AMSE JOURNALS-AMSE IIETA publication-2017-Series: Advances A; Vol. 54; N°2; pp 169-184

Submitted Apr. 04, 2017; Revised Sept. 25, 2017; Accepted Sept. 28, 2017

https://doi.org/10.18280/ama_a.540203

Design and Evaluation of Text Pre-Processor: A Tool for Text Pre-Processing

*Amit Prasad Rauth, **Anjan Pal

*Department of Computer Science & Engineering, OmDayal College of Engineering & Architecture, 39(P) & 39(A), Uluberia, West Bengal-711316, India. (amitrauth1234@gmail.com)

** Department of Computer Science & Engineering, OmDayal College of Engineering & Architecture, 39(P) & 39(A), Uluberia, West Bengal-711316, India. (anjanpal5@gmail.com)

Abstract

This paper introduces the Text Pre-processor, a tool that integrates several text pre-processing tasks such as tokenization, parts-of-speech tagging, and elimination of stop words. These pre-processing tasks are prerequisite for any text processing tasks such as sentiment analysis or text summarization. However, there does not exist any one-stop solution to perform multiple text pre-processing tasks. The Text Pre-processor serves to cover this gap. The tool includes five modules. These include text editor, single file processing, file to file processing, multiple file processing, as well as split and merge files. Informed by the technological acceptance model, a qualitative user study was conducted to evaluate the efficacy of the tool. Participants generally found the tool efficacious.

Key words

Natural language processing, Text processing, Text pre-processing, Text mining tool.

1. Introduction

In the era of ever-growing volume of information, fields such as text mining, natural language processing, and big data analysis are charting a meteoric rise. Both practitioners and researchers working in these fields rely on computational techniques to process huge volumes of information. There has been evolving interest in domains such as computational linguistics, corpus linguistics, information extraction, information retrieval, and discourse processing [1, 2]. These domains entail processing huge volumes of textual data for several text processing tasks such as sentiment analysis, and text summarization [3, 4].

However, text processing tasks are often cumbersome because they entail extensive pre-processing [5, 6]. Text pre-processing refers to transformations that need to be applied on texts before they can be meaningfully parsed and processed. It includes tasks such as tokenization, stop-words elimination, quantification of words, characters, vowels, white space, and sentences as well as parts-of-speech (POS) tagging [7, 8]. These tasks are often necessary before doing any major text processing tasks for further investigation.

Although many advances have been done in text processing tasks, little has been done to automate multiple text pre-processing tasks on a single platform. Prior studies mostly rely on different tools to conduct text pre-processing tasks [9]. For example, some studies used the Stanford Parser's POS tagger to find the proportions of various POS in texts, and some other customized Java programs to compute measures such as characters per word, and words per sentence. Likewise, some studies used the Stanford Parser to obtain POS tagging, while some undisclosed tool was used for lower-casing of text.

Even though there are many distinct tools available that can perform such text pre-processing tasks in isolation, they either perform basic pre-processing tasks such as word count, and upper-lower casing; or advanced pre-processing tasks such as stop-word elimination and POS tagging [10, 11]. Moreover, these tools do not facilitate the pre-processing tasks repeatedly on multiple files. Furthermore, there exists no single tool that integrates many text pre-processing tasks. For these reasons, this paper introduces the Text Pre-processor, a tool that performs several pre-processing tasks through the use of a Graphical User Interface (GUI). The tool was developed using Java Application Programming Interface (Java API). A user study was conducted to examine the efficacy of the tool.

2. Motivation and Significance

Several text mining tools have been developed for mining knowledge from huge amount of text. Holzman presented the design of such a tool for text mining developed in C++ [12]. OpenNLP [13] is a collection of open source tools for NLP. It focuses on feature extraction through annotation. LingPipe is another example of similar text mining tool [14]. Besides, there are a few tools available in the market to pre-process text. Many tools such as notepad++, and editplus can perform some of the pre-processing tasks on text [15, 16]. However, these tools were not designed to support several pre-processing tasks all at one place. This calls for the development of a stand-alone text pre-processing tool.

The significance of this tool is three-fold. First, to the best of the authors' knowledge, this represents one of the earliest attempts to build a tool to integrate multiple text pre-processing tasks. It serves the purposes of various pre-processing tools in an integrated processing unit. Second, this tool serves as an invitation for budding scholars to embark on text mining research. Researchers need not be overly concerned about text pre-processing tasks in handling text corpus. The tool can smoothen the perceived learning curve by offering a one-stop solution for multiple text pre-processing tasks. Third, from the perspective of programmers, this paper demonstrates the use of Java API for handling text under a GUI. The choice of programming language makes the tool platform independent in nature [17].

3. Features of the Tool

The major features of the tool include:

- Provide a user-friendly GUI that can be used by experts and amateurs alike.
- Facilitate an editor to perform different operations such as cut, copy, and paste on the selected file as per user requirements.
- Create new text file.
- Browse an existing text file from any hard drive for processing.
- Save the modified file content with proper extensions.
- Provide different font colors and background colors to make textual content attractive on the editor.
- Support popular keyboard short-cuts such as Ctrl+O for opening the file, Ctrl+S for saving, Alt+F4 for quit.
- Allow quantification of words, vowels, consonants, and sentences on a selected file.
- Eliminate stop-words, periods, and consecutive white spaces from a selected text file.
- Provide POS tagging of text in a file.
- Allow merging of several files into single file.
- Allow to split a large file into several smaller files.
- Facilitate text pre-processing tasks on multiple files.
- Provide a help menu to work with it.

