



DESIGNING A PYTHON BASED TEXT PRE-PROCESSING APPLICATION FOR TEXT CLASSIFICATION

H A Putranto^{1,2}, T Rizaldi^{1,3}, W K Dewanto^{1,4} and W Pebrianto^{1,5}

¹ Information Technology Department, Politeknik Negeri Jember, Mastrip 164 Jember, East Java

² hermawan_ariief@polije.ac.id

³ taufiq_r@polije.ac.id

⁴ wahyu@polije.ac.id

⁵ wahyu.pebrianto1@gmail.com

Abstract. The first step that is always passed by documents in natural language processing is pre-processing text. These steps are needed for transferring text from human language to machine-readable format for further processing. However, not many special applications have been found that function as text pre-processing. This has led to any research on natural language processing having to create its own program code for the pre-processing text phase. The main focus of this research is to create an integrated text pre-processing application that can be accessed by any researcher who needs it. Several issues discussed in this study include the design, implementation, testing and integration of each text pre-processing feature. Text pre-processing which is integrated in this research includes case folding, tokenizing, and feature selection. The tools used in this research are the NLTK library of python and Django framework. The design of the text pre-processing application can be made using the waterfall method. For the application stage, the utilization of the NLTK Library can be applied precisely and systematically. This library also facilitates the implementation phase because of the large number of NLP classes that can be directly applied.

1. Introduction

Text classification or text categorization is a process that automatically places text documents into a predetermined category based on the contents of the text [1]. Today, text classification has become one of the branches of knowledge that is widely studied in various communities such as data mining, machine learning, and information retention. The results of this research have been widely applied in various applications, including applications to detect the topic of a document [2], applications to separate spam and e-mail [3], an application to detect SMS in the form of spam [4] and sentiment analysis [5].

As part of Natural Language Processing, there are special stages in classifying texts that also aim to prepare the text for entry into a classification process called pre-processing text. This stage is a process that must be done so that the computer is able to process input in the form of text into a series of program code. Text pre-processing is one of the key components in many text mining algorithms [4]. Text pre-processing usually consists of several processes namely tokenization, filtering, lemmatization and stemming. Tokenisation is the process of breaking a sentence into its constituent words. Filtering is a process to eliminate words that are not important in the classification process. Lemmatization is the process of classifying words based on specific categories, for example word classes, and stemming returns affixed words into their basic form.

The use of proper text pre-processing will improve accuracy in text classifications. As in the previous research [5] there was a 6% increase in results using text pre-processing techniques. However, each stage that must be done both in text preprocessing and feature selection must be done separately, this

causes impracticality in the process of text classification, so we need a system of integrated feature selection and pre-processing text. In this research, it is proposed an application development that integrates feature selection and text pre-processing, to facilitate text classification. With this research, it is expected that there is an application that integrates feature selection and text preparation so that it can provide convenience in research on text classification.

2. Methodology

This purpose of this research to design a web-based application that can facilitate users in pre-processing text. This simplicity is demonstrated in the form of an application that integrates all stages of pre-processing and feature selection, so users do not need to use different applications for each pre-processing stage. The approach proposed in this study is a qualitative approach and the method used in developing its application is a waterfall.

2.1. Data Collection

The data used in this study are in the form of textual data obtained from news portals on the internet such as detiknews.com and kompas.com. The data retrieval process is done by creating a special bot for each news portal and running it to obtain news content that is suitable for the news category. This process is called scraping. The results of this scraping process are stored in a simple database that is linked to the application to be made.

2.2. Research Process

This research process was developed using the waterfall method. The waterfall method is a software development life cycle where the method illustrates a systematic and sequential approach to software development, starting with the specifications of user needs and then continues through the stages of planning, design or modelling, construction or implementation, and the delivery of the system to customers / user (deployment), which ends with support for the complete software that is produced.

2.3. Application's features

The text preprocessing application that will be designed has several features, including adding news corpus, displaying news corpus, case folding, tokenization, and feature selection. Add news corpus function to add text data to be processed. In the initial scenario, the data used for preprocessing text has been taken directly from the web scraper that is integrated directly with the database. However, so that this application can be used to process text other than news from the web scraper, a feature added a news corpus. Thus, this text preprocessing application can be used more broadly and for a variety of textual data.

News display is used to display textual data that is already in the database, which is ready to be processed. The case folding feature is used to change capital letters to lowercase. This is so that the words at the beginning of the sentence are treated the same as the same words at the end of the sentence. The tokenization feature is used to break a paragraph into its constituent sentences and break a sentence into words. The last is feature selection which aims to retrieve words and sentences that are considered important that affect the meaning of the sentence or paragraph. All features to be developed are developed using the NLTK library from python.

