Normalisation of Indonesian-English Code-Mixed Text and its Effect on Emotion Classification

Evi Yulianti, Ajmal Kurnia, Mirna Adriani
Faculty of Computer Science
Universitas Indonesia
Indonesia

Yoppy Setyo Duto
Faculty of Economy and Business
Universitas Mercu Buana
Indonesia

***Abstract*—Usage of code-mixed text has increased in recent years among Indonesian internet users, who often mix Indonesian-language with English-language text. Normalisation of this code-mixed text into Indonesian needs to be performed to capture the meaning of English parts of the text and process them effectively. We improve a state-of-the-art code-mixed Indonesian-English normalisation system by modifying its pipeline modules. We further analyse the effect of code-mixed normalisation on emotion classification tasks. Our approach significantly improved on a state-of-the-art Indonesian-English code-mixed text normalisation system in both the individual pipeline modules and the overall system. The new feature set in the language identification module showed an improvement of 4.26% in terms of F_1 score. The combination of machine translation and ruleset in the lexical normalisation module improved BLEU score by 25.22% and lowered WER by 62.49%. The use of context in the translation module improved BLEU score by 2.5% and lowered WER by 8.84%. The effectiveness of the overall pipeline normalisation system increased by 32.11% and 33.82%, in terms of BLEU score and WER, respectively. Code-mixed normalisation also improved the accuracy of emotion classification by up to 37.74% in terms of F_1 score.**

***Keywords*—Code-mixed normalisation; Indonesian; English; emotion classification**

I. Introduction

One common form of the phenomenon of multilingualism is code-mixing. It is a linguistic phenomenon that mixes two or more language variations in one utterance [1]. This phenomenon can be found in various contexts, including social media [2], news articles [3], lectures [4], and even sermons [5].

Indonesia is highly multilinguistic, with more than 700 languages spoken across the nation. A 2015 report noted that 57.5% of Indonesian people are bilingual and 17.4% are trilingual, among whom the most popular language combination is Indonesian, English and Javanese.[1] Multilingualistic phenomena such as code-mixing have recently become increasingly common due to more widespread usage of the internet, especially social media.

Recently, code-mixing of Indonesian and English has become popular in Indonesia and has become widely known as *"Bahasa Anak JakSel"*. JakSel stands for "*Jakarta Selatan*" ("South Jakarta"); thus "*Bahasa Anak JakSel*" essentially means "a language of South Jakarta teenagers", so named because combining English and Indonesian words or phrases first become popular amongst teenagers in South Jakarta.

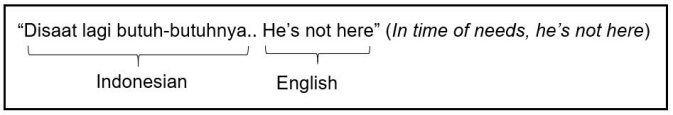

Fig. 1. Example of Indonesian-English Code-mixed Text.

A previous linguistic study found there are four main reasons why Indonesian-Englis code-mixing (i.e. "*Bahasa Anak JakSel*") has become a trend today [6]. The first is language pride: people in Indonesia who can speak or write English are considered prestigious, since not all Indonesians have mastered English. The second is social status and style: some groups frequently use some English words in their conversations because of social demands that arise if many people in the group are also used to speaking English. The third is the existence of English words that can not be translated into Indonesian: some topics are much more familiar and easier to discuss using English terms instead of the equivalent Indonesian expressions. The fourth is the desire to increase English vocabulary use: practicing using English words in conversation or writing helps people master English.

There are no noticeable differences in how code-mixing occurs for Indonesian-English versus other language pairs. English words that are mixed with the other language may be nouns, verbs or adverbs, among other parts of speech (POSs). Like code-mixing in other language pairs, Indonesian-English code-mixing can happen within a sentence (intra-sentential), across different sentences (inter-sentential) or even within a word (intra-lexical/sub-word). The distinction between Indonesian-English code-mixed text and code-mixed text in other language pairs mainly involves the syntactic and semantic aspects of the languages themselves. Fig. 1 illustrates an example of an Indonesian-English code-mixed tweet.

The increase of code-mixed text in social media has invoked research interest in studying various forms of processing such text. However, code-mixed text is more difficult to analyse than monolingual text. Analysing code-mixed text using only one language means that information written in other languages cannot be analysed effectively. On the other hand, analysing code-mixed text using two languages increases the complexity of the model due to the larger amount of vocabulary that needs to be processed and the different

[1]https://www.inlingua-edinburgh.co.uk/how-multilingual-is-your-country/

language characteristics that need to be taken into account. A simple alternative option is to translate the code-mixed text to monolingual text. Translation of code-mixed text has been done previously for various language pairs, including Hindi-English [7], Chinese-English [8], Arabic-English [9] and Indonesian-English [10].