4. Tasks Performed by Text Pre-processor

Before doing any text processing tasks, it is necessary to pre-process the collection of textual data. Text has a rich substrate hierarchically comprising of paragraphs. Each paragraph consists of a sequence of sentences, whereas each sentence consists of a sequence of words that are separated by delimiters. To deal with textual data, the tool has aimed to achieve the following pre-processing tasks:

Tokenization

Tokenization is the process of splitting a text into tokens. The text is converted to tokens through one or more tokenizing functions such as removing all punctuation and extracting word-chunks by grouping characters delimited by spaces [18]. Thus, the tool generates tokens as non-empty sequence of characters excluding spaces and punctuation for fine-grained text processing. Moreover, the tool can generate each sentence as a token for coarse-grained processing of text. Superficial characteristics can be extracted from a text file. Additionally, the tool facilitates calculating the total number of words, lines, vowels, consonants, digits, letters, spaces, and sentences in a file.

Elimination of Consecutive White Spaces, Duplicate Sentences, and Numbers

In case of white space elimination, the tool can substitute consecutive white spaces by single space for the content present in the text file. By eliminating consecutive white spaces, it allows to quantify characters in a file with a high accuracy [19]. The tool can eliminate all duplicate sentences that are redundant from a text file after taking them in the format of collection of sentences. Number elimination facilitates removal of all the numeric digits from a text file in order to improve performance of text clustering algorithm in terms of outlier elimination [20]. It also facilitates extraction of numbers rather than simply eliminating them from a specific text file.

Punctuation and Stop-word Removal

Punctuation removal entails eliminating all types of punctuations such as periods, semicolons and exclamations from the content of a text file. It helps in generating tokens from chunk of text content without accounting for the punctuations, and to normalize text. It also facilitates extracting lexical features such as character counts from the text content. Stop-word removal entails eliminating words such as articles, conjunction and prepositions [11, 21]. Its primary use is to prevent processing of text from being over influenced by very frequent words.

Extraction of Sentences and Paragraphs

Pre-processing tasks include extraction of sentences and paragraphs by matching a specified word given by users. Comparative sentences and paragraphs are retrieved from the text content against a search word for segmentations [22]. These pre-processing tasks are commonly associated with text summarization, sentiment analysis, and knowledge extraction [23].

POS Tagging

Extracting accurate natural language information from a text entails tagging each word with its POS tag [24]. Many text-based tools use POS taggers, which identify POS such as noun, adjective and adverb of a word and tag it accordingly. Then taggers parse the tagged words into grammatical phrases to help distinguish the semantics of the component words. POS taggers for English texts are built using machine learning techniques that depend on the likelihood of a tag given a single word and the tags of its surrounding words in order to contextualize training data. POS tagging is widely used in several natural language processing tasks such as sentiment analysis, text-to-speech conversion, information retrieval, shallow parsing, information extraction, semantics, and translation [25, 26].

Splitting and Merging of Files

Processing text corpus often endeavors processing of either a large single text file or a large number of multiple text files [27]. However, it is often required to split the single large file into multiple files or to merge multiple files into a single one according to the requirement of the processing tasks [28]. The manual process of doing these splitting and merging operations is time consuming and lead to fatigue. Thus, handling a text corpus often lead to such pre-processing tasks.

5. Design and Implementation of Text Pre-processor

The design and implementation of the tool could be broadly classified into two levels: System level design and detailed level design. The system level design is associated with the different modules of Text Pre-processor whereas the detailed level design pertains to the internal implementation details of each module.

5.1 System Level Design

This level tends to identify modules, their specification, and how they interact with each other to implement the desired system. The tool can be functionally divided into the following five modules:

Module 1: Text Editor

Module 2: Single File Processing

Module 3: File to File Processing

Module 4: Multiple File Processing

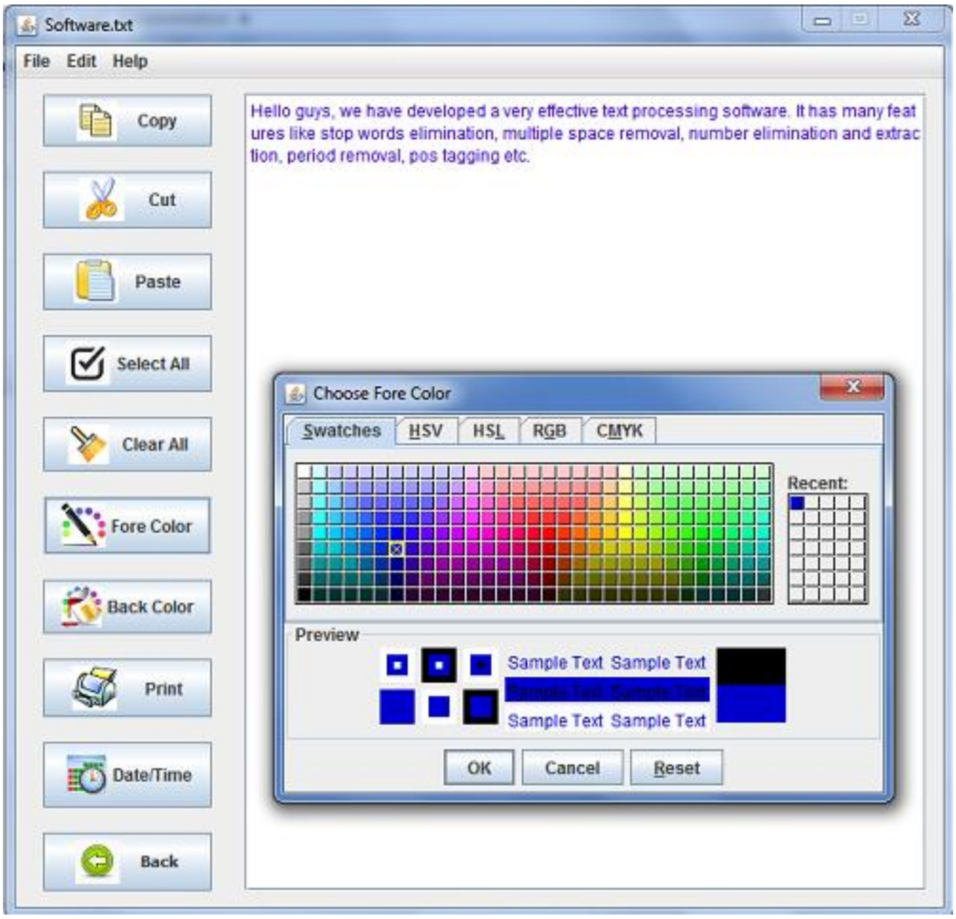
Module 5: Split and Merge Files

5.2 Detailed Level Design

In detailed level design, the internal details of each of the modules are specified. The internal details of all the five modules are given below:

Module 1: Text editor

This module provides an editor to process textual content as shown in Fig. 1. The editor entails the use of a text area from the class of editor pane.



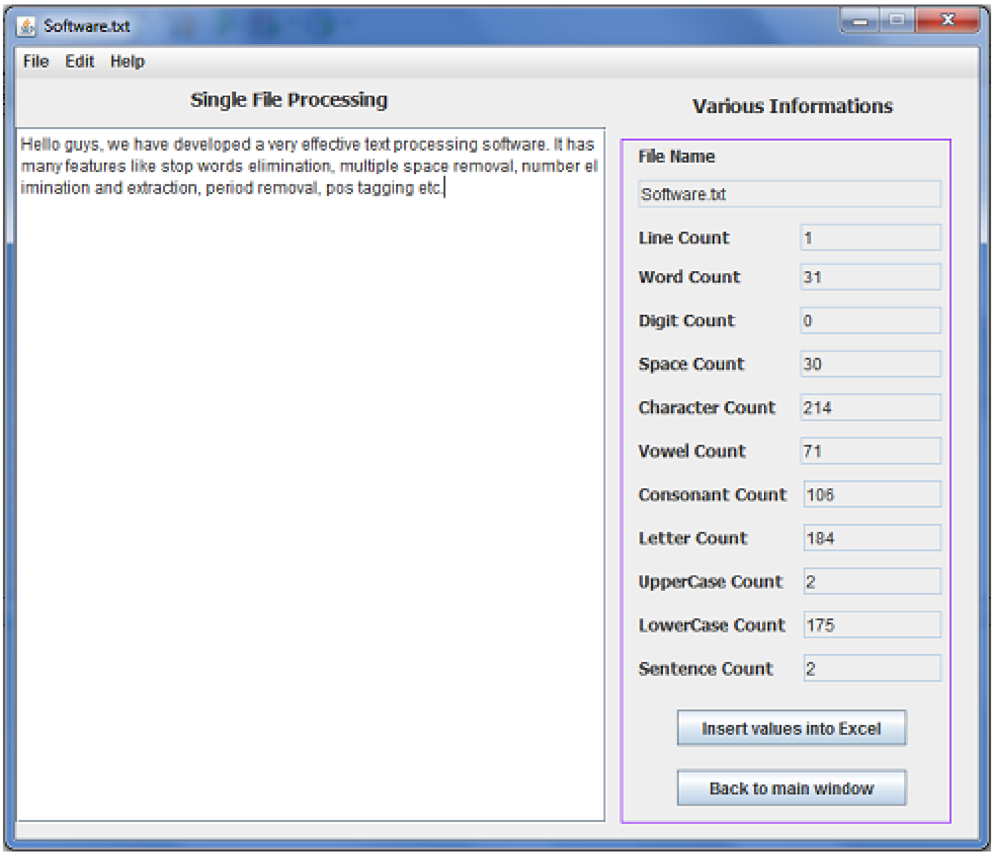
Hello guys, we have developed a very effective text processing software. It has many features like stop words elimination, multiple space removal, number elimination and extraction, period removal, pos tagging etc.

