“COVID-19 Vaccine Types Sentiment Analysis with Data Augmentation and CNN-LSTM”

by

Mathias Yeremia Aryadi (吴智恩)

A Thesis

Submitted to the Faculty of Nanjing Xiaozhuang University

in Partial Fulfillment of the Requirements for the degree of

Bachelor of Software Engineering

School of Information Engineering

Fangshan, Nanjing

May 2022

To:

Dean Xiangjun Zhao

School of Information Engineering

This thesis, written by Mathias Yeremia Aryadi (吴智恩), and entitled “COVID-19 Vaccine Type Sentiment Analysis with Data Augmentation and CNN-LSTM”, having been approved in respect to style and intellectual content, is referred to you for judgment.

I have read this thesis and recommend that it be approved.

Committee A

Committee B

Committee C

Date Defense: May 24, 2022

The thesis of Mathias Yeremia Aryadi (吴智恩) is approved

Dean Xiangjun Zhao

School of Information Engineering

Xxx

Chief Foreign Affairs Office

Nanjing Xiaozhuang University, 2022

DECLARATION OF ORGINALITY

I, the undersigned below:

Student Name	:	MATHIAS YEREMIA ARYADI (吴智恩)
Student ID	:	L20253011 / 2017103126
Study Program	:	SOFTWARE ENGINEERING
Degree	:	BACHELOR'S DEGREE
Faculty	:	SCHOOL OF INFORMATION TECHNOLOGY

Hereby declared that the thesis I wrote with the title:

"**COVID-19 Vaccine Type Sentiment Analysis with Data Augmentation and CNN-LSTM."**

1. It is truly research written and conducted purely by myself, not copying from other published researchers, and not a result of plagiarism.
2. I will allow Nanjing Xiaozhuang University and Kalbis Insitute to manage and keep the copy of this thesis to be used as they deem necessary.

I made this statement of a declaration with full responsibility, and I am willing to accept any consequences according to the rules and regulations should the statement above proved to be wrong in any way.

Jakarta, 22 May 2022

Mathias Yeremia Aryadi

L20253011 / 2017103126

COVID-19 VACCINE TYPE SENTIMENT ANALYSIS

WITH DATA AUGMENTATION AND CNN-LSTM

ABSTRACT

Abstract: *The objectives of this research are to implement the EDA (Easy Data Augmentation) as the data augmentation on the Twitter data for the sentiment analysis models; build and compare the performance difference between the single CNN model, the single LSTM model, and the hybrid CNN-LSTM model with data augmentation on the four COVID-19 vaccine type dataset topics; and build sentiment analysis web application the four COVID-19 vaccine type dataset topics. The incremental model is used to build the COVID-19 vaccine types sentiment analysis that consisted of two incremental processes. The incremental 1 focuses on building and comparing the sentiment analysis models through data crawling, data merging, data filtering, data pre-processing, data labeling, data splitting, data augmentation, and sentiment analysis models building. The incremental 2 focuses on building the sentiment analysis web application using flask framework. The results of the incremental 1 are the single CNN model has worked best on the Sinovac dataset topic with average of 78% on recall, 78% on precision, and 78% on f1-score; and on the Moderna dataset topic with the average of 76% on recall, 77% on precision, and 76% on f1-score. On the other hand, the hybrid CNN-LSTM model has worked best on the AstraZeneca dataset with the average of 82% on recall, 81% on precision, and 81% on f1-score; and on the Pfizer dataset with the average of 71% on recall, 71% on precision, and 71% on f1-score. The result of the incremental 2 is the sentiment analysis web application can perform sentiment analysis through sentiment classification and visualization about each four COVID-19 vaccine type topics.*

Keywords: *Twitter Data, Sentiment Analysis, Data Augmentation, CNN, LSTM, Hybrid CNN-LSTM*

ACKNOWLEDMENT

Praise and gratitude from the researcher give to the Almighty God because of His blessing and grace, the researcher can complete a thesis entitled *"**COVID-19 Vaccine Type Sentiment Analysis with Data Augmentation and CNN-LSTM**"* well and on time. The purpose of the preparation and writing for this thesis is to fulfil one of the requirements to obtain a bachelor's degree in Software Engineering specialization study at Nanjing Xiaozhuang University and bachelor's degree in Computer Science at Kalbis Institute.

With all limitations during preparation and finishing of this thesis, the researcher has received many guidance, supports, loves, and assistance from several persons. Therefore, the researcher would like to express their gratitude and appreciation to:

1. The researcher's family who has always provided loves, supports, encouragements, and prayers so the researcher can finish this thesis very well and on time.
2. The researcher's thesis supervisors both from Nanjing Xiaozhuang University and Kalbis Institute, Mr. Andy Song and Mr. Yulius Denny Prabowo S.T., M.T.I., who has provided a good direction, guidance, and inputs during the process of writing this thesis until completed.
3. Special thanks and gratitude to Mr. Yulius Denny Prabowo S.T., M.T.I. as head of Computer Science Major, Faculty of Computer Science and Design, Kalbis Institute, who has provided the experiment resources such as GPU server for the researcher so the experiment can run smoothly.
4. Mr. Li Qing as head of School of Information Engineering for international student in Nanjing Xiaozhuang University, who has provided a good guidance and knowledge to the researcher.
5. Ms. Sophie Mou and Ms. Wang Jing who has taken care my study as international student in Nanjing Xiaozhuang University for two years.

6. All lecturers both from Nanjing Xiaozhuang University and Kalbis Institute who have shared their knowledges to the researcher during the study.
7. Classmates and colleagues from IT'2017 Kalbis Institute and IT'2018 Nanjing Xiaozhuang University who have supported, encouraged, accompanied, and struggled together to complete the researcher's thesis and bachelor study.

The researcher realizes that this thesis still has limitations, therefore the researcher is open to suggestions and constructive criticism for more improvements. Finally, the researcher hope that this thesis can provide benefits and knowledge to all reader.

Jakarta, 22 May 2022

Mathias Yeremia Aryadi

L20253011 / 2017103126

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1.1 Background

The first case of COVID-19 was reported on Wuhan, People's Republic of China at the end of 2019 and then the disease has been found widely almost around the world, becoming a global wide pandemic. Fortunately, the first COVID-19 vaccine was created and distributed around the world at the end of 2020 after the struggle against the COVID-19 pandemic. The several nations have started to use the COVID-19 vaccine and encourage people to take the shoot. The COVID-19 vaccine topics become a hot discussion around the world, especially in digital platform such as social network; considering there are different types of COVID-19 vaccine: Sinovac, AstraZeneca, Pfizer, and Moderna.

The explosion information in digital platforms, there are plenty of people sharing their opinions or thoughts about COVID-19 vaccine types on social network from 2021 until now. People have been shared opinions about: their decision to take a shoot of certain COVID-19 vaccine type, because some of them resulting some side effects; their feelings after taking a shoot one of the COVID-19 vaccine type; their doubts about COVID-19 vaccine type; and information from a news of particular COVID-19 vaccine type. One of the most used and most popular social networks (media) is Twitter. Based on Statista Research Department statistics, from the top 20 most popular social networks as of October 2021 by number of active users, Twitter is on the rank 16 with 463 million of active users [1]. Twitter was launched in 2006 by Jack Dorsey. In 2021, Twitter already has 2 million active users [2]. This implies everyday Twitter produces huge information that contain people opinions or thoughts and feelings to express their emotions about something. The opinions and feelings may express positive sentiment, negative sentiment, or neutral sentiment.

Sentiment analysis, also known as opinion mining, is the field of study that analyses: people's opinions, sentiments, evaluations, appraisals, attitudes, and emotions towards entities such as products, services, organizations, issues, events,

topics, and their attributes. Sentiment analysis has three level of analysis: document level; sentence level; and entity and aspect level. Document level analyses and classifies the whole opinion in the document whether is positive or negative; sentence level analyses and classifies each opinion through sentence determines whether the opinion sentence is positive, negative, or neutral; and entity and aspect level overcome the limitation of document level and sentence level where analyses and classifies the sentiments on entities or/and their aspects. Sentiment analysis and Twitter datasets has been used along over the time, because Twitter provides an Application Programming Interface (API) access for us to retrieve all the data based on our query. Sentiment analysis allow us to extract the expressed emotion from people opinions or thoughts such as positive, negative, and neutral [3]. Sentiment analysis allow us to summarize the sentiments of opinions from a huge information stream on the social network. Sentiment analysis has been already used together with artificial intelligence such as machine learning and deep learning.

There are several works on sentiment analysis with machine learning, mostly they have used Naïve Bayes. Sentiment analysis on Twitter about anti-LGBT campaign in Indonesia [4], YouTube movie trailer comments [5], and Twitter about COVID-19 vaccine in Indonesia [6]. These works proofed that Naïve Bayes as a machine learning model resulted an accuracy from 81% to 83% and performed better in sentiment analysis compared to another model [4].

However, the machine learning model has several limitations [7]: require a large amount of annotated data to increase the performance, domain-dependent (need to retrain the classifier model for different domain), and set-selection feature limitation (cannot capture phenomena like negation detection or representative feature). Therefore, deep learning model has the capability to learn bigger dataset, capture, and address the representative feature of the data. Several works that related to them are sentiment analysis using both CNN and LSTM on IMDB dataset [8], Word2Vec and LSTM on Indonesian hotel reviews [9], and LSTM on IMDB and Amazon dataset [10]. The deep learning implementations on sentiment analysis

resulted an accuracy from 85% to 88% that is higher than Naïve Bayes as machine learning model.

Additionally, it is also possible to combine multiple deep learning model, for allowing the model to capture the feature data more profound instead of a single deep learning model. Sentiment analysis using Word2Vec-CNN-LSTM on movie reviews [11], multiple word embedding CNN-LSTM on Arabic sentiment analysis [12], Word2Vec-CNN-BiLSTM on Quora question reviews [13], and CNN-LSTM on airline quality reviews. The hybrid model on sentiment analysis resulted a high accuracy from 90% to 91% higher than single learning model.

I also found data augmentation and sentiment analysis can be implemented together based on several works. A document level multi-topic sentiment analysis on email data with Bi-LSTM and random word replacement data augmentation [15]; and sentiment analysis on Vietnamese language with word replacements, word deletion, word swapping, and word insertion data augmentation using classifier models [16].

From the several works mentioned above, the hybrid CNN-LSTM model and the data augmentation can be implemented on the sentiment analysis problem. However, the data augmentation has not been implemented together for the sentiment analysis models such as the single CNN model, the single LSTM model, and the hybrid CNN-LSTM model. Further, sentiment analysis on each COVID-19 vaccine type also have not been implemented yet. Therefore, I interested to build a sentiment analysis on each COVID-19 vaccine types by using data augmentation and CNN-LSTM, to extract what people's sentiment about it.

I built the sentiment analysis models that consisted of the single CNN model, the single LSTM, and the hybrid CNN-LSTM. In contrast, I implemented the data augmentation together with the sentiment analysis models to increase the dataset size. I used tweets about COVID-19 vaccine types that are related to Sinovac, AstraZeneca, Pfizer, and Moderna for the experiment and implementation dataset.

I also did a comparative study the performance between the single CNN model, single LSTM, and the hybrid CNN-LSTM model by using exact same parameters and using the augmented training dataset. The result of this work provides sentiment analysis model performance comparisons between the single CNN model, the single LSTM model, and the hybrid CNN-LSTM model with data augmentation; and sentiment analysis web application for each COVID-19 vaccine type such as Sinovac, AstraZeneca, Pfizer, and Moderna.

1.2 Formulation of Problems

From the background above, I identified several problems that occurred in this research, as follows:

1. How to implement the data augmentation on the sentiment analysis Twitter data?
2. How is the performance difference between the single CNN model, the single LSTM, and the hybrid CNN-LSTM model on the four COVID-19 vaccine type topics with data augmentation for sentiment analysis?
3. How to build a sentiment analysis web application about COVID-19 vaccine types?

1.3 Limitation of Problems

This research has several limitations, described as follows:

1. This research built a sentiment analysis on sentence level
2. This research built a sentiment analysis for positive sentiment and negative sentiment
3. This research used EDA (Easy Data Augmentation) as the data augmentation
4. This research used supervised deep learning model such as single CNN model, single LSTM model, and hybrid CNN-LSTM model for sentiment analysis as binary classification problem
5. This research used four tweet topics about COVID-19 vaccine type such as Sinovac, AstraZeneca. Pfizer, and Moderna

6. This research used four tweet topics about the COVID-19 vaccine type from 1 March 2022 until 21 March 2022 for the experiment dataset
7. This research used four tweet topics about the COVID-19 vaccine type from 22 March 2022 until 31 March 2022 for the implementation dataset
8. This research used split 90% training data and 10% testing data on the four experiment dataset topics
9. This research used an automated tweet sentiments labeling with lexicon based for unlabeled crawled Twitter data
10. This research built the user interface for the sentiment analysis in web application
11. This research used incremental model to define the process of building the sentiment analysis models and the sentiment analysis in web application
12. The incremental 1 produced performance comparisons between the single CNN model, single LSTM mode, and the hybrid CNN-LSTM model with same model parameters on the four dataset experiment topics sentiment analysis
13. The incremental 2 produced sentiment analysis web application for classifying the implementation data using the trained sentiment analysis models and summarizing the sentiment analysis results

1.4 Objectives

Based on the defined problems and its limitations, the objectives of this research are defined as follows:

1. Implement data augmentation using the EDA (Easy Data Augmentation) on the Twitter data for the sentiment analysis models.
2. Build sentiment analysis models such as the single CNN model, the single LSTM model, and the hybrid CNN-LSTM model with data augmentation; and compare the results for each COVID-19 vaccine type topic that are Sinovac dataset topic, AstraZeneca dataset topic, Pfizer dataset topic, and Moderna dataset topic.

3. Build sentiment analysis web application for the COVID-19 vaccine type topics that are Sinovac dataset topic, AstraZeneca dataset topic, Pfizer dataset topic, and Moderna dataset topic.

1.5 Benefits of Research

The benefits of this research are expected to be useful for both theoretically and practically:

1.5.1 Theoretically

1. Science development contribution in artificial intelligence especially in natural language processing and software engineering field
2. Know and discover about how to build COVID-19 vaccine types sentiment analysis with data augmentation and CNN-LSTM
3. Know and discover the performance differences between the single CNN model, the single LSTM model, and the hybrid CNN-LSTM model on the four dataset topics about COVID-19 vaccine types
4. Know and discover about how to build sentiment analysis web application

1.5.2 Practically

For Researcher:

1. Improving practical skills in artificial intelligence, especially on NLP (Natural Language Processing) through building the single CNN model, the single LSTM model, and the hybrid CNN-LSTM model with data augmentation for sentiment analysis
2. Improving practical skills in software engineering through building the sentiment analysis web application

For University:

Research material contribution for future work or research.

For Public:

Provide the sentiment analysis of each COVID-19 type summary for Sinovac, AstraZeneca, Pfizer, and Moderna in web application.

1.6 Schedule

The schedule of this research is estimated in 9 weeks effectively and defined as follows:

Table 1.1 Thesis Schedule of 9 weeks

Activities	Weeks								
	1	2	3	4	5	6	7	8	9
Introduction writing	■								
Literature reviews writing		■							
Research methodology writing			■	■					
Implementation writing					■	■	■		
System Development					■	■	■		
Summary writing								■	■

1.7 Organizational Structure of the Paper

This research is consisted of structural systematic paper writing, divided into five chapters as follows:

CHAPTER 1: INTRODUCTION

This chapter describes about the background of the research, the formulation of problems, the limitation of problems, the benefits of research, the schedule, and the organizational structure of the paper.

CHAPTER 2: THEORITICAL BACKGROUND

This chapter describes about the related works and the related theories such as Twitter data, sentiment analysis, data augmentation, CNN (Convolutional Neural Network), LSTM (Long Short Term-Memory), incremental model, confusion matrix, black box testing. Python, and Flask.

CHAPTER 3: RESEARCH METHODOLOGY

This chapter describes about the research methods that used in this work such as conceptual design, research flow and design, incremental 1, and incremental 2. The incremental 1 focuses on building and comparing the sentiment analysis models through requirement analysis, design, implementation, and testing. The incremental 2 focuses on building the sentiment analysis web application through requirement analysis, design, implementation, and testing.

CHAPTER 4: RESULT AND DISCUSSION

This chapter elaborates the results from the incremental 1 and the incremental 2. The incremental 1 elaborates the dataset result, the sentiment analysis model training results, and the sentiment analysis model testing and evaluation results. The incremental 2 elaborates the sentiment analysis web application user interface results and the sentiment analysis web application black box testing result.

CHAPTER 5: SUMMARY

This chapter summarizes the summary of this research and the suggestion for further research to improve the current state of this research.

CHAPTER 2
THEORITICAL BACKGROUND

2.1 Twitter Data

The Twitter data contains tweet, user or username, mention, retweet, and hashtag. The tweet is a single message posted on the Twitter and it contains maximum 140 characters from personal information or opinion on products or events to others such as links, news, photos, or videos. The user or username is a registered user on the platform and gained access to post tweets. The mention is a message in tweet to mention another user on a post and it uses "@" symbol, followed by the specific username they refer to; for example, "@username". The retweet is the tweet that are shared and it uses its own abbreviation "RT", followed by the author's username; for example, "RT @username". The hashtag labels the relevance of a tweet to a certain topic and it uses "#" symbol, followed by the topic name as the hashtag term on a post; for example, "#topic"; then the users use the hashtags to search and get all the tweets with search tag [14].

2.2 Twitter API

Twitter API has provided developers to have an access to their data programmatically. Twitter API has three developer accesses such as: essential, elevated, and academic research [15].

Table 2.1 Twitter API Product Access

Essential	***Elevated***	***Academic Research***
A free access	*A free access with additional endpoints access and App environment*	*A free access with more endpoints access and App environment*
Retrieve 500,000 tweets per month	*Retrieve 2,000,000 tweets per month*	*Retrieve 10,000,000 tweets per month*
1 project per account	*1 project per account*	*Access to full-archive search and full-archive Tweet counts*
1 App environment per project	*3 App environments per project*	*Access to advanced search operators*

No access to standard v1.1, premium v1.1, or enterprise	*Access to standard v1.1, premium v1.1, or enterprise*

In order to access and retrieve the Twitter data, the Twitter API has provided a search query. The search query matches the Twitter API endpoints with GET request and return a set of historical tweets. This research used the elevated access that only carries 512 characters on the search query. The Table 2.2 defines several Twitter API search query operators.

Table 2.2 Twitter API Search Query Operators

Query	***Description***
Keyword term	*Every keyword on the search query will be tokenized.* *For example: "I like coca cola", will be split into following tokens: "I", "like", "coca", "cola"* *The keyword term is evaluated in a case-insensitive manner.* *For example: a search query term cat will match tweets that contain terms cat, Cat, or Cat.*
Hashtag	*A conditional search query operator to match any tweet that containing a recognized hashtag.* *For example: a search query #newyear will match any tweet that contain #newyear but not the #happynewyear*
AND logic	*A conditional search query operator to match if both conditions are true.* *For example: a search query @andy AND #happynewyear will match any tweet that contain username andy and happynewyear hashtag*

OR logic	*A conditional search query operator to match any tweet if one of the conditions are true.* *For example: a search query @andy OR #happynewyear will match any tweet that at least contain username andy or happynewyear hashtag*
NOT logic	*An operator to negate the search query by using "-" (hyphen) symbol.* *For example: a search query happy -#birthday will match any tweet that contain term happy but without the birthday hashtag*
Grouping	*A search query operator to group operators together with "()" parentheses symbol.* *For example: a search query (happy day) OR (new stuff) will match any tweet that contain either term happy and day or either term new and stuff*

2.3 Sentiment Analysis

Sentiment analysis or usually called as opinion mining, is the field of study of NLP (Natural Language processing) that analyzes and extracts people's opinion sentiments, evaluations, appraisals, attitudes, and emotions towards entities such as products, services, organizations, individuals, issues, events, topics, and their attributes [3]. The Figure 2.1 shows the sentiment analysis has three level applications that are document level, sentence level, and aspect level [7].

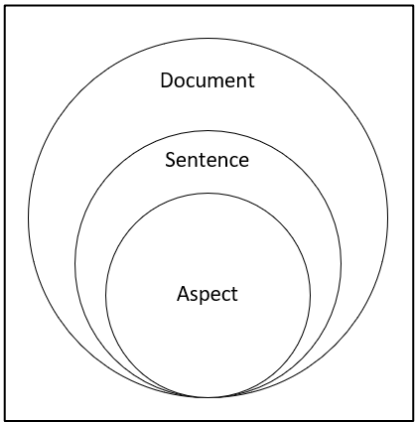


Figure 2.1 The Sentiment Analysis Level

2.3.1 Document Level Sentiment Analysis

The document level sentiment analysis summarizes and classifies whether a whole document expresses negative or positive sentiment. The process of the document level sentiment analysis sums up the scores of each sentence based on weighting rules [7].

Table 2.3 The Document Level Sentiment Analysis Example

Document	***Sentiment***
The movie has great story and cinematic. But some of the casts who are not suitable in playing the characters. Fortunately, the CGI and the lighting of the movie is fantastic.	*Positive*
The movie has bad and confusing story. The cinematic element is very lacking. Only the casts that has good role playing most of the character in the movie. Unfortunately, the CGI and the lighting of the movie also bad.	*Negative*

2.3.2 Sentence Level Sentiment Analysis

The sentence level sentiment analysis summarizes and classifies whether a sentence expresses negative or positive sentiment. The difference between the document level and the sentence level is the document level focuses on the whole document, whereas the sentence level focuses on each sentence [7].

Table 2.4 The Sentence Level Sentiment Analysis Example

Sentence	***Sentiment***
The movie has good story and cinematic.	*Positive*
Fortunately, the CGI and the lighting of the movie is fantastic.	*Negative*
Unfortunately, the CGI and the lighting of the movie also bad.	*Negative*
The cinematic element is very lacking.	*Negative*
The movie has bad and confusing story.	*Negative*

2.3.3 Aspect Level Sentiment Analysis

In contrast of document level and sentence level sentiment analysis, the aspect level sentiment analysis has more precise sentiments or opinions of each entity that existed in the topic [7].

Table 2.5 The Aspect Level Sentiment Analysis Example

Topic	***Aspect***	***Sentiment***
The movie has great story and cinematic. But some of the casts who are not suitable in playing the characters. Fortunately, the CGI and the lighting of the movie is fantastic.	*Story*	*Positive*
	Cinematic	*Negative*
	Cast	*Negative*
	CGI	*Negative*
	Lighting	*Negative*

2.4 Data Augmentation

There are several regularization techniques such as dropout, batch normalization, transfer learning, pre-training, one-shot, and zero-shot learning. These regularization techniques mostly implemented on deep learning model with small dataset [16]. In contrast, data augmentation is also one of the regularization techniques to reduce overfitting when it comes deep learning training by fixing the dataset problem (i.e., small training sample). Data augmentation is a set of algorithms that build synthetic data from an original dataset. Data augmentation tweaks or changes the original dataset with different variant and generated them as additional dataset without changing the original one. Data augmentation can help deep learning model to avoid learning spurious data correlations and memorizing high-frequency patterns. Data augmentation has been implemented on image tasks

with deep learning model. In NLP (Natural Language Processing) tasks, data augmentation shuffles the form of language. There are two data augmentation types on NLP, as follows [17]:

2.4.1 Symbolic Data Augmentation

Symbolic data augmentation uses rule-based, graph-structured decomposition, Mix-up Augmentation, and feature space augmentation to generate new examples of dataset. The rule-based consists of: Easy Data Augmentation (i.e., random swap, random insertion, random deletion, random synonym replacement), regular expression augmentation, and syntactic heuristic (e.g., inverse the subject and the object of the sentence, change active form to passive form and vice versa). The graph-structured decomposition consists of: knowledge graph, WordNet, and syntactic parsing. Mix-up generate new example by mashing up the existing examples by taking a half of one text sequence and concatenate with another sequence. Feature space augmentation manipulates the features at deep neural network space by isolating them and applying noise to generate new examples.

2.4.2 Neural Data Augmentation

Neural data augmentation uses auxiliary neural network to generate new training examples. The neural data augmentation consists of several methods such as back-translation augmentation, style augmentation, and generative augmentation. The back-translation augmentation uses machine translation model to translates text from one language to another and then back from the translation to the original language, that this may lead to the difference structure of the back-translated text. The style augmentation changes the author writing style on the examples by extracting the semantic similarities between them. The generative augmentation uses transfer learning and pre-trained model to generate new training examples such as C-BERT (Conditional-BERT), GPT-3, RAG (Retrieval Augmented Generation), and REALM (Retrieval Augmented Language Model).

2.5 CNN (Convolutional Neural Network)

CNN is one of the deep learning algorithms that learn small regions of the scene rather than the whole scene (i.e., extract the local correlated feature that

available in the input). This implies, the CNN can simplify and speed up the training process of the network. The CNN has three key benefits makes difference from FC (Fully Connected Network): equivalent representations, sparse interactions, and parameter sharing. The Figure 2.2 shows the CNN model architecture several layers that are input layer, convolutional layer (followed by the activation layer), pooling layer, and FC (Fully Connected) layer [18].

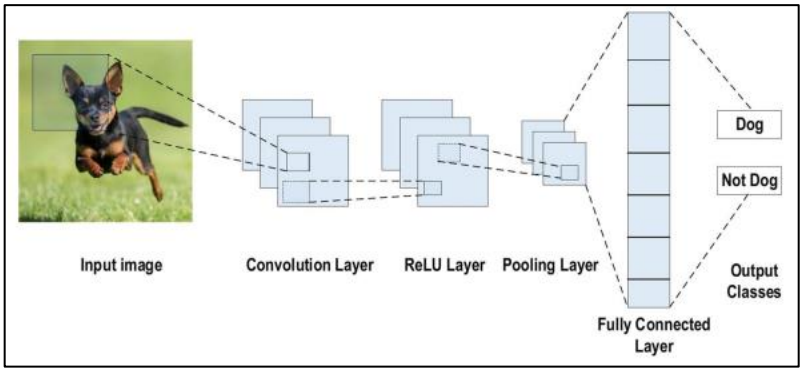


Figure 2.2 The CNN Model Architecture Example [18]

2.5.1 Input Layer

Mainly, the input layer of the CNN organizes the input data x with three dimensions: heigh, width, and depth, they are denoted as $w \times h \times r$, where the width (w) is equal to the height (h) and the r is the depth of the data such as in RGB (Red Green Blue) image has three depths (r) [18].

2.5.2 Convolutional Layer

The convolutional layer is the core of the CNN architecture as its name. The convolutional layer uses convolution filter to convolve through the input data. The convolution filter contains weights that initialized randomly before the training process. The output of this layer is a feature map. The convolutional layer has several hyperparameters such as [18]:

2.5.2.1 Convolution Filter

The filter defines the convolution filter with size of the filter as the depth and size of the kernel as the two dimensional. The convolution filter size is denoted as $k \times k \times f$, where the kernel width (k) is equal to the kernel height (k), however the k value is smaller than the width (w) and the height (h) of the input data, while

the filter size of f is either equal or smaller than the depth (r). The kernel size and the filter size determine the total of the convolution filter weight that will be initialized, for example, the convolution filter size of 3x3x1 means there are 3x3 kernel size, 1 filter size, and total 9 weight values on the convolutional filter. Finally, the output feature map is calculated with one of the activation functions such as sigmoid, tanh, ReLU (Reactified Linear Unit), Leaky ReLU, noisy ReLU, and parametric linear units.

2.5.2.2 Stride

The stride defines how many the vertical or the horizontal steps needed on the convolution process, denoted as s. The bigger stride value (s), the size of the output feature map will have lower dimension.

2.5.2.3 Padding

The padding defines the border size of the input data. The padding size is denoted as $p \times p$, where the padding width (p) is equal to the padding height (p). The bigger padding size (p), the size of the input data and the output feature map will increase.

2.5.3 Pooling Layer

The pooling layer is the next step after the input data is convolved with the filter. The pooling layer performs a sub-sampling by shrinking the output feature data size from the convolutional layer to create a smaller feature map. Several pooling methods such as: tree pooling, gated pooling, average pooling, min pooling, max pooling, GAP (Global Average Pooling), and GMP (Global Max Pooling). The pooling size is denoted as $P \times P$, where the pooling width (P) is equal to the pooling height (P). The bigger pooling size, the sensitivity of capturing certain feature will higher.

2.5.4 Fully Connected (FC) Layer

The FC layer contains neurons that connected each other from the previous and the next layer (i.e., deep neural networks). The convolution and the pooling layer are used as the feature extraction and the FC layer is used as the classifier or

the learning model. The FC layer predicts the output from the CNN model by performing standard deep neural networks process: feed-forward that is multiplication between the flattened feature map and the neuron weights, followed by activation function; and back-propagation to calculate the error, which is, the distance with the predicted output and the expected output and minimize the loss function with optimization algorithm.

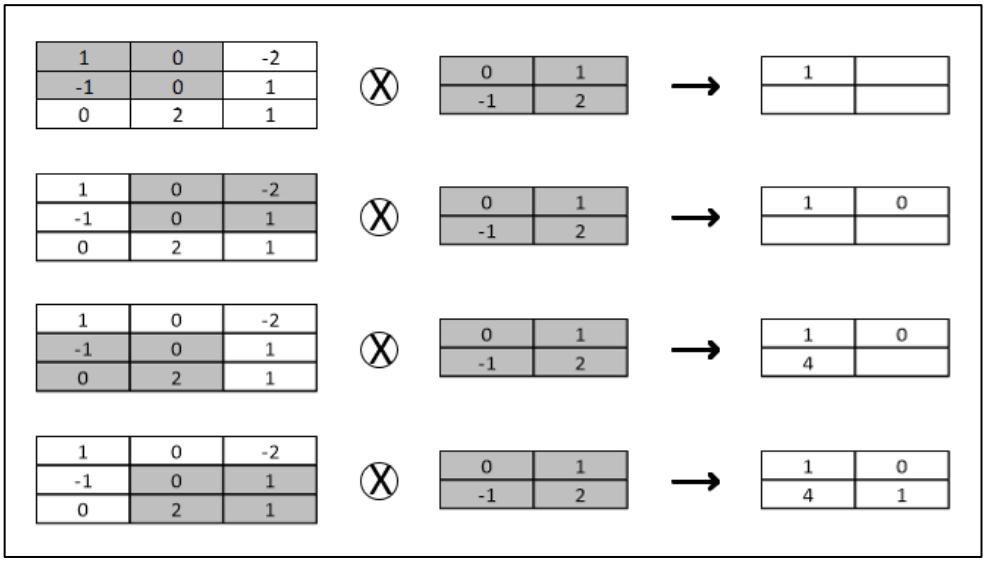


Figure 2.3 Convolution Filter of The Convolutional Layer Calculation Example

The Figure 2.3 shows how the filter process in the convolutional layer on CNN model on 3x3 input data convolved with 2x2 kernel size, filter size of 1 (the convolution filter size will be 2x2x1 with total 4 initialized weights), one stride, and zero padding. The convolution filter slides over the input data to capture the spatial feature as the output feature map by performing a dot product process. The equation 2.1 define the dot product calculation in the filter of convolutional layer.

$$O_{ij} = \sum_{i=1}^{n} X_i F_i \qquad (2.1)$$

The formula above, the dot product is the sum of multiplication between one matrix with another matrix on same rows. The X_i is the input data matrix with row i, the F_i is the convolution filter matrix with row i, and the O_{ij} is the output feature map matrix with row i and column j. The Table 2.5 shows the dot product

of convolutional filter calculation based on the Figure 2.3 with each filter slides calculation.

Table 2.6 The Dot Product of Convolution Filter Calculation Example

Number of Filter Slides	***Dot Product Calculation*** $(\sum_{i=1}^{n} X_i F_i)$	***Output*** (O_{ij})
1	$(1 \times 0) + (0 \times 1) + (-1 \times -1) + (0 \times 2)$ $0 + 0 + 1 + 0$	*1*
2	$(0 \times 0) + (-2 \times 1) + (0 \times -1) + (1 \times 2)$ $0 - 2 + 0 + 2$	*0*
3	$(-1 \times 0) + (0 \times 1) + (0 \times -1) + (2 \times 2)$ $0 + 0 + 0 + 4$	*4*
4	$(0 \times 0) + (1 \times 1) + (2 \times -1) + (1 \times 2)$ $0 + 1 + -2 + 2)$	*1*

The convolved 3x3 input data with 2x2x1 convolution filter resulting in a 2x2x1 (one-dimensional kernel) output feature map (two-dimensional feature map), where the width is 2, the height is 2, and the filter is 1. The equation 2.2 and 2.3 define the convolution filter dimension calculation to achieve the size values [19]:

$$w' = \frac{w - f + s}{s} \tag{2.2}$$

$$h' = \frac{h - f + s}{s} \tag{2.3}$$

The formula above, is how the output feature map size is determined after the convolution process with zero padding. The w' is the output feature map width, the h' is the output feature map height, the w is the input data width, the h is the input data height, the f is the filter size of the convolution filter, and the s is the stride value. The equation 2.4 and 2.5 shows the additional notation p for the padding value in the output feature map size calculation.

$$w' = \frac{w - f + s + p}{s} \tag{2.4}$$

$$h' = \frac{h - f + s + p}{s} \tag{2.5}$$

Based on the formula above, the output feature map size calculation with the 3x3 input data, the 2x2 convolution filter, one stride, and zero padding, are calculated as follows.

$$w' = \frac{3 - 2 + 1 + 0}{1} \tag{2.6}$$

$$w' = \frac{1 + 1 + 0}{1} \tag{2.7}$$

$$w' = \frac{2}{1} \tag{2.8}$$

$$w' = 2 \tag{2.9}$$

As well as the height for the output feature map,

$$h' = \frac{3 - 2 + 1 + 0}{1} \tag{2.10}$$

$$h' = \frac{1 + 1 + 0}{1} \tag{2.11}$$

$$h' = \frac{2}{1} \tag{2.12}$$

$$h' = 2 \tag{2.13}$$

Therefore, the size of the output feature map is 2x2x1.

Then, the pooling layer reduces the convolved output feature map size with the available pooling method that mentioned before. The pooling layer has similar process with the convolution layer, where it slides through the convolved output feature map horizontally and vertically with pooling method and the stride equal to the pooling size. The Figure 2.4 shows a brief example of 2x2 maximum pooling, 2x2 average pooling, GMP (Global Maximum Pooling), and GAP (Global Average Pooling). The maximum pooling finds the maximum value on each slide, the average pooling finds the average value on each slide, while the GMP finds the whole maximum value and the GAP finds the whole average value without sliding. However, both the GMP and the GAP perform the pooling without sliding through the convolved output feature map.

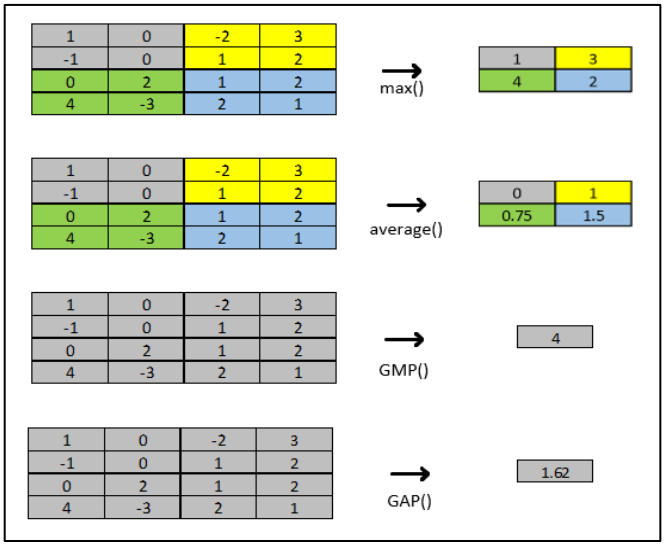


Figure 2.4 The Pooling Layer Process for Maximum Pooling, Average Pooling, GMP (Global Maximum Pooling), GVP (Global Average Pooling)

2.6 LSTM (Long Short-Term Memory)

LSTM (Long Short-Term Memory) or a sequential network is an improved of RNN (Recurrent Neural Network) handles the vanishing gradient problem that suffered from the RNN because the long-term dependency. In contrast, the LSTM has two main components on each neuron in the hidden layer that are memory cell and gates. Additionally, each transition from the previous neuron to the next neuron is considered as timestamp. The memory cell has a cell state that tracks the

information from previous timestamp of hidden layer neurons and updates whether the previous information is need to be remembered or to be forgotten for the next timestamp of memory cell. The gates in each LSTM neuron have several gates that are forget gate, input gate, and output gate [20].