Barik et al. [10] built a system pipeline to normalise and translate Indonesian-English code-mixed tweets. The system pipeline consists of four modules: tokenisation, language identification, lexical normalisation, and translation. We see some potential points of improvement for Barik et al.'s code-mixed normalisation pipeline. First, in the language identification module, the features used in the language identification process could be expanded to improve the identification results. Second, in the lexical normalisation module, Barik et al.'s method – which only relies on a ruleset and a lexicon – was unable to determine whether an ambiguous word was a formal word or slang; their method always considered the word slang if it appeared in the lexicon. To overcome this issue, we propose combining a machine translation (MT) approach with a ruleset to perform lexical normalisation. Finally, in the translation module, Barik et al.'s method translates each token separately, meaning that the system lacks context when performing the translation. This research tackles this problem by translating neighboring tokens simultaneously.

Although extensive research has been performed on code-mixed text normalisation, very few studies have actually investigated the effect of this normalisation on text processing. Goot and Çetinoğlu [11] studied the effect of code-mixed normalisation on a POS-tagging task for Turkish-German text and obtained significant improvement in task performance as a result of normalisation. Singh et al. [12] investigated the effect of code-mixed text normalisation on sentiment analysis and POS-tagging for several language pairs, including Bengali-English, Hindi-English, and Tamil-English. Their results showed that code-mixed normalisation improved performance on both tasks.

No research has yet investigated the effect of Indonesian-English code-mixed text normalisation on a specific text-processing task. Therefore, in this research we also analysed the effects of code-mixed normalisation on an emotion classification task using social media data. Emotion classification tasks often rely on detecting certain phrases or words that signify emotions. Therefore, performing this task on code-mixed data is difficult when the emotion phrases or words are written in different languages. We argue that first normalizing code-mixed text into monolingual text may have benefits for emotion classification. In this work, we examine the extent to which code-mixed normalisation affects the accuracy of emotion classification systems. In general, our research questions are as follows:

1) To what extent the addition of some features to the language identification module, the combination of MT & rule-based approaches in the lexical normalisation module and the addition of context to the translation module can improve the state-of-the-art pipeline system for Indonesian-English code-mixed text normalisation?
2) Does the use of code-mixed text normalisation system affect the accuracy of emotion classification system?

II. Related Work

A. Code-mixed Text Normalisation

The simplest way to normalise and translate code-mixed text is by using an existing MT system to translate the portion of the text that is in a foreign language. Patel and Parikh [13] translated Gujarati-English code-mixed text to Gujarati. The Gujarati token was transliterated using a handcrafted dictionary, whereas the English token was translated using an existing MT system. Dhar et al. [7] proposed using an existing monolingual translation system using the Matrix Language-Frame (MLF) model to translate Hindi-English code-mixed text. Their approach was adapted from the MLF linguistic theory developed by Myers-Scotton [14], which involves first determining the matrix language (dominant language) and embedding language (non-dominant language), then translating the token in the embedding language to the matrix language and finally translating the text to a desired language, if needed.

Another way to translate code-mixed text is by using a parallel corpus of code-mixed and normalised text to train an MT system. Menacer et al. [9] used this approach to translate Arabic-English code-mixed text into English. Mahata et al. [15] used deep learning for Bengali-English code-mixed normalisation, employing a character-level long-short term memory network to perform language identification. Next, the English text was translated using a neural MT system, whilst the Bengali text was transliterated back to its Devanagri form. Finally, bigram and trigram language models were used to reorder the tokens to fix grammatical errors.

Relatively very few studies have explored code-mixed normalisation for Indonesian-English text. The only work that we could find on Indonesian-English code-mixed normalisation was Barik et al.'s study [10], which performed code-mixed normalisation using a pipeline system consisting of four modules: tokenisation, language identification, lexical normalisation, and translation. They used the condition random field (CRF) sequence labelling model for the tokenisation and language identification modules; rule-based, spelling correction and word embedding for the lexical normalisation module; and MLF (as in [7]) for the translation module. In this work, we wanted to improve Barik et al.'s code-mixed normalisation system by modifying its pipeline modules.