Fig.1. Screenshot of Module 1

This module provides three menu options, namely, file, edit, and help associated with other menu items such as open, save, and new under each of them. It allows users to create new text file, open an existing text file, and save current text file. It includes editing options such as cut-paste, copy-paste, and select all. The character encoding scheme of this editor supports UTF-8, a universal character set that allows to process multilingual content. The functionality of browsing existing files into the editor is adjusted in such a way that the user can select only the mentioned file extensions such as .txt, .java, .c, .cpp, .html. Hence, users can not browse other types of files such as image video, audio in the editor that would make the module dysfunction.

Module 2: Single file processing

This module provides different quantitative information for a selected file containing textual data as shown in Fig. 2. Different quantitative information includes total no. of words, lines, vowels, digits, blank spaces, characters, consonants, letters, and sentences present in the file. This module facilitates to import all these lexical features along with the selected file name into an excel. It can process single file at a point of time.



Single File Processing

Hello guys, we have developed a very effective text processing software. It has many features like stop words elimination, multiple space removal, number el imination and extraction, period removal, pos tagging etc.

Various Informations

File Name: Software.txt

Line Count	1
Word Count	31
Digit Count	0
Space Count	30
Character Count	214
Vowel Count	71
Consonant Count	106
Letter Count	184
UpperCase Count	2
LowerCase Count	175
Sentence Count	2

Fig.2. Screenshot of Module 2

Module 3: File to file processing

This module allows to process file to file operations as shown in Fig. 3. It allows users to select a file for different operations such as consecutive white space elimination, duplicate line elimination, and stop word elimination. The two buttons namely, select source file and choose destination file are used to browse a source file and to select destination location along with a file name to store the required output respectively. After processing the selected file, the output is generated in a new file by keeping the original intact. The module also facilitates all the operations by a click on the button named “all function.” To do all the operations at a time, it is required to browse all the destination file locations separately by clicking on the “browse file” button for each operation. This module also allows to search all the sentences and paragraphs from a selected file containing a specific word given by users.

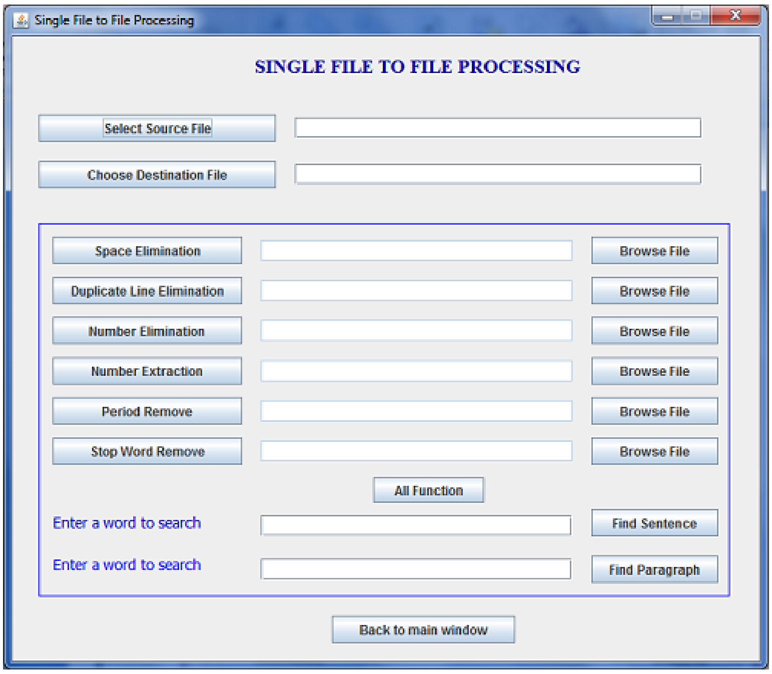
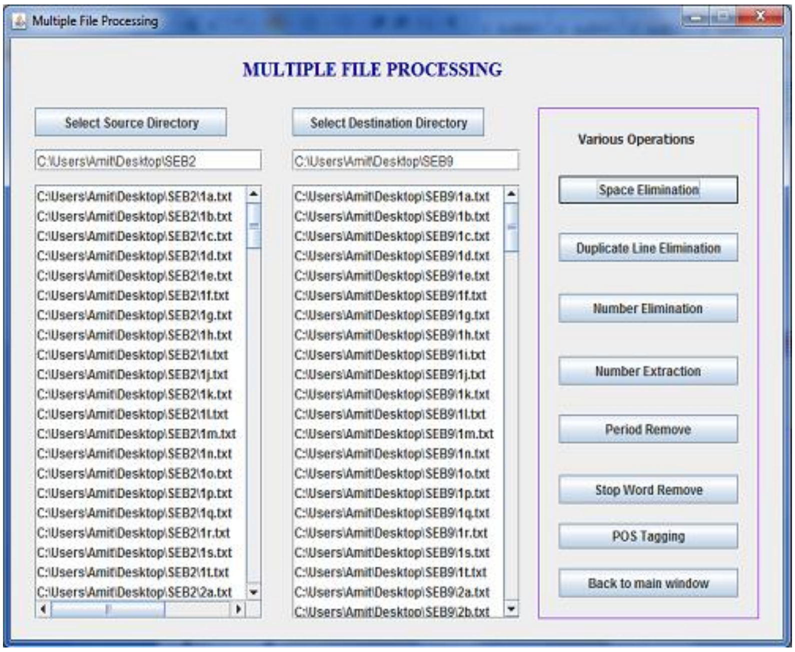