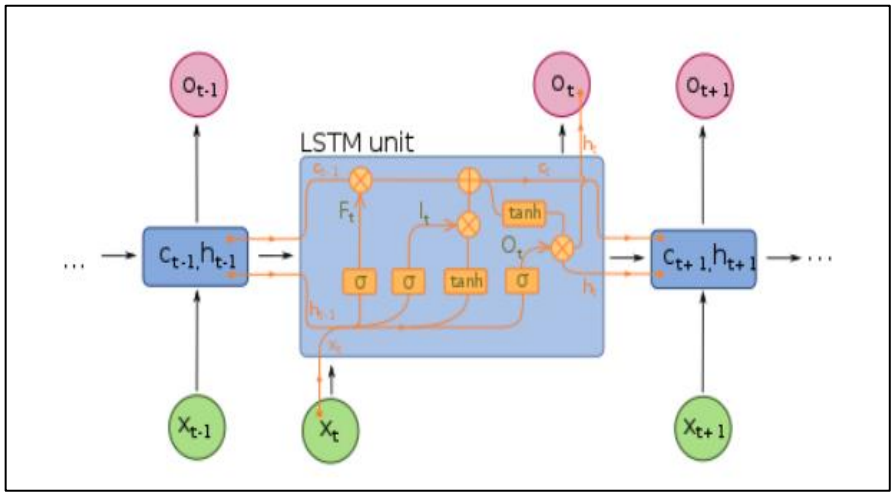


Figure 2.5 The LSTM Architecture [20]

The Figure 2.5 depicts the detail of the LSTM architecture with the notated memory cell and the gates. The basic notation of the Figure 2.5 consisted of t, $t-1$, C, and h, where the t is the current step of the timestamp, the $t-1$ is the previous step of the timestamp, the C is the cell state of the memory cell, and the h is the hidden state at the output gated

2.6.1 Forget Gate

The forget gate decides whether the neuron should keep or forget the information from the previous timestamp output. From the Figure 2.5, the forget gate is defined as follows:

$$F_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \quad (2.14)$$

The equation above shows the calculation behind the forget gate, where the F_t is the forget gate output at the current timestamp t, the σ is the activation function that is sigmoid applied the after calculation inside is completed, the W_f is the forget gate weight, the h_{t-1} is the previous hidden state output from previous

timestamp, the x_t is the input data at the current timestamp t, and the b_f is the forget gate bias.

Based on the equation, the whole process in the forget gate is begun with the multiplication between the forget gate weight W_f with the previous timestamp hidden state output h_{t-1} and the current timestamp input data x_t, then add them with the forget gate bias b_f. The sigmoid activation function applies the output of those calculation and resulting two numbers either 0 or 1 as the current timestamp forget output. Later, the current timestamp cell state will consider the current timestamp forget gate output whether the information should be kept or ignored, by multiplying the forget gate output with the previous timestamp cell state. It is defined the calculation as follows:

$$C_t = F_t * C_{t-1} + \cdots \quad (2.15)$$

The equation above shows the calculation for updating the current timestamp cell state value and later another calculation will be added with input gate section. The current timestamp cell state will forget the previous information if the value of the previous timestamp cell state C_{t-1} is 0, otherwise, the cell state will keep the previous information if the value of the previous timestamp cell state C_{t-1} is 1.

2.6.2 Input Gate

The input gate quantifies the importance of the new information carried by the input. From the Figure 2.5, the input gate is defined as follows:

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \quad (2.16)$$

$$\tilde{C}_t = tanh(W_C[h_{t-1}, x_t] + b_C \quad (2.17)$$

The equation above shows the calculation behind the input gate to calculate the current timestamp input and the new candidate cell state. The cell state uses

them to update the previous timestamp cell state. First, the current timestamp input equation defines the i_t is the input at the current timestamp t, the σ is the activation function that is sigmoid applied after the calculation inside is completed, the W_i is the input gate weight for the input quantification, the h_{t-1} is the previous hidden state output from previous timestamp, the x_t is the input data at the current timestamp t, and the b_i is the input gate bias for the input quantification. Second, the new candidate cell state equation defines the $\tilde{C}_t$ is the new candidate cell state at the current timestamp t, the $tanh$ is the activation function that is tanh that applies the after the calculation inside is completed, the W_C is the input gate weight for the new candidate cell state, the h_{t-1} is the previous hidden state output from previous timestamp, the x_t is the input data at the current timestamp t, and the b_C is the input gate bias for the new candidate cell state.

Based on the equation, the whole process of the input gate is begun with the multiplication between the input gate weight (associated with either W_i and W_C) with the previous timestamp hidden state output h_{t-1} and the current timestamp input data x_t, then add them with the input gate bias (associated with either b_i and b_C). In contrast, the input gate calculation uses sigmoid activation function and the new candidate cell state calculation uses tanh activation function. Later, the current timestamp cell state will update the previous timestamp cell state value, by multiplying the output of the current timestamp input gate with the current timestamp new candidate cell state. Finally, the complete current timestamp updating equation is defined as follows

$$C_t = F_t * C_{t-1} + i_t * \tilde{C}_t \qquad (2.18)$$

The equation above shows the complete calculation for updating the current timestamp cell state value by adding the multiplication in the forget gate output and the multiplication in the input gate output. The C_t is the new cell state output at the current timestamp t. The new value of the C_t will be proceed to the next timestamp (the next neuron) and to the output gate.

2.6.3 Output Gate

The output gate is the last gate of the LSTM neurons to make the final output as the decision or prediction. From the Figure 2.5, the output gate has similar equation from the input gate as follows:

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \quad (2.19)$$

$$h_t = tanh(C_t) * o_t \quad (2.20)$$

The equation above shows the calculation behind the output gates to calculate the output and the new current timestamp hidden state. First, the output equation defines the o_t is the output at the current timestamp t, the σ is the activation function that is sigmoid applied after the calculation inside is completed, the W_o is the output gate weight for the output calculation, the h_{t-1} is the previous hidden state output from previous timestamp, the x_t is the input data at the current timestamp t, and the b_o is the output gate bias for the output calculation. Second, the new current timestamp hidden state equation defines the h_t is the new hidden state at the current timestamp t, the $tanh$ is the activation function that is tanh that applies the after the calculation inside is completed, the C_t is the new or updated current timestamp cell state, and the o_t is the calculated output at the current timestamp t.

Based on the equation, the whole process of the output gate is begun by calculating the output o_t that similar to the input o_t at the input gate, then calculating the new current timestamp hidden state h_t multiplies the output o_t with applied activated function tanh of the updated current timestamp cell state C_t. In the end of the current timestamp, the new hidden state h_t will be also proceed to the next timestamp (the next neuron).

2.7 Related Works

Table 2.7 List of Related Works of Sentiment Analysis with Traditional Machine Learning Model

No.	Paper	Author	Method	Result
1	Sentiment analysis of social media Twitter with case of Anti-LGBT campaign in Indonesia using Naïve Bayes, decision tree, and random forest algorithm [4]	Veny Amilia Fitri, Rachmadita Andreswari, Muhammad Hasibuan Azani	Naïve Bayes, Decision Tree, Random Forest	The Naïve Bayes model gained the accuracy of 83.43% higher than Decision Tree model and Random Forest model

Table 2.8 List of Related Works of Sentiment Analysis with Single Deep Learning Model

No.	Paper	Author	Method	Result
1	Sentiment Analysis on Twitter Data by Using Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) [8]	Usha Devi Gandhi, Priyan Malarvizhi Kumar, Gokulnath Chandra Babu, Gayathri Karthick	CNN, LSTM	The CNN model gained 87.72% accuracy and the LSTM model gained 88.02% accuracy
2	Sentiment Analysis Using Word2vec and Long Short-Term Memory (LSTM) for Indonesian Hotel Reviews [9]	Putra Fissabil Muhammad, Retno Kusumaningrum, Adi Wibowo	LSTM	The LSTM model gained the best accuracy of 85.96%

Table 2.9 List of Related Works of Sentiment Analysis with Hybrid Deep Learning Model

No.	Paper	Author	Method	Result
1	A Hybrid CNN-LSTM Model for Improving Accuracy of Movie Reviews Sentiment Analysis [11]	Anwar Ur Rehman, Ahmad Kamran Malik, Basit Raza, Waqar Ali	CNN-LSTM	The CNN-LSTM model gained 91% higher than traditional machine learning and deep learning models
2	Sentiment analysis using word2vec-cnn-bilstm classification [13]	Wang Yue, Lei Li	Word2Vec, CNN-Bi-LSTM	The CNN-BiLSTM model gained high accuracy of 91.48% from other deep learning models (CNN, LSTM, Bi-

				LSTM, CNN-LSTM)
3	*A Hybrid CNN-LSTM: A Deep Learning Approach for Consumer Sentiment Analysis Using Qualitative User-Generated Contents [21]*	*Praphula Kumar Jain, Vijayalakshmi Saravanan, Rajendra Pamula*	*CNN-LSTM*	*The CNN-LSTM model gained accuracy from 90% and 91% from both datasets, higher other than single machine learning models*

Table 2.10 List of Related Works of Sentiment Analysis with Data Augmentation

No.	***Paper***	***Author***	***Method***	***Result***
1	*Document-level multi-topic sentiment classification of email data with Bi-LSTM and data augmentation [22]*	*Sisi Liu, Kyungmi Lee, Ickjai Lee*	*Data Augmentation, Bi-LSTM*	*The MultiTopic(MT)-Bi-LSTM model gained high accuracy from 78.8% to 91.8% (three datasets) and an improvement of accuracy from 1.5% to 10% with data augmentation*
2	*A data augmentation technique based on text for Vietnamese sentiment analysis [23]*	*Thien Ho Huong, Vinh Truong Hoang*	*Data Augmentation, Naïve Bayes. Random Forest, SVM*	*The Naïve Bayes, Random Forest, and Support Vector Machine models gained average accuracy of 84%, 86%, and 87% from the three datasets, then the average of accuracy improved by 10% on Random Forest model*

Fitri, Veny Amilia *et al.* [4], used several classifier models such as Naïve Bayes, Decision Tree, and Random Forest for sentiment analysis on Anti-LGBT campaign in Indonesia. They used RapidMiner to crawl 3744 comments on Twitter and pre-process with remove irrelevant tweets, case folding, tokenization, stop words removal, and stemming. The result is the Naïve Bayes model gained the accuracy of 83.43% higher than Decision Tree model and Random Forest model.

Gandhi, Usha Devi *et al.* [8], used CNN (Convolutional Neural Network) and LSTM (Long Short-Term Memory) for sentiment analysis on IMDB dataset. They used Python to crawl and process the 50,000 movie reviews; pre-process with data cleaning (e.g., removing unwanted URL, tags, links, spaces, brackets), case folding, tokenization, stop words removal; and extract the future with Word2Vec. The CNN model has input layer, Word2Vec (embedding layer), 1D convolution layer (128 features, 5x6 kernel size, ReLU activation function), Global Max Pooling 1D layer with ReLU activation, Dropout layer with 0.2 rate, and output layer with Softmax activation. Besides that, the LSTM model has input layer, embedding layer, LSTM layer with 128 features size, and output layer with sigmoid activation function. The results are the CNN model gained 87.72% accuracy and the LSTM model gained 88.02% accuracy.

Muhammad, Putra Fissabil et al. [9], used LSTM (Long Short-Term Memory) on Indonesia hotel reviews. They used Selenium and Scrappy Python's libraries to crawl and process 5,000 sentiment texts; pre-process with tokenization, stop words removal, stemming, padding comprises; and extract the future with Word2Vec. The result is the LSTM model gained the best accuracy of 85.96% with the best Word2Vec parameters: Skip-Gram, Hierarchical Softmax evaluation, and vector dimension of 300; and the best LSTM model parameters: 0.2 dropout value, 0.001 learning rate, and average pooling.

Rehman, Anwar Ur *et al.* [11], used CNN-LSTM model for sentiment analysis on IMDB and Amazon movie reviews. They used Python to crawl and process the 40,000 movie reviews on IMDB dataset and 2000 movie reviews on Amazon dataset; pre-process with sentence segmentation, space removal, tokenization, stop words removal, duplicate words removal, stemming; and extract the feature with Word2Vec. The CNN-LSTM model has several layers: Word2Vec (embedding layer with 300 vector size), three convolutional layers (with 256 filter units, ReLU activation function), three global max pooling layers (ReLU activation function), one dense layer (ReLU activation function, 0.2 dropout rate), LSTM layer (256 filter units, 0.2 dropout rate), and output layer (sigmoid activation

function). The result is the CNN-LSTM model gained 91% higher than traditional machine learning and deep learning models with 0,001 learning rate, 20 steps size, SGD (Stochastic Gradient Descent) optimizer, and 64 batch size.

Yue, Wang and Li, Lei [13], used CNN-BiLSTM model for sentiment analysis on Quora dataset. They used 2006 data on Quora, pre-processed stop words removal, low-frequency words removal, and extracted the feature with Word2Vec. The CNN-BiLSTM model has several layers: Word2Vec (embedding layer), convolutional layer (with 2x2 kernel size, 3x3 kernel size, 4x4 kernel size, 5x5 kernel size, ReLU activation function), max pooling layer (2x2 pooling size), Bi-LSTM layer, dropout layer, dense layer, and output layer with softmax activation function. The result is the CNN-BiLSTM model gained high accuracy of 91.48% from other deep learning models (CNN, LSTM, BiLSTM, CNN-LSTM) with 20 epochs, 64 batch size, and ADAM optimization.

Jain, Praphula Kumar *et al.* [21], used CNN-LSTM model for sentiment analysis on airline quality reviews. They used Python to crawl and process the 82,842 reviews on airline quality sentiment data and the 14,640 tweets on Twitter airline data; pre-process with remove unwanted words, symbols, and unknown words, tokenization, padding; and extract the feature with Keras Embedding layer. The CNN-LSTM model has several layers: embedding layer, one convolution layer (4x4 kernel size, 0.3 dropout rate, and ReLU activation function), one 1D max pooling layer (ReLU activation function), dropout layer (0.3 rate), LSTM layer (0.3 dropout rate, ReLU activation function, batch normalization), one dense layer (10 neurons, ReLU activation), and output layer (1 neurons, sigmoid activation function). The result is the CNN-LSTM model gained accuracy from 90% and 91% from both datasets, higher other than single machine learning models (Linear Regression, Support Vector Machine, Naïve Bayes, CNN, and LSTM).

Liu, Sisi *et al.* [22], used Bi-LSTM model and data augmentation for sentiment analysis on multi-topic email document. They used Python to crawl and process the 1,815 texts from three different topic of documents; pre-process with email cleaning, lowercasing, tokenization, spell-checking, stop words removal, and

lemmatization; and extract the feature with word embedding. They also used the random word replacement data augmentation, by generating new text with some words replaced with their synonym or similar words. The result is the MultiTopic(MT)-BiLSTM model gained high accuracy from 78.8% to 91.8% (three datasets) and an improvement of accuracy from 1.5% to 10% with data augmentation.

Ho Huong, Thien et al. [23], used several machine learning models (Naïve Bayes, Support Vector Machine, Random Forest) and data augmentation for sentiment analysis on Vietnamese language. They used Python to crawl and process 56,073 texts from three Vietnamese datasets; pre-process with tokenization, URLs removal, hashtag removal, email removal, symbols removal, emoticons handling, numbers removal, lowercasing, duplicated letters removal, punctuations removal, stop words removal, negation handling; and extract the future with TF-IDF (Term Frequence-Inverse Document Frequency). The result is the Naïve Bayes, Random Forest, and Support Vector Machine models gained average accuracy of 84%, 86%, and 87% from the three datasets, then the average of accuracy improved by 10% on Random Forest model.

This work, I use the single model CNN, the single model LSTM, and the hybrid CNN-LSTM for sentiment analysis and compared their results like [11], [13], and [21]. In contrast, I compare those models with data augmentation from [22] and [23] with EDA (Easy Data Augmentation) for the data augmentation method. In addition, I use different dataset sentiment analysis topic that is Twitter data about the COVID-19 vaccine types such as Sinovac, AstraZeneca, Pfizer, and Moderna. I also build the sentiment analysis web application for each COVID-19 vaccine type sentiment analysis summary; for example, tweets about Sinovac vaccine topic will have its own sentiment analysis summary, tweets about AstraZeneca vaccine topic will also have its own sentiment analysis summary, and so on. I use the pre-trained Glove Twitter for the word embedding with 200 vector dimensions. For the text pre-processing, I used most methods from all mentioned works above, however, I added another text pre-processing methods such as remove non-ASCII (American

Standard Code for Information Interchange) character, contraction expansion, and negation handling.

2.8 Incremental Model

The incremental model is one of SDLC (Software Development Life Cycle) model uses the combination of linear model elements and prototyping iterative. The incremental model builds the iterative prototyping and each iterative or incremental produces an actual product. Each iterative or incremental also has its own progress phase such as design phase, implementation phase, and testing phase [24].

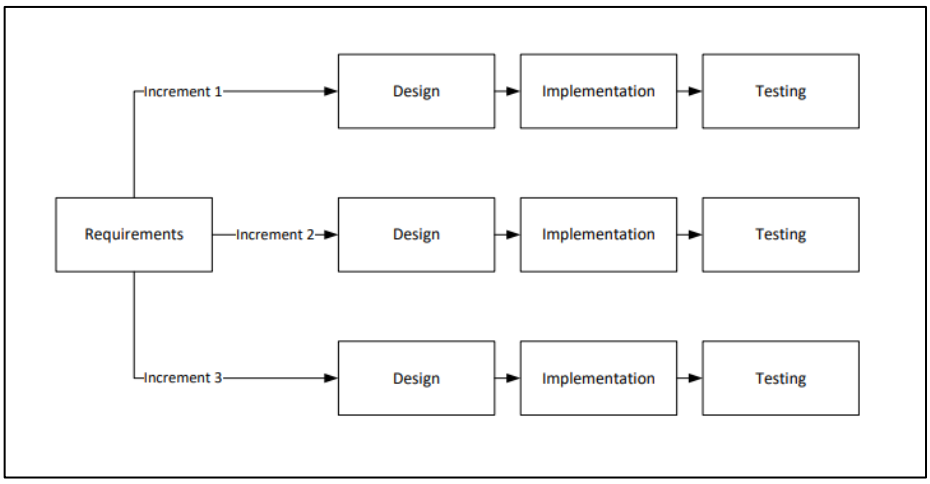


Figure 2.6 The Black Box Process Diagram

2.9 Confusion Matrix

Confusion matrix is a baseline method to measure and summarize the performance of the machine learning on classification problem. The confusion matrix depicts between the predicted values and the actual or true values [25].

Table 2.11 Binary Classification Confusion Matrix Table Example

Predicted	***Actual***	
	Negative	***Positive***
Negative	*TN (True Negative)*	*FP (False Positive)*
Positive	*FN (False Negative)*	*TP (True Positive)*

The Table 2.8 presents the confusion matrix measurement table on binary classification that consisted of TP (True Positive), TN (True Negative), FP (False

Negative), and FN (False Positive). The TP means the actual values are positive and the model predicted as positive. The TN means the values are negative and the model predicted as negative. The FP means the actual values are positive and the model predicted as negative, it is known as type-I error. The FN means the actual values are negative and the model predicted as positive, it is known as type-II error [25]. Based on the confusion matrix summary, the model evaluation metrics such as accuracy, precision, recall, and f1-score can be calculated.

2.9.1 Accuracy

Accuracy is the number of correct predictions (both TP and TN) divided by the number of all samples (all values in the confusion matrix summed up). The accuracy calculation can be defined as follows [26]:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.21)$$

2.9.2 Precision

Precision is the number of samples are predicted as positive that are actually positive, it is known as PPV (Positive Predicted Value). The precision metrics is used to limit and concern the number of the false positive values. The precision calculation can be defined as follows [26]:

$$precision = \frac{TP}{TP + FP} \quad (2.22)$$

2.9.3 Recall

Recall is to measure all positive values is predicted correctly, it is known as sensitivity, hit rate, or TPR (True Positive Rate). The recall metrics is used to limit and concern the number of the false negative values. The recall calculation can be defined as follows [26]:

$$recall = \frac{TP}{TP + FN} \quad (2.23)$$

2.9.4 F1-Score

F1-score is the average overall between precision and recall. The f1-score metrics concerns and takes both precision and recall into account. The F1-score also better in measuring imbalanced binary classification datasets. The f1-score calculation can be defined as follows [26]:

$$f1 - score = 2 \times \frac{precision \times recall}{precision + recall} \quad (2.24)$$

2.10 Black Box Testing

Black box testing is to check and verify the software application from the user's perspective. The black box testing focuses on the input given to the software application and verify the produced output and the expected output. The scope of the black box testing is does not cover the details inside the system such as code, server logic, and development method [27]:

Table 2.12 Black Box Testing Table Example

Test Scenario	***Test Case***	***Expected Output***	***Test Result***	***Summary***
Add New Item	*Input all the item data and click the add button*	*The new item has added*	*Success*	*Normal*
Delete Item	*Choose the item and click the delete button*	*The item has deleted*	*Success*	*Normal*
Update Item Quantity	*Change the item quantity values and click the update button*	*The item quantity has updated*	*Success*	*Normal*

2.11 Flask

Flask is a micro-framework for Python web development developed by Armin Ronacher. Flask has three main dependencies that are WSGI (web Server Gateway Interface) subsystems come from werkzeug, template engine supported by Jinja2, and CLI (Command Line Interface) integration comes from Click. Other than mentioned above, Flask also supports standard feature like many web frameworks such as routing, debugging, database, web forms generation, REST API (Application Programming Interface), and email [28].

CHAPTER 3
RESEARCH METHODOLOGY

3.1 Research Framework

Briefly, this research is begun with the Twitter data crawling from the all the people's sentiment about each COVID-19 vaccine type such as Sinovac, AstraZeneca, Pfizer, and Moderna with Twitter API. The crawled each COVID-19 vaccine type tweets will be merged, filtered, pre-processed, labeled, split, and augmented using Python programming language. The pre-processed and augmented training dataset will be fit to the sentiment analysis models: The single CNN model, the single LSTM model, and the hybrid CNN-LSTM model. Then, the output of the work resulted in sentiment analysis model performance comparison between three of them and sentiment analysis web application on the four dataset topics. The whole processes of this research framework are depicted in Figure 3.1 as below.

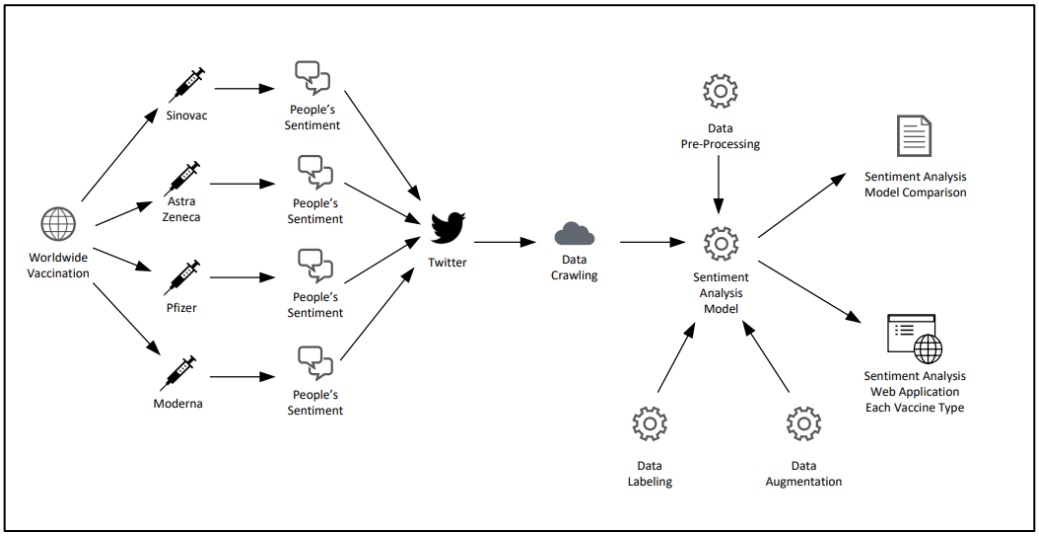


Figure 3.1 Research Framework

3.2 Research Process

The research process is consisted into three phases: the beginning of research phase, software development phase, and research output phase. The beginning of research phase there are problem identification, problem definition, and literature review.

The problem identification identifies the existed problems. Therefore, the problem identification of this research is the sentiment analysis application about each COVID-19 vaccine type using the Twitter data has not been implemented and sentiment analysis model comparisons also has not been implemented with data augmentation.

The problem definition defines the identified problems. Hence, the problem definition of this research is how to build sentiment analysis application on each COVID-19 vaccine type; and how is the performance difference of the sentiment analysis models between the single CNN model, the single LSTM model, and the hybrid CNN-LSTM model with data augmentation on each COVID-19 vaccine type. Then for the literature review, I gathered several information and related works from papers, books, and web references.

The software development phase, I used incremental model for the software development methodology. The incremental model of this work involved two incremental. The incremental 1 focuses on building the sentiment analysis models and model performance comparisons on each COVID-19 vaccine type such as Sinovac, AstraZeneca, Pfizer, and Moderna. The incremental 2 focuses on building the sentiment analysis web application using the trained sentiment analysis models from the incremental 1 on each COVID-19 vaccine type such as Sinovac, AstraZeneca, Pfizer, and Moderna. The process of this research is depicted on the Figure 3.2 below.

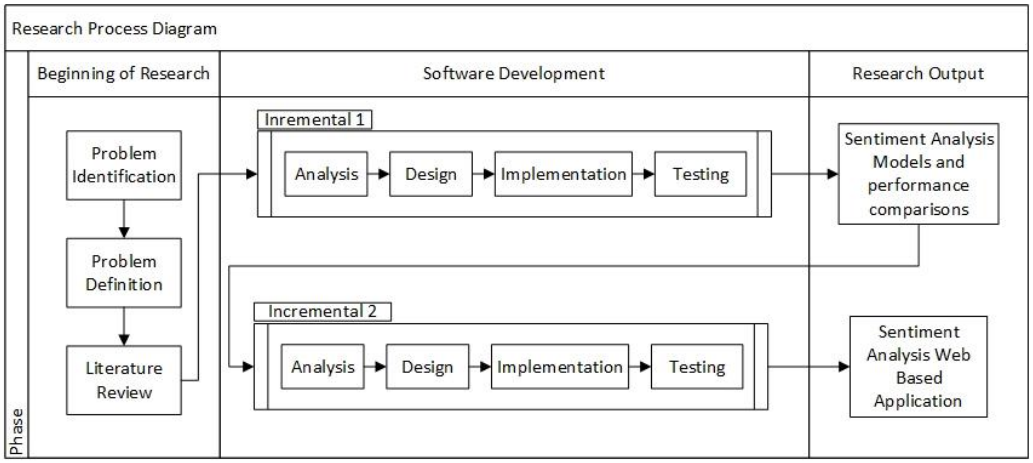


Figure 3.2 Research Process

3.3 Software Development

The software development phase of this research uses incremental model. The incremental 1 focus on building the sentiment analysis models through data crawling, data merging, data filtering, data pre-processing, data labeling, data splitting, data augmentation, and model building. The incremental 1 involved the base SDLC processes such as requirement analysis, design, implementation, and testing.

On the other hand, the incremental 2 focuses on building the sentiment analysis web application as the additional development from the incremental 1. The incremental 2 also involved the base SDLC processes such as requirement analysis, design, implementation, and testing.

3.4 Incremental 1

In the incremental 1 process, I defined the requirements needed to build the sentiment analysis models: the dataset, the pre-processing methods, the data augmentation method, the sentiment analysis models, the hardware, and the libraries. Then, I designed the activity process design to build the sentiment analysis models and the sentiment analysis model architecture designs. After that, I implemented the designed processes through coding and testing the sentiment analysis models to compare the result. The output of the incremental 1 is the sentiment analysis models and the performance comparisons for the single CNN model, the single LSTM model, and the hybrid CNN-LSTM model.

3.4.1 Requirement Analysis

The requirement analysis of the incremental 1 covered several aspects: the dataset, the pre-processing methods, the data augmentation method, the sentiment analysis models, the hardware, and the libraries.

The dataset that will be used is about each COVID-19 vaccine types tweets from Twitter by using Twitter API. Hence, there will be four dataset topics such as Sinovac dataset, AstraZeneca dataset, Pfizer dataset, and Moderna dataset; where contains tweet texts that are related to them. The period of the dataset is taken from 1 March 2022 until 31 March 2022, because the limitation of the elevated access on Twitter API that only can retrieve tweet texts no more than seven days. Therefore, the dataset crawling is carried manually every week to crawl the four COVID-19 vaccine type tweets. The crawled datasets will be divided into two datasets, one will be used for building the sentiment analysis model and the other will be used for implementing the dataset to the sentiment analysis web application. The dataset that will be used for building the sentiment analysis model is the first three weeks of March 2022, which is, from 1 March 2022 until 21 March 2022 as the experiment dataset. On the other hand, the dataset that will be used for implementing the dataset to the sentiment analysis web application is the last one week of March 2022, which is, from 22 March 2022 until 31 March 2022 as the implementation dataset. Therefore, the whole incremental 1's process use the experiment dataset.

The pre-processing process uses several NLP (Natural Language Processing) approaches to handle text data and divided into three parts. The first part focuses on removing HTML (HyperText Markup Language) tags, removing retweets notation "RT", removing URLs (Uniform Resource Allocators), removing mentions that contain username (notated with "@"), removing hashtags (notated with "#") with its term, removing non-ASCII (America Standard Code for Information Interchange) characters, changing the written numbers in words into actual numbers, removing numbers, and case folding. The second part focuses on expanding contractions, replacing negations, removing punctuations, removing stopwords, and lemmatizing. The third part focuses on removing duplicated tweets,

removing empty values, and dividing into experiment dataset and implementation dataset.

Because of the huge raw data, limited time, and resources then it is expensive to label all the datasets manually. Therefore, the labeling process uses lexicon-based method as automated labeling that resulting in positive sentiment, negative sentiment, and neutral sentiment on the unlabeled pre-processed datasets. However, only the positive sentiment and the negative sentiment will be used for the sentiment analysis. Afterwards, the pre-processed labeled datasets will be split into two samples that are training dataset and testing data with 90% and 10% respectively.

The data augmentation will be applied only on the training dataset for each four dataset topics. The data augmentation method used is EDA (Easy Data Augmentation). The EDA method uses synonym replacement, random insertion, random swap, and random deletion. The augmented text data with EDA will be merged to the original datasets, hence will be resulting in bigger training dataset size.

The sentiment analysis models that will be used is a supervised model that are the single CNN model, the single LSTM model, and the hybrid CNN-LSTM model. All of them are compared with each other respect to four dataset topics using same architecture and parameters. The augmented training dataset is fit to the sentiment analysis models. During the training process, the training dataset is implemented with cross validation using 10 k-folds. The pre-trained Glove Twitter word embedding weight will be used to map the pre-trained weight to my own corpus.

Table 3.1 The Hardware Requirement

No.	***Type***	***Name***
1	*Operating System*	*Linux Ubuntu*
2	*Processor*	*Intel i7 6859K 3.60 GHz 12 cores*
3	*Memory*	*62GB*
4	*GPU*	*NVIDIA RTX A6000*

	NVIDIA GeForce 1080 Ti

Table 3.2 The Software Requirement

No.	***Name***	***Type***	***Version***
1	*Jupyter Notebook*	*Integrated Development Environment (IDE)*	*-*
2	*Python*	*Programming Language*	*3.6.9*
3	*tweepy*	*Python Library*	*4.6.0*
4	*pandas*	*Python Library*	*1.1.5*
5	*matplotlib*	*Python Library*	*3.3.4*
6	*regex*	*Python Library*	*2022.1.8*
7	*string*	*Python Library*	*-*
8	*NLTK (Natural Language Tool Kit)*	*Python Library*	*3.6.7*
9	*unicodedata2*	*Python Library*	*-*
10	*BeautifulSoap*	*Python Library*	*4.11.1*
11	*pycontractions*	*Python Library*	*2.0.1*
12	*word2number*	*Python Library*	*1.1*
13	*vaderSetiment*	*Python Library*	*3.3.2*
14	*gensim*	*Python Library*	*4.1.2*
15	*textaugment*	*Python Library*	*1.3.4*
16	*scikit-learn*	*Python Library*	*0.24.2*
17	*tensorflow*	*Python Library*	*2.6.2*
18	*keras*	*Python Library*	*2.6.0*

The Table 3.1 defines all the requirement tools that will be used in this work. I used hosted *Jupyter Notebook* for the Python IDE, a web-based Python notebook, because of the limited resources of my personal computer to build the sentiment analysis model. The *Python* programming language will be used throughout the incremental 1 to produce the sentiment analysis model. The *tweepy* library will be used for the Twitter data crawling. The *pandas* library will be used for reading, filtering, concatenating, and grouping the dataset as dataframe. The *matplotlib* library will be used for visualizing the dataset. The *regex* library will be used for

performing regular expression on removing retweets, removing hashtags, removing mentions, and removing URLs. The *string* library will be used for removing punctuations on the tweet cleaning process. The *NLTK* library will be used for tokenizing texts, lemmatizing words, removing stopwords, and replacing negation with antonyms. The *unicodedata2* library will be used for removing non-ASCII characters. The *BeautifulSoap* library will be used for removing HTML tags. The *pycontractions* library will be used for expanding contractions. The *word2number* library will be used for replacing written numbers in words to actual numbers. The *vaderSentiment* library will be used for labeling the dataset based on its lexicon automatically. The *gensim* library will be used for building the pre-trained word embedding Word2Vec model. The *textaugment* library will be used for augmenting the text dataset with EDA method. The *scikit-learn* library will be used for splitting the datasets and evaluating the models with confusion matrix. The *tensorflow* and *keras* library will be used for building the hybrid CNN-LSTM model, the single CNN model, and the single LSTM model through training, validation, and testing process.

3.4.2 Design

The design of the incremental 1 divided into two designs that are the activity process design and the sentiment analysis model architecture design. The activity process design is designed with flowchart, depicts all the processes in the incremental 1. The sentiment analysis model architecture design designs the sentiment analysis models with diagram about the layers, the neurons, the hyperparameters, and the training parameters.

3.4.2.1 Activity Process Design

All The incremental 1 process are to produce the sentiment analysis models based on the defined requirements as shown in the Figure 3.3 below. The main processes in the incremental 1 are started with crawling the Twitter data; merging the crawled raw data; filtering the merged raw data; pre-processing the filtered raw data; labeling the pre-processed dataset; splitting the pre-processed dataset into training and testing dataset; augmenting the labeled training dataset; loading the

pre-trained Glove Twitter with 200 vectors word embedding; and building the sentiment analysis model such as the single CNN model, the single LSTM, and the hybrid CNN-LSTM model using the loaded pre-trained Glove Twitter with 200 vectors word embedding weight, augmented training data, and testing data.