2.4. NLTK Python Libraries

NLTK stands for Natural Language Toolkit. This toolkit is one of the most powerful NLP libraries which contains packages to make machines understand human language and reply to it with an appropriate response. NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum.

Natural Language Processing with Python provides a practical introduction to programming for language processing. Written by the creators of NLTK, it guides the reader through the fundamentals of writing Python programs, working with corpora, categorizing text, analysing linguistic structure, and more.

3. Result and Discussions

3.1. Requirements Specification

The results of the requirements specification stage are a list of functional and non-functional requirements as well as an overview of the system to be made. Some functional requirements that have been formulated are, the function of connection to the database, the function of adding data to the database, the function of displaying data from the database, the case folding function, the tokenization function, and the feature selection function. The stop word removal feature is not used for the feature selection function, but the TF-IDF calculation. The use of TF-IDF is used so that the feature selection process is not only based on the words contained in the stop word list, but really based on the position of the word in the sentence. Non-functional requirements describe the ideal conditions in which the system can work well. The ideal conditions include computer specifications, software, and network availability in the work environment.

General description of the system is a diagram that illustrates interwoven subsystems that are interrelated in order to produce the expected output. An overview of the system for text processing applications, shown in Figure 1, consists of seven subsystems. The first subsystem is the data extractor, which functions to retrieve data from the internet. The data is then stored in the second subsystem, the database. The third subsystem is data input. This subsystem is different from the data extractor, because the input data is used to enter data from the user through the web interface. The data entered is also stored in the same database that was taken from the internet. The fourth subsystem is the data viewer, which functions to display data from the database. The fifth subsystem is a case folder that functions to change all the letters in the news corpus to lowercase. The sixth is the tokenizer, which functions to break paragraphs into its constituent sentences and breaks each sentence into its constituent words, and the seventh is the feature selector, which functions to delete words that are considered less important in the sentence. Most of these subsystems are built using the Python programming language, using the NLTK library.

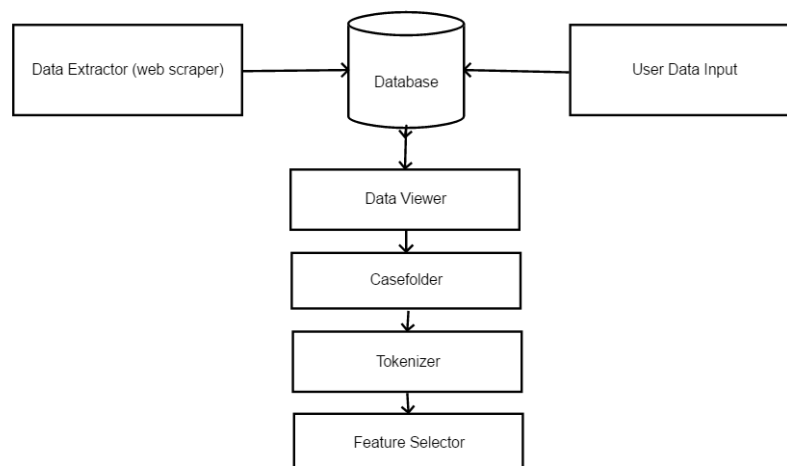


Figure 1. System Block Diagrams

3.2. Designing Application

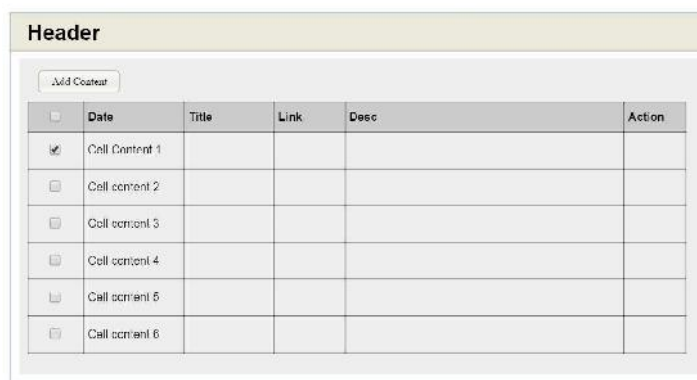
At the application design stage, every basic system requirement that has been previously formulated is outlined in the form of diagrams and layouts. This is done to make it easier for researchers to understand the workflow of the system and provide a real picture when the application is completed.