B. Emotion Classification

Emotion classification is a specific task in natural language processsing to identify emotion contained in a particular document. The dataset used for emotion classification can be in a variety of forms, including video [16], speech [17], text [18], or even electroencephalography signals [19]. Classes or categories for emotion classification tasks are usually limited to basic emotions as defined based on research in the field of psychology. The emotion theory developed by Ekman [20] proposed six basic emotions: anger, joy, sadness, fear, disgust, and surprise. There are also Shaver's [21] six basic emotions, according to which love is categorized as a basic emotion instead of disgust. Another popular emotion theory is Plutchik's [22] eight basic emotions, which are the same as in Ekman's theory but with the addition of anticipation and trust.

A popular approach to performing emotion classification based on textual data is to utilise an emotion lexicon containing

TABLE I. THE STATISTIC OF CODE-MIXED TEXT DATASET FOR CODE-MIXED NORMALIZATION

Item	Value
#Tweet	825
#Token	22,725
#Character	105,955
#Indonesian Token	11,200
#English Token	5,608
Code Mixing Index (CMI)	19.4

a list of words and the emotion associated with each word. The lexicon can be either semi-automatically generated or manually handcrafted [23], [24]. The emotion label of a document can be determined by computing the point mutual information (PMI) value of affect words [25]. The lexicon can be used as a feature for supervised machine learning models in combination with other features, such as POS tags.

Saputri et al. [18] performed emotion classification on a dataset of Indonesian tweets. They classified each tweet as belonging to one of five emotion categories: angry, happy, sad, fear, and love. They explored various features, including emotion lexicon, sentiment lexicon, bag of words, word embedding, POS tags and morphological information. In this work, we use the five emotion labels used by Saputri et al. [18] in our emotion classification task.

III. DATASET

The dataset used in this research is taken from previous research [10]. It consists of 825 Indonesian-English code-mixed tweets along with annotations for all steps in their pipeline normalisation system (i.e. tokenisation, language identification, lexical normalisation and translation). The statistics of the dataset are presented in Table I. The data consist of 22,725 tokens and 105,955 characters. The average tweet lengths is 27.54 tokens and the average token length is 4.66 characters. There are 11,200 tokens in Indonesian (49.28% of overall tokens) and 5,608 in English (24.67% of overall tokens).

The code-mixing index (CMI) value for this dataset was 19.44. CMI is a metric used to measure how often code mixing occurs in a text [26]. This CMI value is rather high; almost 20% of the overall non-neutral language tokens in this dataset are code-mixed text.

To analyse the effect of code-mixed normalisation on emotion classification, we needed a ground truth of emotion labels for each Indonesian-English code-mixed tweet in our dataset. For this purpose, human annotation was performed to assign emotion labels to the tweets. This process was conducted by two annotators (both master's students in computer science who were familiar with the annotation tasks). Before performing the annotations, the annotators were given guidance on how to label the emotions for each tweet. Our annotation procedure followed previous research on emotion classification using Indonesian tweets [18]. Each tweet in the dataset was assigned to one of seven possible emotion labels: anger, joy, sadness, fear, love, mixed and neutral. Tweets labeled as mixed or neutral were filtered out and were not used in the emotion classification task; neutral tweets were those not associated with any emotion, whereas mixed label was assigned to tweets to which multiple emotion labels could be applicable, which is beyond the scope of this task. The annotators achieved a Cohen's kappa coefficient of 64.72 before consolidating the final label for each tweet, representing substantial inter-rater agreement according to Landis and Koch [27]. Our code-mixed dataset for emotion classification has been made available for research purposes.[2]

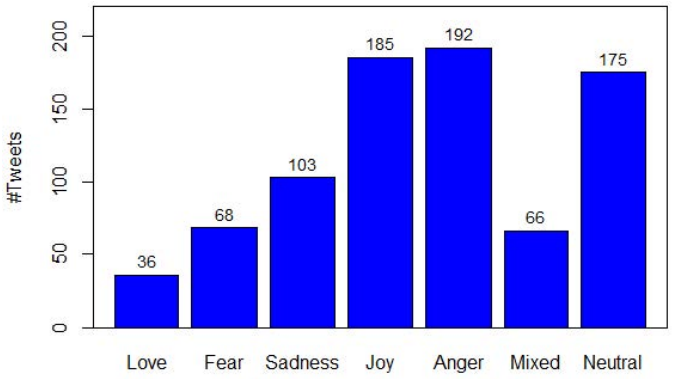

Fig. 2. The Distribution of Emotion Labels in Our Dataset.

