CHAPTER 3

RESEARCH METHODOLOGY

3.1 Introduction

This chapter will provide a comprehensive explanation of techniques and process that will be implement in this research project. Subsequently, the library and packages used will be listed. This project framework consists of phases starting from identification of problem to data collection, data preprocessing, sentiment analysis model development, model hyperparameter tuning, until model evaluation measure and verification phase. However, the phase 1 research gap will not need be discussed in this methodology as it had been discussed in detailed in chapter 2 literature review. The detailed flow of the research work phase starting from data collection until model evaluation will be shown to ensure systematic workflow and resulting in effective completion of the research.

3.2 Phase 2: Data Collection

The second phase of this project is to collect a primary dataset in field regarding electric vehicle discourse. The process in this stage is to achieve objective 1. For the dataset collection, the data for this study were sources from Reddit and YouTube. Both of it were a global platform with diverse individuals from different social and economical backgrounds. Besides, both the social media environment is anonymous and real-time updated which provide a platform for more objective public opinions. To collect the project data, web scraping will be implemented to crawl posts and comments containing the keyword ‘electric vehicle’ and specific region of ‘Malaysia’. The Beautiful Soup python package is used. The data collected is as table 3.1 below. The architecture flowchart of web scarping is shown in figure 3.1 below.

Table 3.1 Description of Web Scrap Data

Variable	Meaning
timestamp	Date and time when the comment/post was published
post_title	Title of the Reddit post (for context in sentiment analysis).
video_title	Title of the YouTube video associated with the comment.
comment_text	Text content of the comment/post

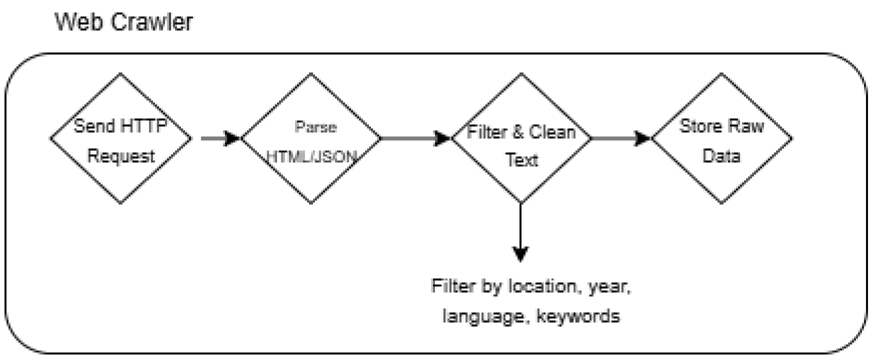

Figure 3.1 Architecture of Web Crawler

3.3 Phase 3: Data Preprocessing

In this phase, the collected data will be pre-processed and the objective 1 will also be fulfilled. The process of data preprocessing holds great significance in the field of natural language processing (NLP) by having a clean and structured data to ensure only relevant information is considered. Text data is often messy, noisy, and unstructured, which can affect the quality and accuracy of your NLP models learning algorithms. In this phase, the data cleaning techniques that required for text cleaning will be shown.

3.3.1 Remove Unwanted Character

Remove unwanted character such as punctuation, numbers, symbols, HTML tags, excessive whitespace, mentions and emojis. The characters will cause noise and ambiguity to the data, and may not relevant for the sentiment task. Regular expression can be used to remove the non-alphabetical by import the re module to use the 're.sub()' function. However, emoji and emoticon contain rich sentiment indicators but it requires the conversion of these to text for Bert to process. BERT tokenizer will split the emojis which converted to text into sub words as a sentiment clues. Hence, the retain or removal of some unwanted character is considerable but the irrelevant one will be prior to be remove.

3.3.2 Language Filtering

As the data were web scrap from platform which can be access globally, language filtering is required to only cater preferred language for this project. The preferred language for this project is scope to be English only. However, as Malaysia is a multilingual country with diverse population, the discourse in Malaysia sometimes includes a mix of Malay and English or Chinese with English. Hence, language detection is applied to either remove unmeaningful text that is not English language or retain meaningful sentiment value by translating non-English text to English language.