Fig.3. Screenshot of Module 3



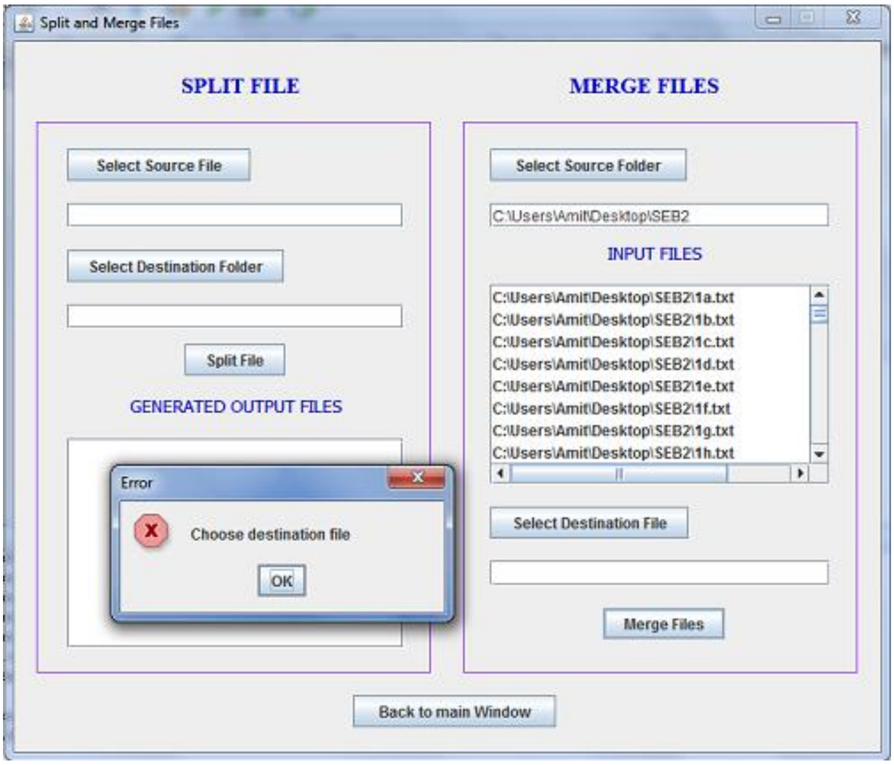
```
C:\Users\Amit\Desktop\SEB2
C:\Users\Amit\Desktop\SEB2\1a.txt
C:\Users\Amit\Desktop\SEB2\1b.txt
C:\Users\Amit\Desktop\SEB2\1c.txt
C:\Users\Amit\Desktop\SEB2\1d.txt
C:\Users\Amit\Desktop\SEB2\1e.txt
C:\Users\Amit\Desktop\SEB2\1f.txt
C:\Users\Amit\Desktop\SEB2\1g.txt
C:\Users\Amit\Desktop\SEB2\1h.txt
C:\Users\Amit\Desktop\SEB2\1i.txt
C:\Users\Amit\Desktop\SEB2\1j.txt
C:\Users\Amit\Desktop\SEB2\1k.txt
C:\Users\Amit\Desktop\SEB2\1l.txt
C:\Users\Amit\Desktop\SEB2\1m.txt
C:\Users\Amit\Desktop\SEB2\1n.txt
C:\Users\Amit\Desktop\SEB2\1o.txt
C:\Users\Amit\Desktop\SEB2\1p.txt
C:\Users\Amit\Desktop\SEB2\1q.txt
C:\Users\Amit\Desktop\SEB2\1r.txt
C:\Users\Amit\Desktop\SEB2\1s.txt
C:\Users\Amit\Desktop\SEB2\1t.txt
C:\Users\Amit\Desktop\SEB2\2a.txt
```

```
C:\Users\Amit\Desktop\SEB9
C:\Users\Amit\Desktop\SEB9\1a.txt
C:\Users\Amit\Desktop\SEB9\1b.txt
C:\Users\Amit\Desktop\SEB9\1c.txt
C:\Users\Amit\Desktop\SEB9\1d.txt
C:\Users\Amit\Desktop\SEB9\1e.txt
C:\Users\Amit\Desktop\SEB9\1f.txt
C:\Users\Amit\Desktop\SEB9\1g.txt
C:\Users\Amit\Desktop\SEB9\1h.txt
C:\Users\Amit\Desktop\SEB9\1i.txt
C:\Users\Amit\Desktop\SEB9\1j.txt
C:\Users\Amit\Desktop\SEB9\1k.txt
C:\Users\Amit\Desktop\SEB9\1l.txt
C:\Users\Amit\Desktop\SEB9\1m.txt
C:\Users\Amit\Desktop\SEB9\1n.txt
C:\Users\Amit\Desktop\SEB9\1o.txt
C:\Users\Amit\Desktop\SEB9\1p.txt
C:\Users\Amit\Desktop\SEB9\1q.txt
C:\Users\Amit\Desktop\SEB9\1r.txt
C:\Users\Amit\Desktop\SEB9\1s.txt
C:\Users\Amit\Desktop\SEB9\1t.txt
C:\Users\Amit\Desktop\SEB9\2a.txt
C:\Users\Amit\Desktop\SEB9\2b.txt
```