TABLE II. THE STATISTIC OF CODE-MIXED TEXT DATASET FOR EMOTION CLASSIFICATION

Item	Value
#Tweet	584
#Token	16,939
#Character	77,041
#Indonesian Token	8,539
#English Token	4,289
Code Mixing Index (CMI)	24.26

The distribution of emotion labels after the annotation process is shown in Fig. 2. In total, 66 tweets were labelled as mixed and 175 as neutral, meaning that 241 of the 825 tweets from the original dataset could not be used in the emotion classification task. Fig. 2 also indicates imbalances in the emotion labels in the dataset. The emotion associated with the smallest number of tweets is love, with 36 tweets. In comparison, tweets expressing anger and joy were associated with the highest number of tweets (192 and 185, respectively). Meanwhile, 68 tweets were labelled as expressing fear and 103 as expressing sadness.

The summary statistics of the final dataset for emotion classification task are shown in Table II. The emotion classification dataset was reduced to 584 tweets with 16,939 tokens (including 8,539 Indonesian tokens and 4,289 English tokens) and 77,041 characters. The CMI value increased to 24.26, meaning that, on average, code-mixing occurred more often in this dataset compared to the original dataset (see Table I for comparison).

The other dataset used in this research was a corpus of Indonesian-English code-mixed tweets, consisting of 900,000 Indonesian-language tweets and 1.6 million English-language tweets. This corpus was taken from Barik et al.'s work and was used as training data to build word embedding for the emotion classification process.

[2]https://github.com/ir-nlp-csui/CodeMixedEmotion

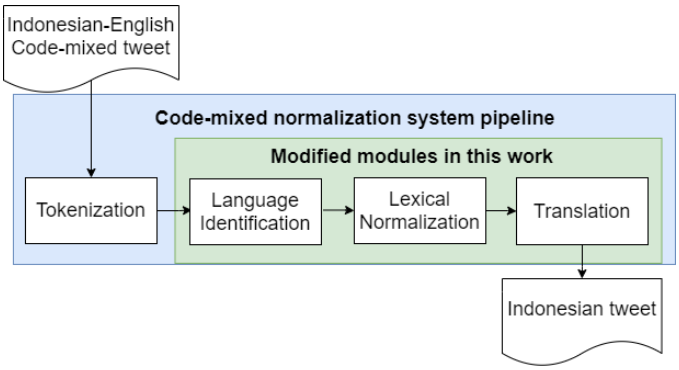

Fig. 3. The Framework of Indonesian-English Code-mixed Normalisation Pipeline.

IV. Methods

A. Code-Mixed Normalisation

Given Indonesian-English code-mixed text, our normalisation system will transform the text into Indonesian. In general, our approach follows Barik et al.'s [10] pipeline system, which consists of four sequential modules: tokenisation, language identification, lexical normalisation, and translation. We choose to improve their pipeline because it is the most recent work in Indonesian-English code-mixed normalisation. They proposed the use of four modules in an attempt to perform comprehensive preprocessing (by adding dedicated tokenisation and lexical normalisation modules) before translating code-mixed text into one language. Some previous researchers have used simpler methods than Barik et al.'s, including Patel and Parikh [13], Dhar et al. [7], and Menacer et al. [9], who essentially omitted the special tokenisation (by tokenising simply using space characters), lexical normalisation, and/or language identification modules. We decided to use four modules in our system because Barik et al. has reported that the use of all four modules was more effective than using only some modules. Three of their modules are modified in this work with the aim to improve their system. Briefly, our modifications are as follows: (i) adding some features of the language identification module, (ii) using different approaches by combining rule-based and MT approaches in the lexical normalisation module, and (iii) adding context to the MLF approach in the translation module.

The flow of our system's pipeline is illustrated in Fig. 3. The modules of Barik et al.'s pipeline are shown in blue box. The modules modified in this work are shown in the green box. The following subsections explain the details of each module in our code-mixed normalisation pipeline system and how it differs from previous research.

1) Tokenisation: Tokenisation is the process of splitting text into tokens, which can be a single word or multi-word expression. Our tokenisation method follows the same procedure as Barik et al.; in other words, we did not make any changes to this module. The strategy is to classify each character in the text to a beginning–inside–outside (BIO) tag label. The label is then used to split the text into tokens. The label B (for "beginning") represents a character at the beginning of a token, the label I (for "inside") represents characters in the middle of a token, and the label O (for "outside") represents characters not in any tokens. For example, the BIO tag label for the text "top text" is "BIIOBIII", so the sequence token result is "top" and "text". This approach has some advantages compared with the common tokenisation method of using whitespace as the delimiter of a token. This is supported by Barik et al.'s results, which showed that this approach was more effective than tokenisation using whitespace and a ruleset (e.g. the TweetTokenizer module from the nltk package) for informal text with high levels of noise. In total, there are 8 features used in this module.