The language detection library in Python ‘langdetect’ which is port of Google’s language detection library can be install to identify the language of a given text. The detect() function of ‘langdetect’ library is capable in identification of 55 languages. Besides, there is also another open-source python library for language detection, which is ‘lingua_py’. It can detect a total of 75 languages. Next, the detected non-English text can be translated by using translation library such as ‘googletrans’ or ‘DeepL’. Google translation is optimal for general translation while DeepL is suitable for technical translation. There is research shows that DeepL translation library accuracy is more highly accurate and nuanced translation as compared to Google translation library.

3.3.3 Text Normalizations

Normalization is process in converting text data into standardized and consistent format. It helps reduce the complexity and variability of the text. The normalization process involves stemming, lemmatization, stop word remover and lowercase. Stemming and lemmatization can help group words with similar meanings thus reduce the size and complexity of the vocabulary. Stop word remover can reduce the redundancy of the text by removing the most frequently uninformative word such as articles, prepositions, conjunctions and pronouns. Besides, lowercase can reduce vocabulary size as different algorithms will classifies uppercase letter and lowercase letter such as ‘Hello’ and ‘hello’ as 2 different words. Those normalization process can be done through import of Natural Language Toolkit (NLTK) or spaCy library; which is advance Natural Language Processing in Python.

However, that text normalisation is needed before implementation of model training using approach such as traditional machine learning model or deep learning model. Whereas, for Bert-based model some text normalisation is not needed. The implementation and method for BERT in text normalization is shown as below:
1. For Word lemmatization and stemming:

In BERT, lemmatization and stemming is not needed as BERT implemented a tokenization method named WordPiece Tokenization. This is particularly useful for handling unseen words and deal effective with compound or morphologically complex words which could maintain sematic details than stemming. Breaking down the words into subword units rather than relying on whole words. Handling subwords by initially splitting words into characters with a ## prefix for all.

For example the word ‘hugging’ is token into ‘h’, ‘##u’, ‘##g’, ‘##g’, ‘##i’, ‘##n’, ‘##g’. Then it merged subword pairs iteratively and used the formula:

$$score = \frac{(freq\_of\_pair)}{(freq\_of\_first\_element \times freq\_of\_second\_element)} \tag{3.1}$$

If given a corpus with (’hug’, 10), (’pug ’, 5), (’pun ’, 12), (’bun’, 4) and (‘hugs’, 5), word piece start with the token such as ‘h’, ‘##u’, ‘##g’ and learn merge such as (‘##u’, ‘##s’) → ‘##gs’, then (‘h’, ‘##u’) → ‘hu’, followed by (‘hu’, ‘##g’) → ‘hug’, until it form full sub words like hugging through optimized merges.

2. For Lowercasing or Uppercasing:

During preprocessing for lowercasing and uppercasing, need to encounter whether or not to retain the casing before training using Bert-based model. As some Bert-based model such as BERT-based-cased, BERT-based-uncased and BERT-based-multilingual-cased had different implementation requirement on capitalization.

For BERT-based-cased, it will train on task with retaining its capitalization either uppercase or lowercase. It ensure better performance for named entity recognition. While for BERT-based-uncased, it will convert the text to lowercase itself. For BERT-based-multilingual-cased, lowercasing of word is not applied.

3. For Stopword Remover:

During preprocessing for informative stopword remover is not beneficial especially for sentiment analysis using BER-based model. As Bert is a deep contextual language model, which capable in understanding the meaning of a word based on its surrounding context. Stopword such as is or the often had syntactic or grammatical importance that helps BERT to capture the sentence structure. Besides, stopword is semantically for sentiment classification. For example, removing of the word ‘not’ in the sentence would influence the semantic meaning of the sentence. Nevertheless, BERT’s WordPiece tokenizer can effectively handle the stopword as it often short and frequent.