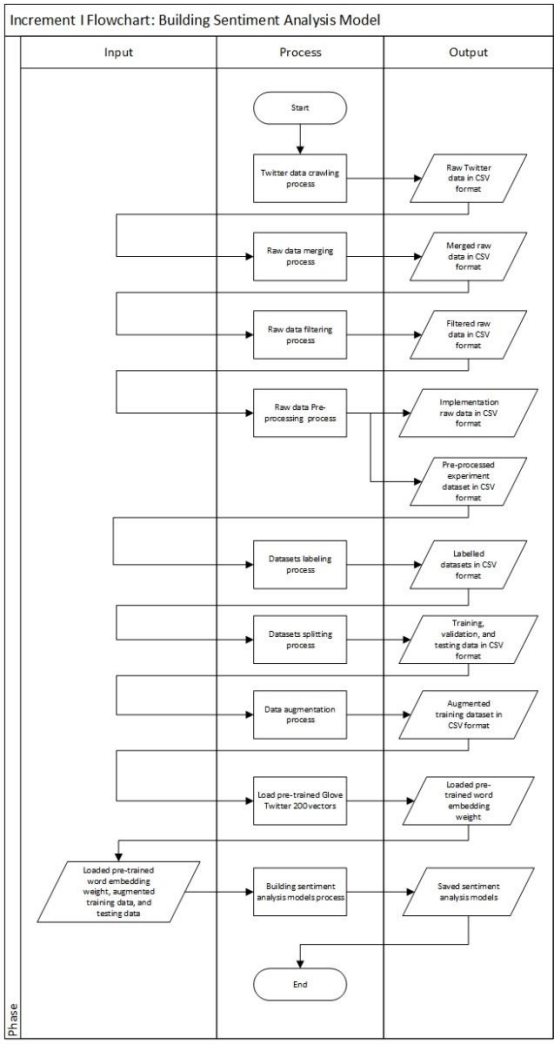


Figure 3.3 Building the Sentiment Analysis Model Flowchart

The Twitter data crawling process crawls the Twitter data and transforms them into datasets as shown in the Figure 3.4 below. The Twitter data crawling process uses the API token and the API key to access the Twitter API. Then, built the Twitter API search query to search for tweets that are related to each COVID-19 vaccine types: Sinovac, AstraZeneca, Pfizer, and Moderna. The search query used keyword terms, hashtag, and negation operation. The whole processes are

done by five times from 1 March 2022 until 31 March 2022. Finally, the stored Twitter data in dataframes is saved into CSV format.

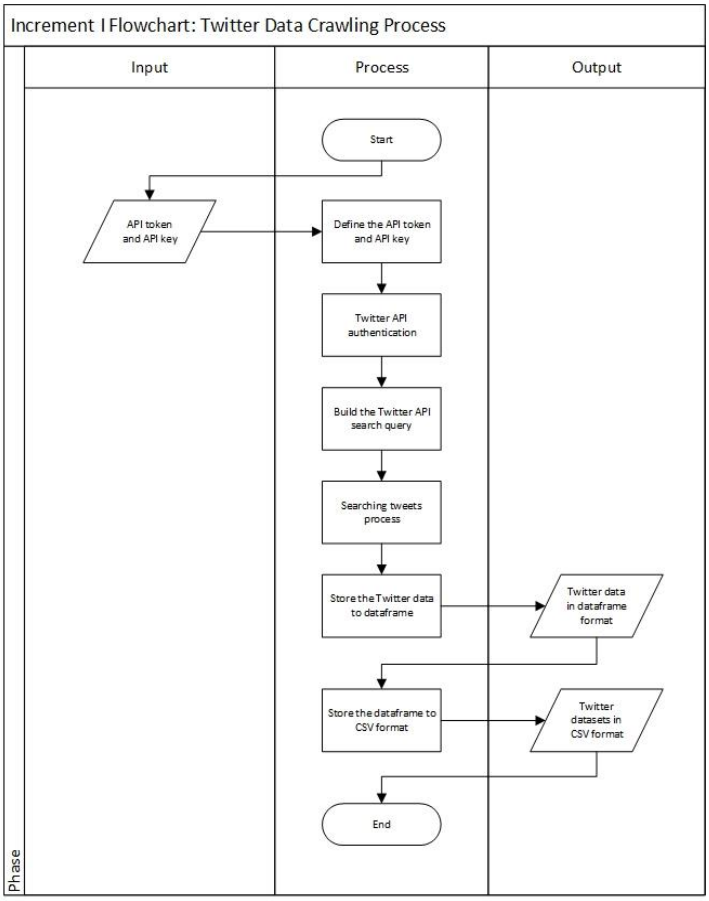


Figure 3.4 Twitter Data Crawling Process Flowchart

The dataset merging process merges the crawled Twitter data on each week into four new raw data respected to their topics as shown in the Figure 3.5 below. For example, the raw data about Sinovac that has five datasets are merged into one Sinovac raw data. Therefore, the result of the raw data merging process will be produced four new raw data topics in CSV format.

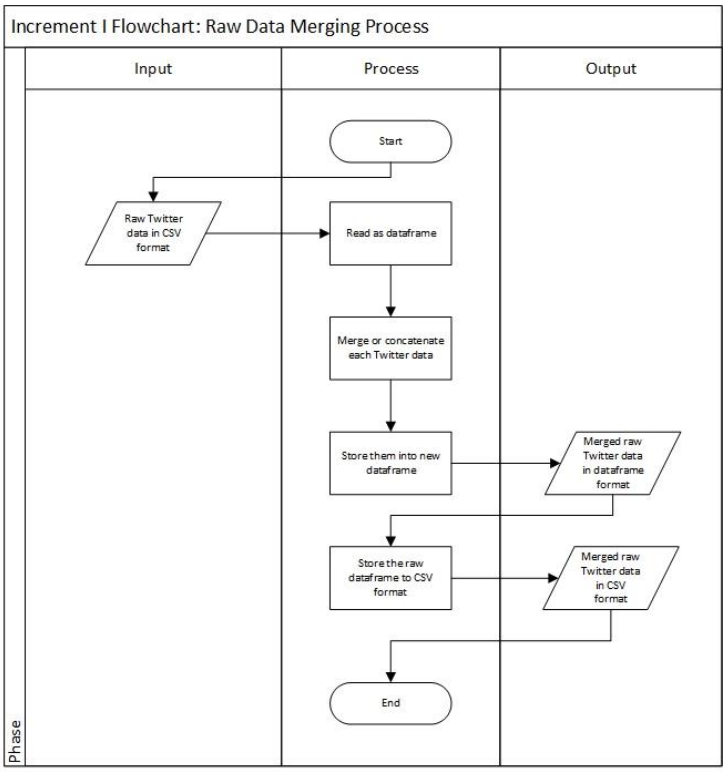


Figure 3.5 Raw Data Merging Process Flowchart

The raw data filtering process filters the four raw data topics by the date, in order to retrieve all raw data only in March 2022 period as shown in the Figure 3.6 below. Therefore, the result of the raw data filtering process produced four new raw data topics in CSV format with only March 2022 period.

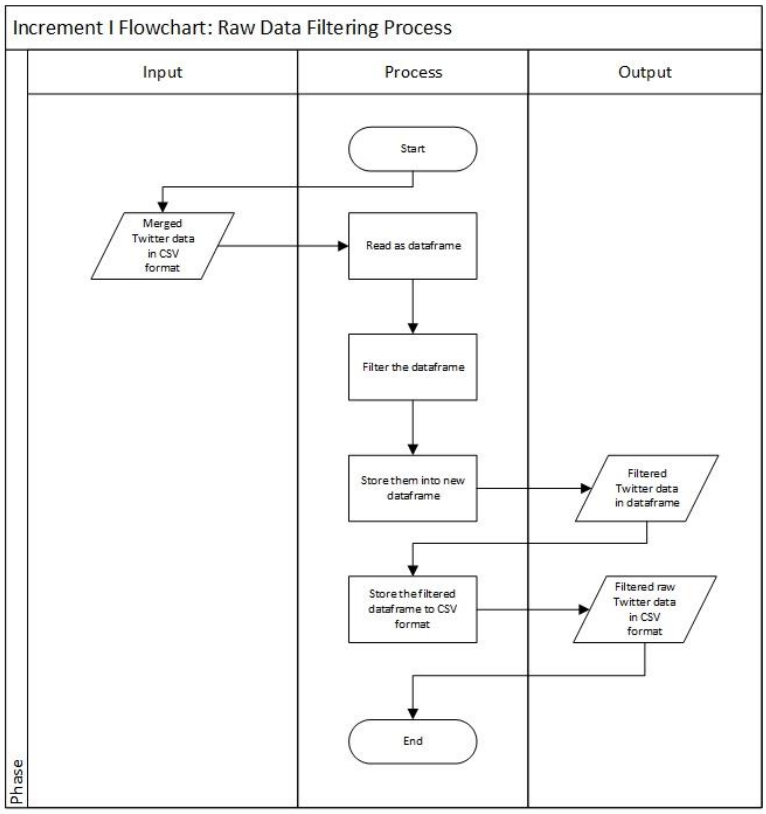


Figure 3.6 Raw Data Filtering Process Flowchart

The first part raw data pre-processing process cleans the noises or unnecessary characters or symbols or terms in the Twitter data as shown in the Figure 3.7 below. The *BeautifulSoup* library will be used to remove all HTML tags in text using. The *regex* library will be used to remove retweet notations "RT"; remove URLs; remove all mention notations "@" and its usernames; remove hashtags and its terms in text. The *unicodedata2* library will be used to remove all non-ASCII characters in text. The *word2number* library will be used to change all numbers that written in words into actual numbers using in text and then the numbers are removed. Finally, all the texts will be case folded to lower case all the words in the text. The result of the first part raw data pre-processing is continued to the second part raw data pre-processing.

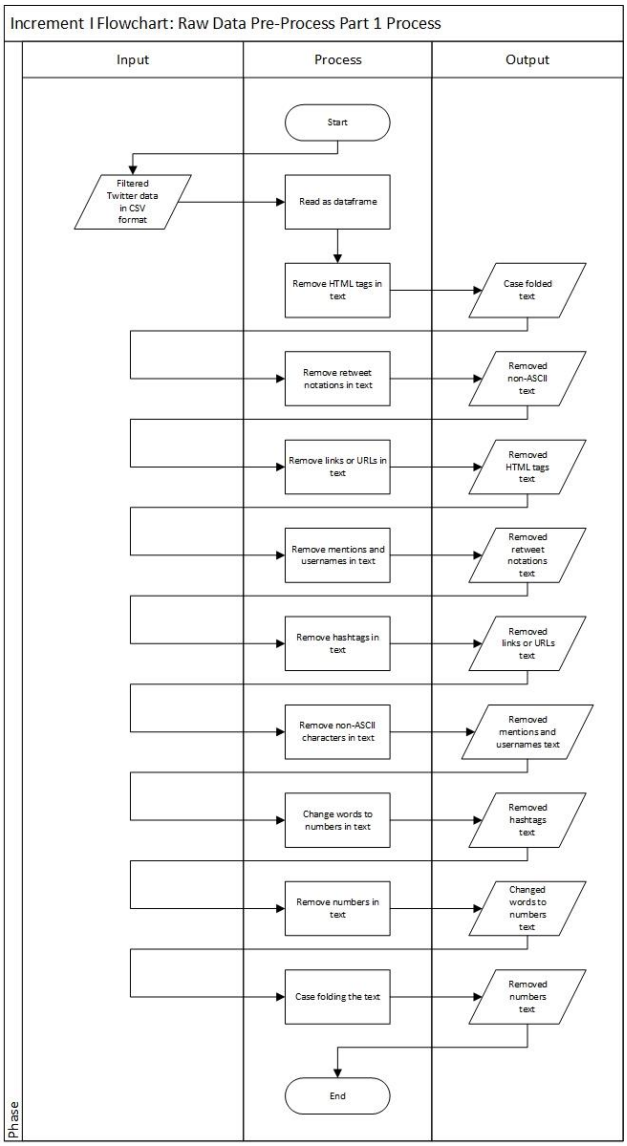


Figure 3.7 Raw Data Pre-Processing Part 1 Process Flowchart

The second part raw data pre-processing process normalizes the cleaned texts from the first part pre-processing as shown in the Figure 3.8 below. The *pycontraction* library will be used to expand all contractions in text. The *NLTK wordnet* library to replace all negations with their antonym. The *string* library will be used to remove punctuations. The *NLTK stopwords* library will be used to remove all stopwords in text. The *NLTK WordNetLemmatizer* and *NLTK POST tag* library will be used to lemmatize all words in text. Finally, several words that only has 2 characters as the residual from the text pre-processing will be removed. The

lemmatized text as the result of the second part raw data pre-processing is continued to the third part raw data pre-processing.

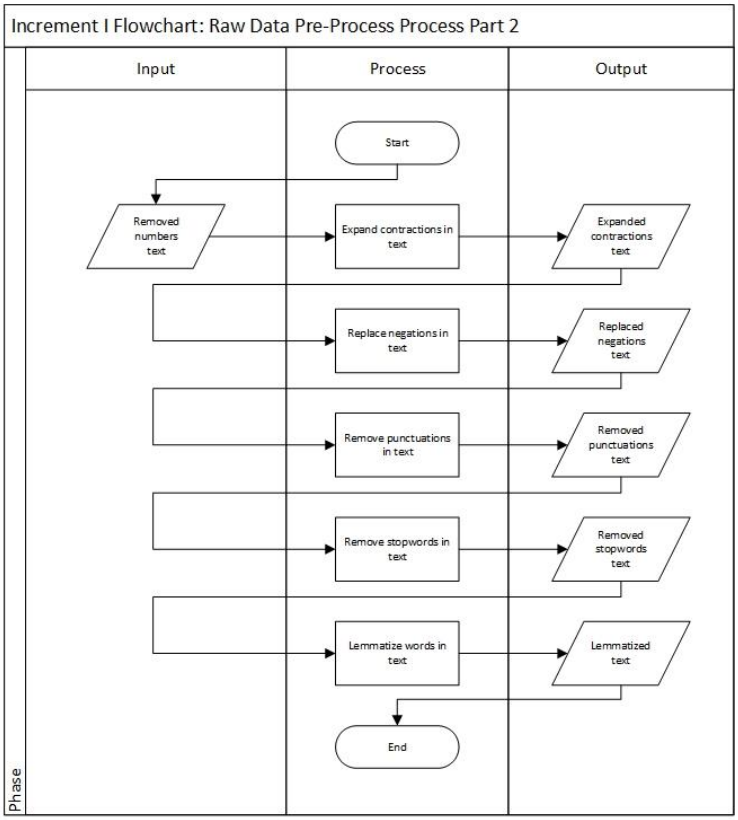


Figure 3.8 Raw Data Pre-Processing Part 2 Process Flowchart

The third part raw data pre-processing process finalizes the pre-processed texts as shown in the Figure 3.9 below. The duplicated text will be removed, but the first occurrence is kept. Then, the experiment dataset and the implementation dataset will be chosen. The experiment dataset will be used for training and testing the sentiment analysis models. On the other hand, the implementation dataset will be used for the sentiment analysis web application. In contrast, the implementation dataset only used the raw or not pre-processed tweet texts. All empty values in the experiment dataset and the implementation dataset will be removed. Finally, the result of the dataset per-processing is four pre-processed experiment dataset topics are used to build the sentiment analysis model and four raw implementation dataset topics are used as in the sentiment analysis web application.

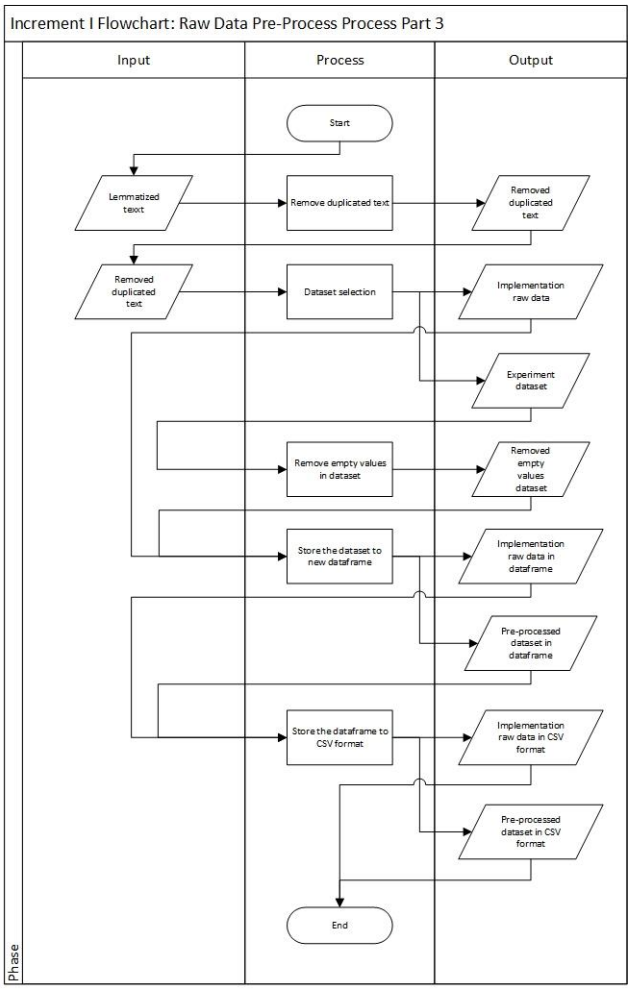


Figure 3.9 Dataset Pre-Processing Part 3 Process Flowchart

The dataset labeling process labels the four unlabeled pre-processed experiment dataset topics as shown in the Figure 3.10 below. The *vaderSentiment* library will be used because it has performed better than other lexicon-based sentiment analysis libraries like *TextBlob* library on social media text [29] [30] [31]. The result of the dataset labeling process using *vaderSentiment* library produce positive sentiment, negative sentiment, and neutral sentiment. However, only the positive sentiment and the negative sentiment will be used and saved into four new dataset topics in CSV format.

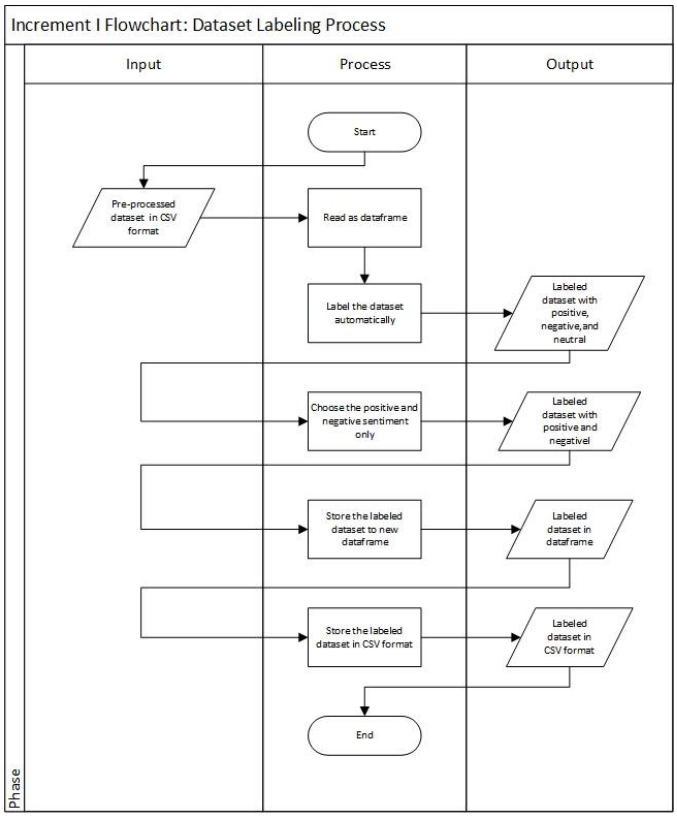


Figure 3.10 Dataset Labeling Process Flowchart

The dataset splitting process splits the four labeled dataset topics into training data and testing data as shown in the Figure 3.11 below. The scikit-learn library will be used to split the datasets into training data and testing data. Before the dataset split, the number of words in the tweet texts that below five will be removed, because it is considered as short text. The splitting portions will be used are 90% training and 10% testing data. Finally, the results of the dataset splitting process are saved into CSV format, where each of the four dataset topics has training data and testing data.

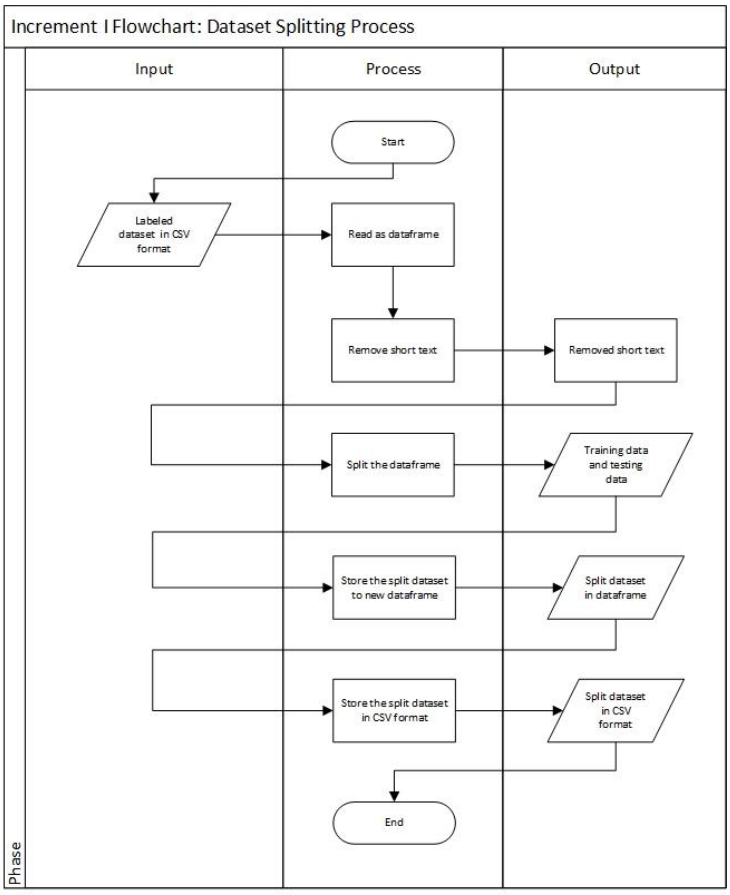


Figure 3.11 Dataset Splitting Process Flowchart

The dataset augmentation process generates new text data for the training data as shown in the Figure 3.12 below. First, read the four training dataset topics in CSV format to dataframe. Then, they will be applied with EDA using *textaugment* library. The EDA method parameters is set as it is suggested from [32] that are the number of words to be replaced randomly with its synonym is 10% from the total length of the text, the number of words to be inserted randomly with other words is 10% from the total length of the text, the number of swap times to swap randomly between two words is 10% from the total length of the text, and the probability to delete words randomly is 10% from the total length of the text. Then, the results of the dataset augmentation process are saved into CSV format, that the four dataset topics have contained augmented training data.

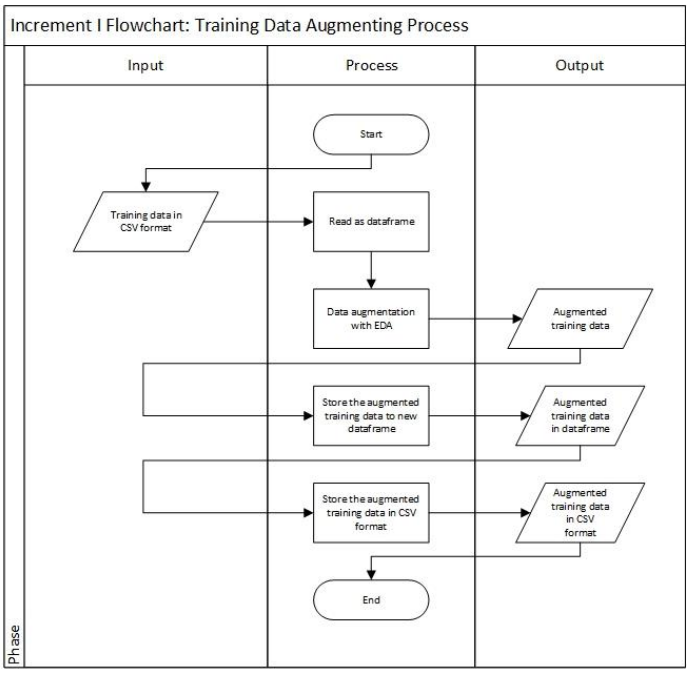


Figure 3.12 Training Data Augmentation Process Flowchart

The building sentiment analysis model process builds, trains, and tests the single CNN model, the single LSTM model, and the hybrid CNN-LSTM model on each four dataset topics as shown in the Figure 3.13 below. The four augmented training and testing dataset topics are used to train, test, and evaluate the sentiment analysis models. The *keras* library is used to build, train, test, and evaluate the sentiment analysis models. The four augmented training dataset topics and the four testing dataset topics are tokenized and transformed into sequence of positive integers respect to its word uniquely. The four training and testing label topics are changed into categorical one-hot encoding. Further, sequence padding is applied to equalize the sequence length, because the input of the sentiment analysis models requires same sequence length. The mapped pre-trained word embedding matrix and the padded augmented training data will be used in the sentiment analysis model training processes. The training processes will be done separately for the four dataset topics and split into the training data and validation data with 10 k-folds cross validation. The result of the sentiment analysis models produced on the four topic datasets are saved so it can be used in the sentiment analysis web application.

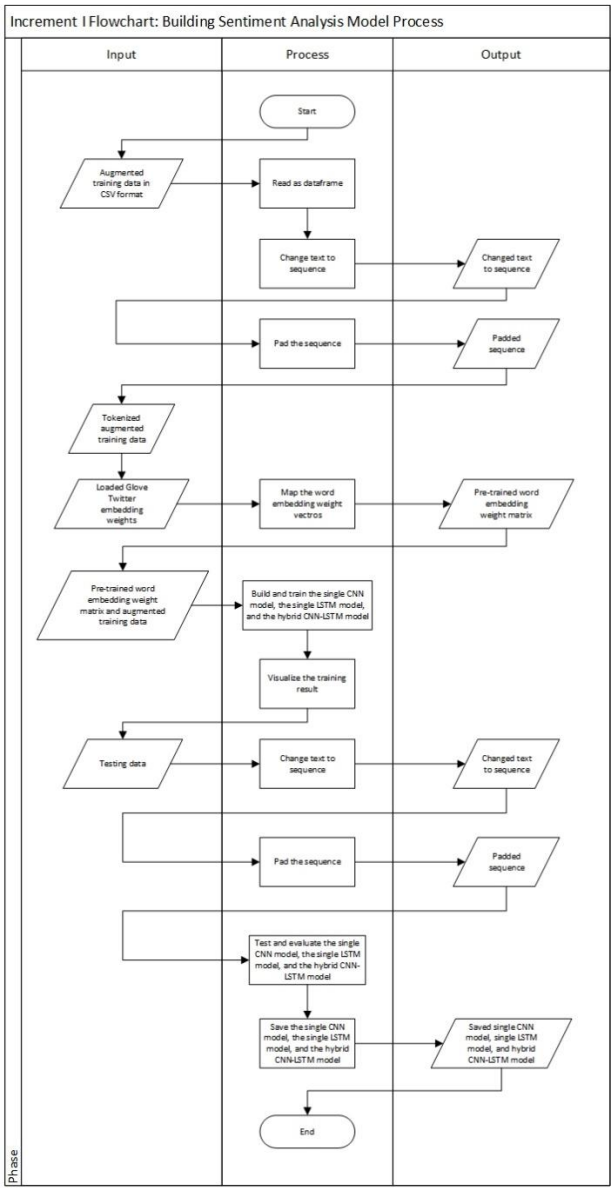


Figure 3.13 Building Sentiment Analysis Model Process Flowchart

3.4.2.2 Sentiment Analysis Model Architecture Design

This section depicts and defines the model architecture design for the single CNN model, the single LSTM model, and the hybrid CNN-LSTM model. The model architecture design covers the number of layers, the number of neurons, and the model hyperparameters depicted with diagram. The three sentiment analysis models used exactly same training parameters that are 64 batch size, 50 epochs, categorical loss entropy, and ADAM optimizer with 0,001 learning rate,

3.4.2.2.1 Single CNN Model Architecture Design

The single CNN model architecture has word embedding layer, one convolutional layer, one max pooling layer, and two dense layers as shown in the Figure 3.14 below. First, the word embedding layer weight uses the trained custom Word2Vec model with my own corpus. The mapped pre-trained Glove Twitter word embedding with own corpus has 200-dimension vectors. Then, the dropout layer is applied after the word embedding layer with 50% dropout rate. The CNN model consisted of one convolution layer and one max pooling layer. The 1D convolution layer and 1D max pooling layer are used. The 1D convolution layer has 64 filter units, 3 kernel size, 1 stride, valid padding, and ReLU activation function. The 1D max pooling layer has 2 pool size and 1 stride. Then, the dropout layer is applied after the CNN model with 50% dropout rate. Further, the CNN model output will be fit to the two layers classifier dense (basic neural network) model. The first dense layer has 128 units and ReLU activation function, followed by dropout layer with 50% dropout rate. The second layer has 2 units and softmax activation function as the output layer.

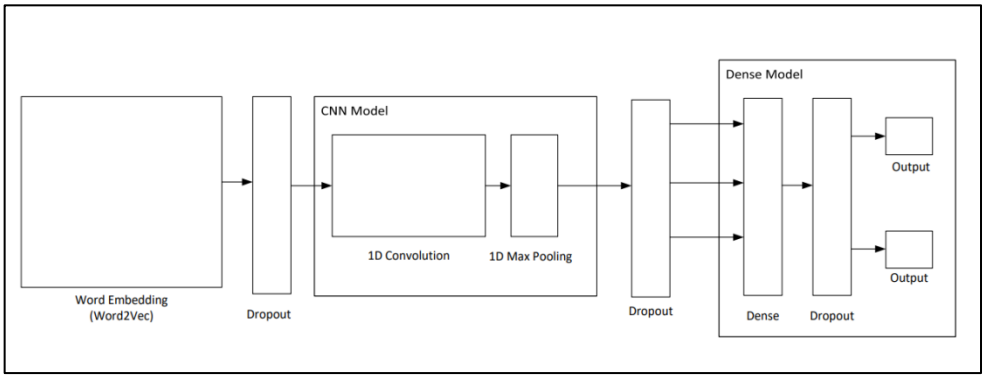


Figure 3.14 The Single CNN Model Architecture Design

3.4.2.2.2 Single LSTM Model Architecture Design

The single LSTM model architecture has word embedding layer, one LSTM layer, and two dense layers as shown in the Figure 3.15 below. First, the word embedding layer weight uses the trained custom Word2Vec model with my own corpus. The mapped pre-trained Glove Twitter word embedding with own corpus has 200-dimension vectors. Then, the dropout layer is applied after the word embedding layer with 50% dropout rate. Then, the LSTM layer has 128 units, 50%

dropout rate, and tanh activation function. Finally, the output from the LSTM model will be fit to the two layers classifier dense (basic neural network) model. The first dense layer has 128 units and ReLU activation function, followed by dropout layer with 50% dropout rate. The second layer has 2 units and softmax activation function as the output layer.

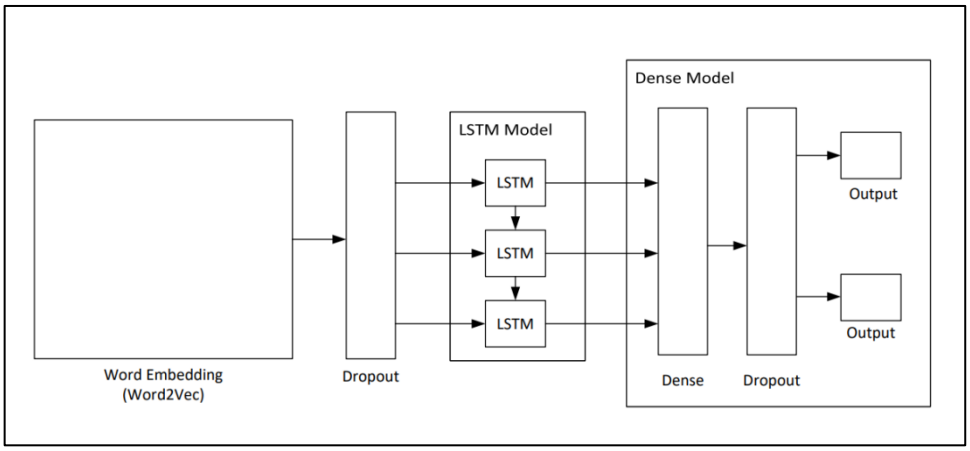


Figure 3.15 The Single LSTM Model Architecture Design

3.4.2.2.3 Hybrid CNN-LSTM Model Architecture Design

The hybrid CNN-LSTM model architecture combines the single CNN model architecture and the single LSTM model architecture with exact same layers, neurons, and parameters as shown in the Figure 16 below. First, the word embedding layer weight uses the trained custom Word2Vec model with my own corpus. The mapped pre-trained Glove Twitter word embedding with own corpus has 200-dimension vectors. Then, the dropout layer is applied after the word embedding layer with 50% dropout rate. The hybrid CNN-LSTM model architecture is similar to [21]. The CNN model consisted of one convolution layer and one max pooling layer. The 1D convolution layer and 1D max pooling layer are used. The 1D convolution layer has 64 filter units, 3 kernel size, 1 stride, valid padding, and ReLU activation function. The 1D max pooling layer has 2 pool size and 1 stride. Then, the dropout layer is applied after the CNN model with 50% dropout rate. Further, the CNN model output will be fit to the LSTM model that has one layer LSTM. The LSTM layer has 128 units, 50% dropout rate, and tanh activation function. Finally, the output from the LSTM model will be fit to the two

layers classifier dense (basic neural network) model. The first dense layer has 128 units and ReLU activation function, followed by dropout layer with 50% dropout rate. The second layer has 2 units and softmax activation function as the output layer.