2) Language Identification: This module identifies the language for each token. This process is important because, in this code-mixed normalisation work, the English part of the Indonesian-English code-mixed text will be translated into Indonesian. Therefore, we need to identify which tokens are in Indonesian and which are in English in the code-mixed text. The language identification method used by Barik et al. applied a sequence labelling approach. A CRF model is trained to identify the language for each token based on the tokenisation result. The language labels are "id", "en", and "un". The label "id" represents a token in Indonesian, "en" represents a token in English, and "un" refers to a language-neutral token (i.e. a token that cannot be identified as either Indonesian or English, such as punctuation, named entities, emoji, and tokens in other languages).

This research expands the language identification features from Barik et al.'s work. The window token feature was expanded to window size of 4 to capture more contextual information and it showed good performance in our early experimental results. Additional questions such as "is title case" and "has punctuation" were added to the token morphology feature to enrich the morphology information. The previous research only used the character n-gram feature with the size of 5. We believe that this restriction prevents the model from capturing the variety of unique character sequences and other sub-word information that is unique in each language, such as affixes. Affixes in English are very different from affixes in Indonesian. To capture this information, we decided to add more character n-grams to the feature set to add more sub-word information. Specifically, we used character n-grams ranging from unigram through 6-gram as features. Lastly, we added one feature, called morphology sequence, to better understand how the morphology of each character changed in a token. This feature is obtained by converting all lowercase and uppercase alphabet characters to "a" and "A" (respectively), all numeric characters to "0", all whitespace characters to " ", all punctuation to "-" and removing character repetition. In total, 19 features were used in this module.

3) Lexical Normalisation: Lexical normalisation is the process of transforming informal or slang words into their formal variants. The previous model by Barik et al. used a combination of rule-based and spelling-correction approaches for this process. Their ruleset was composed of six rules that normalise tokens containing specific language. Spelling correction normalises tokens that are not in the formal lexicon using edit distance and word embedding. Our model replaces the spelling-correction part of the model with an MT approach.

Our method for lexical normalisation combines rule-based and MT approaches. The rule-based method was applied as a pre-normalisation stage before the text was inputted into the

TABLE III. RULESET FOR LEXICAL NORMALISATION

No	Rule Description	Lang	Example
1	Reduce character repetition to maximum 2	id, en	"*aaamiiin*" => "*aamiin*" (amen)
2	Reduplicate token ends with character "2"	id	"*baik2*" => "*baik-baik*" ("fine")
3	Add "-" in the middle of a duplicate token	id	"*baik baik*" => "*baik-baik*" ("fine")
4	Extend contracted words using Kooten	en	"I'm" => "I am"
5	Delete prefix "*nge-*"	en	"*ngevote*" => "vote"
6	Delete suffix "*-nya*" and add "the" at the start of the token	en	"*jobnya*" => "the job"

MT stage. This idea is supported by previous work [28], [29], [30]. Veliz et al. [28] found that a ruleset could be used for lexical normalisation as a pre-normalisation step before the text was processed using MT. Kurnia and Yulianti [29] confirmed that a ruleset could be applied and proved to be useful for lexical normalisation. Yulianti et al. [30] applied rule-based MT (RBMT) before the text was inputted into statistical MT (SMT), and showed that the resulting hybrid MT was more effective than using RBMT or SMT alone.

The ruleset used in this work is taken from Barik et al.'s work. The details of the ruleset are listed in Table III. In total, six rules are used in our ruleset, which only applies to alphanumeric tokens with a minimum of one alphabet character. It does not apply to emoji, emoticons, URLs, hashtags, and user tags. This restriction was implemented to avoid transforming already-formal tokens, known as over-normalisation.

The goal of the first rule is to reduce the amount of character repetition variations of a token. This rule applies to all non-restricted tokens, regardless of language. The second and third rules normalise informal reduplicated words (applied only to tokens in the Indonesian language). The fourth rule extends contracted English words. The fifth and sixth rules normalise sub-word code-mixed tokens. For the Indonesian-English language pair, sub-word code-mixing occurs when combining English words with Indonesian affixes.

