2009.05.006 http://www.sciencedirect.com/science/article/pii/S01663 6150900102X

[2] R. A. Saravanan., & M. R. Babu. Text mining techniques and its applications: a survey. International Journal of Computer Science and Technology. (2017). 8(3), pp.33-34. ISSN: 0976-8491. Obtenido el día 26 de febrero de 2019 desde http://www.ijcst.com/vol8/8.3/ver1/7-r-annamalai-saravanan.pdf

[3] J. Quillo-Espino., R. M. Romero-González., & A. Lara-Guevara. Advantages of using a spell checker in text mining pre-processes. (2018). Journal of Computer and Communications. (6)11, pp.43-54. DOI: 10.4236/jcc.2018.611004 Obtenido el 25 de noviembre de 2018 desde https://www.scirp.org/journal/PaperInformation.aspx?PaperID=88461

[4] S. Kannan. & V. Gurusamy. Preprocessing techniques for text mining. (2015). Conference paper, obtenido el 19 de febrero de 2019 desde https://www.researchgate.net/publication/273127322_Preprocessing_Techniques_for_Text_Mining

[5] M. Allahyari, S. Pouriyeh., M. Assefi., S. Safaei., E. D., Trippe., J. B. Guitierrez., & K. Kochut. A brief survey of text mining, classification, clustering and extraction techniques. (2017). In Proceedings of KDD Bigdas, obtenido el 13 de mayo de 2019 desde https://arxiv.org/pdf/1707.02919.pdf

[6] A. Hotto. A brief survey of text mining. Journal for Computational Linguistics and Language Technology. (2005). LVD Forum (20), pp. 19-62, obtenido el 30 de marzo de 2019 desde https://www.researchgate.net/publication/215514577_A_Brief_Survey_of_Text_Mining

[7] K. L. Sumanthy., & M. Chidambaran. Text mining: concepts, applications, tools and issues- an overview. (2013). 80(4), p. 30. DOI: 10.5120/13851-1685, obtenido el 22 de febrero de 2019 desde http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.403.2426&rep=rep1&type=pdf

[8] Porter, M. F. (1980), An algorithm for suffix stripping, Program, 14(3) pp 130−137. Obtenido el 26 de abril de 2019 desde https://tartarus.org/martin/PorterStemmer/index.html

[9] Porter, M. F., Miles, P., Aksyonoff, A., Bartunov,O. (2006). Snowball Algorithm. Obtenido el 27 de abril de 2019 desde http://snowball.tartarus.org

[10] P. V. Baradad., & A-M. Mugabushaka. Corpus specific stop words to improve the textual analysis in scientometrics. (2015). ISSI, obtenido el 15 de abril de 2019 desde https://pdfs.semanticscholar.org/618d/a4f6d5cd329d3bc498d9457f575cbdfaf53d.pdf
[11] V. Gupta., & G. S. Lehal. A survey of text mining techniques and applications. (2009). Journal of emerging technologies in web intelligence. (1)1, pp.60-76, obtenido el 23 de marzo de 2019 desde http://learnpunjabi.org/pdf/gslehal-pap18.pdf
[12] A. S. Nayak., A. P. Kanive., N. Chandavekar. & R. Balasubramani. Survey on pre-processing techniques for text mining. (2016). International journal of engineering and computer science. 5(6). pp. 16875-16879. ISSN: 2319-7242, obtenido el 18 de abril de 2019 desde https://www.researchgate.net/publication/319716444_Survey_on_Pre-Processing_Techniques_for_Text_Mining
[13] S. Bird., E. Loper., & E. Klein. Natural Language processing with python. (2009). O' Reilly Media Inc. 17(3), p.33. ISBN: 978-0-596-51649-9, obtenido el 24 de abril de 2019 desde http://www.datascienceassn.org/sites/default/files/Natural%20Language%20Processing%20with%20Python.pdf
[14] K. R. Srinath., Python. The fastest growing programming language. (2017). International Research Journal of Engineering and Technology. 04(12), pp.354-357, obtenido el 25 de abril de 2019 desde https://www.irjet.net/archives/V4/i12/IRJET-V4I1266.pdf
[15] A. Bogdanchikov., M. Zhaparov., & R. Suliyev. Python to learn programming. (2013). Journal of physics. doi:10.1088/1742-6596/423/1/012027, obtenido el 28 de abril de 2019 desde https://www.researchgate.net/publication/258800484_Python_to_learn_programming
[16] Fuente de consulta para nltk obtenido el 21 de abril de 2019 desde https://stackoverflow.com/
[17] Fuente de consulta para Python obtenida el 22 de abril de 2019 desde https://www.python.org/
[18] G. V. Rossum. Python tutorial. (1995). Stiching mathematisch centrum, Amsterdam, The Netherlands, obtenido el 7 de marzo de 2019 desde https://gvanrossum.github.io/Publications.html.