3.2.1 *Database design.* The database used consists of one table, which contains date data, titles, hyperlinks and content descriptions from the news. This is caused by the need for storage that is used does not require a relationship, so it is enough to use just one table. Following is the database design used by the system shown in Figure 2.

<input type="checkbox"/>	dt_berita
<input checked="" type="checkbox"/>	date
<input type="checkbox"/>	judul
<input type="checkbox"/>	link
<input type="checkbox"/>	deskripsi

Figure 2. Database Design

3.2.2 *Interface design.* Interface design is used to ensure the availability of all components needed by the user to run the application. Interface design here includes web page layouts, headers, buttons, information windows and footers. The layout design for the initial page of the text preprocessing application is shown in Figure 3.



<input type="checkbox"/>	Date	Title	Link	Desc	Action
<input checked="" type="checkbox"/>	Cell content 1				
<input type="checkbox"/>	Cell content 2				
<input type="checkbox"/>	Cell content 3				
<input type="checkbox"/>	Cell content 4				
<input type="checkbox"/>	Cell content 5				
<input type="checkbox"/>	Cell content 6				

Figure 3. Interface design

3.3. Application development

The next stage in this research is to change the functions and features described in functional requirements into applications that can be used by the user. The development of features and functions is made using the Python programming language. Some of the results of developing this application are shown in the following figure.

```

87
88 def index(request):
89     #print(math.log(3))
90     # queryset
91     posts = Post.objects.all()
92     context = {
93         'Title': 'Proses Preprocessing',
94         'Heading': 'Preprocessing',
95         'Posts': posts,
96     }
97     return render(request, 'preprocessing/index.html', context)
98

```

Figure 4. Data viewer implementations

Figure 4 shows the results of implementing a data calling script that is already stored in the database. Data that has been called is then displayed in a table on the index page.

```
def hasil_casefolding(data):
    data = text_lower(data)
    data = hapus_punct(data)
    data = hapus_tanda_harga(data)
    data = hapus_spasi(data)
    return data

def casefolding(request):
    lower_case = hasil_casefolding(data)
    context = {}
    title = 'Proses Preprocessing'
    heading = 'Preprocessing'
    data = lower_case
    return render(request, 'preprocessing/tampil_casefolding.html', context)
```

Figure 5. Case folder implementations

Figure 5 shows the results of the implementation of the case folder feature. This feature changes the entire text in sentences and paragraphs to lowercase, and clears paragraphs of punctuation that does not have any function in the sentence.

3.4. Application deployment

The first step to run a pre-processing text application is to select a document that is already stored in the database. If the data sought is not found, it can be added to the database using the add data button. After that, the selected data will be displayed by Data Viewer.

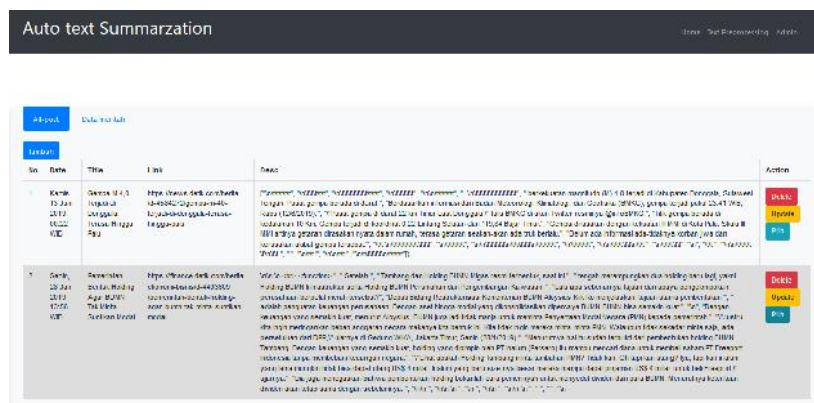


Figure 6. Front page applications

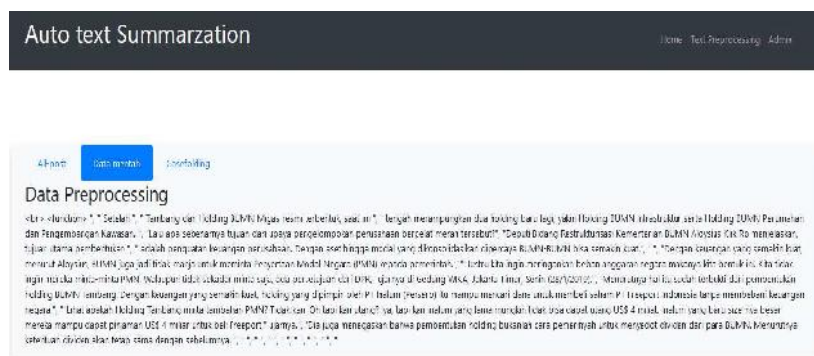


Figure 7. Viewing selected data

The next process is case folding. At this stage, in addition to changing to the lowercase form, also carried out the stage of cleansing the text of punctuation that is not necessary or not related to news content. This punctuation is usually the result of a web scraper, which still leaves the html code in the search results.