3.4 Phase 4: Feature Extraction

In text classification and prediction, Natural Language Processing (NLP) is one of the fields in computer science essential to use for linguistic machine learning study. Vectorization or text representation is one of the ways used to extract features from text and hence undergo text vectorization which converts text into a vectorized numerical value and is used as input for model training of the features this method is usually called feature extraction.

3.4.1 BERT-based feature extraction

The feature extraction process in BERT start from sequence encoding , which the raw text is transformed into structured numerical inputs. After tokenization of the sentence using WordPiece tokenization, the input is represented as input_ids. Input_ids is the input sentence in the form of sequence of integer indices. Special tokens such as CLS (classification), SEP (separator), and PAD (padding) are embedded within these indices to mark sequence boundaries and fill shorter inputs. CLS are at beginning of the sentence while SEP are at the separate segment of sentence. To differentiate real tokens from padding, an attention_mask is generated, assigning 1 to valid tokens and 0 to PAD placeholders. This is to ensure the model ignores padded positions during self-attention calculations. In addition, for tasks involving paired sequences such as question-answering, token_type_ids are introduced: 0 for tokens in the first sentence and 1 for the token in the second sentence. This would enabling the model to distinguish between it.

These sequence which are input_ids, attention_mask and token_type_ids is then been processed into token embeddings. It are summed with positional embeddings which are learned absolute position indicators and segmentation embeddings which are sentence identifiers. This combined input representation is the forward into BERT's based which contain 12 encoder layers where multi-head self-attention dynamically weights contextual relationships. For example, linking the token 'love' to 'Tesla' in 'I love Tesla'. While the feed-forward networks refine local interactions. The final state which is particularly the CLS token's representation, it encode task-specific features and ready for downstream classification. Hence, this structured encoding ensures the model captures nuanced semantics while handling variable-length inputs efficiently.

3.5 Phase 5: Build Bert-based Electric Vehicle Sentiment Analysis Model

In this stage, the model in classifying the sentiment analysis of electric vehicle discourse will be build using pre-trained BERT-based model. This phase is implement to achieve objective 2 by comparison of which BERT-based model is better in performance of sentiment classification for topic regarding electric vehicle discourse.

3.5.1 BERT-based uncased

The BERT-based uncased model consist of 12 transformer encoder layers that processed the lowercase input sentence using token, position and segmentation embedding. Each layer apply the multi-head self-aatttention and a feed-forward network to produce contextual embedding. The final CLS token representing the whole sentence. The CLS vector is then passed through a dense layer with softmax activation and optimised using cross entropy loss for sentiment classification. The architecture flow of BERT-based uncased model is shown as figure below.

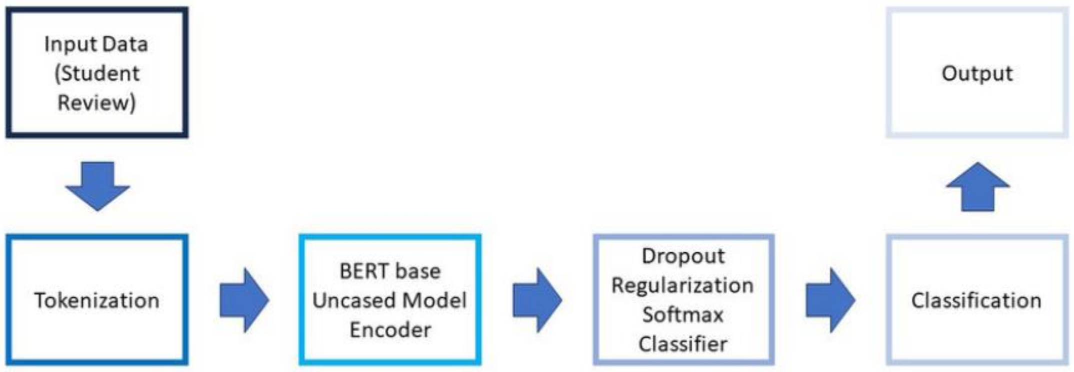
Input Data (Student Review)
Tokenization
BERT base Uncased Model Encoder
Dropout Regularization Softmax Classifier
Classification
Output

Figure 3.2 Example of BERT-based uncased model

3.6 Phase 6: Experiment on Hyperparameter

In this phase will show the hyperparameter tuning that can be implemented for increased the performance of the model.