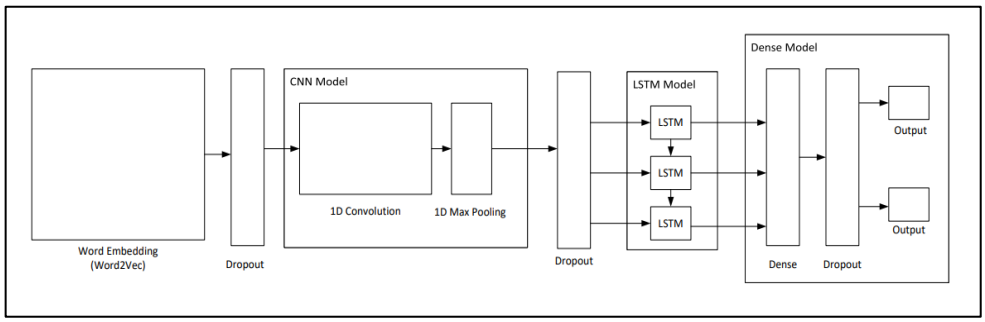


Figure 3.16 The Hybrid CNN-LSTM Model Architecture Design

3.4.3 Implementation

The crawling Twitter data code as shown in the Figure 3.17 below, uses the API token and the API key; accesses the Twitter API authentication with *OAuthHandler()* object and *set_access_token()*; defines Twitter API search query by using keyword terms, hashtag, and negation operations to search topics about Sinovac vaccine, Moderna vaccine, Pfizer vaccine, and AstraZeneca vaccine respectively; and stores into *pandas* dataframe by taking the tweet texts, tweet dates, and tweet languages as the columns.

```
"""# **Crawling Twitter Dataset**"""

ACCESS_TOKEN = '1491321301377904640-3f34SrhEWeXTNchIPVp7P1ZETxaoN0'
ACCESS_TOKEN_SECRET = 'sc3mB96PjXkR53e5K4g8binVxuohShV1WYZsuxkbofxCH'
CONSUMER_KEY = '77fSLVCOYcm2pX5rwxieKYEI4'
CONSUMER_SECRET = 'nikfO3dn95p5em3hgXUO7SYFuHpH3DS7S0WByOvoDaE3L9zbD1'

auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
auth.set_access_token(ACCESS_TOKEN, ACCESS_TOKEN_SECRET)
api = tweepy.API(auth, wait_on_rate_limit=True, wait_on_rate_limit_notify=True)

search_sinovac_keywords = '(sinovac OR #sinovac) -moderna -pfizer -astrazeneca lang:en -is:retweet'
search_astrazeneca_keywords = '(astrazeneca OR #astrazeneca) -moderna -pfizer -sinovac lang:en -is:retweet'
search_pfizer_keywords = '(pfizer OR #pfizer) -sinovac -moderna -astrazeneca lang:en -is:retweet'
search_moderna_keywords = '(moderna OR #moderna) -sinovac -pfizer -astrazeneca lang:en -is:retweet'

tweets = tweepy.Cursor(api.search, q=search_astrazeneca_keywords, tweet_mode='extended', lang='en').items(2000)

tweet_texts = []
tweet_dates = []
tweet_langs = []
for tweet in tweets:
  tweet_texts.append(tweet.full_text)
  tweet_dates.append(tweet.lang)
  tweet_langs.append(tweet.created_at)
```

Figure 3.17 The Twitter Data Crawling Code

The dataset merging code merges all the crawled Twitter data into four dataset topics: Sinovac dataset, AstraZeneca dataset, Pfizer dataset, and Moderna

dataset as shown in the Figure 3.18 below. I used the *pandas concat()* function to merge all the crawled Twitter data into one dataframe.

```
## **Concate Dataset: Sinovac**
"""

df_sinovac1 = pd.read_csv('raw/sinovac1.csv')
df_sinovac2 = pd.read_csv('raw/sinovac2.csv')
df_sinovac3 = pd.read_csv('raw/sinovac3.csv')
df_sinovac41 = pd.read_csv('raw/sinovac41.csv')
df_sinovac42 = pd.read_csv('raw/sinovac42.csv')

df_sinovac = pd.concat([df_sinovac1, df_sinovac2, df_sinovac3, df_sinovac41, df_sinovac42], ignore_index=True)

"""## **Concate Dataset: AstraZeneca*****"""

df_astrazeneca1 = pd.read_csv('raw/astrazeneca1.csv')
df_astrazeneca2 = pd.read_csv('raw/astrazeneca2.csv')
df_astrazeneca3 = pd.read_csv('raw/astrazeneca3.csv')
df_astrazeneca41 = pd.read_csv('raw/astrazeneca41.csv')
df_astrazeneca42 = pd.read_csv('raw/astrazeneca42.csv')

df_astrazeneca = pd.concat([df_astrazeneca1, df_astrazeneca2, df_astrazeneca3, df_astrazeneca41, df_astrazeneca42], ignore_index=True)

"""## **Concate Dataset: Pfizer*****"""

df_pfizer1 = pd.read_csv('raw/pfizer1.csv')
df_pfizer2 = pd.read_csv('raw/pfizer2.csv')
df_pfizer3 = pd.read_csv('raw/pfizer3.csv')
df_pfizer41 = pd.read_csv('raw/pfizer41.csv')
df_pfizer42 = pd.read_csv('raw/pfizer42.csv')

df_pfizer = pd.concat([df_pfizer1, df_pfizer2, df_pfizer3, df_pfizer41, df_pfizer42], ignore_index=True)

"""## **Concate Dataset: Moderna*****"""

df_moderna1 = pd.read_csv('raw/moderna1.csv')
df_moderna2 = pd.read_csv('raw/moderna2.csv')
df_moderna3 = pd.read_csv('raw/moderna3.csv')
df_moderna41 = pd.read_csv('raw/moderna41.csv')
df_moderna42 = pd.read_csv('raw/moderna42.csv')

df_moderna = pd.concat([df_moderna1, df_moderna2, df_moderna3, df_moderna41, df_moderna42], ignore_index=True)
```

Figure 3.18 The Dataset Merging Code

The dataset filtering code uses *drop()* function from the *pandas* library to filter the four dataset topics only on March 2022 period and in English language as shown in the Figure 3.19 below as the example taken from the Sinovac dataset merging code.

```
# Filter only March
df_sinovac = df_sinovac.drop(df_sinovac[df_sinovac['date'] < '2022-03-01 00:00:00'].index)
df_sinovac = df_sinovac.drop(df_sinovac[df_sinovac['date'] > '2022-03-31 24:59:59'].index)
df_sinovac = df_sinovac.drop(df_sinovac[df_sinovac['lang'] != 'en'].index)
df_sinovac = df_sinovac.reset_index(drop=True)
```

Figure 3.19 The Dataset Filtering Code

The removing HTML tags code uses the *BeautifulSoup()* class from the *BeautifulSoup* library by passing the given text and the HTML parser rule *lxml* as shown in the Figure 3.20 below.

```
# Remove html tags in text
def remove_html_tags(self, text):
    cleaned_text = BeautifulSoup(text, 'lxml')
    return cleaned_text.get_text()
```

Figure 3.20 The Dataset Pre-Processing: Removing HTML Tags Code

The removing retweets code uses *sub()* function from the *regex* library by passing the regular expression pattern to remove the retweets notation "RT" at the beginning of the text as shown in the Figure 3.21 below.

```
# Remove 'RT' or 'rt' in text
def remove_retweets(self, text):
    cleaned_text = re.sub(r'\bRT\b', '', text)
    return cleaned_text
```

Figure 3.21 The Dataset Pre-Processing: Removing Retweets Code

The removing URLs code uses *sub()* function from the *regex* library by passing the regular expression pattern to remove the URLs as shown in the Figure 3.22 below.

```
# Remove URLs in text
def remove_urls(self, text):
    cleaned_text = re.sub(
    '(?:http?\:\/\/|http?\:\/|http?\:|https?\:\/\/|https?\:\/|https?\:|www)\S+', '', text)
    return cleaned_text
```

Figure 3.22 The Dataset Pre-Processing: Removing URLs Code

The removing mentions code uses *sub()* function from the *regex* library by passing the regular expression patterm to remove the mention notation "@" with its username as shown in the Figure 3.23 below.

```
# Remove username with '@' in text
def remove_mentions(self, text):
  cleaned_text = re.sub('@[^\s]+', '', text)
  return cleaned_text
```

Figure 3.23 The Dataset Pre-Processing: Removing Mentions Code

The removing hashtags code uses *replace()* function by passing the regular expression rule to remove the hashtag notation "#" as shown in the Figure 3.24 below.

```
# Remove hashtags '#' in text
def remove_hashtags(self, text):
  cleaned_text = text.replace('#', '')
  return cleaned_text
```

Figure 3.24 The Dataset Pre-Processing: Removing Hashtags Code

The removing non-ASCII characters code uses *normalize()* function from the *unicodedata2* library to change the non-ASCII characters into the normal form as shown in the Figure 3.25 below.

```
# Normalized the accented character in text
def remove_non_ascii(self, text):
    cleaned_text = unicodedata.normalize('NFKD', text).encode('ascii', 'ignore').decode('utf-8',
    'ignore')
    # Remove turncate symbol
    cleaned_text = re.sub('[\.]{3,}', '', cleaned_text)
    return cleaned_text
```

Figure 3.25 The Dataset Pre-Processing: Removing Non-ASCII Characters Code

The changing word to number code uses *word_to_num()* function from the *word2number* library to change each number that written in word to an actual number, but ignoring an actual word as shown in the Figure 3.26 below.

```
# Change the written number in word to actual number in text
def change_word_to_number(self, text):
    changed_text = []
    for word in text.split():
        try:
            changed_text += [str(w2n.word_to_num(word))]
        except ValueError:
            changed_text += [word]
    changed_text = ' '.join(changed_text)
    return changed_text
```

Figure 3.26 The Dataset Pre-Processing: Changing Word to Number Code

The removing numbers code uses *isdigit()* function to remove if it is a digit and ignore the non-digit as shown in the Figure 3.27 below.

```
# Remove numbers in text
def remove_numbers(self, text):
    cleaned_text = ''.join([word for word in text if not word.isdigit()])
    return cleaned_text
```

Figure 3.27 The Dataset Pre-Processing: Removing Numbers Code

The case folding code uses *lower()* function to change all words into lower case as shown in the Figure 3.28 below.

```
# Lower case all words in text
def case_folding(self, text):
    return text.lower()
```

Figure 3.28 The Dataset Pre-Processing: Case Folding Code

The expanding contractions code uses *expand_texts()* function from the *pycontractions* library to expand all the existed contraction words using the pre-trained Google News Word2Vec model as shown in the Figure 3.29 below.

```
# Expand the existing contractions in text
def expand_contractions(self, text, contractions_expander):
    expanded_text = list(contractions_expander.expand_texts([text], precise=True))
    expanded_text = ' '.join(expanded_text)
    return expanded_text
```

Figure 3.29 The Dataset Pre-Processing: Expanding Contractions Code

The getting antonym word code uses several functions from the *NLTK wordnet* library to find the antonym and return the first existed antonym from the current word as shown in the Figure 3.30 below.

```
# Get the correspond antonym to the given word
def get_antonym(self, word):
    word_antonyms = set()
    # Get the synsets to the given word
    for syn in wordnet.synsets(word):
        # Get the correspond lemmas to the given word
        for lemma in syn.lemmas():
            # Get all the antonyms to the given word
            for antonym in lemma.antonyms():
                word_antonyms.add(antonym.name())

    # Get the relevant antonym
    if len(word_antonyms) != 0:
        return word_antonyms.pop()
    else:
        return None
```

Figure 3.30 The Dataset Pre-Processing: Getting Antonym Word Code

The replacing negations code uses the *get_antonym()* function to find the antonym of each negation word that previously has “not” or “no” word and ignore the non-negation words as shown in the Figure 3.31.

```
# Replace the negation words with the antonym in text
def replace_negation(self, text):
    replaced_text = []
    index = 0
    tokenized_text = self.tokenize(text)
    token_length = len(tokenized_text)
    while index < token_length:
        current_word = tokenized_text[index]
        # Check if word is negation
        if (current_word == 'not' or current_word == 'no') and index+1 < token_length:
            current_antonym = self.get_antonym(tokenized_text[index+1])

            # Replace if negation word
            if current_antonym:
                # Store the replaced negation word with the antonym
                replaced_text.append(current_antonym)
                index += 2
                continue
        # Store the non-negation word
        replaced_text.append(current_word)
        index += 1

    replaced_text = ' '.join(replaced_text)

    return replaced_text
```

Figure 3.31 The Dataset Pre-Processing: Replacing Negations Code

The removing punctuations code uses *maketrans()* function from the *string* library to define the translator rule using the *string.punctuation* and uses *translate()* function to remove the each existed punctuation as shown in the Figure 3.32 below.

```
# Remove punctuations in text
def remove_punctuations(self, text):
    english_punctuations = string.punctuation
    translator = str.maketrans('', '', english_punctuations)
    cleaned_text = text.translate(translator)
    return cleaned_text
```

Figure 3.32 The Dataset Pre-Processing: Removing Punctuations Code

The tokenizing sentence code uses *word_tokenize()* function from the *NLTK tokenize* library to split each word in a sentence into token of words as shown in the Figure 3.33 below.

```
# Tokenize sentence into word
def tokenize(self, text):
    tokenized_text = word_tokenize(text)
    return tokenized_text
```

Figure 3.33 The Dataset Pre-Processing: Tokenizing Sentence Code

The removing stopwords code uses *stopwords.words()* from the *NLTK stopwords* library to define the English stopwords dictionary and ignores the exsited stopwords in the English stopwords dictionary as shown in the Figure 3.34 below.

```
# Remove stopwords in text
def remove_stopwords(self, text):
    stop_words = stopwords.words('english')
    cleaned_text =  [word for word in self.tokenize(text) if word not in stop_words]
    cleaned_text = ' '.join(cleaned_text)
    return cleaned_text
```

Figure 3.34 The Dataset Pre-Processing: Removing Stopwords Code

The getting word's POS tag uses *nltk.post_tag()* function from the *NLTK* library to find and return the correct POS tag from the current word as shown in the Figure 3.35 below.

```
# Get the correspond word tag
def get_pos_tag(self,word):
    current_word_tag = nltk.pos_tag([word])[0][1][0].upper()
    tag_map = {'J': wordnet.ADJ,
               'N': wordnet.NOUN,
               'V': wordnet.VERB,
               'R': wordnet.ADV}

    return tag_map.get(current_word_tag, wordnet.NOUN)
```

Figure 3.35 The Dataset Pre-Processing: Getting Word's POS Tag Code

The lemmatizing words code uses *lemmatize()* function from the *NLTK WordNetLemmatizer* library to lemmatize each word based on its POST tag. Then, several words that below two characters as pre-processing residual are removed as shown in the Figure 3.36 below.

```
# Lemmatize each word in text
def lemmatize(self, text):
    lemmatizer = WordNetLemmatizer()
    lematized_text = [lemmatizer.lemmatize(word, self.get_pos_tag(word)) for word in self.tokenize(
    text)]
    lematized_text = [word for word in lematized_text if len(word) > 2]
    lematized_text = ' '.join(lematized_text)
    return lematized_text
```

Figure 3.36 The Dataset Pre-Processing: Lemmatizing Words Code

The implementation of the dataset pre-processing uses *apply()* function from the *pandas* library on each tweet text with each pre-processing function as shown in the Figure 3.37 below as the example taken from the Sinovac dataset pre-processing implementation code.

```
"""## **Sinovac Dataset Pre-Process**"""

df_sinovac['removed_html_tags'] = df_sinovac['text'].apply(lambda sentiment: text_cleaner.remove_html_tags(sentiment))
df_sinovac['removed_retweets'] = df_sinovac['removed_html_tags'].apply(lambda sentiment: text_cleaner.remove_retweets(sentiment))
df_sinovac['removed_urls'] = df_sinovac['removed_retweets'].apply(lambda sentiment: text_cleaner.remove_urls(sentiment))
df_sinovac['removed_mentions'] = df_sinovac['removed_urls'].apply(lambda sentiment: text_cleaner.remove_mentions(sentiment))
df_sinovac['removed_hashtags'] = df_sinovac['removed_mentions'].apply(lambda sentiment: text_cleaner.remove_hashtags(sentiment))
df_sinovac['removed_non_ascii'] = df_sinovac['removed_hashtags'].apply(lambda sentiment: text_cleaner.remove_non_ascii(sentiment))
df_sinovac['changed_word_to_number'] = df_sinovac['removed_non_ascii'].apply(lambda sentiment: text_cleaner.change_word_to_number(sentiment))
df_sinovac['removed_numbers'] = df_sinovac['changed_word_to_number'].apply(lambda sentiment: text_cleaner.remove_numbers(sentiment))
df_sinovac['case_folding'] = df_sinovac['removed_numbers'].apply(lambda sentiment: text_cleaner.case_folding(sentiment))
df_sinovac['expanded_contractions'] = df_sinovac['case_folding'].apply(lambda sentiment: text_cleaner.expand_contractions(sentiment,
contractions_expander))
df_sinovac['replaced_negation'] = df_sinovac['expanded_contractions'].apply(lambda sentiment: text_cleaner.replace_negation(sentiment))
df_sinovac['removed_punctuations'] = df_sinovac['replaced_negation'].apply(lambda sentiment: text_cleaner.remove_punctuations(sentiment))
df_sinovac['removed_stopwords'] = df_sinovac['removed_punctuations'].apply(lambda sentiment: text_cleaner.remove_stopwords(sentiment))
df_sinovac['lemmatized'] = df_sinovac['removed_stopwords'].apply(lambda sentiment: text_cleaner.lemmatize(sentiment))
```

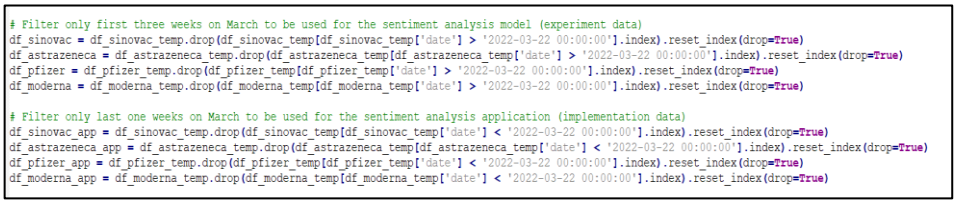
Figure 3.37 The Dataset Pre-Processing: Dataset Implementation Code

The removing duplicated texts code uses *drop_duplicates()* function from the *pandas* library to remove the duplicated tweet texts by keeping the first occurrence as shown in the Figure 3.38 below.

```
# Remove duplicated tweets
df_sinovac_unique = df_sinovac.drop_duplicates(['lemmatized'], keep='first', ignore_index=True)
df_astrazeneca_unique = df_astrazeneca.drop_duplicates(['lemmatized'], keep='first', ignore_index=True)
df_pfizer_unique = df_pfizer.drop_duplicates(['lemmatized'], keep='first', ignore_index=True)
df_moderna_unique = df_moderna.drop_duplicates(['lemmatized'], keep='first', ignore_index=True)
```

Figure 3.38 The Dataset Pre-Processing: Removing Duplicated Text Code

The dataset selection code uses *drop()* function from the *pandas* library to select the experiment dataset on 1 March 2022 until 21 March 2022 period and the implementation dataset on 22 March 2022 until 31 March 2022 period.



```
# Filter only first three weeks on March to be used for the sentiment analysis model (experiment data)
df_sinovac = df_sinovac_temp.drop(df_sinovac_temp[df_sinovac_temp['date'] > '2022-03-22 00:00:00'].index).reset_index(drop=True)
df_astrazeneca = df_astrazeneca_temp.drop(df_astrazeneca_temp[df_astrazeneca_temp['date'] > '2022-03-22 00:00:00'].index).reset_index(drop=True)
df_pfizer = df_pfizer_temp.drop(df_pfizer_temp[df_pfizer_temp['date'] > '2022-03-22 00:00:00'].index).reset_index(drop=True)
df_moderna = df_moderna_temp.drop(df_moderna_temp[df_moderna_temp['date'] > '2022-03-22 00:00:00'].index).reset_index(drop=True)

# Filter only last one weeks on March to be used for the sentiment analysis application (implementation data)
df_sinovac_app = df_sinovac_temp.drop(df_sinovac_temp[df_sinovac_temp['date'] < '2022-03-22 00:00:00'].index).reset_index(drop=True)
df_astrazeneca_app = df_astrazeneca_temp.drop(df_astrazeneca_temp[df_astrazeneca_temp['date'] < '2022-03-22 00:00:00'].index).reset_index(drop=True)
df_pfizer_app = df_pfizer_temp.drop(df_pfizer_temp[df_pfizer_temp['date'] < '2022-03-22 00:00:00'].index).reset_index(drop=True)
df_moderna_app = df_moderna_temp.drop(df_moderna_temp[df_moderna_temp['date'] < '2022-03-22 00:00:00'].index).reset_index(drop=True)
```

Figure 3.39 The Dataset Pre-Processing: Dataset Selection Code

The removing empty values code uses *astype()* function from the pandas library to ignore the empty value that evaluated as *False* for both the experiment dataset and the implementation dataset as shown in the Figure 3.40 below.

```
df_sinovac = df_sinovac[df_sinovac['lemmatized'].astype(bool) == True]
df_astrazeneca = df_astrazeneca[df_astrazeneca['lemmatized'].astype(bool) == True]
df_pfizer = df_pfizer[df_pfizer['lemmatized'].astype(bool) == True]
df_moderna = df_moderna[df_moderna['lemmatized'].astype(bool) == True]

df_sinovac_app = df_sinovac_app[df_sinovac_app['lemmatized'].astype(bool) == True]
df_astrazeneca_app = df_astrazeneca_app[df_astrazeneca_app['lemmatized'].astype(bool) == True]
df_pfizer_app = df_pfizer_app[df_pfizer_app['lemmatized'].astype(bool) == True]
df_moderna_app = df_moderna_app[df_moderna_app['lemmatized'].astype(bool) == True]
```

Figure 3.40 The Dataset Pre-Processing: Removing Empty Value Code

The dataset labeling code uses *polarity_scores()* function from the *vaderSentiment* library function to retrieve the sentiment scores. The sentiment label is determined through the compound value, where the sentiment is positive if the compound value equal or bigger than 0,05; the sentiment is neutral if the compound value is bigger than -0,05 and smaller than 0,05; and the sentiment is negative if the compound value is equal or smaller than -0,05 as shown in the Figure 3.41 below.

```
def get_sentiment_label(text):
    vader = SentimentIntensityAnalyzer()
    vader_result = vader.polarity_scores(text)

    if vader_result['compound'] >= 0.05:
        labeled_sentiment = 'positive'
    elif vader_result['compound'] > -0.05 and vader_result['compound'] < 0.05:
        labeled_sentiment = 'neutral'
    else:
        labeled_sentiment = 'negative'

    return labeled_sentiment
```

Figure 3.41 The Dataset Labeling: Labeling Function Code

The implementation of the dataset labeling code uses *apply()* function from the *pandas* library on each removed hashtag tweet texts, because *vaderSentiment*

library already has contraction and negation handling as shown in the Figure 3.37 below.

```
# Applying VADER sentiment label
df_sinovac['sentiment'] = df_sinovac['removed_hashtags'].apply(lambda sentiment: get_sentiment_label(sentiment))
df_sinovac = df_sinovac[df_sinovac['sentiment'] != 'neutral']
df_sinovac = df_sinovac[['lemmatized', 'sentiment']].rename(columns={'lemmatized': 'text'})

df_astrazeneca['sentiment'] = df_astrazeneca['removed_hashtags'].apply(lambda sentiment: get_sentiment_label(sentiment))
df_astrazeneca = df_astrazeneca[df_astrazeneca['sentiment'] != 'neutral']
df_astrazeneca = df_astrazeneca[['lemmatized', 'sentiment']].rename(columns={'lemmatized': 'text'})

df_pfizer['sentiment'] = df_pfizer['removed_hashtags'].apply(lambda sentiment: get_sentiment_label(sentiment))
df_pfizer = df_pfizer[df_pfizer['sentiment'] != 'neutral']
df_pfizer = df_pfizer[['lemmatized', 'sentiment']].rename(columns={'lemmatized': 'text'})

df_moderna['sentiment'] = df_moderna['removed_hashtags'].apply(lambda sentiment: get_sentiment_label(sentiment))
df_moderna = df_moderna[df_moderna['sentiment'] != 'neutral']
df_moderna = df_moderna[['lemmatized', 'sentiment']].rename(columns={'lemmatized': 'text'})
```

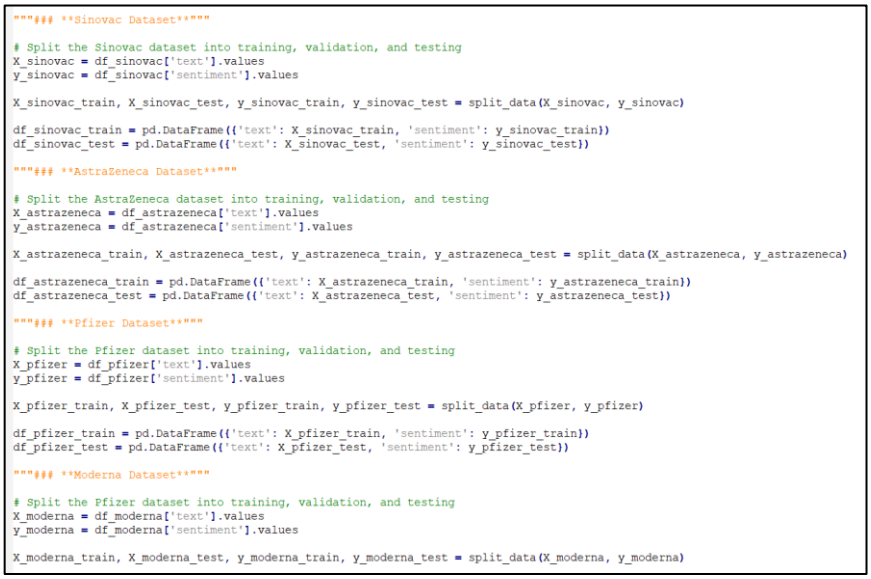
Figure 3.42 The Dataset Labeling: Applying Labeling Function Code

The dataset splitting code uses *train_test_split()* function from the *scikit-learn* library to split 90% training data and 10% testing data with stratified splitting for balance dataset as shown in the Figure 3.43 below.

```
def split_data(X, y):
    X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                        train_size=0.9,
                                                        test_size=0.1,
                                                        stratify=y,
                                                        shuffle=True,
                                                        random_state=0)


    return X_train, X_test, y_train, y_test
```

Figure 3.43 The Dataset Splitting: Splitting Function Code

The implementation of the dataset splitting code uses *split_data()* by passing the text data as the feature and the label data as the target prediction as shown in the Figure 3.43 below.



```
"""### **Sinovac Dataset**"""

# Split the Sinovac dataset into training, validation, and testing
X_sinovac = df_sinovac['text'].values
y_sinovac = df_sinovac['sentiment'].values

X_sinovac_train, X_sinovac_test, y_sinovac_train, y_sinovac_test = split_data(X_sinovac, y_sinovac)

df_sinovac_train = pd.DataFrame({'text': X_sinovac_train, 'sentiment': y_sinovac_train})
df_sinovac_test = pd.DataFrame({'text': X_sinovac_test, 'sentiment': y_sinovac_test})

"""### **AstraZeneca Dataset**"""

# Split the AstraZeneca dataset into training, validation, and testing
X_astrazeneca = df_astrazeneca['text'].values
y_astrazeneca = df_astrazeneca['sentiment'].values

X_astrazeneca_train, X_astrazeneca_test, y_astrazeneca_train, y_astrazeneca_test = split_data(X_astrazeneca, y_astrazeneca)

df_astrazeneca_train = pd.DataFrame({'text': X_astrazeneca_train, 'sentiment': y_astrazeneca_train})
df_astrazeneca_test = pd.DataFrame({'text': X_astrazeneca_test, 'sentiment': y_astrazeneca_test})

"""### **Pfizer Dataset**"""

# Split the Pfizer dataset into training, validation, and testing
X_pfizer = df_pfizer['text'].values
y_pfizer = df_pfizer['sentiment'].values

X_pfizer_train, X_pfizer_test, y_pfizer_train, y_pfizer_test = split_data(X_pfizer, y_pfizer)

df_pfizer_train = pd.DataFrame({'text': X_pfizer_train, 'sentiment': y_pfizer_train})
df_pfizer_test = pd.DataFrame({'text': X_pfizer_test, 'sentiment': y_pfizer_test})

"""### **Moderna Dataset**"""

# Split the Pfizer dataset into training, validation, and testing
X_moderna = df_moderna['text'].values
y_moderna = df_moderna['sentiment'].values

X_moderna_train, X_moderna_test, y_moderna_train, y_moderna_test = split_data(X_moderna, y_moderna)
```

Figure 3.44 The Dataset Splitting: Applying Splitting Function Code

The dataset augmentation code uses several functions from the *textaugment* library such as *synonym_replacement()* function to perform synonym replacement, *random_insertion()* function to perform random insertion, *random_swap()* function to perform random swap, and *random_deletion()* to perform random delete as shown in the Figure 3.45 below.

```
class TextAugmentation:
    def __init__(self):
        self.augmenter = EDA()

    def replace_synonym(self, text):
        augmented_text_portion = max(1, int(len(text)*0.1))
        synonym_replaced = self.augmenter.synonym_replacement(text, n=augmented_text_portion)
        return synonym_replaced

    def random_insert(self, text):
        augmented_text_portion = max(1, int(len(text)*0.1))
        random_inserted = self.augmenter.random_insertion(text, n=augmented_text_portion)
        return random_inserted

    def random_swap(self, text):
        augmented_text_portion = max(1, int(len(text)*0.1))
        random_swaped = self.augmenter.random_swap(text, n=augmented_text_portion)
        return random_swaped

    def random_delete(self, text):
        random_deleted = self.augmenter.random_deletion(text, p=0.1)
        return random_deleted

text_augmentation = TextAugmentation()
```

Figure 3.45 The Dataset Augmentation: EDA Class Code

The implementation of the data augmentation code uses *apply()* function from the *pandas* library on each tweet text with EDA function as shown in the

Figure 3.46 below as the example taken from the Sinovac data augmentation implementation code.

```
# Data augmentation for positive class in Sinovac training data
synonym_replacement_train_positive = df_sinovac_train_positive['text'].apply(lambda text: text_augmentation.replace_synonym(text))
random_insertion_train_positive = df_sinovac_train_positive['text'].apply(lambda text: text_augmentation.random_insert(text))
random_swap_train_positive = df_sinovac_train_positive['text'].apply(lambda text: text_augmentation.random_swap(text))
random_deletion_train_positive = df_sinovac_train_positive['text'].apply(lambda text: text_augmentation.random_delete(text))

# Data augmentation for negative class in Sinovac training data
synonym_replacement_train_negative = df_sinovac_train_negative['text'].apply(lambda text: text_augmentation.replace_synonym(text))
random_insertion_train_negative = df_sinovac_train_negative['text'].apply(lambda text: text_augmentation.random_insert(text))
random_swap_train_negative = df_sinovac_train_negative['text'].apply(lambda text: text_augmentation.random_swap(text))
random_deletion_train_negative = df_sinovac_train_negative['text'].apply(lambda text: text_augmentation.random_delete(text))
```

Figure 3.46 The Dataset Augmentation: Applying EDA Function Code

The encoding label data code encodes the “negative” label into label 0 and the “positive” label into label 1 as shown in the Figure 3.47 below.

```
def encode_label(sentiment):
    if sentiment == 'negative':
        return 0
    elif sentiment == 'positive':
        return 1
```

Figure 3.47 The Sentiment Analysis Model Building: Encoding Label Function Code

The data transformation code uses several functions from the *keras preprocessing* library such as *fit_on_texts()* function tokenizes each word uniquely, *texts_to_sequence()* function encodes the text sequence into sequence of positive integer based, and *pad_sequences()* function pads the sequence into same size. Then, the label training and testing data is encoded using the *encode_label()* function as shown in the Figure 3.48 as the example taken from the Sinovac dataset transformation code.

```
# Tokenize the sequence
sinovac_max_length = 100
sinovac_tokenizer = Tokenizer()
sinovac_tokenizer.fit_on_texts(df_sinovac['text'].values)

X_sinovac_train_tokenized = sinovac_tokenizer.texts_to_sequences(df_sinovac_train['text'].values)
X_sinovac_test_tokenized = sinovac_tokenizer.texts_to_sequences(df_sinovac_test['text'].values)

# Pad the sequence
X_sinovac_train_padded = pad_sequences(X_sinovac_train_tokenized, maxlen=sinovac_max_length)
X_sinovac_test_padded = pad_sequences(X_sinovac_test_tokenized, maxlen=sinovac_max_length)

sinovac_num_words = len(sinovac_tokenizer.word_index) + 1

# Encode label
df_sinovac_train['sentiment'] = df_sinovac_train['sentiment'].apply(lambda x: encode_label(x))
df_sinovac_test['sentiment'] = df_sinovac_test['sentiment'].apply(lambda x: encode_label(x))

y_sinovac_train_category = df_sinovac_train['sentiment'].values
y_sinovac_test_category = to_categorical(df_sinovac_test['sentiment'])

print('The maximum sequence length : ', sinovac_max_length)
print('Total unique words : ', sinovac_num_words)
print('Padded training data : ', X_sinovac_train_padded.shape)
print('Padded testing data : ', X_sinovac_test_padded.shape)
print('Train label size : ', y_sinovac_train_category.shape)
print('Test label size : ', y_sinovac_test_category.shape)
```

Figure 3.48 The Sentiment Analysis Model Building: Data Transformation Code

The sentiment analysis model building code uses *load_word_embedding()* function to load and extract the pre-trained Glove Twitter word embedding weight vectors and uses the *map_word_embedding()* function to map the weight vectors with my own corpus as shown in the Figure 3.49 below.

```
def load_word_embedding(file_name):
    embeddings_weight_vector = {}

    with open(os.path.join('', file_name)) as file:
        for line in file:
            values = line.split();
            word = values[0]
            weights = np.asarray(values[1:], dtype='float32')
            embeddings_weight_vector[word] = weights;

    return embeddings_weight_vector

def map_word_embedding(word_index, num_words, embedding_weight_vectors):
    embedding_matrix = np.zeros((num_words, 200))
    for word, index in word_index.items():
        embedding_vector = embedding_weight_vectors.get(word)
        if embedding_vector is not None:
            embedding_matrix[index] = embedding_vector

    return embedding_matrix
```

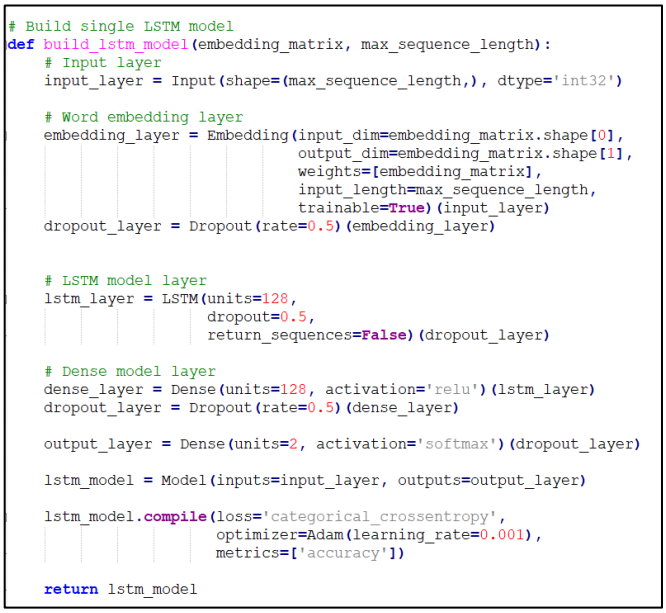
Figure 3.49 The Sentiment Analysis Model Building: The Loading and Mapping Pre-Trained Glove Twitter Word Embedding Function Code

The sentiment analysis model building code for the single CNN model uses several classes from the *keras* library to define and build the input layer, embedding layer, dropout layer, 1D convolution layer, 1D max pooling layer, flatten layer, and dense layer. The single CNN model uses categorical cross entropy loss function and ADAM optimization with 0,001 learning rate as shown in the Figure 3.50 below.

```
# Build single CNN model
def build_cnn_model(embedding_matrix, max_sequence_length):
    # Input layer
    input_layer = Input(shape=(max_sequence_length,))

    # Word embedding layer
    embedding_layer = Embedding(input_dim=embedding_matrix.shape[0],
                                output_dim=embedding_matrix.shape[1],
                                weights=[embedding_matrix],
                                input_length=max_sequence_length,
                                trainable=True)(input_layer)
    dropout_layer = Dropout(rate=0.5)(embedding_layer)


    # CNN model layer
    cnn_layer = Conv1D(filters=64,
                       kernel_size=3,
                       strides=1,
                       activation='relu')(dropout_layer)
    cnn_layer = MaxPooling1D(pool_size=2)(cnn_layer)
    cnn_layer = Dropout(rate=0.5)(cnn_layer)

    flatten = Flatten()(cnn_layer)

    # Dense model layer
    dense_layer = Dense(units=128, activation='relu')(flatten)
    dropout_layer = Dropout(rate=0.5)(dense_layer)

    output_layer = Dense(units=2, activation='softmax')(dropout_layer)

    cnn_model = Model(inputs=input_layer, outputs=output_layer)

    cnn_model.compile(loss='categorical_crossentropy',
                      optimizer=Adam(learning_rate=0.001),
                      metrics=['accuracy'])

    return cnn_model
```