The MT approach to lexical normalisation works by translating text in informal language to formal language. This approach has been used to normalise text in various languages, such as English [31] Dutch [28], and Indonesian [32]. The MT model used in this research is SMT with a phrase translation unit, also known as phrase-based statistical MT (PBMT). The PBSMT model was implemented with the mosesdecoder tool [33]. The alignment model used for the PBSMT model is IBM Model 5 with MGIZA [34]. The language models (3-gram, 4-gram, and 5-gram) were trained with a normalised corpus on training data using KenLM [35]. Tuning was performed in batches with the margin-infused relax algorithm (MIRA) using bilingual evaluation understudy (BLEU) as a tuning metric [36]. Special tokens – such as URLs, hashtags, and user tags – were converted to "[URL]", "[HASHTAG]", and "[USER]" in the training and tuning process.

4) Translation: The translation module translates code-mixed text to monolingual text – here, Indonesian-English code-mixed text to Indonesian-language text. We slightly modified the MLF approach used by Barik et al. by grouping neighbouring tokens with the same language in the text as one language segment; thus the translation is not performed for each token, but for each segment, similar to the approach taken in [15]. Because Barik et al. separately applied translation to individual tokens, it potentially produces incorrect translation results since it prevents the MT system to know the correct context of the token. In our method, we added context to the MT by translating a group of neighboring tokens together. The use of language partitions serves as context during translation to improve translation results.

Given the code-mixed text input, we first determine the text's matrix and embedding languages. Neighboring tokens with similar language are then grouped to form one segment. Using MLF method, each segment from the embedding language is translated to the matrix language. If the embedding language is English and the matrix language is Indonesian, translate the English segment into Indonesian. If the embedding language is Indonesian and the matrix language is English, first translate the Indonesian segment into English, then translate the overall text into Indonesian (This is important because the goal of this work is to normalise Indonesian-English code-mixed text into Indonesian).

The difference between the MLF method used in previous work by Barik et al. and that used in this work is illustrated in Fig. 4. Previous work separately applied translation to individual tokens, which may result in translation errors. In our method, we added context to the MT by translating a sequence of tokens together. As shown in the figure, using our modified MLF, the English text "It is not that hard" is translated as a unit and produces an accurate translation result in Indonesian, whereas the MLF used in previous work produces a translation error as a result of translating each token separately.

B. Emotion Classification

The emotion classification process begins by preprocessing data using a code-mixed normalisation system. In this research, we compared two different classification systems. The first uses a classic approach, following the research in [18]. The second is a more modern approach based on deep learning.

The first system uses word embedding as a feature to represent each sentence. We take the average word embedding of all the tokens in a sentence as a feature when training the classifier model. Based on the results of previous research, we used the word embedding Word2Vec: a dense representation of a word based on its context [37]. This representation is obtained using a neural network that attempts to predict a word based on its context or vice-versa. We chose a word embedding size of 300 based on our early experimental results. The emotion classifier for the first system is SVM, based on the results in [18] as well as our early experimental results.

The second system fine-tunes a deep pretrained language model on emotion classification task. The popularity of deep pretrained language model has been on the rise since the original release of BERT [38]. Such model has been extensively researched, and have become the state of the art for many text-processing tasks, including classification. Deep pretrained language models are deep learning models (usually transformers) trained on a large corpus with tasks that the dataset can easily generate. For example, BERT is pretrained on a masked language model task and a next sequence prediction task (see