3.6.1 Epoch

Epoch is the tuning of the number of complete passes through the training dataset. This is to act as a control how many timed the model can learn from the entire dataset. It cannot be set too low to avoid underfitting, or either too many epochs which would lead to overfitting.

3.6.2 Learning Rate

Learning rate is the size of step at each iteration to update the model weights. It is used to determine how quickly the model converges. It is typicaaly in range of 2e-5 to 5e-5 for BERT. Besides, learning rate scheduler such as linear decay and warm up can also be implement. The learning rate formula is as shown below.

$$\theta = \theta - \eta \cdot \nabla \theta L(\theta) \quad (3.2)$$

Where θ = model parameter, η = learning rate, ∇θL = gradient of loss

3.6.3 Optimizer

An optimizer such as BERTAdam (AdamW) optimizer can be implemented. The formula is shown below.

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \\ \widehat{m_t} &= \frac{m_t}{1-\beta_1^t}, \widehat{v_t} = \frac{v_t}{1-\beta_1^t} \\ \theta_t &= \theta_{t-1} - \eta \times \frac{\widehat{m_t}}{\sqrt{\widehat{v_t}} + \in} \end{aligned} \quad (3.3)$$

Where:

- θt: Parameters (weights) at iteration t
- gt: Gradient of the loss function L(θt)
- η: Learning rate (step size)
- $\beta 1, \beta 2$: Exponential decay rates for first and second moments

- ϵ: Small constant to avoid division by zero (numerical stability)
- λ: Weight decay coefficient (L2 regularization)

3.7 Phase 7: Model Evaluation and Performance Metrics Evaluation

In this phase, the evaluation measurement will be used to evaluated the experiment output from phase 6 and 7. The evaluated output of each model comparison and hyperparameter tuning of each model will be recorded and further analysis for each respective performance based on performance metric will be conducted. This phase is to achieve objective 3, where the analysis on the sentiment insight of Malaysian discourse about the electric vehicle can be evaluated. Evaluate how well the BERT model performs in identifying positive, negative and neutral sentiment.

The common evaluation metric for sentiment analysis classifier is accuracy, precision, recall and F1 score, confusion matrix and Matthew's correlation coefficient. The definition and formula for the evaluation metric is show as below:

(a) Accuracy:

Accuracy indicates the proportion of correctly predicted instance over the total number of predictions. It is mathematically defined as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.4}$$

(b) Precision:

Precision metric is employed to address the inadequacies of Accuracy. It indicates how accurate the percentage of positive predictions. High precision indicates a low false positive rate. The formula for calculating Precision of binary and multi-class classification is as below:

Binary Classification Equation:

$$Precision = \frac{TP}{TP + FP} \tag{3.5}$$

Multi-Class Precision (Averaged over all classes):

$$Precision_{macro} = \frac{1}{N}\sum_{i=1}^{N}\frac{TP_i}{TP_i + FP_i} \tag{3.6}$$

(c) Recall

Recall is the ratio of how many correctly predicted objects to the total number of predicted objects. This metric is utilized to evaluate the model's capacity to forecast positive outcomes.

Binary Classification Equation:

$$Recall = \frac{TP}{TP + FN} \tag{3.7}$$

Multi-class Recall Equation:

$$Recall_{macro} = \frac{1}{N}\sum_{i=1}^{N}\frac{TP_i}{TP_i + FN_i}\frac{TP}{TP + FN} \tag{3.8}$$

(a) F1-score

F1-score is the average of precision and recall, considering both false positive and false negative. It is more useful than accuracy when dealing with imbalance class distribution. Accuracy is only ideal when the false positive and false negative are similar.