Figure 3.50 The Sentiment Analysis Model Building: The Building Single CNN Model Function Code

The sentiment analysis model building code for the single LSTM model also uses several classes from the *keras* library to define and build the input layer, embedding layer, dropout layer, LSTM layer, and dense layer. The single LSTM model also uses categorical cross entropy loss function and ADAM optimization with 0,001 learning rate as shown in the Figure 3.50 below.



```
# Build single LSTM model
def build_lstm_model(embedding_matrix, max_sequence_length):
    # Input layer
    input_layer = Input(shape=(max_sequence_length,), dtype='int32')

    # Word embedding layer
    embedding_layer = Embedding(input_dim=embedding_matrix.shape[0],
                                output_dim=embedding_matrix.shape[1],
                                weights=[embedding_matrix],
                                input_length=max_sequence_length,
                                trainable=True)(input_layer)
    dropout_layer = Dropout(rate=0.5)(embedding_layer)


    # LSTM model layer
    lstm_layer = LSTM(units=128,
                      dropout=0.5,
                      return_sequences=False)(dropout_layer)

    # Dense model layer
    dense_layer = Dense(units=128, activation='relu')(lstm_layer)
    dropout_layer = Dropout(rate=0.5)(dense_layer)

    output_layer = Dense(units=2, activation='softmax')(dropout_layer)

    lstm_model = Model(inputs=input_layer, outputs=output_layer)

    lstm_model.compile(loss='categorical_crossentropy',
                       optimizer=Adam(learning_rate=0.001),
                       metrics=['accuracy'])

    return lstm_model
```

Figure 3.51 The Sentiment Analysis Model Building: The Building Single LSTM Model Function Code

The sentiment analysis model building code for the hybrid CNN-LSTM model also uses several classes from the *keras* library to define and build the input layer, embedding layer, dropout layer, 1D convolutional layer, 1D max pooling layer, LSTM layer, and dense layer with same parameters from the single CNN model and the single LSTM model. The hybrid CNN-LSTM model also uses categorical cross entropy loss function and ADAM optimization with 0,001 learning rate as shown in the Figure 3.50 below.

```python
# Build hybrid CNN-LSTM model
def build_cnn_lstm_model(embedding_matrix, max_sequence_length):
    # Input layer
    input_layer = Input(shape=(max_sequence_length,))

    # Word embedding layer
    embedding_layer = Embedding(input_dim=embedding_matrix.shape[0],
                                output_dim=embedding_matrix.shape[1],
                                weights=[embedding_matrix],
                                input_length=max_sequence_length,
                                trainable=True)(input_layer)
    dropout_layer = Dropout(rate=0.5)(embedding_layer)

    # CNN model layer
    cnn_layer = Conv1D(filters=64,
                       kernel_size=3,
                       strides=1,
                       activation='relu')(dropout_layer)
    cnn_layer = MaxPooling1D(pool_size=2)(cnn_layer)
    cnn_layer = Dropout(rate=0.5)(cnn_layer)

    # LSTM model layer
    lstm_layer = LSTM(units=128,
                      dropout=0.5,
                      return_sequences=False)(cnn_layer)

    # Dense model layer
    dense_layer = Dense(units=128, activation='relu')(lstm_layer)
    dropout_layer = Dropout(rate=0.5)(dense_layer)

    output_layer = Dense(units=2, activation='softmax')(dropout_layer)

    cnn_lstm_model = Model(inputs=input_layer, outputs=output_layer)

    cnn_lstm_model.compile(loss='categorical_crossentropy',
                           optimizer=Adam(learning_rate=0.001),
                           metrics=['accuracy'])

    return cnn_lstm_model
```

Figure 3.52 The Sentiment Analysis Model Building: The Building Hybrid CNN-LSTM Model Function Code

The sentiment analysis model building code uses visualize_training() function to visualize the training process result of each k-folds such as the training accuracy, the training loss, the validation accuracy, and the validation loss as shown in the Figure 3.53 below.

```python
# Visualize training result
def visualize_training(histories, file_name):

    metrics=['loss', 'accuracy', 'val_accuracy', 'val_loss']
    nrows = (len(metrics)-1) // 2 + 1

    fig, axes = plt.subplots(nrows=nrows, ncols=2, figsize=(16, 16))
    axes = axes.reshape(nrows, 2)

    for index, metric in enumerate(metrics):
        axes[(index+2)//2 - 1, 1 - (index+1)%2].set_title(metric)
        axes[(index+2)//2 - 1, 1 - (index+1)%2].set_xlabel('epochs')
        axes[(index+2)//2 - 1, 1 - (index+1)%2].set_ylabel(metric)

        for history in histories:
            axes[(index+2)//2 - 1, 1 - (index+1)%2].plot(history[metric])
            axes[(index+2)//2 - 1, 1 - (index+1)%2].legend([index+1 for index in range(len(histories))])

    fig.savefig(os.path.join('assets/', file_name))
```

Figure 3.53 The Sentiment Analysis Model Building: The Training Result Visualization Function Code

The sentiment analysis model building code for training the single CNN model uses *split()* function from the *scikit-learn* library to perform shuffled and stratified cross validation with 10 k-folds, the *to_categorical()* function from the *keras* library function to one-hot encode the train and validation label data, the

build_cnn_model() function to build the single CNN model, the *fit()* from the *keras* library to train the single CNN model, and the *visualize_training()* function to visualize the training result as shown in the Figure 3.54 below, as the example taken from the single CNN model training for Sinovac dataset code.

```
# Training single CNN model
sinovac_cnn_histories = []
sinovac_cnn_scores = []

with tf.device('/device:GPU:0'):
    k = 1
    for train, val in kfold.split(X_sinovac_train, y_sinovac_train):
        X_train, X_val = X_sinovac_train[train], X_sinovac_train[val]
        y_train, y_val = to_categorical(y_sinovac_train[train]), to_categorical(y_sinovac_train[val])

        sinovac_cnn_model = build_cnn_model(SINOVAC_EMBEDDING_MATRIX, SINOVAC_MAX_SEQUENCE)

        sinovac_cnn_history = sinovac_cnn_model.fit(x=X_train,
                                                    y=y_train,
                                                    batch_size=64,
                                                    validation_data=(X_val, y_val),
                                                    epochs=20)

        sinovac_cnn_histories.append(sinovac_cnn_history.history)
        sinovac_cnn_scores.append(sinovac_cnn_model.evaluate(X_val, y_val, verbose=0))
        print('\n')
        k += 1

print('Average validation accuracy: ', np.asarray(sinovac_cnn_scores)[:, 1].mean())
print('Average validation loss: ', np.asarray(sinovac_cnn_scores)[:, 0].mean())
sinovac_cnn_model.summary()

# Visualize training result
visualize_training(sinovac_cnn_histories, 'sinovac_training_cnn.jpg')
```

Figure 3.54 The Sentiment Analysis Model Building: The Training Single CNN Model Code

The sentiment analysis model building code for training the single LSTM model uses *split()* function from the *scikit-learn* library to perform shuffled and stratified cross validation with 10 k-folds, the *to_categorical()* function from the *keras* library function to one-hot encode the train and validation label data, the *build_lstm_model()* function to build the single LSTM model, the *fit()* from the *keras* library to train the single LSTM model, and the *visualize_training()* function to visualize the training result as shown in the Figure 3.55 below, as the example taken from the single LSTM model training for Sinovac dataset code.

```
"""### **Training Model: CNN**"""

# Training single CNN model
sinovac_cnn_histories = []
sinovac_cnn_train_scores = []
sinovac_cnn_val_scores = []

with tf.device('/device:GPU:0'):
    k = 1
    for train, val in kfold.split(X_sinovac_train, y_sinovac_train):
        X_train, X_val = X_sinovac_train[train], X_sinovac_train[val]
        y_train, y_val = to_categorical(y_sinovac_train[train]), to_categorical(y_sinovac_train[val])

        sinovac_cnn_model = build_cnn_model(SINOVAC_EMBEDDING_MATRIX, SINOVAC_MAX_SEQUENCE)

        sinovac_cnn_history = sinovac_cnn_model.fit(x=X_train,
                                                    y=y_train,
                                                    batch_size=64,
                                                    validation_data=(X_val, y_val),
                                                    epochs=20)

        sinovac_cnn_histories.append(sinovac_cnn_history.history)
        sinovac_cnn_train_scores.append(sinovac_cnn_model.evaluate(X_train, y_train, verbose=0))
        sinovac_cnn_val_scores.append(sinovac_cnn_model.evaluate(X_val, y_val, verbose=0))
        print('\n')
        k += 1

print('Average train accuracy: {:.2f}%'.format(np.asarray(sinovac_cnn_train_scores)[:, 1].mean()*100))
print('Average train loss: {:.2f}'.format(np.asarray(sinovac_cnn_train_scores)[:, 0].mean()))
print('Average validation accuracy: {:.2f}%'.format(np.asarray(sinovac_cnn_val_scores)[:, 1].mean()*100))
print('Average validation loss: {:.2f}\n'.format(np.asarray(sinovac_cnn_val_scores)[:, 0].mean()))
sinovac_cnn_model.summary()

# Visualize training result
visualize_training(sinovac_cnn_histories, 'sinovac_training_cnn.jpg')
```

Figure 3.55 The Sentiment Analysis Model Building: The Training Single LSTM Code

The sentiment analysis model building code for training the hybrid CNN-LSTM model uses *split()* function from the *scikit-learn* library to perform shuffled and stratified cross validation with 10 k-folds, the *to_categorical()* function from the *keras* library function to one-hot encode the train and validation label data, the *build_cnn_lstm_model()* function to build the single LSTM model, the *fit()* from the *keras* library to train the hybrid CNN-LSTM model, and the *visualize_training()* function to visualize the training result as shown in the Figure 3.56 below, as the example taken from the hybrid CNN-LSTM model training for Sinovac dataset code.

```
"""### **Training Model: CNN-LSTM**"""

# Training hybrid CNN-LSTM model
sinovac_cnn_lstm_histories = []
sinovac_cnn_lstm_scores = []

with tf.device('/device:GPU:0'):
    k = 1
    for train, val in kfold.split(X_sinovac_train, y_sinovac_train):
        X_train, X_val = X_sinovac_train[train], X_sinovac_train[val]
        y_train, y_val = to_categorical(y_sinovac_train[train]), to_categorical(y_sinovac_train[val])

        sinovac_cnn_lstm_model = build_cnn_lstm_model(SINOVAC_EMBEDDING_MATRIX, SINOVAC_MAX_SEQUENCE)

        sinovac_cnn_lstm_history = sinovac_cnn_lstm_model.fit(x=X_train,
                                                              y=y_train,
                                                              batch_size=64,
                                                              validation_data=(X_val, y_val),
                                                              epochs=20)

        sinovac_cnn_lstm_histories.append(sinovac_cnn_lstm_history.history)
        sinovac_cnn_lstm_scores.append(sinovac_cnn_lstm_model.evaluate(X_val, y_val, verbose=0))
        print('\n')
        k += 1

print('Average validation accuracy: ', np.asarray(sinovac_cnn_lstm_scores)[:, 1].mean())
print('Average validation loss: ', np.asarray(sinovac_cnn_lstm_scores)[:, 0].mean())
sinovac_cnn_lstm_model.summary()

# Visualize training result
visualize_training(sinovac_cnn_lstm_histories, 'sinovac_training_cnn_lstm.jpg')
```

Figure 3.56 The Sentiment Analysis Model Building: The Training Hybrid CNN-LSTM Code

The sentiment analysis building code for the test and evaluation trained sentiment analysis models use *visualize_confusion_matrix()* to visualize the confusion matrix and *evaluate_model()* to generate evaluation metrics such as recall, precision, accuracy, and f1-score as shown in the Figure 3.57 below.

```
"""# **Testing & Evaluation Model**"""

def visualize_confussion_matrix(y_true, y_pred, title, file_name):

    classes = ['Negative', 'Positive']
    cm = confusion_matrix(y_true, y_pred)

    cmap = plt.cm.Blues
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], 'd'), horizontalalignment='center', color='white' if cm[i, j] > thresh else 'black')

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')


    plt.savefig(os.path.join('assets/', file_name))
    plt.show()

def evaluate_model(y_true, y_pred):
    target_names = ['Negative', 'Positive']
    classification_eval = metrics.classification_report(y_true, y_pred, target_names=target_names)
    print(classification_eval)
```

Figure 3.57 The Sentiment Analysis Model Building: Visualizing Testing and Evaluation Result Function Code

The sentiment analysis model building code for single CNN model testing and evaluation uses the *evaluate()* function from the *keras* library to test the model with unseen data, the *visualize_confusion_matrix()* function to visualize the confusion matrix, the *evaluate_model()* function to generate the evaluation metrics,

and *save()* function from the *keras* library to save the model as shown in the Figure 3.58 below, as the example for the single CNN model testing and evaluation code on the Sinovac dataset that the rest of the models and the rest of the datasets code for testing and evaluation use the exact same functions and code structures.

```
with tf.device('/device:GPU:0'):
    result = sinovac_cnn_model.evaluate(X_sinovac_test, y_sinovac_test, verbose=1)

print('Accuracy test : {:.2f}%'.format(result[1]*100))
print('Loss test : {:.3}'.format(result[0]))

with tf.device('/device:GPU:0'):
    y_sinovac_pred_raw = sinovac_cnn_model.predict(X_sinovac_test)
    y_sinovac_pred_cnn = np.argmax(y_sinovac_pred_raw, axis=1)
    y_sinovac_true = np.argmax(y_sinovac_test, axis=1)

    visualize_confussion_matrix(y_sinovac_pred_cnn,
                                y_sinovac_true,
                                'CNN Model Confussion Matrix',
                                'sinovac_cnn_confussion.jpg')

    evaluate_model(y_sinovac_pred_cnn, y_sinovac_true)

sinovac_cnn_model.save('model/sinovac_cnn_model.h5')
```

Figure 3.58 The Sentiment Analysis Model Building: The Testing and Evaluation Code

3.4.4 Testing

The testing of the incremental 1 is to test and evaluate the sentiment analysis models performance with the unseen data. The confusion matrix is used to evaluate model through TP (True Positive), FN (False Negative), FP (False Positive), and TN (True Negative) and produce evaluation metrics such as recall, precision, and f1- score. The confusion matrix will be map into correspond sentiment labels that are positive sentiment and negative sentiment.

Table 3.3 The Sentiment Analysis Model Recall, Precision, and F1-Score Metrics Example

Model	***Label***	***Metrics***		
		Recall	***Precision***	***F1-Score***
Hybrid CNN-LSTM	*Negative*			
	Positive			
Average				
Single CNN	*Negative*			
	Positive			
Average				
Single LSTM	*Negative*			

	Positive
Average	

3.5 Incremental 2

In the incremental 2 process, I defined the requirements needed to build the sentiment analysis web application that includes the datasets, the functions, the hardware, and the libraries. Then, I designed the sentiment analysis application function flows and the user interface design. After that, I implemented the process according to the design through coding and testing the sentiment analysis model web application with black-box testing. The output of the incremental 2 is the sentiment analysis web application.

3.5.1 Requirement Analysis

The sentiment analysis web application uses the implementation dataset that already selected from the incremental 1. The experiment dataset consisted from 22 March 2022 until 31 March 2022 period.

The main functions for the sentiment analysis web application are uploading the implementation data, data pre-processing, and displaying sentiment analysis result. The uploading implementation dataset will be used file input field and selection input field, where the file input field is used to upload the implementation dataset and the selection input field is used to choose a sentiment analysis model. The data pre-processing has the exact same processes from the incremental 1 such as case folding or lowercasing all the words in tweets, removing non-ASCII characters, HTML (HyperText Markup Language) tags, removing retweets notation "RT", removing URLs, removing mentions that contain username (notated with "@"), removing hashtag (notated with "#"), changing the written numbers in words into actual numbers, removing numbers, removing punctuations, expanding contractions, replacing negations, removing stopwords, lemmatizing, and tokenizing. Then, All the four uploaded and pre-processed implementation datasets will be implemented with the chosen sentiment analysis model such as the single CNN model, the single LSTM model, or the hybrid CNN-LSTM model. Finally,

the classified four implementation dataset results will be displayed as sentiment analysis summary.

The sentiment analysis web application pages will be consisted of three main pages such as home page, about page, and sentiment analysis page. The home page will be contained an overview definition that briefly explained the topics to the user about the domain. The about page will be explained of all the technologies and tools behind the system. The sentiment analysis page will be contained the main functions of the sentiment analysis web application; it consisted of six subpages. The first subpage is the sentiment analysis welcome page that gives the instruction to the user about how to use the system. The second subpage is the Sinovac sentiment analysis page that allows the user to upload the Sinovac implementation dataset, choose a sentiment analysis model, and process it. After the Sinovac implementation dataset is processed, the system will show the Sinovac sentiment analysis result page to show the sentiment analysis summary to the user. The third subpage is the AstraZeneca sentiment analysis page that allows the user to upload the Sinovac implementation dataset, choose a sentiment analysis model, and process it. After the AstraZeneca implementation dataset is processed, the system will show the AstraZeneca sentiment analysis result page to show the sentiment analysis summary to the user. The fourth subpage is the Pfizer sentiment analysis page that allows the user to upload the Pfizer implementation dataset, choose a sentiment analysis model, and process it. After the Pfizer implementation dataset is processed, the system will show the Pfizer sentiment analysis result page to show the sentiment analysis summary to the user. The fifth subpage is the Moderna sentiment analysis page that allows the user to upload the Moderna implementation dataset, choose a sentiment analysis model, and process it. After the Moderna implementation dataset is processed, the system will show the Moderna sentiment analysis result page to show the sentiment analysis summary to the user. The sixth last page is the back to the home page that will reset all the sentiment analysis result for the user to quit the main function of the system.

Table 3.4 The Hardware Requirement

No.	Type	Name
1	Operating System	Windows 10
2	Processor	Intel i7 9400f 2.90 GHz 6 cores
3	Memory	16GB
4	GPU	NVIDIA GeForce GTX 1050Ti

Table 3.5 The Software Requirement

No.	Name	Type	Version
1	Visual Studio Code	Integrated Development Environment (IDE)	-
2	Python	Programming Language	3.6.9
3	Pipenv	Python Library	2022.4.8
4	pandas	Python Library	1.4.2
5	regex	Python Library	2022.4.2
6	string	Python Library	-
7	NLTK (Natural Language Tool Kit)	Python Library	3.7
8	unicodedata2	Python Library	14.0.0
9	BeautifulSoap	Python Library	4.11.1
10	pycontractions	Python Library	2.0.1
11	word2number	Python Library	1.1
12	tensorflow	Python Library	2.8.0
13	keras	Python Library	2.8.0
14	flask	Python Library	2.1.2
15	flask-wtf	Python Library	1.0.1
16	plotly	Python Library	5.7.0

The Table 3.6 defines all the requirement libraries that will be used in this work. I used *Visual Studio Code* for the *Python* and *flask* IDE, which is, a general code editor. The *Python* programming language will be used throughout the incremental 2 to produce the sentiment analysis web application. The *pipenv* library will be used to create a *Python* virtual environment. The *pandas* library will be used

for reading, filtering, concatenating, and grouping the implementation dataset as dataframe. The *regex* library will be used for performing regular expression on removing retweets, removing hashtags, removing mentions, and removing URLs. The *string* library will be used for removing punctuations on the tweet cleaning process. The *NLTK* library will be used for tokenizing texts, lemmatizing words, removing stopwords, and replacing negation with antonyms. The *unicodedata2* library will be used for removing non-ASCII characters. The *BeautifulSoap* library will be used for removing HTML tags. The *pycontractions* library will be used for expanding contractions. The *word2number* library will be used for replacing written numbers in words to actual numbers. The *tensorflow* and *keras* library will be used for loading the trained sentiment analysis models from the incremental 1 such as the hybrid CNN-LSTM model, the single CNN model, and the single LSTM model. The *flask* library will be used as a Python web development framework. The *flask-wtf* (what the form) library will be used as the *flask* form handling for form processing and form validation. The *plotly* library will be used to visualize the sentiment analysis result with bar plot and pie plot.

3.5.2 Design

The design of the incremental 2 divided into two designs that are the functional flow design and user interface design. The functional flow design is designed with flowchart, depicted all the function flow processes in the sentiment analysis web application. The sentiment analysis web application user interface design will be designed with wireframe design.

3.5.2.1 Function Flow Design

The home page function shows the home page of the sentiment analysis web application, where the flow is shown in the Figure 3.59 below. First, the user clicks the “Home” button at the navigation bar at top of the page. Second, the system renders the home view through defined route. Then, the home page is displayed on the user interface.

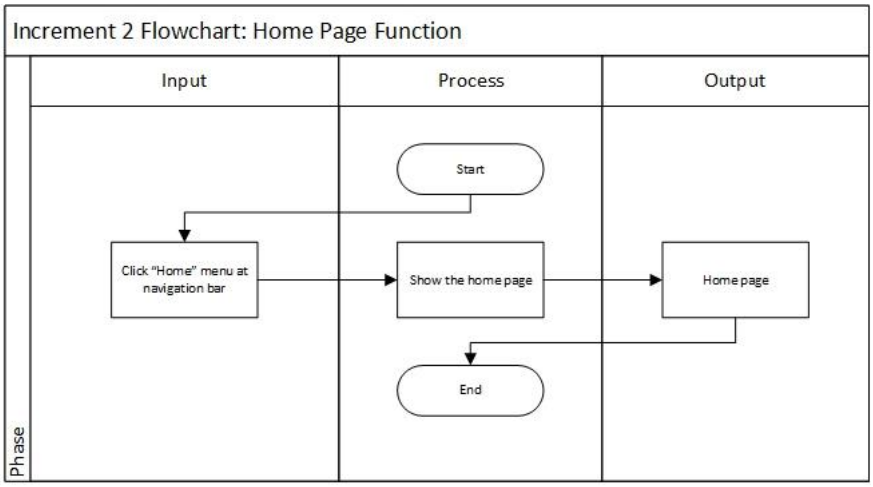


Figure 3.59 The Sentiment Analysis Web Application Home Page Function Flowchart

The about page function shows the about page of the sentiment analysis web application, where the flow is shown in the Figure 3.60 below. First, the user clicks the "About" button at the navigation bar at top of the page. Second, the system renders the about view based on defined route. Then, the about page is displayed on the user interface.

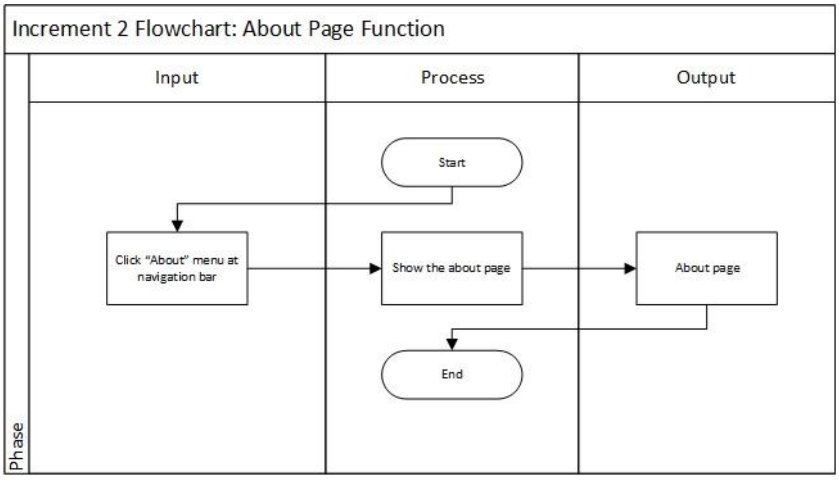


Figure 3.60 The Sentiment Analysis Web Application About Page Function Flowchart

The sentiment analysis function shows the sentiment analysis welcome page of the sentiment analysis web application, where the flow is shown in the Figure 3.61 below. First, the user clicks the "Sentiment Analysis" button at the navigation bar at top of the page. Second, the system renders the sentiment analysis view based on defined route. Then, the sentiment analysis page is displayed on the user interface.

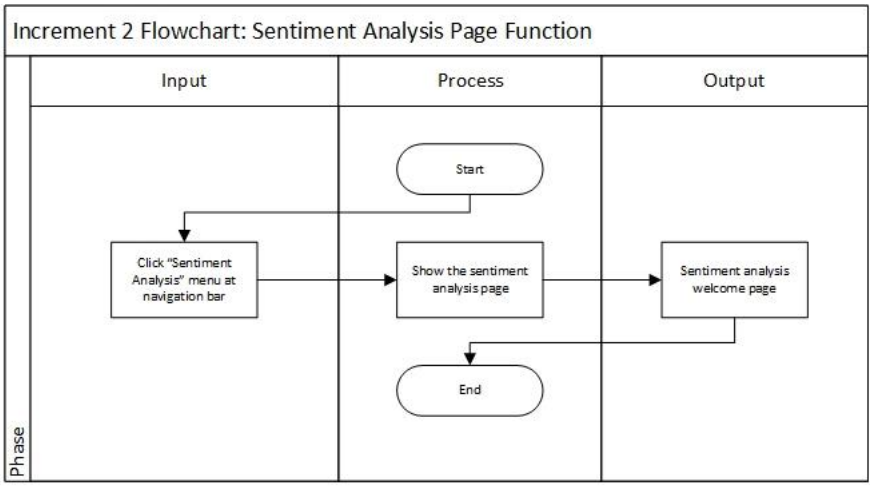


Figure 3.61 The Sentiment Analysis Web Application Sentiment Analysis Page Function Flowchart

The Sinovac sentiment analysis function shows the Sinovac sentiment analysis page of the sentiment analysis web application, where the flow is shown in the Figure 3.62 below. First, the user clicks the “Sinovac” menu at the side bar on the left of the page. Second, the system renders the Sinovac sentiment analysis view through defined route. Third, the sentiment analysis page is displayed on the user interface with form field. Forth, the user uploads the Sinovac implementation dataset through the file input field. Fifth, the user chooses one sentiment analysis model between the hybrid CNN-LSTM, single CNN model, and single LSTM model through the selection input field to perform a sentiment classification. Sixth, the system validates the user inputs, if the user inputs are invalid, then display validation error messages, otherwise, the system performs data pre-processing. Then, the system saves the pre-processed Sinovac implementation dataset in CSV format as the output.

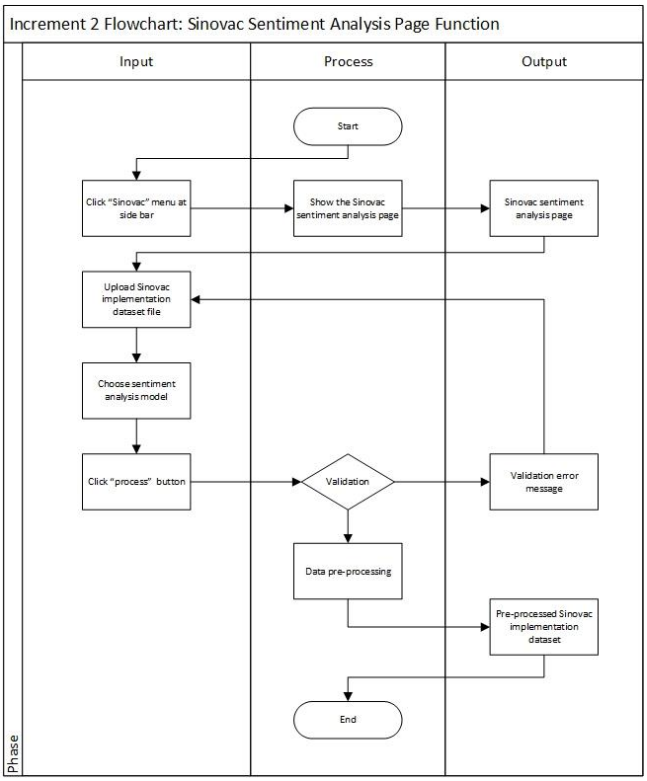


Figure 3.62 The Sentiment Analysis Web Application Sinovac Sentiment Analysis Page Function Flowchart

The Sinovac sentiment analysis result function shows the Sinovac sentiment analysis result page of the sentiment analysis web application, where the flow is shown in the Figure 3.63. First, after the Sinovac implementation dataset has pre-processed and saved successfully, the system immediately reads the saved pre-processed Sinovac implementation dataset. Second, the system tokenizes and pads the sequences in the pre-processed Sinovac implementation dataset. Third, the system loads the chosen sentiment analysis model and performs sentiment classification. Forth, the system decodes the classified sentiment label such as negative sentiment for 0 classified label and positive for 1 classified label. Fifth, the system retrieves sentiment analysis result by filtering, aggregating, grouping, and counting for the sentiment analysis summarization. Sixth, the system renders Sinovac sentiment analysis view based on defined route. Then, the Sinovac sentiment analysis result page is displayed on the user interface with panel data, prediction example results, and sentiment analysis visualizations as the sentiment analysis summary.

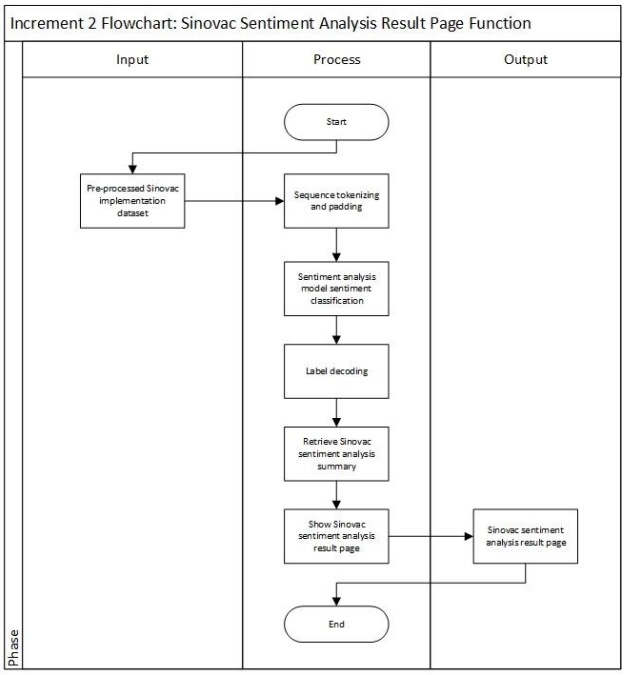


Figure 3.63 The Sentiment Analysis Web Application Sinovac Sentiment Analysis Result Page Function Flowchart

The AstraZeneca sentiment analysis function shows the AstraZeneca sentiment analysis page of the sentiment analysis web application, where the flow is shown in the Figure 3.64 below. First, the user clicks the "AstraZeneca" menu at the side bar on the left of the page. Second, the system renders the AstraZeneca sentiment analysis view through defined route. Third, the sentiment analysis page is displayed on the user interface with form field. Forth, the user uploads the AstraZeneca implementation dataset through the file input field. Fifth, the user chooses one sentiment analysis model between the hybrid CNN-LSTM, single CNN model, and single LSTM model through the selection input field to perform a sentiment classification. Sixth, the system validates the user inputs, if the user inputs are invalid, then display validation error messages, otherwise, the system performs data pre-processing. Then, the system saves the pre-processed AstraZeneca implementation dataset in CSV format as the output.

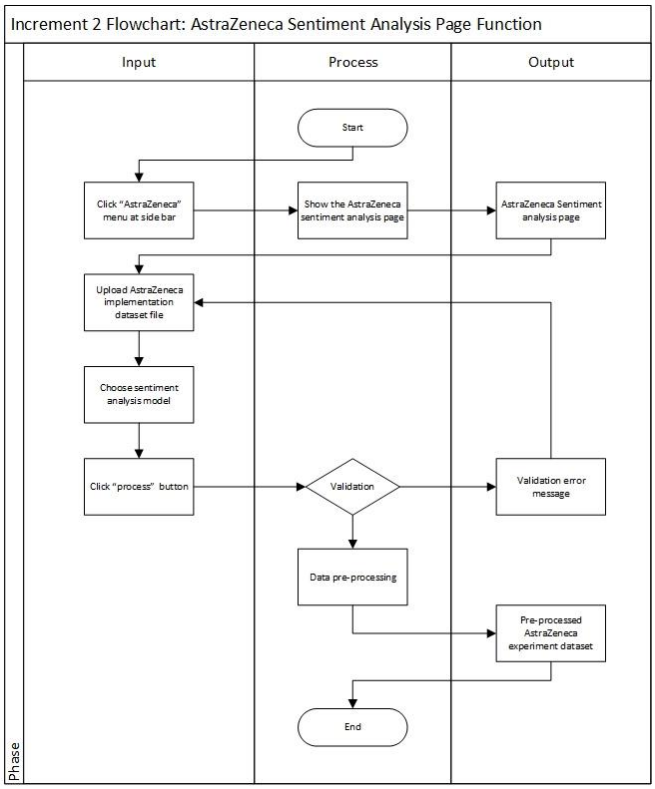


Figure 3.64 The Sentiment Analysis Web Application AstraZeneca Sentiment Analysis Page Function Flowchart

The AstraZeneca sentiment analysis result function shows the AstraZeneca sentiment analysis result page of the sentiment analysis web application, where the flow is shown in the Figure 3.65 below. First, after the AstraZeneca implementation dataset has pre-processed and saved successfully, the system immediately reads the saved pre-processed AstraZeneca implementation dataset. Second, the system tokenizes and pads the sequences in the pre-processed AstraZeneca implementation dataset. Third, the system loads the chosen sentiment analysis model and performs sentiment classification. Forth, the system decodes the classified sentiment label such as negative sentiment for 0 classified label and positive for 1 classified label. Fifth, the system retrieves sentiment analysis result by filtering, aggregating, grouping, and counting for the sentiment analysis summarization. Sixth, the system renders AstraZeneca sentiment analysis view based on defined route. Then, the Sinovac sentiment analysis result page is displayed on the user interface with panel data, prediction results, and sentiment analysis visualizations as the sentiment analysis summary.

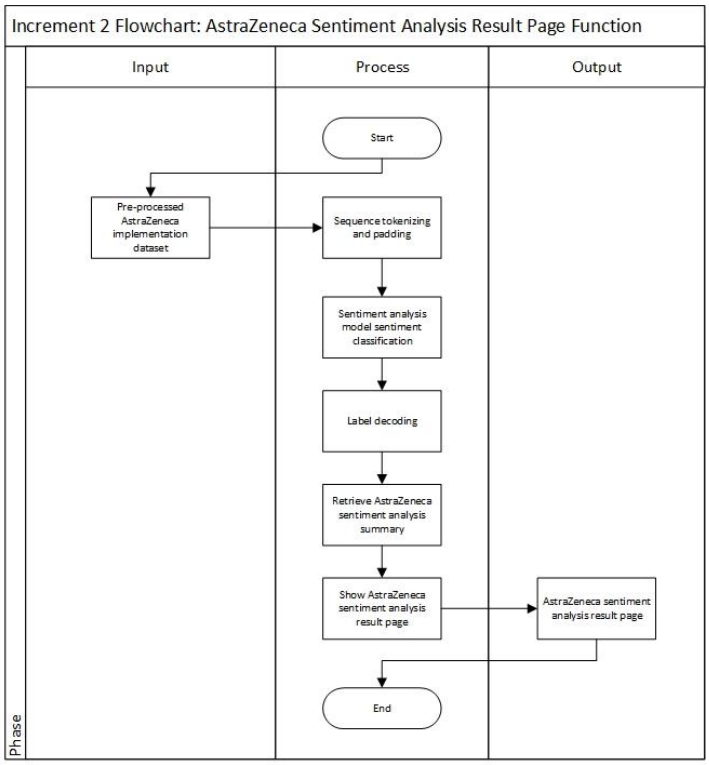


Figure 3.65 The Sentiment Analysis Web Application AstraZeneca Sentiment Analysis Result Page Function Flowchart

The Pfizer sentiment analysis function shows the Pfizer sentiment analysis page of the sentiment analysis web application, where the flow is shown in the Figure 3.66 below. First, the user clicks the "Pfizer" menu at the side bar on the left of the page. Second, the system renders the Pfizer sentiment analysis view through defined route. Third, the sentiment analysis page is displayed on the user interface with form field. Forth, the user uploads the Pfizer implementation dataset through the file input field. Fifth, the user chooses one sentiment analysis model between the hybrid CNN-LSTM, single CNN model, and single LSTM model through the selection input field to perform a sentiment classification. Sixth, the system validates the user inputs, if the user inputs are invalid, then display validation error messages, otherwise, the system performs data pre-processing. Then, the system saves the pre-processed Pfizer implementation dataset in CSV format as the output.