Figure 8. Case folding results

Furthermore, data that is already clean and already has lowercase letters goes directly to the tokenization process. The paragraph is broken down into sentences and then given an index to determine the order of the sentence. The results of the tokenization process are shown in Figure 9.

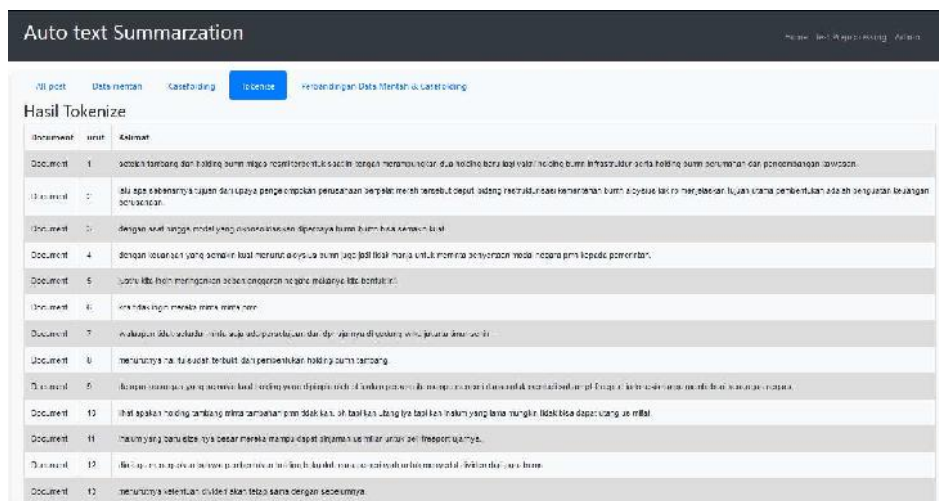
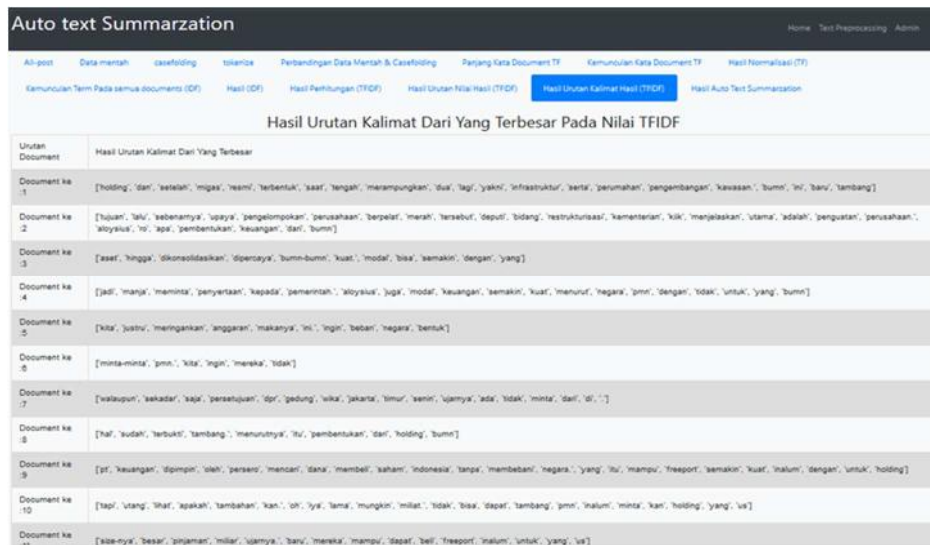


Figure 9. Tokenization results

At this stage, documents have become an easy form to be processed for various NLP needs. Starting from the classification of the text, determining the topic of a paragraph, information retrieval and various other forms of NLP. However, to improve the accuracy of text classification, one more step is needed, namely feature selection. The feature selection process used in this application uses the TFIDF calculation approach. Where every word in the sentence is weighted based on the appearance of the word in each sentence and each document.

Every word that has a known weight will be sorted by highest value to lowest value. The word with the highest weight will be considered as an important word, so it will be stored, while the word with the lowest weight will be discarded because it is considered an insignificant word. Words that appear in many sentences in one paragraph will be given a small weight, while words that appear several times in a sentence, but not all paragraphs will be given a large weight and are considered important words. The results of the feature selection process are shown in Figure 10.