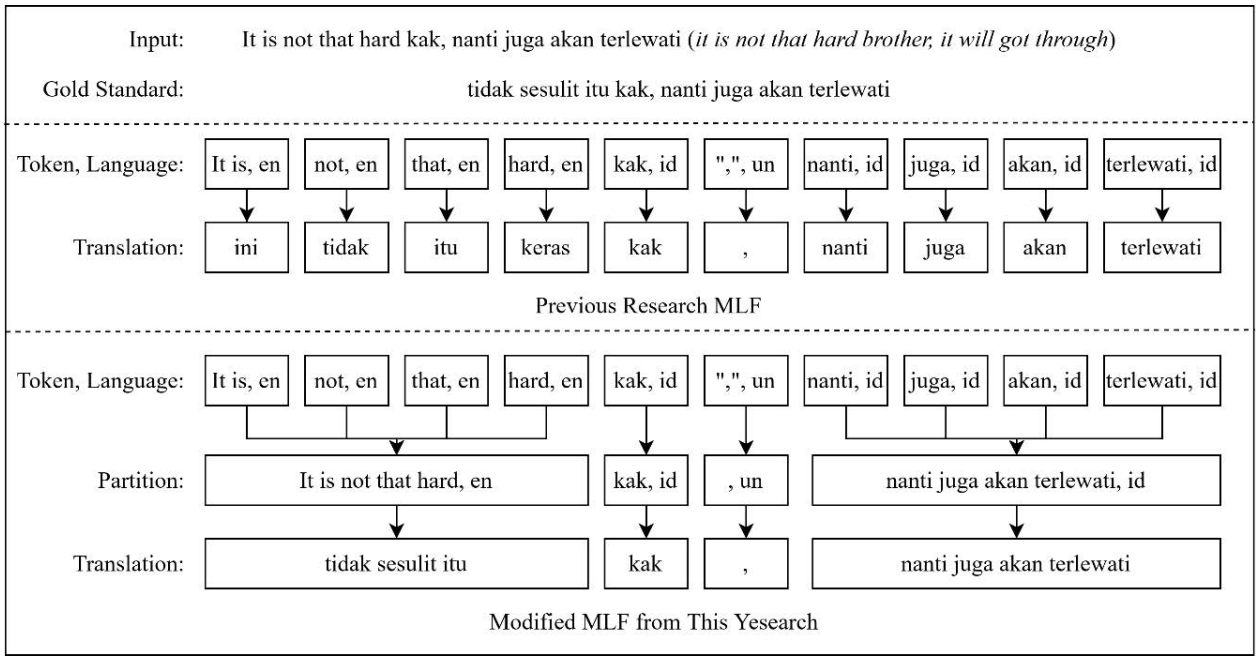

Fig. 4. Example of Indonesian-English Code-mixed Text.

TABLE IV. EXPERIMENT SCENARIO

No	Experiment	Baseline	Measure
1	Code-Mixed Normalisation (individual modules)		
	a. Language Identification	CRF [10], CNN, RNN, CNN+CRF	Accuracy, Precision, Recall, F_1 score
	b. Lexical Normalisation	raw data, ruleset + spell correction [10], SMT	BLEU, WER
	c. Translation	Direct translation, MLF[10]	BLEU, WER
2	Code-Mixed Normalisation (the whole pipeline)	raw data, Barik et al.'s system [10]	BLEU, WER
3	Using Code-mixed Normalisation for Emotion Classification	tokenisation-only, simple preprocessing	Accuracy, Precision, Recall, F_1 score

the original paper for a more in-depth explanation). The pre-trained language model used in this experiment is IndoBERT, a BERT-based model trained on Indonlu corpus (Indonesian corpus containing approximately four billion words) [39].

V. EXPERIMENT

Three experiments were performed in this research. All models in our experiments were trained with five-fold cross validation. Significance was measured using t-test with a 0.05 significance level. A summary of all of our experiments, including the baseline methods and evaluation measures, is presented in Table IV.

The **first experiment** separately tested the individual modules of the code-mixed text normalisation system. This experiment was performed to examine the effectiveness of our modifications to each module (except tokenisation, as we did not make any changes to the tokenisation module). During this experiment, the input for each module used gold-standard or reference data from the previous step in the pipeline. This configuration makes it possible to safely assume that the input from the previous normalisation step is free of errors.

The modified language identification model is compared with four other models as baselines. They include the Conditional Random Field (CRF) model from Barik et al.'s work, Convolutional Neural Network (CNN) on character level [40], Recurrent Neural Network (RNN) [41], and CRF model using CNN as feature (CNN+CRF). The CNN architecture used in this work consist of one convolution layer and one fully connected layer. The CNN+CRF method replaces the fully connected layer with CRF.

Next, the modified lexical normalisation module is compared wth three other models as baselines. They include using unprocessed input text (raw data), the combination of rule-based and spell correction model from Barik et al., and SMT model alone. We used SMT model alone as one of our baselines to analyse the effect of ruleset in our lexical normalisation module.

At last, the modified translation module is compared with two baselines: MLF method used by Barik et al., and direct translation method using machine translations. In direct translation method, we simply translate the Indonesian-English code-mixed text into Indonesian language using two popular machine translations: Microsoft[3] and Google. Translate[4]

The **second experiment** tested the entire code-mixed text normalisation system to normalise the given Indonesian-English code-mixed text into Indonesian, which is the main goal of the system. The input for this experiment was code-mixed Indonesian-English tweets, which were put through the tokenisation, language identification, lexical normalisation,

[3]https://translator.microsoft.com/
[4]https://translate.google.com/

and translation modules, resulting in normalised Indonesian-language tweets. Baselines for this experiment included unprocessed input text (raw data), and the code-mixed normalisation pipeline system from Barik et al.'s work.