Fig.4. Screenshot of Module 4

Module 4: Multiple File Processing

This module allows users to process multiple files from a selected source directory in order to speed up the repetitive tasks for those operations that have been done in the Module 3 as a single file processing unit. After browsing the source directory where the files are kept for processing, and the destination directory where the files need to be stored after processing, users can click any of the operations that are to be done on such multiple files. All the generated output files take the names of their corresponding input files while being created during the processing. Moreover, this module additionally includes the POS tagging feature that have been done commonly in many NLP tasks such as sentiment analysis and text summarization. It uses the Stanford POS tagging parser in the back-end. The screenshot of this module is shown in Fig. 4.



C:\Users\Amit\Desktop\SEB2

C:\Users\Amit\Desktop\SEB2\1a.txt
C:\Users\Amit\Desktop\SEB2\1b.txt
C:\Users\Amit\Desktop\SEB2\1c.txt
C:\Users\Amit\Desktop\SEB2\1d.txt
C:\Users\Amit\Desktop\SEB2\1e.txt
C:\Users\Amit\Desktop\SEB2\1f.txt
C:\Users\Amit\Desktop\SEB2\1g.txt
C:\Users\Amit\Desktop\SEB2\1h.txt

Choose destination file

Fig.5. Screenshot of Module 5

Module 5: Split and Merge Files

This module performs splitting and merging operations on files. It helps to split a large file into multiple files having smaller sizes. To do this, users need to browse the source file that needs to be split, and the destination directory that stores the multiple files generated during processing. Conversely, it is possible to merge several files into a single file. The screenshot of this module is shown in Fig. 5.

6. Evaluation of Text Pre-processor

A user study was conducted to evaluate the efficacy of the tool. A total of 20 participants were recruited based on convenience sampling. Among them, 10 participants had undergraduate degrees in computer science with a mean age of 24.40 years. The rest hailed from other disciplines such as business and information studies with a mean age of 25.10 years. All of them had interest and preliminary experience in text processing.

The user study comprised three steps. In the first step, a set of pre-compiled deck of slides was used to demonstrate the tool to the participants. Slides contained a brief introduction to the Text Pre-Processor followed by descriptions of the pre-processing tasks that are facilitated by the tool. Next, participants were shown slides that described the different modules of the tool. This was followed by video demonstration of a few tasks associated with each module.

In the second step, the participants were given a set of instructions containing 10 pre-processing tasks—two from each module, and some sample text files for performing the tasks. They had to complete at least five pre-processing tasks, one from each of the five modules. The tasks are listed in Table 1.

Table 1. List of Text Pre-Processing Tasks

Module	Text pre-processing tasks
1	Task 1: Browse a sample file to delete the existing textual content and write your name and save it.
	Task 2: Change the back color of the editor.
2	Task 3: Browse a sample text file, and report number of words and sentences in the content.
	Task 4: Browse a sample text file, and report number of characters and vowels in the content.
3	Task 5: Find all the sentences containing a specific search word from a given sample file, and save the generated output file in a given location.
	Task 6: Eliminate all punctuations from a sample text file, and save the generated output file in a given location.
4	Task 7: POS tagging for the textual content of all the sample files and save the corresponding output files in a given location.
	Task 8: Eliminate the stop words from all the given sample files and save the corresponding output files in a given location.
5	Task 9: Merge all the given sample files into a single file.

	Task 10: Split a sample file into multiple files.

In the last step, a questionnaire was administered to the participants regarding perceived usefulness and perceived ease of use of the tool as shown in Table 2. Additionally, both positive and negative feedback about the tool was sought. The questions were derived from Davis' technology acceptance model (TAM) that explains the acceptance of the new developed products and services at users' end [29, 30]. TAM has been widely used to explain users' perceptions regarding adoption of new tools and technologies [31, 32].

According to TAM, users' perceptions can be measured by using two variables that impact adoptions of new tools and technologies. The two variables are perceived usefulness and perceived ease of use. Perceived usefulness refers to the degree to which a user believes that using a particular tool or technology would enhance his or her job performance [32]. Perceived ease of use refers to the degree to which a user believes that using a particular tool or technology would not be cognitively challenging [32].

Table 2. Questionnaire Items Used in the Study

Construct	Questionnaire items
Perceived usefulness	What is the potential of the tool to enhance your performance in text pre-processing tasks?
Perceived ease of use	Does the tool make it easier for you to pre-process text? Why?
General feedback	What did you like and dislike about the tool?