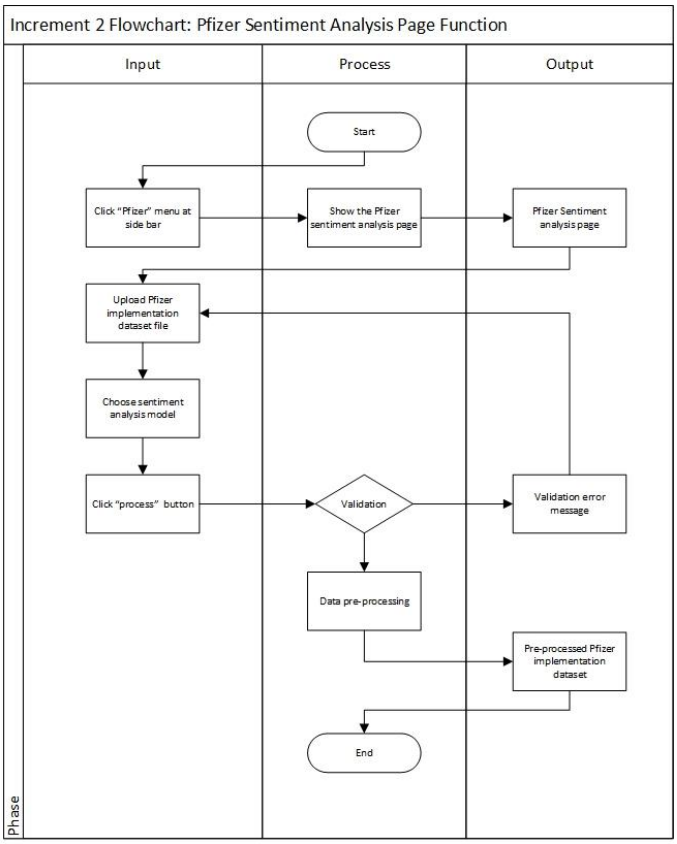


Figure 3.66 The Sentiment Analysis Web Application Pfizer Sentiment Analysis Page Function Flowchart

The Pfizer sentiment analysis result function shows the Pfizer sentiment analysis result page of the sentiment analysis web application, where the flow is shown in the Figure 3.67 below. First, after the Pfizer implementation dataset has pre-processed and saved successfully, the system immediately reads the saved pre-processed Pfizer implementation dataset. Second, the system tokenizes and pads the sequences in the pre-processed Pfizer implementation dataset. Third, the system loads the chosen sentiment analysis model and performs sentiment classification. Forth, the system decodes the classified sentiment label such as negative sentiment for 0 classified label and positive for 1 classified label. Fifth, the system retrieves sentiment analysis result by filtering, aggregating, grouping, and counting for the sentiment analysis summarization. Sixth, the system renders Pfizer sentiment analysis view based on defined route. Then, the Pfizer sentiment analysis result page is displayed on the user interface with panel data, prediction results, and sentiment analysis visualizations as the sentiment analysis summary.

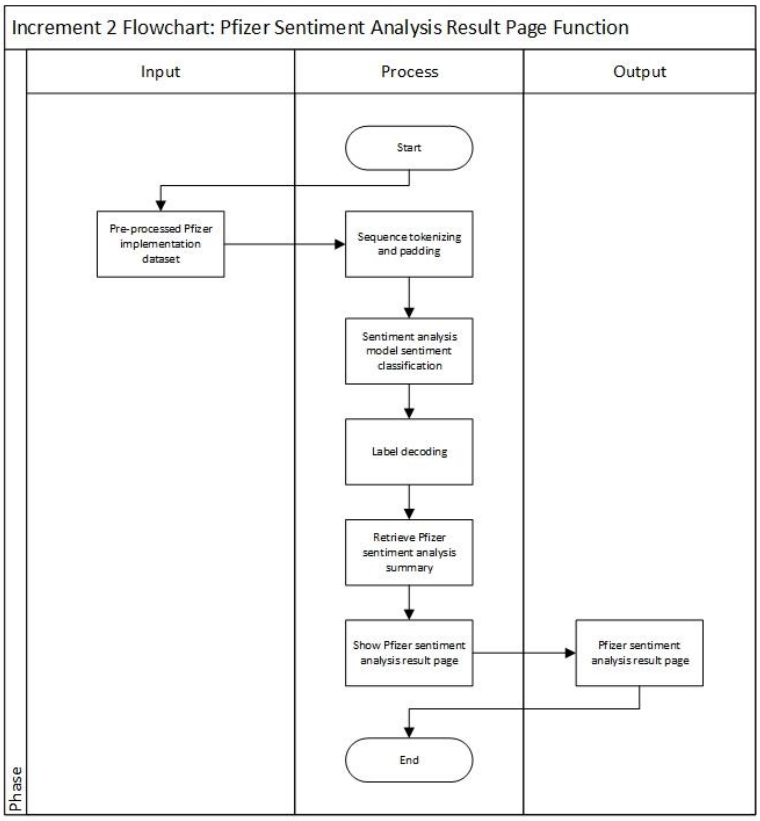


Figure 3.67 The Sentiment Analysis Web Application Pfizer Sentiment Analysis Result Page Function Flowchart

The Moderna sentiment analysis function shows the Moderna sentiment analysis page of the sentiment analysis web application, where the flow is shown in the Figure 3.68 below. First, the user clicks the "Moderna" menu at the side bar on the left of the page. Second, the system renders the Moderna sentiment analysis view through defined route. Third, the sentiment analysis page is displayed on the user interface with form field. Forth, the user uploads the Moderna implementation dataset through the file input field. Fifth, the user chooses one sentiment analysis model between the hybrid CNN-LSTM, single CNN model, and single LSTM model through the selection input field to perform a sentiment classification. Sixth, the system validates the user inputs, if the user inputs are invalid, then display validation error messages, otherwise, the system performs data pre-processing. Finally, the system saves the pre-processed Moderna implementation dataset in CSV format as the output.

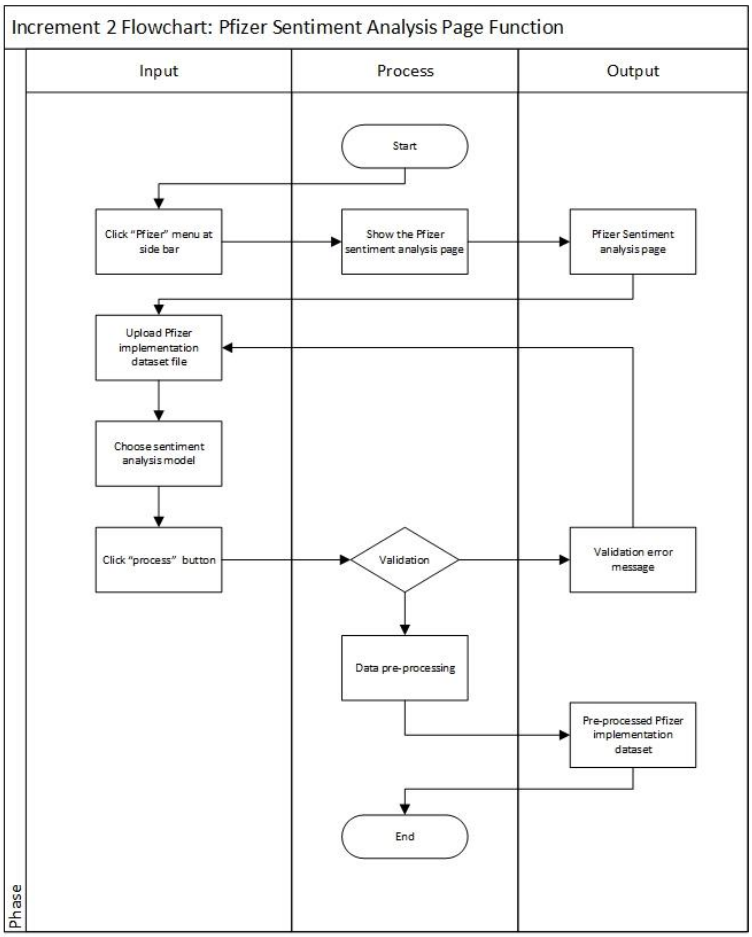


Figure 3.68 The Sentiment Analysis Web Application Moderna Sentiment Analysis Page Function Flowchart

The Moderna sentiment analysis result function shows the Moderna sentiment analysis result page of the sentiment analysis web application, where the flow is shown in the Figure 3.69 below. First, after the Moderna implementation dataset has pre-processed and saved successfully, the system immediately reads the saved pre-processed Moderna implementation dataset. Second, the system tokenizes and pads the sequences in the pre-processed Moderna implementation dataset. Third, the system loads the chosen sentiment analysis model and performs sentiment classification. Forth, the system decodes the classified sentiment label such as negative sentiment for 0 classified label and positive for 1 classified label. Fifth, the system retrieves sentiment analysis result by filtering, aggregating, grouping, and counting for the sentiment analysis summarization. Sixth, the system renders Moderna sentiment analysis view based on defined route. Finally, the

Moderna sentiment analysis result page is displayed on the user interface with panel data, prediction results, and sentiment analysis visualizations as the sentiment analysis summary.

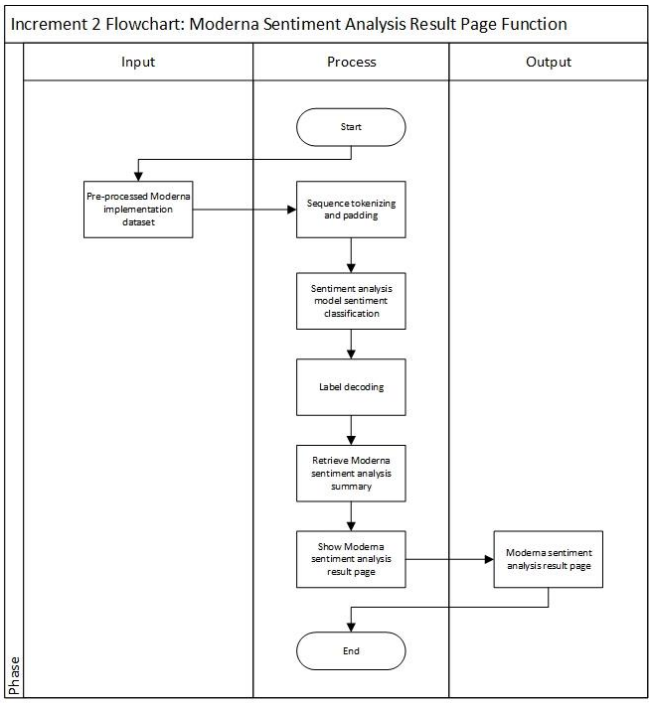


Figure 3.69 The Sentiment Analysis Web Application Moderna Sentiment Analysis Result Page Function Flowchart

The resetting sentiment analysis results function removes all the sentiment analysis summaries and redirect the user to the home page of the sentiment analysis web application, where the flow is shown in the Figure 3.70 below. First, the user clicks the “Back to Home” menu at the side bar on the left of the page. Second, the system shows the quit confirmation modal to alert the user before quitting the sentiment analysis page. Third, if the user chooses “cancel”, then it stays on the sentiment analysis page, otherwise, the system resets all the sentiment analysis results. Forth, the system removes all the pre-processed and uploaded implementation datasets. Fifth, the system renders home view based on defined route. Finally, the home page is displayed on the user interface.

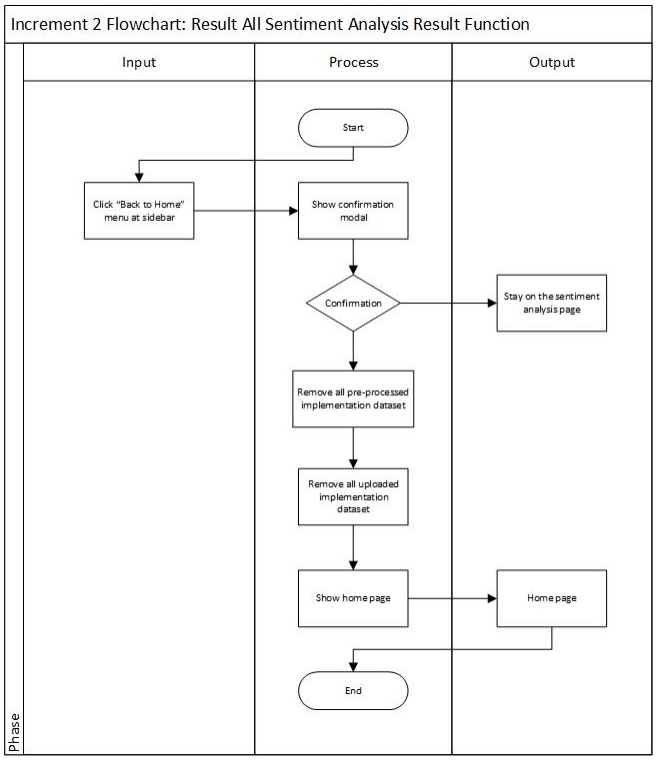


Figure 3.70 The Sentiment Analysis Web Application Reset All Sentiment Analysis Result Function Flowchart

3.5.2.2 User Interface Wireframe Design

The home page in the sentiment analysis web application follows the user interface wireframe design as shown in the Figure 3.71 below that has navigation bar at the top, sliding carousels, contents, and footer at the bottom. The home page content explains and presents about the domains to the user such as NLP, sentiment analysis, Sinovac, AstraZeneca, Pfizer, and Moderna.

Figure 3.71 The Sentiment Analysis Web Application Home Page User Interface Wireframe Design

The about page in the sentiment analysis web application follows the user interface wireframe design as shown in the Figure 3.72 below that has navigation bar at the top, sliding carousels, contents, and footer at the bottom. The about page content explains and presents about the technologies and tools behind the sentiment analysis web application to the user such as AI, NLP, Twitter data, data augmentation, *flask* framework, and *plotly* visualization.

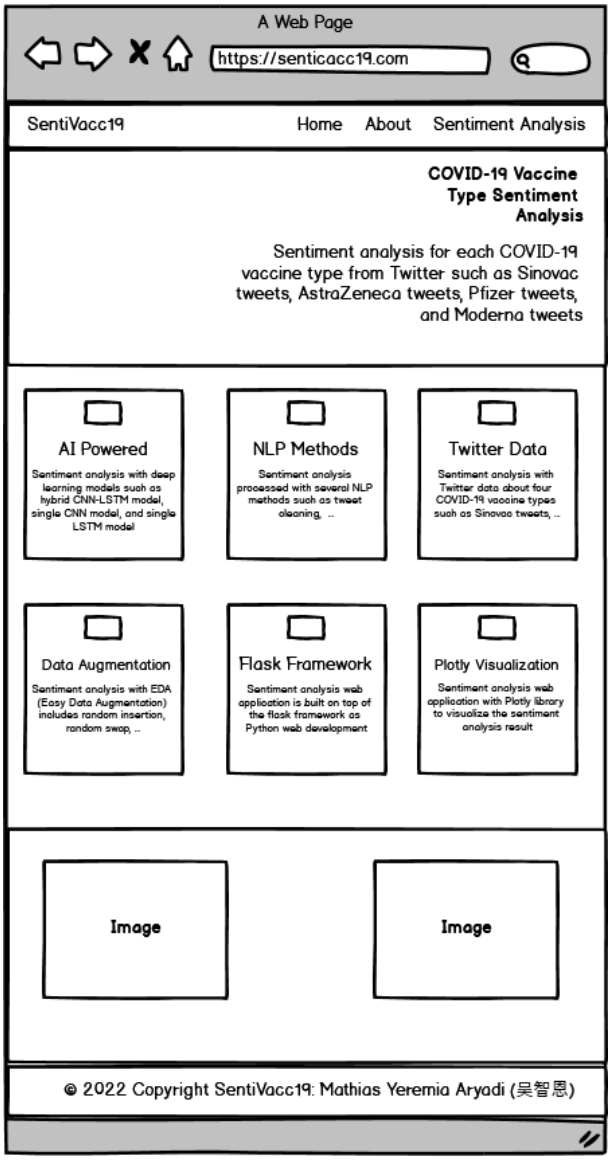


Figure 3.72 The Sentiment Analysis Web Application About Page User Interface Wireframe Design

The sentiment analysis page in the sentiment analysis web application follows the user interface wireframe design as shown in the Figure 3.73 below that has side bar on the left, sliding carousels, content, and footer at the bottom. The sentiment analysis page content explains and presents about the instructions to use the sentiment analysis web application to the user.

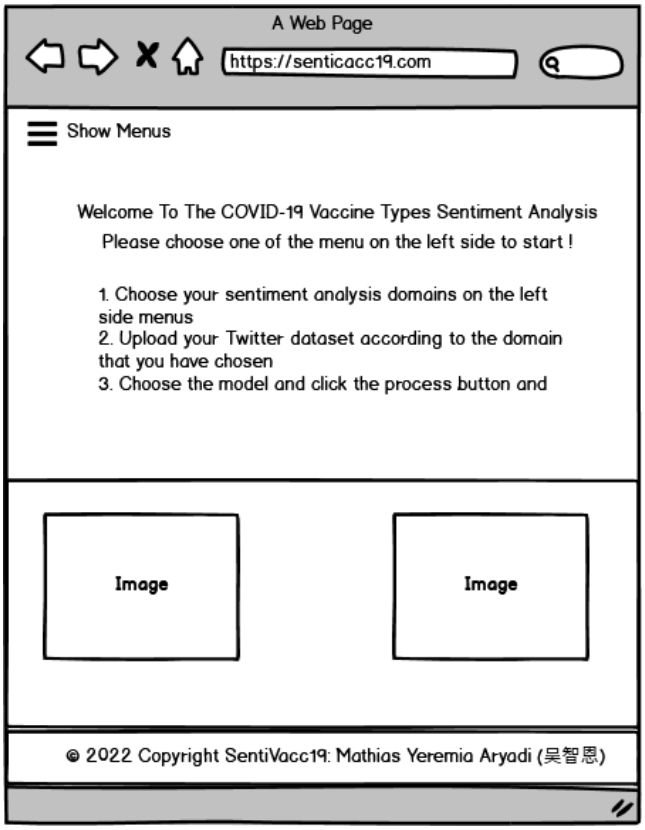


Figure 3.73 The Sentiment Analysis Web Application Sentiment Analysis Page User Interface Wireframe Design

The sentiment analysis page with opened side bar in the sentiment analysis web application follows the user interface wireframe design as shown in the Figure 3.74 below. The side bar shows all the subpages in the sentiment analysis page such as welcome page, Sinovac sentiment analysis page, AstraZeneca sentiment analysis page, Pfizer sentiment analysis page, Moderna sentiment analysis page, and reset all sentiment analysis result menu.

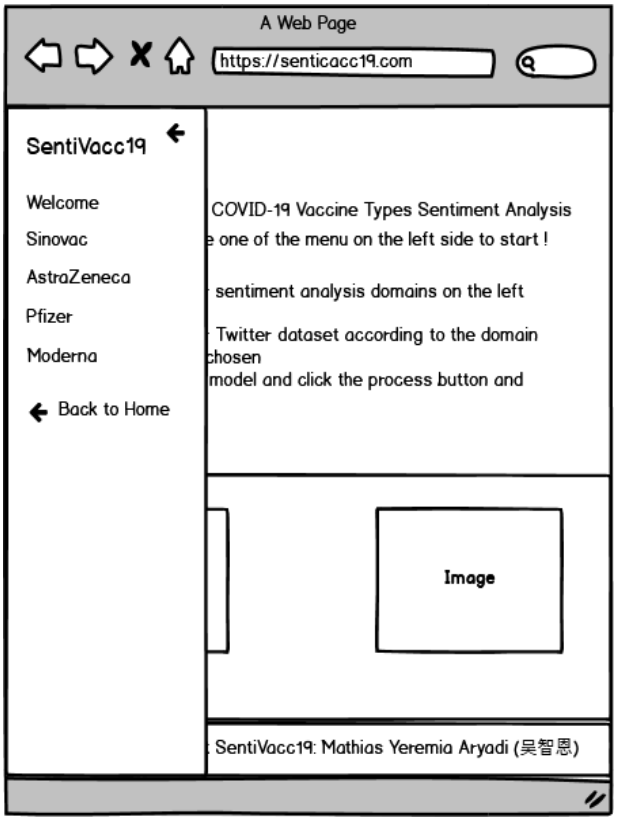


Figure 3.74 The Sentiment Analysis Web Application Sentiment Analysis Page with Opened Sidebar User Interface Wireframe Design

Each topic sentiment analysis page follows the user interface wireframe design as shown in the Figure 3.75 below, because they have the exact same component that is the form field with file input field, selection input field, and submit button. The user uploads an implementation dataset topic file through the file input field and select a sentiment analysis model through the selection input field.

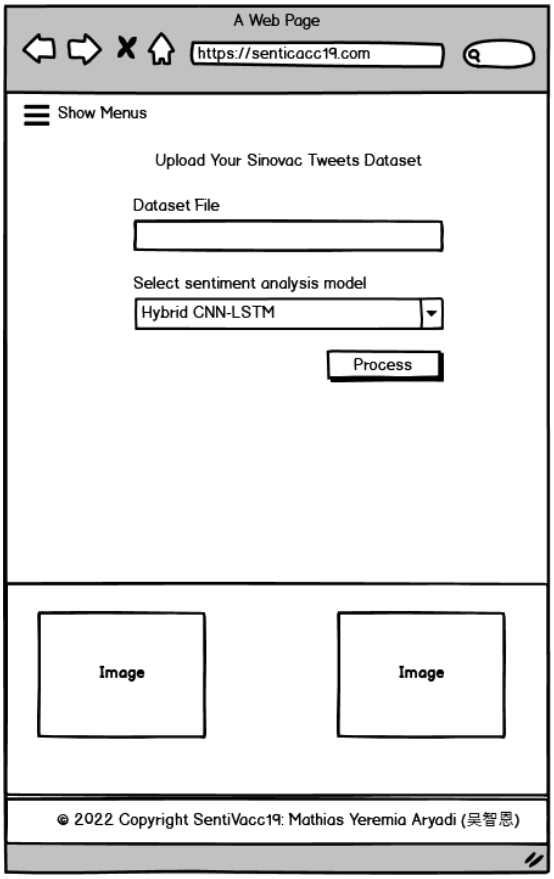


Figure 3.75 The Sentiment Analysis Web Application Sentiment Analysis Page User Interface Wireframe Design

Each topic sentiment analysis result page follows the user interface wireframe design as shown in the Figure 3.76 below, because they have the exact same components that are three data panels, classification table result, and classified sentiment visualization. The three data panels show the number of tweets, the number of classified positive tweets, and the number of classified negative tweets according to the topic sentiment analysis page. The classification table result shows the classification result in table with ten row examples and two columns for the raw tweet texts and the classified sentiment. The classified sentiment visualization visualizes the sentiment analysis results in bar plot on the left side and pie plot on the right side.

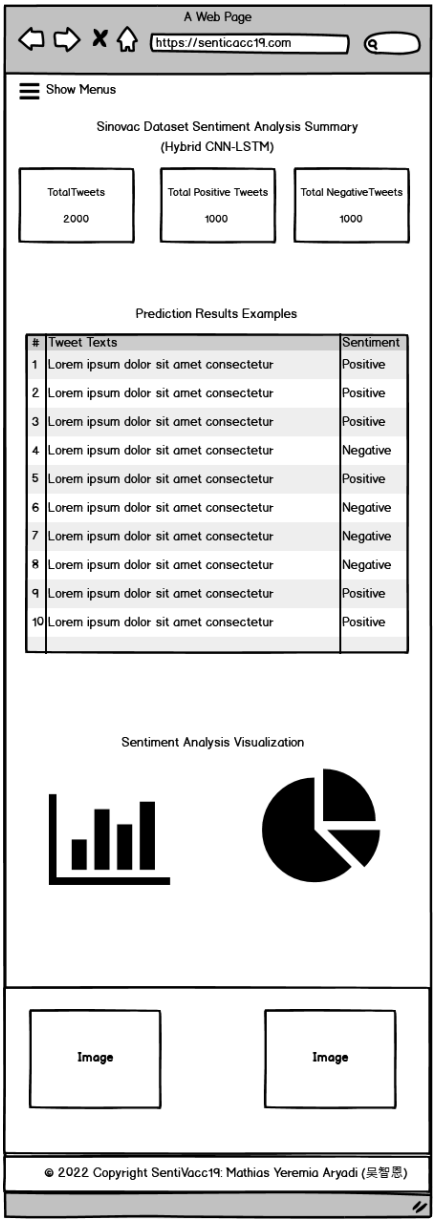


Figure 3.76 The Sentiment Analysis Web Application Sinovac Sentiment Analysis Result Page User Interface Wireframe Design

The resetting all sentiment analysis results in the sentiment analysis web application follows the user interface wireframe design as shown in the Figure 3.77 below that the user can quit the main sentiment analysis function to the home page through the appeared modal confirmation and reset or remove all the sentiment analysis results.

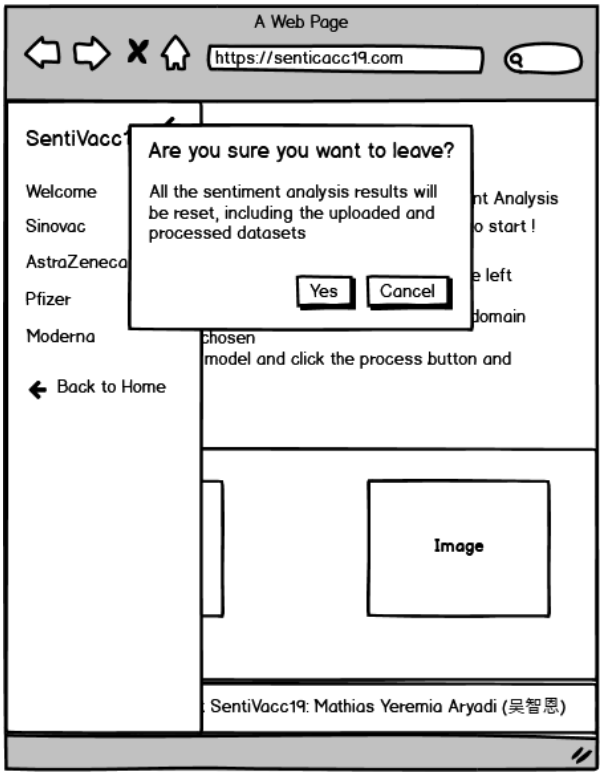


Figure 3.77 The Sentiment Analysis Web Application Reset All Sentiment Analysis Confirmation User Interface Wireframe Design

3.5.3 Implementation

The *flask* project structure of the sentiment analysis web application has several directories as shown in the Figure 3.78 below. The *datasets* directory contains all the uploaded and pre-processed implementation datasets in CSV format. The *models* directory contains all the trained sentiment analysis models such as the hybrid CNN-LSTM model, the single CNN model, and the single LSTM model in H5 format for four dataset topics. The *static* directory contains all the static resource for the web application such as *CSS*, *JavaScript*, and images. The *templates* directory contains all the sentiment analysis web application pages in HTML file. The *app.py* file contains all the *flask* main web application logic code such as routes, form, and handling the implementation dataset. The *pre_process.py* contains all the data pre-processing functions.

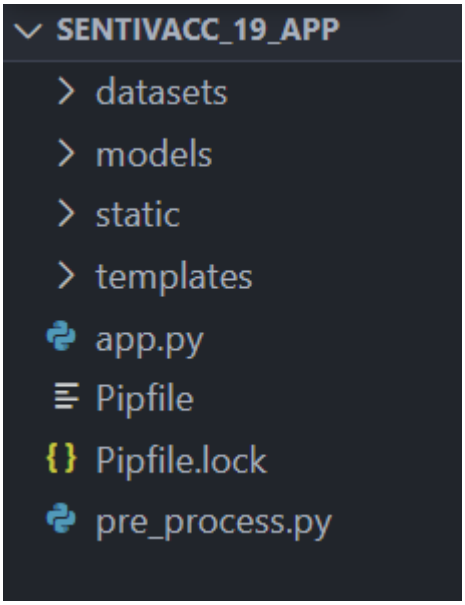


Figure 3.78 The Sentiment Analysis Web Application Flask Web Project Structure

The model selection and prediction function code use several functions to set the selected sentiment analysis model, set the selected topic dataset, transform the dataset, and classify the sentiment as shown in the Figure 3.79 below. First, the selected model text is set based on the chosen sentiment analysis model. Second, the sentiment analysis model is loaded according to the dataset topic. Third, the pre-processed implementation dataset is transformed into token and sequence of positive integers using *transform_data()* function so the model can predict with the proper data form. Forth, the predicted sentiments using *predict()* function are set to the implementation dataset new column and decode the predicted label into "Positive" and "Negative" using *decode_label()* function. Then, the implementation dataset is filtered by taking the raw tweet texts column and the predicted sentiments column to be shown in the sentiment analysis result page.

```
def select_model_predict(df):
    selected_model_text = ''
    model = None

    if df['model'].iloc[0] == 'cnn-lstm':
        selected_model_text = 'Hybrid CNN-LSTM'
        file_name = df['topic'].iloc[0] + '_cnn_lstm_model.h5'
        model = load_model(os.path.join(app.root_path, 'models', file_name))
        X_test = transform_data(df)
        y_pred_raw = model.predict(X_test)
        y_pred = np.argmax(y_pred_raw, axis=1)
        df['sentiment'] = y_pred
        df['sentiment'] = df['sentiment'].apply(lambda x: dencode_label(x))
        df = df[['text', 'sentiment']]
    elif df['model'].iloc[0] == 'cnn':
        selected_model_text = 'Single CNN Model'
        file_name = df['topic'].iloc[0] + '_cnn_model.h5'
        model = load_model(os.path.join(app.root_path, 'models', file_name))
        X_test = transform_data(df)
        y_pred_raw = model.predict(X_test)
        y_pred = np.argmax(y_pred_raw, axis=1)
        df['sentiment'] = y_pred
        df['sentiment'] = df['sentiment'].apply(lambda x: dencode_label(x))
        df = df[['text', 'sentiment']]
    else:
        selected_model_text = 'Single LSTM Model'
        file_name = df['topic'].iloc[0] + '_lstm_model.h5'
        model = load_model(os.path.join(app.root_path, 'models', file_name))
        X_test = transform_data(df)
        y_pred_raw = model.predict(X_test)
        y_pred = np.argmax(y_pred_raw, axis=1)
        df['sentiment'] = y_pred
        df['sentiment'] = df['sentiment'].apply(lambda x: dencode_label(x))
        df = df[['text', 'sentiment']]

    return df, selected_model_text
```

Figure 3.79 The Sentiment Analysis Web Application Sentiment Analysis Model Implementation Code

The sentiment analysis web application for implementing the home page code uses *render_index_view()* function to show the home page as *flask* route handler function with "/" URL and *GET* request method as shown in the Figure 3.80 below.

```
# Home view
@app.route('/', methods=['GET'])
def render_index_view():
    title = 'SentiVacc19 | Home'
    return render_template('index.html', title=title)
```

Figure 3.80 The Sentiment Analysis Web Application Home Page Route Code

The sentiment analysis web application for implementing the about page code uses *render_about_view()* function as *flask* route handler function with "/about" URL and *GET* request method as shown in the Figure 3.81 below.

```python
# About view
@app.route('/about', methods=['GET'])
def render_about_view():
    title = 'SentiVacc19 | About'
    return render_template('about.html', title=title)
```

Figure 3.81 The Sentiment Analysis Web Application About Page Route Code

The sentiment analysis web application for implementing the sentiment analysis page code uses *render_sentiment_analysis_view()* function as *flask* route handler function with "/sentiment-analysis" URL and *GET* request method as shown in the Figure 3.82 below.

```python
# Sentiment analysis views
@app.route('/sentiment-analysis', methods=['GET'])
def render_sentiment_analysis_view():
    title = 'SentiVacc19 | Sentiment Analysis'
    return render_template('sentiment-analysis.html', title=title)
```

Figure 3.82 The Sentiment Analysis Web Application Sentiment Analysis Page Route Code

Each topic sentiment analysis page code has the exact same code structure such as checking the session (if the sentiment analysis result is already existed), validating the form fields, uploaded dataset pre-processing, topic and model selection. In addition, the *flask* route handler function with its URL endpoint follows the topic name, for examples, *render_sinovac_view()* function with URL "/sentiment-analysis/sinovac" and *render_astrazeneca_view()* function with URL "/sentiment-analysis/astrazeneca". In contrast, they are use *GET* and *POST* request methods, because this page also has form processing as shown in the Figure 3.83 below, as the example taken from the Sinovac sentiment analysis page code.

```python
# Sinovac views
@app.route('/sentiment-analysis/sinovac', methods=['GET', 'POST'])
def render_sinovac_view():
    if 'sinovac_session' in session:
        return redirect(url_for('render_sinovac_result_view'))

    title = 'SentiVacc19 | Sentiment Analysis: Sinovac'
    form = SentimentForm()

    if form.validate_on_submit():
        f = form.dataset_file_field.data
        filename = secure_filename(f.filename)
        f.save(os.path.join(app.root_path, 'datasets', filename))

        selected_model = form.model_select_field.data
        df_sinovac = pd.read_csv(os.path.join(app.root_path, 'datasets', filename))
        df_sinovac = set_sentiment_analysis_model(selected_model, df_sinovac)
        df_sinovac = set_topic('sinovac', df_sinovac)
        pre_process(df_sinovac, 'sinovac_cleaned.csv')
        session['sinovac_session'] = 1

        return redirect(url_for('render_sinovac_result_view'))

    return render_template('pages/sinovac.html', title=title, form=form)
```

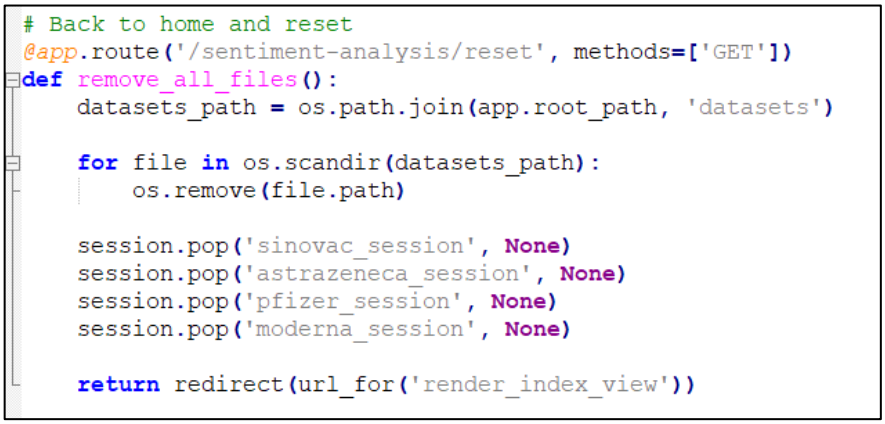
Figure 3.83 The Sentiment Analysis Web Application Sentiment Analysis Page Route Code

Each topic sentiment analysis result page code also has the exact same code structure such as checking the session (if the sentiment analysis result is not existed), the loading the chosen sentiment analysis model, sentiment class cation, sentiment analysis classification result filtering and aggregating, and sentiment analysis result visualization. In addition, the *flask* route handler function with its URL endpoint follows the topic name and use *GET* request method, for examples, *render_sinovac_result_view()* function with URL "/sentiment-analysis/sinovac/result" and *render_astrazeneca_result_view()* function with URL "/sentiment-analysis/astrazeneca/result", as shown in the Figure 3.84 below, as the example taken from the Sinovac sentiment analysis result page code.

```
# Result
@app.route('/sentiment-analysis/sinovac/result', methods=['GET'])
def render_sinovac_result_view():
    if 'sinovac_session' not in session:
        return redirect(url_for('render_sinovac_view'))

    title = 'SentiVacc19 | Sentiment Analysis: Sinovac Result'
    df_sinovac = pd.read_csv(os.path.join(app.root_path, 'datasets', 'sinovac_cleaned.csv'))

    df_sinovac, selected_model_text = select_model_predict(df_sinovac)
    sinovac_data = df_sinovac.copy()
    sinovac_count = sinovac_data['sentiment'].value_counts().rename_axis('sentiment').reset_index(name='count')
    sinovac_tweets = sinovac_data.shape[0]
    sinovac_positive = sinovac_count[sinovac_count['sentiment'] == 'Positive']['count'].values[0]
    sinovac_negative = sinovac_count[sinovac_count['sentiment'] == 'Negative']['count'].values[0]
    sinovac_data = sinovac_data.head(10)
    sinovac_bar_plot = plot_bar(sinovac_count)
    sinovac_pie_plot = plot_pie(sinovac_count)

    return render_template(
        'pages/sinovac-result.html',
        title=title,
        selected_model_text=selected_model_text,
        sinovac_tweets=sinovac_tweets,
        sinovac_positive=sinovac_positive,
        sinovac_negative=sinovac_negative,
        sinovac_data=sinovac_data,
        sinovac_bar_plot=sinovac_bar_plot,
        sinovac_pie_plot=sinovac_pie_plot
    )
```

Figure 3.84 The Sentiment Analysis Web Application Sentiment Analysis Result Page Route Code

The sentiment analysis web application for implementing the resetting sentiment analysis result code uses *remove_all_files()* function as *flask* route handler function with "/sentiment-analysis/reset" URL and *GET* request method as shown in the Figure 3.85 below.



```
# Back to home and reset
@app.route('/sentiment-analysis/reset', methods=['GET'])
def remove_all_files():
    datasets_path = os.path.join(app.root_path, 'datasets')

    for file in os.scandir(datasets_path):
        os.remove(file.path)

    session.pop('sinovac_session', None)
    session.pop('astrazeneca_session', None)
    session.pop('pfizer_session', None)
    session.pop('moderna_session', None)

    return redirect(url_for('render_index_view'))
```

Figure 3.85 The Sentiment Analysis Web Application Reset All Sentiment Analysis Result Route Code

The sentiment analysis web application for the home page HTML code shows several description or definition about the domains such as NLP, sentiment analysis, Sinovac, AstraZeneca, Pfizer, and Moderna as shown in the Figure 3.86 and Figure 3.87 below.