Binary Classification Equation:

Multiclass F1-score Equation:

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{3.9}$$

$$F1 - score_{macro} = \frac{1}{N} \sum_{i=1}^{N} 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{3.10}$$

(b) Matthew's Correlation Coefficient (MCC)

Matthew's correlation coefficient (MCC) or phi coefficient is the performance metrics and model evaluation measurement of the correlation of the true classes with the predicted labels. This metric is used in the machine learning field as binary and multiclass classification quality measurement. The MCC

$$MCC = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{3.11}$$

measurement is distributed unevenly when the datasets are imbalanced (Zhu, 2020) and for the unequal prediction class frequencies. (Jones & Ward, 2003)

(c) Confusion Matrix

Confusion matrix is a two-dimensional matrix which is used to evaluate and inspect errors in classification cases and is also used as the tuning parameters for threshold detection. Confusion matrices can lead to multi-class calculation with $n \times n$ matrices. The misclassified items for each pair of the class elements are encoded using a confusion matrix. From Figure there are True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) together to form a matrix configuration. The TP and TN show that the model correctly predicts and the actual conditions are positive and negative respectively. However, the FP refer to the models that provide wrong predictions of the negative class (predicted condition as a positive result while actual condition as a negative result) while the FN refer to the models that provide wrong predictions of the positive class (predicted condition as a negative result while actual condition as a positive result).

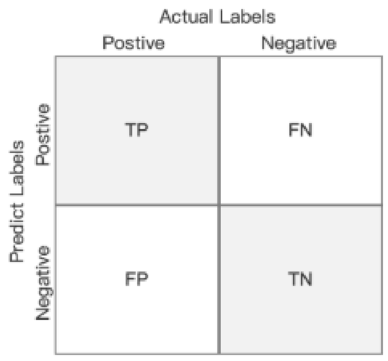

Figure 3.3 Confusion Matrix

3.8 Phase 8: Final Report and Project Wrap Up

Lastly, the whole project report will be wrapped up. Future work and limitation of this project will also be discussed.

3.9 Conclusion

This chapter provided a detailed flow of each phases in the methodology for building sentiment classifier od electric vehicle using BERT-based model. The flowchart and project framework of the methodology is shown in figure 3.4 and figure 3.5 below.

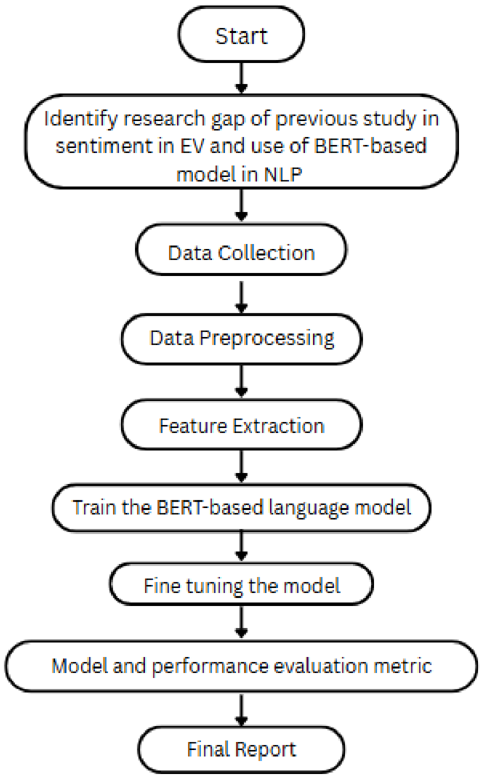
Start
Identify research gap of previous study in sentiment in EV and use of BERT-based model in NLP
Data Collection
Data Preprocessing
Feature Extraction
Train the BERT-based language model
Fine tuning the model
Model and performance evaluation metric
Final Report