With respect to perceived usefulness, majority of the participants perceived Text Pre-processor to be a useful tool. In particular, its perceived usefulness stems from its functionality for various test pre-processing tasks performed on multiple files. For instance, participant 2 expressed that the tool "could be effective to handle multiple files" while participant 7 liked the concept of the automated file naming system while doing work on multiple file processing module. Some participants also suggested that the tool could be useful for pre-processing tasks on text while handling a text corpus. For example, participants 15 and 19 related that the tool can "encourage researchers" to "solve their text pre-processing difficulties" that are often cumbersome before conducting any text processing tasks.

With respect to perceived ease of use, it was unanimously agreed that the interfaces of all the modules was easy to learn and use. In particular, participant 12 shared that "the interface is quite user-friendly with features that can be easily understood." Participants also felt that the navigation was unambiguous. For example, participant 1 complimented, “It is pretty easy to navigate while performing multiple file processing tasks, and requires maximum of three-four clicks to complete the task.” However, few participants stated that the interface could be made a bit more attractive.

In terms of positive feedback, the majority of the participants perceived the Text Pre-processor to be a useful tool in the field of text pre-processing. In particular, its perceived usefulness stems from its power to handle multiple files for a single operation which is tedious to perform separately for each file to them. For instance, participant 3 stated, “The tool is helpful because it works not only on single file but also it helps in handling multiple files. It really takes the hurdle of handling monotonous tasks for multiple files.” Participant 5 was impressed with the facility of “performing several pre-processing tasks under a single tool.” Moreover, on the response of perceived ease of use it was unanimously agreed that the tool has relatively simple interface, and that it was easy to learn and use. In particular, participant 14 shared that “the tool is quite user-friendly with all features. All the tasks starting from selecting a file, perform operations and after that saving the file can be easily done.”

In terms of negative feedback, among the most common dislikes include the unattractive interface. For example, participant 4 indicated, “the graphical interface can be made a bit more attractive.” Some of the participants also suggested to add more functionalities related to some of their fields such as stemming and other customized processing of text. For example, participants 17 commented that “the tool does not perform operations such as stemming.”

7. Conclusion

Recent years have been witnessing a growing interest in text processing tasks [1, 5]. However, these tasks are cumbersome because of extensive text pre-processing tasks that are seldom conducted using a single tool. Hence, this paper introduced Text Pre-processor, a tool to pre-process text.

This paper represents one of the earliest attempts to build a tool in order to integrate multiple text pre-processing tasks that are prerequisite for almost all text processing applications. The tool serves the purposes by providing many text pre-processing tasks to both practitioners and researchers. Moreover, the tool makes it easier for users to process multiple files that could otherwise have been time-consuming. The tool can smoothen the perceived learning curve by offering a one-stop solution for multiple text pre-processing tasks.

The development of the tool relied on Java API. The choice of this programming language makes the tool platform independent where the adoption of new technological trends often makes the change in the platform itself. Thus, it facilities a cross platform solution. Since Java program code is machine independent, and is designed to run in an identical manner on target platforms ranging from embedded systems to high-end workstations, the tool has the potential to work with other systems. Furthermore, Java API allows to facilitate the design of user-friendly GUI that makes the tool to be used by amateur also.

The paper is constrained by two limitations. First, it used only qualitative data collection to obtain participants' opinion about the Text Pre-processor. Second, the sample size was small. Using a larger sample coupled with both qualitative and quantitative data collections might have yielded richer insights.

In response to the users' study while evaluating the tool, this paper creates two opportunities for future research. First, the modules of this tool could be extended by further adding more functionalities such as stemming and other customized pre-processing tasks to meet further users' requirements identified during the evaluation phase. This will further enrich the tool to make the set of the pre-processing tasks complete.

Second, the follow-up version could conduct a user study to test the technology acceptance model with a large set of participants. It will help to explain the factors such as perceived ease of use and perceived usefulness that affects users' perceptions and acceptance of the tool. Furthermore, the look-and-feel of the GUI could be made more attractive than the present version.

Acknowledgment

The authors would like to thank Mullick Kabirul Huda, Subhendu Maity, Subhajit Das, and Sk. Abdul Nasim for their help in this project.

References

1. I. Pollach, Taming textual data: The contribution of corpus linguistics to computer-aided text analysis, Organizational Research Methods, vol. 15, 2012, pp. 263-287.
2. S. Pletscher-Frankild, A. Pallejà, K. Tsafou, J. X. Binder, L. J. Jensen, DISEASES: Text mining and data integration of disease–gene associations, Methods, vol. 74, 2015, pp. 83-89.
3. S.A. Crossley, L.K. Allen, K. Kyle, D. S. McNamara, Analyzing discourse processing using a simple natural language processing tool, Discourse Processes, vol. 51, 2014, pp. 511-534.