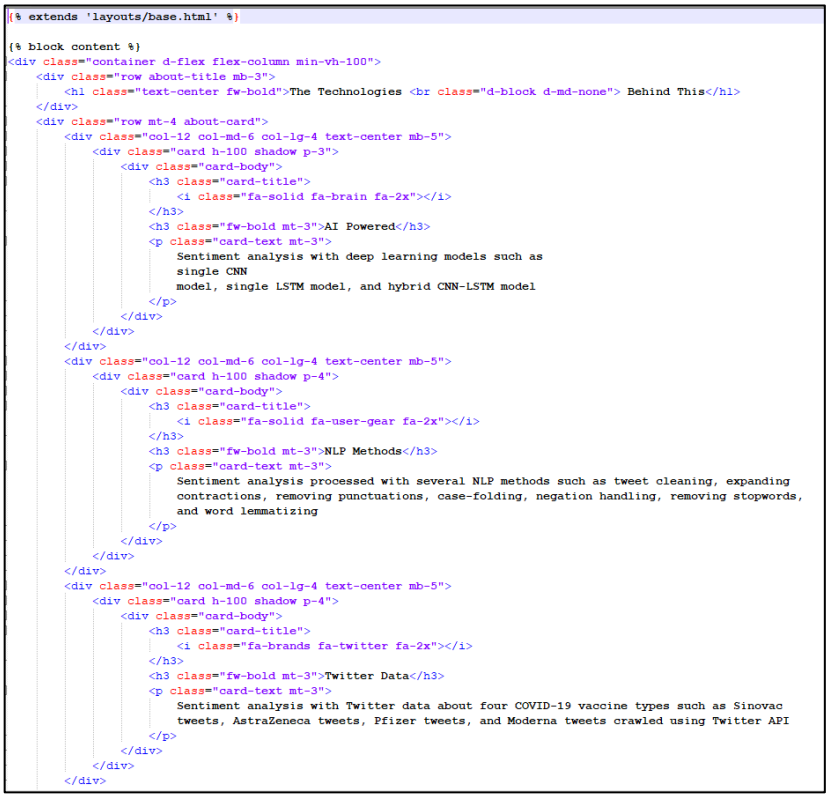
```
{% extends 'layouts/base.html' %}

{% block content %}
<div class="container mb-5">
    <div class="row gy-5 mt-5">
        <div class="col-12 col-md-6 text-center text-md-start">
            <h1 class="fw-bold">Natural Language Processing</h1>
            <p class="fs-4 px-3 px-md-0">
                Refers to the branch of computer science—and more specifically, the branch of artificial
                intelligence or AI—concerned with giving computers the ability to understand text and spoken
                words in much the same way human beings can.
            </p>

        </div>
        <div class="col-12 col-md-6">
            <img src="{{ url_for('static', filename='img/natural-language-processing.png') }}"
                class="img-fluid d-none d-md-block" alt="natural-language-processing">
        </div>
    </div>

    <div class="row gy-5 mt-md-5">
        <div class="col-12 col-md-6 justify-content-center d-none d-md-block">
            <img src="{{ url_for('static', filename='img/sentiment-analysis.jpg') }}" class="img-fluid"
                alt="sentiment-analysis">
        </div>
        <div class="col-12 col-md-6 text-center text-md-end">
            <h1 class="fw-bold">Sentiment Analysis</h1>
            <p class="fs-5 px-3 px-md-0 mt-3">
                Sentiment analysis, also called opinion mining, is the field of study that
                analyzes people's opinions, sentiments, evaluations, appraisals, attitudes,
                and emotions towards entities such as products, services, organizations,
                individuals, issues, events, topics, and their attributes.
            </p>
        </div>
    </div>
</div>

<div class="container middle-carousel">
    <div class="row">
        <div class="col-12 text-center">
            <h1 class="fw-bold">Which are types of COVID-19 vaccine topics <br> that used in this
                sentiment analysis?</h1>
            <p class="fs-4 px-3 px-md-0 mt-3 mt-md-3">
                The sentiment analysis topics are about <br> Sinovac vaccine, AstraZeneca
                    vaccine, Pfizer vaccine, and Moderna vaccine
            </p>
        </div>
    </div>
</div>
```

Figure 3.86 The Sentiment Analysis Web Application Home Page Code Part 1

```
<div class="vaccine-section bg-dark d-flex flex-column min-vh-100 bg-dark text-white pb-5">
    <div class="container">
        <div class="row gy-5 mt-5">
            <div class="col-12 text-center">
                <h1 class="fw-bold">What is Sinovac Vaccine?</h1>
                <p class="fs-4 px-3 px-md-5 mt-3 mt-md-5">
                    Sinovac is a vaccine produced by a Chinese biopharmaceutical company, Sinovac BioTech. This vaccine,
                    which has another name, CoronaVac, is a type of whole virus vaccine that uses the inactivated
                    SARS-CoV-2 virus. The inactivated virus is no longer able to infect the body, but it can trigger the
                    formation of immunity in our body.
                </p>
            </div>
            <div class="col-12 text-center">
                <h1 class="fw-bold">What is AstraZeneca Vaccine?</h1>
                <p class="fs-4 px-3 px-md-5 mt-3 mt-md-5">
                    The AstraZeneca or Oxford-AstraZeneca vaccine is a vaccine produced by a British biopharmaceutical
                    company and the University of Oxford. This vaccine is a type of viral vector vaccine that utilizes
                    the chimpanzee Adenovirus (which is weakened until it's harmless), to deliver the spike protein from
                    Covid-19 into our cells and trigger it to build antibodies.
                </p>
            </div>

            <div class="col-12 text-center">
                <h1 class="fw-bold">What is Pfizer Vaccine?</h1>
                <p class="fs-4 px-3 px-md-5 mt-3 mt-md-5">
                    The Pfizer-BioNTech COVID-19 vaccine also contains a harmless piece of messenger RNA (mRNA). The
                    COVID-19 mRNA teaches cells in the body how to create an immune response to the virus that causes
                    COVID-19. This response helps protect you from getting sick with COVID-19 in the future.
                </p>
            </div>

            <div class="col-12 text-center mb-5">
                <h1 class="fw-bold">What is Moderna Vaccine?</h1>
                <p class="fs-4 px-3 px-md-5 mt-3 mt-md-5">
                    Moderna uses the same mRNA technology as Pfizer-BioNTech and had a similarly high efficacy at preventing symptomatic disease when the
                    companies applied for authorization; it also needs to be stored in freezer-level temperatures.
                </p>
            </div>
        </div>
    </div>
</div>

{% endblock %}
```

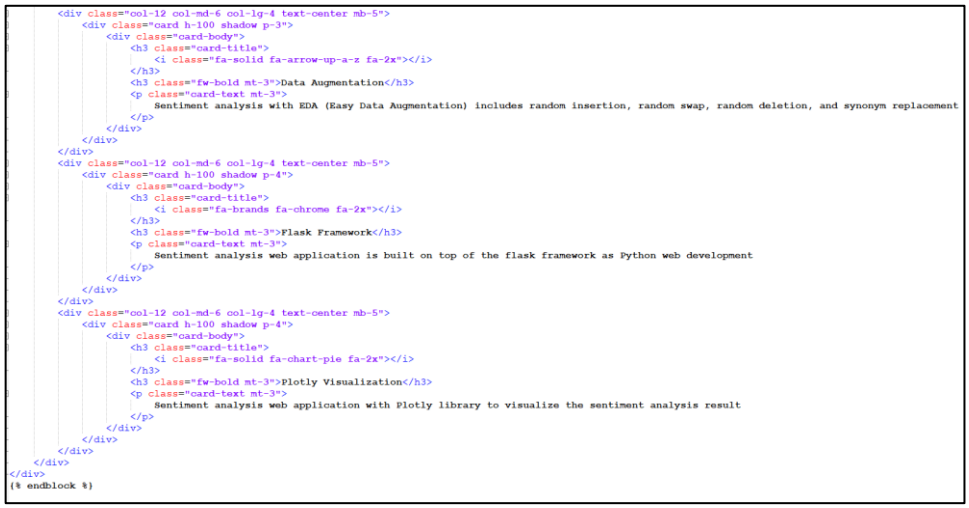
Figure 3.87 The Sentiment Analysis Web Application Home Page Code Part 2

The sentiment analysis web application for the about page HTML code shows several description or definition about the technologies, tools, or data used behind the application such as AI, NLP, Twitter data, data augmentation, *flask* framework, and *plotly* visualization as shown in the Figure 3.88 and Figure 3.89 below.



```
{% extends 'layouts/base.html' %}

{% block content %}
<div class="container d-flex flex-column min-vh-100">
    <div class="row about-title mb-3">
        <h1 class="text-center fw-bold">The Technologies <br class="d-block d-md-none"> Behind This</h1>
    </div>
    <div class="row mt-4 about-card">
        <div class="col-12 col-md-6 col-lg-4 text-center mb-5">
            <div class="card h-100 shadow p-3">
                <div class="card-body">
                    <h3 class="card-title">
                        <i class="fa-solid fa-brain fa-2x"></i>
                    </h3>
                    <h3 class="fw-bold mt-3">AI Powered</h3>
                    <p class="card-text mt-3">
                        Sentiment analysis with deep learning models such as
                        single CNN
                        model, single LSTM model, and hybrid CNN-LSTM model
                    </p>
                </div>
            </div>
        </div>
        <div class="col-12 col-md-6 col-lg-4 text-center mb-5">
            <div class="card h-100 shadow p-4">
                <div class="card-body">
                    <h3 class="card-title">
                        <i class="fa-solid fa-user-gear fa-2x"></i>
                    </h3>
                    <h3 class="fw-bold mt-3">NLP Methods</h3>
                    <p class="card-text mt-3">
                        Sentiment analysis processed with several NLP methods such as tweet cleaning, expanding
                        contractions, removing punctuations, case-folding, negation handling, removing stopwords,
                        and word lemmatizing
                    </p>
                </div>
            </div>
        </div>
        <div class="col-12 col-md-6 col-lg-4 text-center mb-5">
            <div class="card h-100 shadow p-4">
                <div class="card-body">
                    <h3 class="card-title">
                        <i class="fa-brands fa-twitter fa-2x"></i>
                    </h3>
                    <h3 class="fw-bold mt-3">Twitter Data</h3>
                    <p class="card-text mt-3">
                        Sentiment analysis with Twitter data about four COVID-19 vaccine types such as Sinovac
                        tweets, AstraZeneca tweets, Pfizer tweets, and Moderna tweets crawled using Twitter API
                    </p>
                </div>
            </div>
        </div>
```

Figure 3.88 The Sentiment Analysis Web Application About Page Code Part 1



```
        <div class="col-12 col-md-6 col-lg-4 text-center mb-5">
            <div class="card h-100 shadow p-3">
                <div class="card-body">
                    <h3 class="card-title">
                        <i class="fa-solid fa-arrow-up-a-z fa-2x"></i>
                    </h3>
                    <h3 class="fw-bold mt-3">Data Augmentation</h3>
                    <p class="card-text mt-3">
                        Sentiment analysis with EDA (Easy Data Augmentation) includes random insertion, random swap, random deletion, and synonym replacement
                    </p>
                </div>
            </div>
        </div>
        <div class="col-12 col-md-6 col-lg-4 text-center mb-5">
            <div class="card h-100 shadow p-4">
                <div class="card-body">
                    <h3 class="card-title">
                        <i class="fa-brands fa-chrome fa-2x"></i>
                    </h3>
                    <h3 class="fw-bold mt-3">Flask Framework</h3>
                    <p class="card-text mt-3">
                        Sentiment analysis web application is built on top of the flask framework as Python web development
                    </p>
                </div>
            </div>
        </div>
        <div class="col-12 col-md-6 col-lg-4 text-center mb-5">
            <div class="card h-100 shadow p-4">
                <div class="card-body">
                    <h3 class="card-title">
                        <i class="fa-solid fa-chart-pie fa-2x"></i>
                    </h3>
                    <h3 class="fw-bold mt-3">Plotly Visualization</h3>
                    <p class="card-text mt-3">
                        Sentiment analysis web application with Plotly library to visualize the sentiment analysis result
                    </p>
                </div>
            </div>
        </div>
    </div>
</div>
{% endblock %}
```

Figure 3.89 The Sentiment Analysis Web Application About Page Code Part 2

The sentiment analysis web application for the sentiment analysis page HTML code shows the instruction for the user to use the application as welcoming page. The instructions are listed with list group item as shown in the Figure 3.90 below.

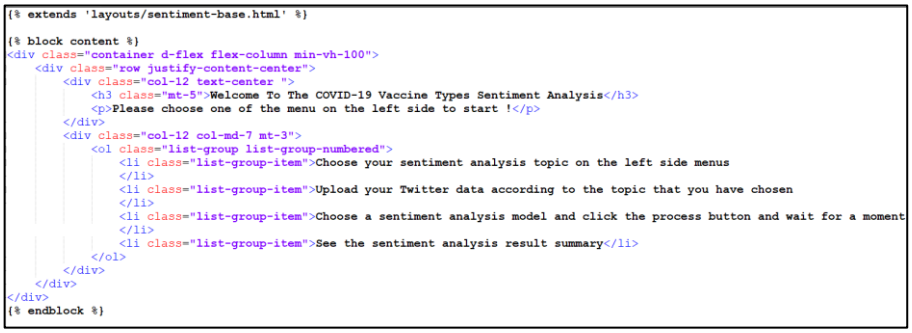


Figure 3.90 The Sentiment Analysis Web Application Sentiment Analysis Page Code

Each topic sentiment analysis page HTML code has the exact same code structures that render the form field with file input field, selection input field, and submit button from the *main.py* module using the *flask-wtf* library, as shown in the Figure 3.91 below, as the example taken from the sentiment analysis page HTML code for the Sinovac topic. The user uploads the implementation dataset through the file input field and selects the sentiment analysis model through the selection input field correspond to the topic.

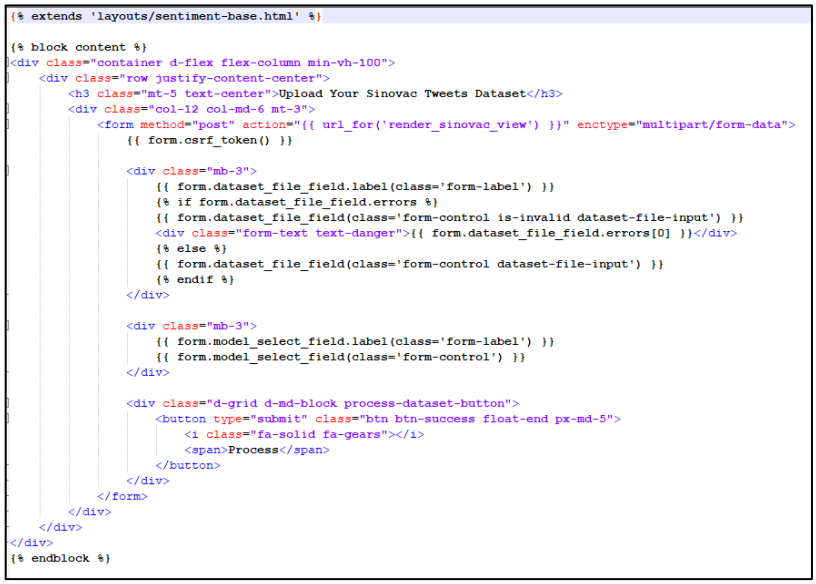


Figure 3.91 The Sentiment Analysis Web Application Sentiment Analysis Page Code

The first part of each topic sentiment analysis result page HTML code has the exact same code structures that render the three summary panels to present the number of tweets on the first panel data, the number of classified positive tweets on the second panel data, and the number of classified negative tweets on the third

panel data as shown in the Figure 3.92 below, as the example taken from the first part sentiment analysis result page HTML code for the Sinovac topic.

```
{% extends 'layouts/sentiment-base.html' %}

{% block content %}
<div class="container d-flex flex-column min-vh-100 sentiment-result">
    <div class="row justify-content-center">
        <h3 class="mt-5 text-center fw-bold">Sinovac Dataset Sentiment Analysis Summary <br> ({{ selected_model_text }})</h3>
        <div class="col-12 col-md-4 mt-3">
            <div class="card text-white text-center bg-primary mb-3 h-100">
                <div class="card-header h-50">
                    <h5>Total Tweet Texts</h5>
                </div>
                <div class="card-body">
                    <p class="card-text">
                        {{ sinovac_tweets }}
                    </p>
                </div>
            </div>
        </div>

        <div class="col-12 col-md-4 mt-3">
            <div class="card text-white text-center bg-success mb-3 h-100">
                <div class="card-header h-50">
                    <h5>Total Positive Tweets</h5>
                </div>
                <div class="card-body">
                    <p class="card-text">
                        {{ sinovac_positive }}
                    </p>
                </div>
            </div>
        </div>

        <div class="col-12 col-md-4 mt-3">
            <div class="card text-white text-center bg-danger mb-3 h-100">
                <div class="card-header h-50">
                    <h5>Total Negative Tweets</h5>
                </div>
                <div class="card-body">
                    <p class="card-text">
                        {{ sinovac_negative }}
                    </p>
                </div>
            </div>
        </div>
    </div>
```

Figure 3.92 The Sentiment Analysis Web Application Sentiment Analysis Result Page Code Part 1

The second part of each topic sentiment analysis result page HTML code has the exact same code structures that render the sentiment prediction or classification result in table with ten examples and two columns for the tweet texts and the classified sentiment as shown in the Figure 3.93 below, as the example taken from the second part sentiment analysis result page HTML code for the Sinovac topic.

```
<div class="row justify-content-center mt-5">
    <h3 class="mt-5 text-center fw-bold">Prediction Results Examples</h3>
    <div class="col-12 col-md-12 mt-4">
        <div class="card mb-3 shadow">
            <div class="card-body table-responsive">
                <table class="table table-hover">
                    <thead>
                        <tr>
                            <th scope="col">#</th>
                            <th scope="col">Tweet Texts</th>
                            <th scope="col">Sentiment</th>
                        </tr>
                    </thead>
                    <tbody>
                        {% for index, data in sinovac_data.iterrows() %}
                        <tr>
                            <th scope="row">{{ index+1 }}</th>
                            <td>{{ data['text'] }}</td>
                            <td>
                                {% if data['sentiment'] == 'Positive' %}
                                <span class="badge bg-success">{{ data['sentiment'] }}</span>
                                {% else %}
                                <span class="badge bg-danger">{{ data['sentiment'] }}</span>
                                {% endif %}
                            </td>
                        </tr>
                        {% endfor %}
                    </tbody>
                </table>
            </div>
        </div>
    </div>
</div>
```

Figure 3.93 The Sentiment Analysis Web Application Sentiment Analysis Result Page Code Part 2

The third part of each topic sentiment analysis result page HTML code has the exact same code structures that visualize the sentiment prediction or classification result in bar plot and pie plot using the *plotly* library as shown in the Figure 3.94 below, as the example taken from the third part sentiment analysis result page HTML code for the Sinovac topic.

```
    <div class="row justify-content-center mt-5 mb-5">
        <h3 class="mt-5 text-center fw-bold">Sentiment Analysis Visualization</h3>
        <div class="col-12 col-lg-6 mt-3">
            <div class="card text-center mb-3 shadow">
                <div class="card-body">
                    <div id="sinovac-bar-plot"></div>
                </div>
            </div>
        </div>

        <div class="col-12 col-lg-6 mt-3">
            <div class="card text-center mb-3 shadow">
                <div class="card-body">
                    <div id="sinovac-pie-plot"></div>
                </div>
            </div>
        </div>
    </div>
</div>

<script src="https://cdn.plot.ly/plotly-2.11.1.min.js"></script>
<script>
    window.onresize = () => {
        Plotly.Plots.resize('sinovac-bar-plot')
        Plotly.Plots.resize('sinovac-pie-plot')
    }

    const sinovac_bar_graph = {{ sinovac_bar_plot | safe }}
    Plotly.newPlot('sinovac-bar-plot', sinovac_bar_graph, {}, { responsive: true })

    const sinovac_pie_graph = {{ sinovac_pie_plot | safe }}
    Plotly.newPlot('sinovac-pie-plot', sinovac_pie_graph, {}, { responsive: true })
</script>
{% endblock %}
```

Figure 3.94 The Sentiment Analysis Web Application Sentiment Analysis Result Page Code Part 3

3.5.4 Testing

The testing of the incremental 2 is to test and evaluate the sentiment analysis web application functions. The basic black-box testing table will be used to test each function by define the test case name, the input test, the expected output, the test result, and the summary.

Table 3.6 The Sentiment Analysis Web Application Black-Box Testing Table

Test Scenario	***Test Case***	***Expected Output***	***Test Result***	***Summary***
Home Page	*Open the home page by clicking the home menu in navigation bar*	*Show the home page*		
About Page	*Open the about page by clicking the about menu in navigation bar*	*Show the about page*		
Sentiment Analysis Page	*Open the about page by clicking the sentiment*	*Show the sentiment*		

	analysis menu in navigation bar	*analysis welcome page*
Sinovac Sentiment Analysis Page	*Open the Sinovac sentiment analysis page by clicking the Sinovac menu in sidebar*	*Show the Sinovac sentiment analysis page*
Sinovac Sentiment Analysis Form Validation	*Make the file input field empty and upload a non-CSV format*	*Show the form validation error message*
Sinovac Sentiment Analysis Result Page	*Upload the proper Sinovac dataset file, choose the sentiment analysis model, and clicked the process button*	*Show the Sinovac sentiment analysis result page with overview panels, example of classified sentiment, and the visualization*
AstraZeneca Sentiment Analysis Page	*Open the AstraZeneca sentiment analysis page by clicking the AstraZeneca menu in sidebar*	*Show the AstraZeneca sentiment analysis page*
AstraZeneca Sentiment Analysis Form Validation	*Make the file input field empty and upload a non-CSV format*	*Show the form validation error message*
AstraZeneca Sentiment	*Upload the proper AstraZeneca*	*Show the AstraZeneca sentiment*

Analysis Result Page	*dataset file, choose the sentiment analysis model, and clicked the process button*	*analysis result page with overview panels, example of classified sentiment, and the visualization*
Pfizer Sentiment Analysis Page	*Open the Pfizer sentiment analysis page by clicking the Pfizer menu in sidebar*	*Show the Pfizer sentiment analysis page*
Pfizer Sentiment Analysis Form Validation	*Make the file input field empty and upload a non-CSV format*	*Show the form validation error message*
Pfizer Sentiment Analysis Result Page	*Upload the proper Pfizer dataset file, choose the sentiment analysis model, and clicked the process button*	*Show the Pfizer sentiment analysis result page with overview panels, example of classified sentiment, and the visualization*
Moderna Sentiment Analysis Page	*Open the Moderna sentiment analysis page by clicking the Moderna menu in sidebar*	*Show the Moderna sentiment analysis page*
Moderna Sentiment Analysis Form Validation	*Make the file input field empty and upload a non-CSV format*	*Show the form validation error message*

Moderna Sentiment Analysis Result Page	*Upload the proper Moderna dataset file, choose the sentiment analysis model, and clicked the process button*	*Show the Moderna sentiment analysis result page with overview panels, example of classified sentiment, and the visualization*
Reset Sentiment Analysis Result and Back to the Home Page	*Reset all the sentiment analysis result by clicking the back to home menu in sidebar*	*Delete all the uploaded, pre-processed, and classified data and file, then open the home page*

CHAPTER 4
RESULT AND DISCUSSION

4.1 Incremental 1

This section covers the results from the incremental 1 process that are the dataset crawling process result, the dataset pre-processing process result, the dataset labeling process result, the dataset splitting process result, the dataset augmentation process result, the training sentiment analysis model process result, and the testing and evaluation sentiment analysis model process result. Then, this section also elaborates the single CNN model, the single LSTM model, and the hybrid CNN-LSTM model training and testing result comparison.

4.1.1 Dataset

The Twitter data crawling process resulting in each of the topic has five raw unmerged data in CSV format, so the number of the raw data files are 20 files as shown in the Figure 4.1 below.

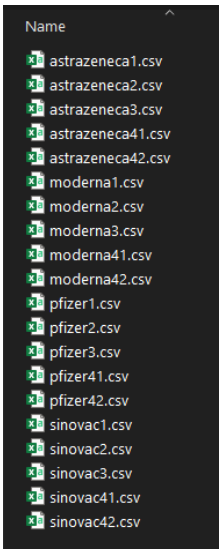


Figure 4.1 Twitter Data Crawling Process Result

In detail, the Sinovac topic has 9697 raw tweet texts, the AstraZeneca topic has 8651 raw tweet texts, the Pfizer topic has 10000 raw tweet texts, and the Moderna topic has 10000 raw tweet texts as shown in the Table 4.1 below. Hence, the Pfizer topic and the Moderna topic has the largest amount of the raw tweets.

Table 4.1 The Number of Tweets of The Dataset Merging Process Result

No.	Dataset	Number of Tweets
1	Sinovac	9697
2	AstraZeneca	8651
3	Pifzer	10000
4	Moderna	10000

Unfortunately, there is a limitation from Twitter API, where only the retweets are truncated with "…" symbol, therefore there are several tweet texts that are not in full text. The truncated tweets are calculated for each four dataset topics and resulting in 15% truncated tweets in the Sinovac topic, 23% truncated tweets in the AstraZeneca topic, 31% truncated tweets in the Pfizer topic, and 21% truncated tweets in the Moderna topic. From the calculated truncated tweets, I decided to keep the all the tweet texts from the four raw data topics because there is not too much truncated text (the threshold is 70%).

The crawled raw data on the four topics are implemented with the raw data pre-processing in order to reduce noises in the data from words or characters that are useless and meaningless for the sentiment analysis models. The Table 4.2 below shows the example of the raw data pre-processing results from the Sinovac topic, the AstraZeneca topic, the Pfizer topic, and the Moderna topic.

Table 4.2 The Raw Data Pre-Processing Result Examples

No.	Raw Tweets	Pre-Processed Tweets
1	@LycosAustralis @ekecheirion In America Sputnik will never get approved but Sinovac probably will eventually	america sputnik never get approve sinovac probably eventually
2	@JKJAVMY hi, may i know this station got sinovac for 9 years old kid ?? thanks	hi may know station get sinovac year old kid thanks
3	RT @Jenna_bee__: @GullyJudith @truth8775 When I was injured is when I learned doctors here aren't ALLOWED to write exemptions. Also, advers…	injured learn doctor disallow write exemption also advers
4	RT @DrJohnB2: Longitudinally extensive transverse myelitis	longitudinally extensive transverse myelitis inflammation

	(inflammation of the spinal cord) after #CovidVaccination (AstraZeneca): https:/…	*spinal cord covidvaccination astrazeneca*
5	*RT @theysayitsrare: Nothing to see here ... just a $2.8M wire transfer from Pfizer to the FDA 😉 https://t.co/ucxdFXbWBk*	*nothing see wire transfer pfizer fda*
6	*RT @LfcJmr: @DrLoupis Pfizer Could Lose Liability Protection and Be Sued if the Deaths Are Proven as Willful\n\nThomasRenz: "I don't know how…*	*pfizer could lose liability protection sue death proven willful thomasrenz ignore*
7	*RT @SAHealth: A half dose of the Moderna COVID-19 vaccine is available for children aged 6 to 11 at selected pharmacies and GPs.\n\nCheck wit…*	*half dose moderna covid vaccine available child age select pharmacy gps check wit*
8	*RT @JeanRees10: DEATH - BOOSTER\n\n27 yo female - Foreign\n\n20 days post 3rd dose Moderna\n\n"This regulatory authority case concerns a 27-year-…*	*death booster female foreign day post dose moderna regulatory authority case concern year*

The pre-processed four topic datasets will be selected for the experiment dataset and the implementation dataset. The experiment dataset also being implemented with removing the number of words that below five to avoid error during the data augmentation. In contrast, the implementation dataset only uses the raw data for the sentiment analysis web application to be classified the sentiment. The Table 4.3 below shows the number of tweets in the experiment dataset and the implement dataset in detail.

Table 4.3 The Number of Tweets for The Experiment Dataset and The Implementation Dataset

No.	***Dataset***	***Number of Tweets for Experiment Dataset***	***Number of Tweets for Implementation Dataset***
1	*Sinovac*	*2271*	*1094*
2	*AstraZeneca*	*1487*	*1421*
3	*Pfizer*	*1351*	*1270*
4	*Moderna*	*1986*	*1767*

The pre-processed experiment dataset on the four topics will be labeled with the *vaderSentimet* library as the lexicon-based labeling, because this research performs supervised learning. The results of the dataset labeling process with *vaderSentiment* library are positive sentiment, neutral sentiment, and negative sentiment. However, this research focuses on the positive sentiment and the negative sentiment. The Table 4.4 below shows the dataset labeling result example from the Sinovac topic, the AstraZeneca topic, the Pfizer topic, and the Moderna topic.

Table 4.4 The Sinovac Dataset Labeling Process Result Examples

No.	***Tweets***	***Sentiment***
1	*america sputnik never get approve sinovac probably eventually*	*Negative*
2	*may know station get sinovac year old kid thanks*	*Positive*
3	*injured learn doctor disallow write exemption also advers*	*Negative*
4	*shane astrazeneca jab many boxer heart trouble cut weight shane professional athlete nutritional knowledge experience*	*Positive*
5	*pfizer could lose liability protection sue death proven willful thomasrenz ignore*	*Negative*
6	*pfizer admits vaccine ineffective child married governor stop inject child*	*Negative*
7	*death booster female foreign day post dose moderna regulatory authority case concern year*	*Negative*
8	*death female foreign day post dose moderna cardiac arrest pulmonary thromboembolism cardiac arrest ast*	*Negative*

The labeled experiment dataset on the four topics will be split into the training dataset and the testing dataset by 90% and 10% respectively. Before split the dataset, the number of words that below five on each tweet texts in the experiment dataset are removed to avoid error during the data augmentation. The Table 4.5 below shows the number of tweets after the splitting dataset into the training dataset and the testing dataset in detail.

Table 4.5 The Number of Tweets of The Dataset Splitting Process Result

No.	***Dataset***	***Number of Tweets for Training Dataset***	***Number of Tweets for Testing Dataset***
1	*Sinovac*	*1330*	*333*
2	*AstraZeneca*	*736*	*184*
3	*Pfizer*	*729*	*183*
4	*Moderna*	*1060*	*265*

Finally, the data augmentation is applied on the four training dataset topics to increase variation on the training dataset. The result of the data augmentation process increases the number of tweets in the training dataset as shown in the Table 4.6 below, because it adds four additional new tweet texts from the EDA method and the original tweet texts. Hence the original tweet texts are four times increased.

Table 4.6 The Number of Tweets Before and After The Data Augmentation Process Result

No.	***Dataset***	***Number of Tweets in Training Dataset***	
		Before Data Augmentation	***After Data Augmentation***
1	*Sinovac*	*1330*	*6650*
2	*AstraZeneca*	*736*	*3680*
3	*Pfizer*	*729*	*3645*
4	*Moderna*	*1060*	*5300*

4.1.2 Training Result: Sinovac Dataset Model

The training 6650 tweet texts process results of the sentiment analysis models for the Sinovac dataset topic with 20 epochs and 10 k-folds cross validation

has descent results. The single CNN model has the best training performance compared to the single LSTM model and the hybrid CNN-LSTM model on the Sinovac dataset topic. Overall, the training performance results of the sentiment analysis models for the Sinovac dataset topic are not heavily overfit, because the difference between the training and the validation are not significantly big as shown in the Table 4.7 below.

Table 4.7 The Training Performance Results of The Sentiment Analysis Models for The Sinovac Dataset Topic

No.	***Sentiment Analysis Models***	***Average Training Performance Results 10 K-Folds Cross Validation***			
		Training Accuracy	***Training Loss***	***Validation Accuracy***	***Validation Loss***
1	*Single CNN*	***99,96%***	***0,00***	***95,16%***	***0,16***
2	*Single LSTM*	*99,82%*	*0,01*	*95,08%*	*0,19*
3	*Hybrid CNN-LSTM*	*99,86%*	*0,01*	*94,90%*	*0,19*

4.1.3 Training Result: AstraZeneca Dataset Model

The training 3680 tweet texts process results of the sentiment analysis models for the AstraZeneca dataset topic with 20 epochs and 10 k-folds cross validation also has descent results. The single CNN model also has the best training performance compared to the single LSTM model and the hybrid CNN-LSTM model on the AstraZeneca dataset topic. Overall, the training performance results of the sentiment analysis models for the AstraZeneca dataset topic also are not heavily overfit, because the difference between the training and the validation are not significantly big as shown in the Table 4.8 below.