Figure 3.4 Flowchart of Methodology Phases

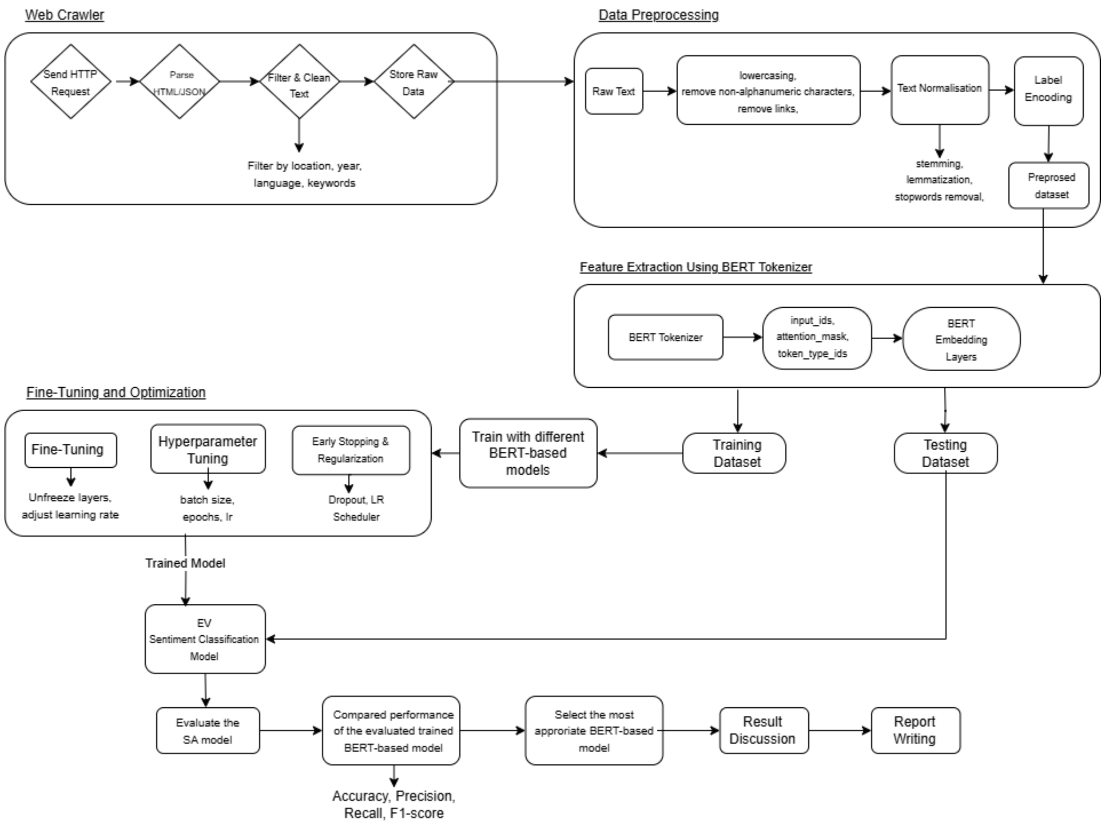
Web Crawler
Send HTTP Request
Parse HTML/JSON
Filter & Clean Text
Store Raw Data
Filter by location, year, language, keywords
Data Preprocessing
Raw Text
lowercasing, remove non-alphanumeric characters, remove links,
Text Normalisation
stemming, lemmatization, stopwords removal,
Label Encoding
Preprosed dataset
Feature Extraction Using BERT Tokenizer
BERT Tokenizer
input_ids, attention_mask, token_type_ids
BERT Embedding Layers
Training Dataset
Testing Dataset
Train with different BERT-based models
Fine-Tuning and Optimization
Fine-Tuning
Unfreeze layers, adjust learning rate
Hyperparameter Tuning
batch size, epochs, lr
Early Stopping & Regularization
Dropout, LR Scheduler
Trained Model
EV Sentiment Classification Model
Evaluate the SA model
Compared performance of the evaluated trained BERT-based model
Accuracy, Precision, Recall, F1-score
Select the most approriate BERT-based model
Result Discussion
Report Writing

Figure 3.5 Project Framework