The screenshot shows a web application titled "Auto text Summarization". The main heading is "Hasil Urutan Kalimat Dari Yang Terbesar Pada Nilai TFIDF". Below this, there is a table with 11 rows, each representing a document and its selected words based on TFIDF values.

Urutan Document	Hasil Urutan Kalimat Dari Yang Terbesar
Document ke 1	[holding, dari, setelah, migas, resmi, terbentuk, saaf, tengah, merampungkan, dua, lagi, yakni, infrastruktur, serta, perumahan, pengembangan, kawasan, bumi, hi, baru, tambang]
Document ke 2	[tolak, itu, bebannya, upaya, pengumpulan, perusahaan, berpelai, merah, tersebut, deputi, bidang, restrukturisasi, Kementerian, Niki, melakukan, utama, adalah, pengutan, perusahaan, sloyaku, hi, apa, pembentukan, keuangan, dari, bumi]
Document ke 3	[saaf, hingga, dikonsolidasikan, dipercaya, bumi-bumi, kuat, modal, bisa, semakin, dengan, yang]
Document ke 4	[jadi, mang, meminta, panyertaan, kepada, pemerintah, sloyaku, juga, modal, keuangan, semakin, kuat, menurut, negara, gmi, dengan, tidak, untuk, yang, bumi]
Document ke 5	[kita, justru, menginginkan, anggaran, makanya, hi, ingin, bebar, negara, bentuk]
Document ke 6	[minta-minta, gmi, kita, ingin, mereka, tidak]
Document ke 7	[walaupun, sakadar, apa, persetujuan, dir, gedung, kita, jakarta, timur, senin, ujanya, ada, tidak, minta, dari, di, ...]
Document ke 8	[haf, sudah, terbukti, tambang, menurutnya, itu, pembentukan, dari, holding, bumi]
Document ke 9	[pr, keuangan, dipinai, oleh, penera, mencari, dana, membeli, saham, indonesia, tanpa, membebani, negara, yang, itu, mampu, freport, semakin, kuat, dalam, dengan, untuk, holding]
Document ke 10	[tagi, yang, biar, apakah, tambahan, kan, hi, itu, lama, mungkin, tidak, bisa, dapat, tambang, gmi, dalam, minta, kan, holding, yang, hi]
Document ke 11	[saya-sya, besar, pinjaman, milar, ujanya, baru, mereka, mampu, dapat, beli, freport, dalam, untuk, yang, hi]

Figure 10. Feature selection results

4. Conclusion

The design of the text pre-processing application can be made using the waterfall method, because this application is a form of software, and the waterfall method is one of the methods of software development. For the application stage, the utilization of the NLTK Library can be applied precisely and systematically. This library also facilitates the implementation phase because of the large number of NLP classes that can be directly applied.

This application also makes it easy for users who want to pre-processing text, because each pre-process stage is integrated into one in an application. In addition, because each process is carried out in stages, the user can also observe the results of each stage and then analyse it. For future research, this application can be embedded with a feature to display the results of pre-processing text into other formats such as JSON to make it more flexible.

5. References

- [1] M. Allahyari, S. Pouriyeh, S. Assefi, S. Safaei, E. Trippe, J. Gutierrez and K. Kochut, "A Brief Survey of Text Mining: Classification, Clustering and Extraction Techniques," *arXiv preprint*, 2017.
- [2] M. Ghiassi, M. Olschimke, B. Moon and P. Arnaudo, "Automated text classification using a dynamic artificial neural network model," *Expert Systems with Applications*, pp. 10967-10976, 2012.
- [3] S. Gunal, S. Ergin, M. Gulmezoglu and O. N. Gerek, "On feature extraction for spam e-mail detection," *Lecture Notes in Computer Science*, p. 635-642, 2006.
- [4] A. K. Uysal and S. Gunal, "The Impact of Preprocessing on Text Classification," *Information Processing and Management*, pp. 104-112, 2013.
- [5] H. A. Putranto, O. Setyawati and Wijono, "Pengaruh Phrase Detection dengan POS-Tagger terhadap Akurasi Klasifikasi Sentimen menggunakan SVM," *Jurnal Nasional Teknik Elektro dan Teknologi Informasi*, vol. 5, no. 4, pp. 252-259, 2016.

Acknowledgements

This research was fully supported by PNBPN Funding from Politeknik Negeri Jember. We thank our colleagues from Information Technology Politeknik Negeri Jember who provided insight and expertise that greatly assisted the research.