Table 4.8 The Training Performance Results of The Sentiment Analysis Models for The AstraZeneca Dataset Topic

No.	***Sentiment Analysis Models***	***Average Training Performance Results 10 K-Folds Cross Validation***			
		Training Accuracy	***Training Loss***	***Validation Accuracy***	***Validation Loss***
1	*Single CNN*	***99,99%***	***0,00***	***95,87%***	***0,13***

2	*Single LSTM*	*99,96%*	*0,00*	*96,14%*	*0,14*
3	*Hybrid CNN-LSTM*	*99,91%*	*0,00*	*96,20%*	*0,14*

4.1.4 Training Result: Pfizer Dataset Model

The training 3645 tweet texts process results of the sentiment analysis models for the Pfizer dataset topic with 20 epochs and 10 k-folds cross validation also has descent results. The single CNN model also has the best training performance compared to the single LSTM model and the hybrid CNN-LSTM model on the Pfizer dataset topic. Overall, the training performance results of the sentiment analysis models for the Pfizer dataset topic also are not heavily overfit, because the difference between the training and the validation are not significantly big as shown in the Table 4.9 below.

Table 4.9 The Training Performance Results of The Sentiment Analysis Models for The Pfizer Dataset Topic

No.	***Sentiment Analysis Models***	***Average Training Performance Results 10 K-Folds Cross Validation***			
		Training Accuracy	***Training Loss***	***Validation Accuracy***	***Validation Loss***
1	*Single CNN*	***99,99%***	***0,00***	***96,13%***	***0,12***
2	*Single LSTM*	*99,83%*	*0,01*	*95,31%*	*0,15*
3	*Hybrid CNN-LSTM*	*99,98%*	*0,00*	*95,94%*	*0,14*

4.1.5 Training Result: Moderna Dataset Model

The training 5300 tweet texts process results of the sentiment analysis models for the Moderna dataset topic with 20 epochs and 10 k-folds cross validation also has descent results. The single CNN model also has the best training performance compared to the single LSTM model and the hybrid CNN-LSTM model on the Moderna dataset topic. Overall, the training performance results of the sentiment analysis models for the Moderna dataset topic also are not heavily overfit, because the difference between the training and the validation are not significantly big as shown in the Table 4.10 below.

Table 4.10 The Training Performance Results of The Sentiment Analysis Models for The Moderna Dataset Topic

No.	***Sentiment Analysis Models***	***Average Training Performance Results 10 K-Folds Cross Validation***			
		Training Accuracy	***Training Loss***	***Validation Accuracy***	***Validation Loss***
1	*Single CNN*	***99,99%***	***0,00***	***95,92%***	***0,13***
2	*Single LSTM*	*99,93%*	*0,00*	*95,96%*	*0,16*
3	*Hybrid CNN-LSTM*	*99,98%*	*0,00*	*96,02%*	*0,15*

4.1.6 Testing and Evaluation Result: Sinovac Dataset Model

The testing on 333 unseen tweet texts resulting in the single CNN model has the best testing performance compared to the single LSTM model and the hybrid CNN-LSTM model on the Sinovac dataset topic as shown in the Table 4.11 below.

Table 4.11 The Testing Results of The Sentiment Analysis Models for The Sinovac Dataset Topic

Sentiment Analysis Models	***Testing Performance***	
	Accuracy	***Loss***
Single CNN	***78,38%***	***0,89***
Single LSTM	*76,88%*	*0,95*
Hybrid CNN-LSTM	*75,68%*	*1,2*

The evaluation metrics strongly proofed that the single CNN model has the best overall performance with the average of 78% on recall, 78% on precision, and 78% on f1-score compared to the single LSTM model and the hybrid CNN-LSTM model on the Sinovac dataset topic as shown in the Table 4.11 below.

Table 4.12 The Evaluation Results of The Sentiment Analysis Models for The Sinovac Dataset Topic

Sentiment Analysis Models	***Label***	***Metrics***		
		Recall	***Precision***	***F1-Score***
Single CNN	*Negative*	*77%*	*81%*	*79%*
	Positive	*79%*	*76%*	*77%*

Average		***78%***	***78%***	***78%***
Single LSTM	*Negative*	*81%*	*77%*	*79%*
	Positive	*72%*	*77%*	*74%*
Average		*77%*	*77%*	*77%*
Hybrid CNN-LSTM	*Negative*	*76%*	*78%*	*77%*
	Positive	*76%*	*73%*	*74%*
Average		*76%*	*76%*	*76%*

4.1.7 Testing and Evaluation Result: AstraZeneca Dataset Model

The testing on 184 unseen tweet texts resulting in the single CNN model has the lowest loss but the hybrid CNN-LSTM has the highest accuracy on the AstraZeneca dataset topic as shown in the Table 4.13 below.

Table 4.13 The Testing Results of The Sentiment Analysis Models for The AstraZeneca Dataset Topic

Sentiment Analysis Models	***Testing Performance***	
	Accuracy	***Loss***
Single CNN	*80,43%*	***0,78***
Single LSTM	*81,52%*	*1,1*
Hybrid CNN-LSTM	***81,52%***	*0,86*

The evaluation metrics strongly proofed that the hybrid CNN-LSTM model has the best overall performance with the average of 82% on recall, 81% on precision, and 81% on f1-score even though the loss is higher compared to the single CNN model on the AstraZeneca dataset topic as shown in the Table 4.14 below.

Table 4.14 The Evaluation Results of The Sentiment Analysis Models for The AstraZeneca Dataset Topic

Sentiment Analysis Models	***Label***	***Metrics***		
		Recall	***Precision***	***F1-Score***
Single CNN	*Negative*	*75%*	*81%*	*75%*
	Positive	*85%*	*80%*	*83%*
Average		*80%*	*80%*	*80%*
Single LSTM	*Negative*	*76%*	*82%*	*79%*

	Positive	*86%*	*81%*	*84%*
Average		*81%*	*82%*	*81%*
Hybrid CNN-LSTM	*Negative*	*82%*	*78%*	*80%*
	Positive	*81%*	*85%*	*83%*
Average		***82%***	***81%***	***81%***

4.1.8 Testing and Evaluation Result: Pfizer Dataset Model

The testing on 183 unseen tweet texts resulting in the hybrid CNN-LSTM has the best testing performance compared to the single CNN model and the single LSTM model on the Pfizer dataset topic as shown in the Table 4.15 below.

Table 4.15 The Testing Results of The Sentiment Analysis Models for The Pfizer Dataset Topic

Sentiment Analysis Models	***Testing Performance***	
	Accuracy	***Loss***
Single CNN	*68,85%*	*1,14*
Single LSTM	*69,95%*	*1,08*
Hybrid CNN-LSTM	***71,04%***	***1,17***

The evaluation metrics strongly proofed that the hybrid CNN-LSTM model has the best overall performance with the average of 71% on recall, 71% on precision, and 71% on f1-score compared to the single CNN model and the single LSTM model on the AstraZeneca dataset topic as shown in the Table 4.16 below.

Table 4.16 The Evaluation Results of The Sentiment Analysis Models for The Pfizer Dataset Topic

Sentiment Analysis Models	***Label***	***Metrics***		
		Recall	***Precision***	***F1-Score***
Single CNN	*Negative*	*74%*	*71%*	*72%*
	Positive	*62%*	*66%*	*64%*
Average		*68%*	*68%*	*68%*
Single LSTM	*Negative*	*75%*	*72%*	*73%*
	Positive	*63%*	*68%*	*65%*
Average		*69%*	*70%*	*69%*
Hybrid CNN-LSTM	*Negative*	*75%*	*73%*	*74%*

	Positive	*66%*	*68%*	*67%*
Average		***71%***	***71%***	***71%***

4.1.9 Testing and Evaluation Result: Moderna Dataset Model

The testing on 265 unseen tweet texts resulting in the single CNN model has the best testing performance compared to the single LSTM model and the hybrid CNN-LSTM model on the Sinovac dataset topic as shown in the Table 4.17 below.

Table 4.17 The Testing Results of The Sentiment Analysis Models for The AstraZeneca Dataset Topic

Sentiment Analysis Models	***Testing Performance***	
	Accuracy	***Loss***
Single CNN	***76,98%***	***1,02***
Single LSTM	*75,09%*	*1,07*
Hybrid CNN-LSTM	*73,96%*	*1,12*

The evaluation metrics strongly proofed that the single CNN model has the best overall performance with the average of 76% on recall, 77% on precision, and 76% on f1-score compared to the single LSTM model and the hybrid CNN-LSTM model on the Sinovac dataset topic as shown in the Table 4.18 below.

Table 4.18 The Evaluation Results of The Sentiment Analysis Models for The AstraZeneca Dataset Topic

Sentiment Analysis Models	***Label***	***Metrics***		
		Recall	***Precision***	***F1-Score***
Single CNN	*Negative*	*70%*	*76%*	*73%*
	Positive	*82%*	*78%*	*80%*
Average		***76%***	***77%***	***76%***
Single LSTM	*Negative*	*66%*	*75%*	*70%*
	Positive	*82%*	*75%*	*79%*
Average		*74%*	*75%*	*74%*
Hybrid CNN-LSTM	*Negative*	*64%*	*74%*	*69%*
	Positive	*82%*	*74%*	*78%*
Average		*73%*	*74%*	*73%*

4.1.10 Discussion

Based on the described results, the single CNN model has dominant training performances on the Sinovac dataset topic, the AstraZeneca dataset topic, the Pfizer dataset topic, and the Moderna dataset topic compared to the single LSTM model and the hybrid model. However, in terms of testing and evaluation, the single CNN model has worked best on the Sinovac dataset topic with average of 78% on recall, 78% on precision, and 78% on f1-score; and on the Moderna dataset topic with the average of 76% on recall, 77% on precision, and 76% on f1-score. On the other hand, the hybrid CNN-LSTM model has worked best on the AstraZeneca dataset with the average of 82% on recall, 81% on precision, and 81% on f1-score; and on the Pfizer dataset with the average of 71% on recall, 71% on precision, and 71% on f1-score.

Unfortunately, the overall the sentiment analysis model performances have significantly difference performances between the training process and the testing process that leads to the overfitting. There are several main causes that can be suspected in order to explain the overfitting on the testing dataset.

First, some tweet texts are truncated, which, reduced the whole information; this may affect the lexicon-based labeling process that is possible to label the wrong sentiment because of the reduced information on the truncated tweet texts. Thus, there is some bias on the labeling process where several most words in positive sentiment also exist in the negative sentiment and made the sentiment analysis models "confuse" to generalize the unseen data. As I described previously, there are 15% truncated tweet texts on the Sinovac topic, 23% truncated tweet texts on the AstraZeneca topic, 31% truncated tweet texts on the Pfizer topic, and 21% truncated tweet texts on the Moderna topic.

Second, the biased labeled data also may affect the training data that some of it not completely represent the feature of the unseen data because some of the mislabeled tweet texts.

Third, some words do not available in the pre-trained Glove Twitter word embedding because the pre-trained Glove Twitter word embedding corpus may not contain words that are exist in the COVID-19 vaccine trend as in each four corpus topics as shown in the Table 4.19 below. This also cause the word embedding ignored some important words. Another, that also can be considered is the current sentiment analysis models may have not reached the optimized parameters.

Table 4.19 Each Four Corpus Topic's Word Examples That Are Not Existed In The Pre-Trained Glove Twitter Word Embedding Corpus

No.	Sinovac Corpus Topic	AstraZeneca Corpus Topic	Pfizer Corpus Topic	Moderna Corpus Topic
1	sinovac	covid	covid	covid
2	covid	vaccinum	vaccinum	vaccinum
3	vaccinum	evusheld	pfizers	vaxxed
4	biontech	lynparza	bourla	fauci
5	sinopharm	antibodybased	vaxxed	bancel
6	vaxxed	astrazenecas	vaxx	twelvemonth
7	inoculation	fasenra	fauci	myocarditis
8	immunise	oxfordastrazeneca	twelvemonth	sarscov
9	coronavac	subvariant	ukrayina	phizer
10	chinaware	masovia	invermectin	immunodeficiency

4.2 Incremental 2

This section covers the results from the incremental 2 processes that are the sentiment analysis web application about each COVID-19 vaccine type such as Sinovac, AstraZeneca, Pfizer, and Moderna. In detail, this section shows the sentiment analysis web application user interface page that has built with *flask* framework for *Python* web development and the black box testing result for each function.

4.2.1 Sentiment Analysis Web Application User Interface

The home page in the sentiment analysis web application shows a brief domain introduction about NLP (Natural Language Processing), sentiment analysis,

Sinovac vaccine, AstraZeneca vaccine, Pfizer vaccine, and Moderna vaccine according to the user interface wireframe design as shown in the Figure 4.2 below.

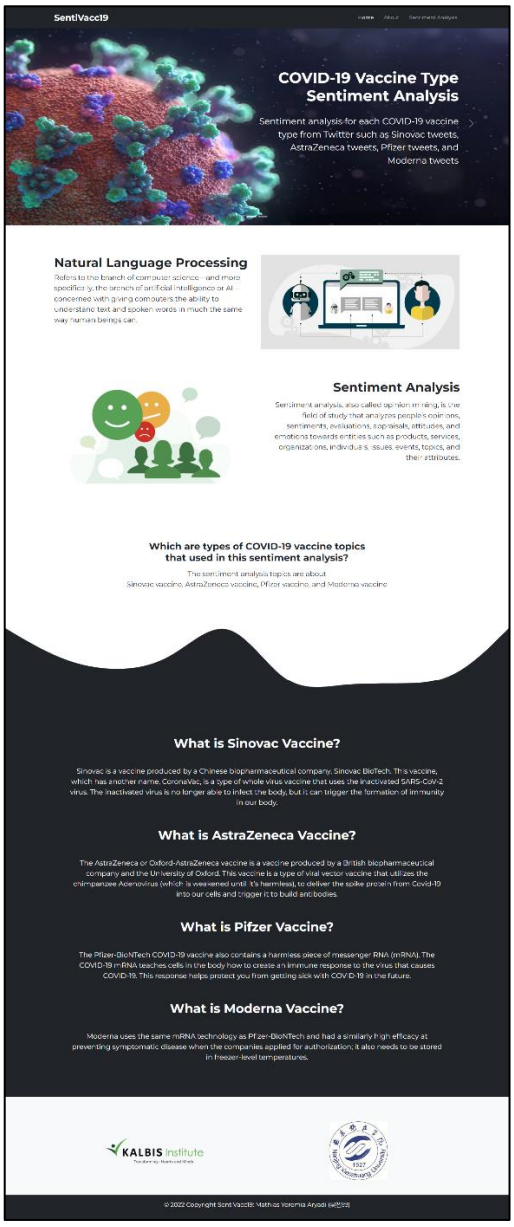


Figure 4.2 Sentiment Analysis Web Application: Home Page

The about page in the sentiment analysis web application tells the user about what are technologies and methods behind the sentiment analysis for COVID-19 vaccine types according to the user interface wireframe design as shown in the Figure 4.3 below.

Figure 4.3 Sentiment Analysis Web Application: About Page

The sentiment analysis welcome page in the sentiment analysis web application tells user the instruction to use the sentiment analysis web application according to the user interface wireframe design as shown in the Figure 4.4 below.

Figure 4.4 Sentiment Analysis Web Application: Sentiment Analysis Page

The sentiment analysis welcome page in the sentiment analysis web application when the side bar on the left is active to show the subpages of the sentiment analysis page according to the user interface wireframe design as shown in the Figure 4.5 below.

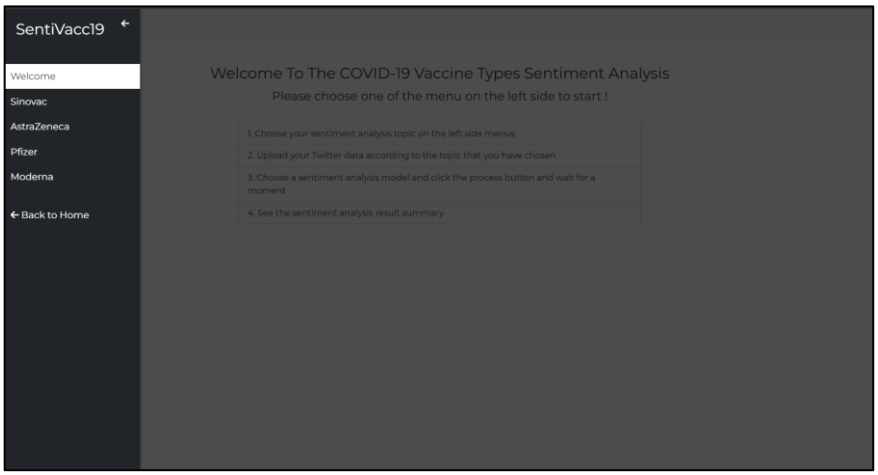


Figure 4.5 Sentiment Analysis Web Application: Sidebar Menu Triggered at Sentiment Analysis Page

Each topic sentiment analysis page in the sentiment analysis web application has the exact same component that show the form field with file input

field, selection input field, and submit button according to the user interface wireframe design as shown in the Figure 4.6 below.

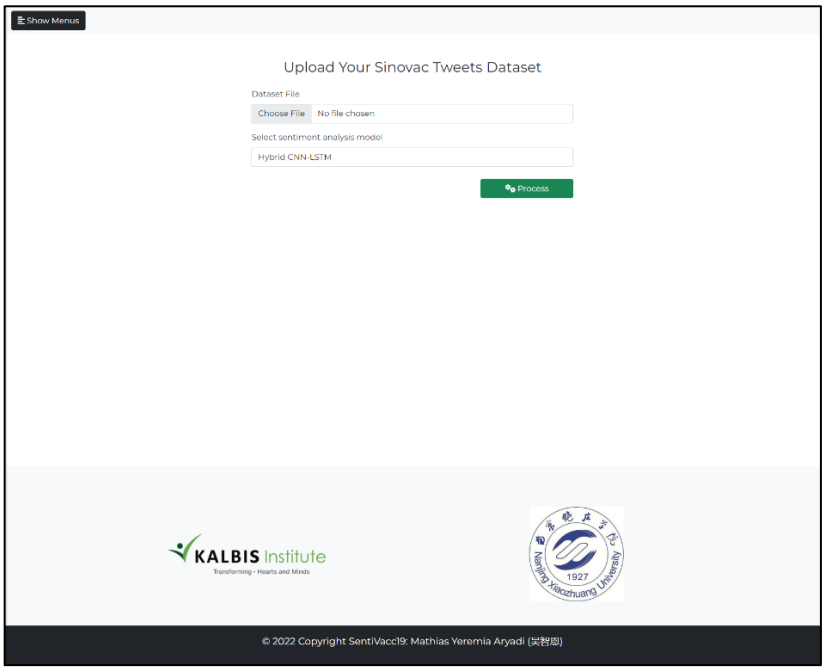


Figure 4.6 Sentiment Analysis Web Application: Sinovac Sentiment Analysis Page

Each topic sentiment analysis result page in the sentiment analysis web application has the exact same components that show the three data panels, the classification table result, and the classified sentiment visualization according to the user interface wireframe design as shown in the Figure 4.7 below.

Figure 4.7 Sentiment Analysis Web Application: Sinovac Sentiment Analysis Result Page

The resetting all the sentiment analysis results in the sentiment analysis web application shows the confirmation modal to the users before they quit the sentiment analysis page and back to the home page according to the user interface wireframe design as shown in the Figure 4.8 below, that the system will reset all the sentiment analysis results by removing all the uploaded, pre-processed, and classified file dataset.

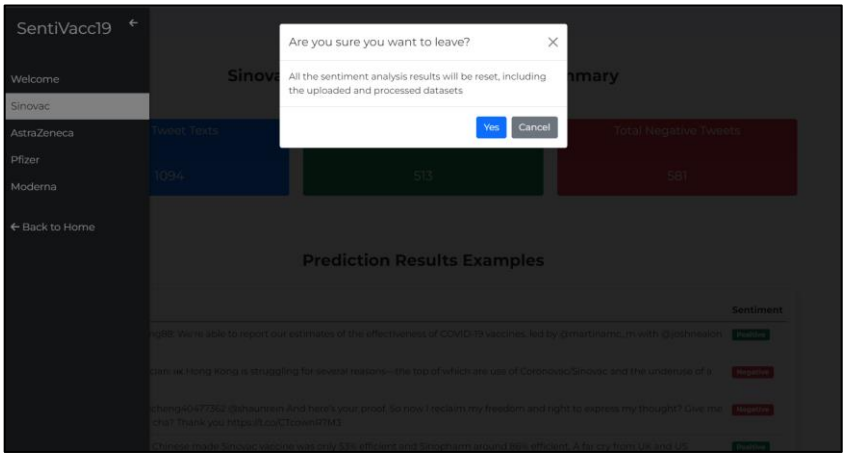


Figure 4.8 Sentiment Analysis Web Application: Reset All Sentiment Analysis Results

4.2.2 Back Box Testing

Table 4.20 The Sentiment Analysis Web Application Black-Box Testing Result

Test Scenario	***Test Case***	***Expected Output***	***Test Result***	***Summary***
Home Page	*Open the home page by clicking the home menu in navigation bar*	*Show the home page*	*Success*	*Normal*
About Page	*Open the about page by clicking the about menu in navigation bar*	*Show the about page*	*Success*	*Normal*
Sentiment Analysis Page	*Open the about page by clicking the sentiment analysis menu in navigation bar*	*Show the sentiment analysis welcome page*	*Success*	*Normal*
Sinovac Sentiment Analysis Page	*Open the Sinovac sentiment analysis page by clicking the Sinovac menu in sidebar*	*Show the Sinovac sentiment analysis page*	*Success*	*Normal*

Sinovac Sentiment Analysis Form Validation	*Make the file input field empty and upload a non-CSV format*	*Show the form validation error message*	*Success*	*Normal*
Sinovac Sentiment Analysis Result Page	*Upload the proper Sinovac dataset file, choose the sentiment analysis model, and clicked the process button*	*Show the Sinovac sentiment analysis result page with overview panels, example of classified sentiment, and the visualization*	*Success*	*Normal*
AstraZeneca Sentiment Analysis Page	*Open the AstraZeneca sentiment analysis page by clicking the AstraZeneca menu in sidebar*	*Show the AstraZeneca sentiment analysis page*	*Success*	*Normal*
AstraZeneca Sentiment Analysis Form Validation	*Make the file input field empty and upload a non-CSV format*	*Show the form validation error message*	*Success*	*Normal*
AstraZeneca Sentiment Analysis Result Page	*Upload the proper AstraZeneca dataset file, choose the sentiment analysis model, and clicked the process button*	*Show the AstraZeneca sentiment analysis result page with overview panels, example of classified sentiment, and the visualization*	*Success*	*Normal*

Pfizer Sentiment Analysis Page	*Open the Pfizer sentiment analysis page by clicking the Pfizer menu in sidebar*	*Show the Pfizer sentiment analysis page*	*Success*	*Normal*
Pfizer Sentiment Analysis Form Validation	*Make the file input field empty and upload a non-CSV format*	*Show the form validation error message*	*Success*	*Normal*
Pfizer Sentiment Analysis Result Page	*Upload the proper Pfizer dataset file, choose the sentiment analysis model, and clicked the process button*	*Show the Pfizer sentiment analysis result page with overview panels, example of classified sentiment, and the visualization*	*Success*	*Normal*
Moderna Sentiment Analysis Page	*Open the Moderna sentiment analysis page by clicking the Moderna menu in sidebar*	*Show the Moderna sentiment analysis page*	*Success*	*Normal*
Moderna Sentiment Analysis Form Validation	*Make the file input field empty and upload a non-CSV format*	*Show the form validation error message*	*Success*	*Normal*
Moderna Sentiment Analysis Result Page	*Upload the proper Moderna dataset file, choose the sentiment analysis model, and clicked the process button*	*Show the Moderna sentiment analysis result page with overview panels, example of classified*	*Success*	*Normal*

		sentiment, and the visualization		
Reset Sentiment Analysis Result and Back to the Home Page	*Reset all the sentiment analysis result by clicking the back to home menu in sidebar*	*Delete all the uploaded, pre-processed, and classified data and file, then open the home page*	*Success*	*Normal*

4.2.3 Discussion

Based on the black-box testing, the sentiment analysis web application that has built using *flask* framework is working as expected and fulfilled the requirements and the designs. All the functions in the sentiment analysis web application have worked normally by producing the expected outputs. The *flask* framework has made easier to implement the trained sentiment analysis models because of the same programming language, which is, *Python*. The sentiment analysis web application has successfully performed sentiment analysis on the four dataset topics and has fulfilled the research objective.

CHAPTER 5
SUMMARY

5.1 Summary

In this research, I experimented with three deep learning models such as the single CNN model, the single LSTM model, and the hybrid CNN-LSTM model with EDA on the four different sentiment analysis topics. The topics are about COVID-19 vaccine types such as Sinovac, AstraZeneca, Pfizer, and Moderna in the Twitter data. Briefly, the objectives of this research are to implement the EDA as the data augmentation on the Twitter data for the sentiment analysis models; build and compare the performance difference between the single CNN model, the single LSTM model, and the hybrid CNN-LSTM model with data augmentation on the four dataset topics; and build sentiment analysis web application for the four dataset topics. Therefore, this research is summarized into several key notes:

1. This research has succeeded in implementing the data augmentation on the sentiment analysis models using EDA method on the Twitter data for sentiment analysis.
2. This research has succeeded in building the single CNN model, the single LSTM model, and the hybrid CNN-LSTM model with data augmentation on the four dataset topics through model training, testing, and evaluation.
3. This research has succeeded in comparing the single CNN model, the single LSTM model, and the hybrid CNN-LSTM model performances with data augmentation on the four dataset topics.
4. In terms on the training results, the single CNN model has dominant performance on the four dataset topics with the highest accuracy and the lowest loss on the average of the training accuracy, training loss, validation accuracy, and validation loss through 10 k-folds cross validation. On the Sinovac dataset topic, the model has achieved 99,96% training accuracy average; 0,00 training loss average; 95,16% validation accuracy average; and 0,16 validation loss average. On the AstraZeneca

dataset topic, the model has achieved 99,99% training accuracy average; 0,00 training loss average; 95,87% validation accuracy average; and 0,13 validation loss average. On the Pfizer dataset topic, the model has achieved 99,99% training accuracy average; 0,00 training loss average; 96,13% validation accuracy average; and 0,12 validation loss average. On the Moderna dataset topic, the model has achieved 99,99% training accuracy average; 0,00 training loss average; 95,92% validation accuracy average; and 0,13 validation loss average.

5. In terms on the testing and evaluation results, the single CNN model has worked best on the Sinovac dataset topic with average of 78% on recall, 78% on precision, and 78% on f1-score; and on the Moderna dataset topic with the average of 76% on recall, 77% on precision, and 76% on f1-score. On the other hand, the hybrid CNN-LSTM model has worked best on the AstraZeneca dataset with the average of 82% on recall, 81% on precision, and 81% on f1-score; and on the Pfizer dataset with the average of 71% on recall, 71% on precision, and 71% on f1-score.
6. This research has succeeded in building the sentiment analysis web application using *flask* framework for *Python* web development that the sentiment analysis web application can perform sentiment analysis on each four topics.
7. The drawbacks in this research are explained with limitations in the datasets that there are tweet texts are truncated: 15% truncated tweet texts in the Sinovac topic, 23% truncated tweet texts in the AstraZeneca topic, 31% truncated tweet texts in the Pfizer topic, and 21% truncated tweet texts in the Moderna topic; bias in automated lexicon-based labeling; lack of feature representation in the training data for the unseen data; limitations in the pre-trained Glove Twitter word embedding; and the sentiment analysis model parameters tuning may not optimal yet.

5.2 Suggestion

Based on the summaries, there are several suggestions for further research to minimize the drawbacks that existed in this research:

1. Improve the Twitter API search query so it can retrieve the non-truncated tweet texts.
2. It is highly suggested to label the text data manually with plenty of time and resources, because human generalization more accurate than automated lexicon-based data labeling in order to reduce the biases on the labeling process, even on the truncated text.
3. Increase more datasets for the Twitter data so the training dataset feature can be more representative resulting in better sentiment analysis model generalization.
4. Implement other data augmentations such as BERT (Bidirectional Encoder Representations from Transformers), GAN (Generative Adversarial Network), or ELMo (Embedding from Language Model) methods.
5. Implement other pre-trained word embeddings or use different vector dimensions or train custom word embedding by using own corpus.
6. Improve the current sentiment analysis models by more parameters tuning until reach the optimal parameters.
7. Implement other machine learning or deep learning models to enrich the model comparison for sentiment analysis.

LIST OF REFERENCES

[1] Research Department, Statista, "Most popular social networks worldwide as of October 2021, ranked by number of active users," 26 November 2021. [Online]. Available: https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/. [Diakses 2021 December 26].

[2] Research Development, Statista, "Number of monetizable daily active Twitter users (mDAU) worldwide from 1st quarter 2017 to 3rd quarter 2021," 2021 November 01 . [Online]. Available: https://www.statista.com/statistics/970920/monetizable-daily-active-twitter-users-worldwide/. [Diakses 27 December 2021].

[3] B. Lui, "Sentiment Analysis: A Fascinating Problem," dalam *Sentiment Analysis and Opinion Mining,* California, Morgan & Claypool, 2012, p. 7.

[4] V. A. Fitri, R. Andreswari dan M. A. Hasibuan, "Sentiment analysis of social media Twitter with case of Anti-LGBT campaign in Indonesia using Naïve Bayes, decision tree, and random forest algorithm," dalam *Procedia Computer Science*, Bandung, 2019.

[5] R. Novendri, A. S. Callista, D. N. Pratama dan C. E. Puspita, "Sentiment Analysis of YouTube Movie Trailer Comments Using Naïve Bayes," *Bulletin of Computer Science and Electrical Engineering,* vol. 1, no. 1, pp. 26-32, 2020.

[6] Pristiyono, M. Ritonga, M. A. A. Ihsan, A. Anjar dan F. H. Rambe, "Sentiment analysis of COVID-19 vaccine in Indonesia using Naïve Bayes Algorithm," *IOP Conference Series: Materials Science and Engineering,* vol. 1088, no. 1, pp. 12-45, 2021.

[7] M. Birjali, M. Kasri dan A. Beni-Hssane, "A comprehensive survey on sentiment analysis: Approaches, challenges and trends," *Knowledge-Based Systems,* vol. 226, 2021.

[8] U. D. Gandhi, P. Malarvizhi Kumar, G. Chandra Babu dan G. Karthick, "Sentiment Analysis on Twitter Data by Using Convolutional Neural Network (CNN) and Long Short Term Memory (LSTM)," *Wireless Personal Communications,* 2021.

[9] P. F. Muhammad, R. Kusumaningrum dan A. Wibowo, "Sentiment Analysis Using Word2vec and Long Short-Term Memory (LSTM) for Indonesian Hotel Reviews," dalam *Procedia Computer Science*, Semarang, 2021.

[10] S. prabha.K.S dan P. N. Karthikayan, "For Movie Reviews, A Sentiment Analysis using Long Short Term Memory Networks," *Turkish Journal of Computer and Mathematics Education,* vol. 12, no. 9, pp. 1758-1766, 2021.

[11] A. U. Rehman, A. K. Malik, B. Raza dan W. Ali, "A Hybrid CNN-LSTM Model for Improving Accuracy of Movie Reviews Sentiment Analysis," *Multimedia Tools and Applications,* vol. 78, no. 18, pp. 26597-26613, 2019.

[12] A. H. Ombabi, W. Ouarda dan A. M. Alimi, "Deep learning CNN–LSTM framework for Arabic sentiment analysis using textual information shared in social networks," *Social Network Analysis and Mining,* vol. 10, no. 1, 2020.

[13] W. Yue dan L. Li, "Sentiment analysis using word2vec-cnn-bilstm classification," dalam *2020 7th International Conference on Social Network Analysis, Management and Security, SNAMS 2020*, 2020.

[14] A. Giachanou dan F. Crestani, "Like it or not: A survey of Twitter sentiment analysis methods," *ACM Computing Surveys,* vol. 49, no. 2, 2016.

[15] Twitter, "Twitter API," 21 November 2021. [Online]. Available: https://developer.twitter.com/en/docs/twitter-api. [Diakses 07 March 2022].

[16] C. Shorten dan T. M. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning," *Journal of Big Data,* vol. 6, no. 1, 2019.

[17] C. Shorten, T. M. Khoshgoftaar dan B. Furht, "Text Data Augmentation for Deep Learning," *Journal of Big Data,* vol. 8, no. 1, 2021.

[18] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie dan L. Farhan, "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," *Journal of Big Data,* vol. 8, no. 1, 2021.

[19] S. Khan, H. Rahmani, S. A. A. Shah dan M. Bennamoun, "A Guide to Convolutional Neural Networks for Computer Vision," *Synthesis Lectures on Computer Vision,* vol. 8, no. 1, pp. 1-207, 2018.

[20] M. Bernico, "Deep Learning Quick Reference Useful hacks for training and optimizing deep neural networks with TensorFlow and Keras," dalam *Long Short Term Memory Networks*, Birmingham, Packt, 2018, pp. 290-293.

[21] P. K. Jain, V. Saravanan dan R. Pamula, "A Hybrid CNN-LSTM: A Deep Learning Approach for Consumer Sentiment Analysis Using Qualitative User-Generated Contents," *ACM Transactions on Asian and Low-Resource Language Information Processing,* vol. 20, no. 5, pp. 1-15, 2021.

[22] S. Liu, K. Lee dan I. Lee, "Document-level multi-topic sentiment classification of Email data with BiLSTM and data augmentation," *Knowledge-Based Systems,* vol. 197, 2020.

[23] T. Ho Huong dan V. Truong Hoang, "A data augmentation technique based on text for Vietnamese sentiment analysis," *Association for Computing Machinery,* 2020.

[24] B. Acharya dan K. Sahu, "Software Development Life Cycle Models: A Review Paper," *International Journal of Advanced Research in Engineering and Technology,* vol. 11, no. 12, pp. 169-176, 2020.

[25] Deepanshi, "In-depth understanding of Confusion Matrix," Analytics Vidhya, 18 May 2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/05/in-depth-understanding-of-confusion-matrix/. [Diakses 03 May 2022].

[26] A. C. Müller dan S. Guido, "Model Evaluation and Improvement," dalam *Introduction to Machine Learning with Python,* The United States of America, O'Reilly Media, Inc., 2017, pp. 282-284.

[27] ReqTest, "Black Box Testing – Understanding the Basics," 08 August 2019. [Online]. Available: https://reqtest.com/testing-blog/black-box-testing/#:~:text=Black%20box%20testing%20checks%20scenarios,on%20entering%20an%20incorrect%20password.. [Diakses 15 March 2022].

[28] M. Griberg, Flask Web Development: Developing Web Applications With Python, United States of America: O'Reilly Media Inc., 2018.

[29] V. Bonta, N. Kumaresh dan N. Janardhan, "A Comprehensive Study on Lexicon Based Approaches for Sentiment Analysis," *Asian Journal of Computer Science and Technology,* vol. 8, no. S2, pp. 1-6, 2019.

[30] F. Illia, M. P. Eugenia dan S. A. Rutba, "Sentiment Analysis on PeduliLindungi Application Using TextBlob and VADER Library," *ICDCSOS,* pp. 278-288, 2021.

[31] S. Sanyal dan M. Kumar Barai, "Comparative Study on Lexicon-based sentiment analysers over Negative sentiment," *International Journal of Electrical, Electronics and Computers,* vol. 6, no. 6, pp. 1-13, 2021.

[32] J. Wei dan K. Zou, "EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks," *arXiv preprint arXiv:1901.11196,* 2019.

[33] H. Bhasin, Python Basics: A Self-Teaching Introduction, Dulles: Mercury Learning and Information LLC., 2019.