4. P. Velardi, P. Fabriani, M. Missikoff, Using text processing techniques to automatically enrich a domain ontology, Proceedings of the International Conference on Formal Ontology in Information Systems-Volume 2001, 2001, pp. 270-284.
5. E. Haddi, X. Liu, Y. Shi, The role of text pre-processing in sentiment analysis, Procedia Computer Science, vol. 17, 2013, pp. 26-32.
6. D. Munková, M. Munk, M. Vozár, Data pre-processing evaluation for text mining: Transaction/sequence model, Procedia Computer Science, vol. 18, 2013, pp. 1198-1207.
7. Z. Ceska, C. Fox, The influence of text pre-processing on plagiarism detection, Proceedings of the Association for Computational Linguistics, 2011, pp. 55-59.
8. N.A. Ghani, S.S.M. Kamal, A sentiment-based filteration and data analysis framework for social media, Proceedings of the International Conference on Computing and Informatics, 2015, pp. 632-637.
9. S. Banerjee, A.Y.K. Chua, J.J. Kim, Using supervised learning to classify authentic and fake online reviews, Proceedings of the International Conference on Ubiquitous Information Management and Communication, 2015, pp. 88:1-7.
10. M. Ott, C. Cardie, J.T. Hancock, Finding deceptive opinion spam by any stretch of the imagination, Proceedings of the Association for Computational Linguistics, 2011, pp. 309-319.
11. W.J. Wilbur, K. Sirotkin, The automatic identification of stop words, Journal of Information Science, vol. 18, 1992, pp. 45-55.
12. L.E. Holzman, T.A. Fisher, L.M. Galitsky, A. Kontostathis, W.M. Pottenger, A software infrastructure for research in textual data mining, International Journal on Artificial Intelligence Tools, vol. 13, 2004, pp. 829-849.
13. D. Ho, Notepad++, 2011, Retrived from: http://notepad-plus-plus. org.
14. R. Godwin-Jones, Emerging technologies: Web-writing 2.0: Enabling, documenting, and assessing writing online, 2008, Language Learning & Technology, vol. 12, pp. 7-13.
15. C.C. von Bastian, A. Locher, M. Ruflin, Tatool, A Java-based open-source programming framework for psychological studies, 2013, Behavior Research Methods, vol. 45, pp. 108-115.
16. G. Neumann, J. Piskorski, A shallow text processing core engine, Computational Intelligence, vol. 18, 2002, pp. 451-476.

17. V. Tunali, T.T. Bilgin, PRETO: A high-performance text mining tool for preprocessing Turkish texts, 2012, Proceedings of the International Conference on Computer Systems and Technologies, pp. 134-140.
18. K. Shi, L. Li, High performance genetic algorithm based text clustering using parts of speech and outlier elimination, 2013, Applied Intelligence, vol. 38, pp. 511-519.
19. A.K. Uysal, S. Gunal, The impact of preprocessing on text classification, 2014, Information Processing & Management, vol. 50, pp. 104-112.
20. M.K. Dalal, M.A. Zaveri, Automatic classification of unstructured blog text, Journal of Intelligent Learning Systems and Applications, vol. 5, 2013, pp. 108-114.
21. N. Jindal, B. Liu, Identifying comparative sentences in text documents, Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval, 2006, pp. 244-251.
22. M. Pontiki, D. Galanis, J. Pavlopoulos, H. Papageorgiou, I. Androutsopoulos, S. Manandhar, Semeval-2014 task 4: Aspect based sentiment analysis, Proceedings of the International Workshop on Semantic Evaluation, 2014, pp. 27-35.
23. M. Mitray, A. Singhalz, C. Buckleyyy, Automatic text summarization by paragraph extraction, Compare, vol. 22215, 1997, pp. 26:1-11.
24. R. Feldman, Techniques and applications for sentiment analysis, Communications of the ACM, vol. 56, 2013, pp. 82-89.
25. S.N. Kim, O. Medelyan, M.Y. Kan, T. Baldwin, Automatic key phrase extraction from scientific articles, Language Resources and Evaluation, vol. 47, 2013, pp. 723-742.
26. E.W. Brown, J.P. Callan, W.B. Croft, Fast incremental indexing for full-text information retrieval, Proceedings of the Very Large Data Bases Conference, 1994, Internet: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.46.6221&rep=rep1&type=pdf [Dec 26, 2015].
27. T. Erjavec, C. Ignat, B. Pouliquen, R. Steinberger, Massive multi lingual corpus compilation: Acquis communautaire and totale, Archives of Control Science, vol. 15, 2005, pp. 529-540.
28. S.C. Lewis, R. Zamith, A. Hermida, Content analysis in an era of big data: A hybrid approach to computational and manual methods, Journal of Broadcasting & Electronic Media, vol. 57, 2013, pp. 34-52.
29. F.D. Davis, R.P. Bagozzi, P.R. Warshaw, User acceptance of computer technology: A comparison of two theoretical models, Management Science, vol. 35, 1989, pp. 982-1003.

30. L.G. Wallace, S.D. Sheetz, The adoption of software measures: A technology acceptance model (TAM) perspective, Information & Management, vol. 51, 2014, pp. 249-259.
31. G.H. Subramanian, A replication of perceived usefulness and perceived ease of use measurement, Decision Sciences, vol. 25, 1994, pp. 863-874.
32. F.D. Davis, Perceived usefulness, perceived ease of use, and user acceptance of information technology, MIS Quarterly, vol. 13, 1989, pp. 318-340.