The **third experiment** performed emotion classification using a code-mixed text normalisation system to preprocess the data. The aim of this experiment was to analyse the effect of code-mixed normalisation on effectiveness in the emotion classification task. For this purpose, we compared the results of emotion classification methods that used code-mixed normalisation in the preprocessing step and the baseline methods that did not use code-mixed normalisation (tokenisation-only and simple preprocessing methods). In the tokenisation-only method, the tweets are simply tokenised before an emotion classification method is applied. We used Tweet-Tokenizer from NLTK library[5] to perform this tokenisation. In the other baseline, i.e. simple preprocessing method, common preprocessing data methods including tokenisation, stopword removal, punctuation removal, and character repetition removal were performed. The stopword list for Indonesian and English languages were taken from NLTK library .

This research uses varieties of evaluation metrics to measure the system performance on each experiment. Accuracy, precision, recall, and F_1 score are common metrics to compare a set of labels between gold standard (reference) and the prediction from the model. In this work, they are used for evaluating language identification module in the first experiment, and emotion classification in the third experiment. Accuracy measures the rate of correct prediction over the entire test data. Precision measures how many predictions of a relevant classes that are actually correct. Recall measures how many relevant classes the model could find. F_1 score is a harmonic mean between precision and recall. The formula to calculate Accuracy, Precision, Recall, and F-1 measures are as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$F_1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{4}$$

TP (True Positive) defines the correctly predicted positive values. TN (True Negative) defines the correctly predicted negative values. FP (False Positive) defines the value of incorrect positive predictions. FN (False Negative) defines the value of incorrect negative predictions.

Bilingual Evaluation Understudy (BLEU) and Word Error Rate (WER) are used to evaluate task that produce a correctness of a free form text compared to gold standard text. BLEU score measures the similarity of the text produced by the model and the gold standard. While this measure is commonly used for evaluating machine translation system, previous researches also use this metric to measure lexical normalisation performances that use machine translation [31], [42], [43]. Since there is translation task in our lexical normalisation module, translation module, and overall code-mixed text normalisation system, then we use BLEU for the experiments on these systems. WER measures how many edits (substitution, deletion, insertion) it takes from the predicted model to the gold standard with respect to the length of the gold standard text. In other words, WER measures the error of a system. This metric is commonly used on speech recognition task and expanded to measure performance on other task such as translation and lexical normalisation [42], [28]. The formula to calculate BLEU and WER are as follows:

$$BLEU = p * bp \tag{5}$$

$$WER = \frac{ed}{N} \tag{6}$$

BLEU score is computed by multiplying the geometric mean of the corpus precision scores (p) by the exponential brevity penalty factor (bp). For more detailed explanation, please refer to the original paper of BLEU metric [44]. WER is computed by taking the ratio between edit distance (ed) and the number of token in the reference text (N). Here, edit distance is measured by finding the minimum number of insertion, deletion, and substitution operations required to perform the alignment between the predictive text and the reference text.

TABLE V. THE RESULTS OF LANGUAGE IDENTIFICATION EXPERIMENT

Model	Precision	Recall	F_1 score	Accuracy
CRF[10]	91.22	88.60	89.72	90.40
CNN	92.58	91.70	92.08	92.70
CNN+CRF	92.00	91.63	91.79	92.42
RNN	85.71	84.38	84.77	86.09
CRF++	$94.56^{\star\dagger\ddagger\times}$	$93.53^{\star\dagger\ddagger\times}$	$93.98^{\star\dagger\ddagger\times}$	$94.41^{\star\dagger\ddagger\times}$

Note: Significant differences of our method (CRF++) against CRF[10], CNN, CNN+CRF, and RNN are denoted by $\star$, $\dagger$, $\ddagger$, and $\times$, respectively.

VI. RESULTS AND ANALYSIS

A. Effectiveness of Individual Modules of Code-Mixed Normalisation System

1) Results of Language Identification Module: Table V shows the experimental results for the language identification module. The model created in this research achieved the highest score, outperforming the CRF model used by Barik et al. by 3.33% on precision, 4.92% on recall, 4.26% on F_1 score, and 4.01% on accuracy. This improvement comes from the new feature set for language identification. The deep learning baseline was unable to outperform our CRF++ model. The RNN model achieved the worst performance. One possible cause for this was the low amount of sequences in the training data. The data that the RNN processed included entire tweets or the whole sequence of a token, and there are only 825 tweets in the dataset. The same problem did not occur with the CNN model because CNN attempts to classify each token separately; thus the processed data are individual tokens. Note that there are 22,725 tokens available in the dataset.

[5]https://www.nltk.org/

TABLE VI. THE RESULTS OF LEXICAL NORMALISATION EXPERIMENT

Model	Precision	Recall
Raw Data	47.79	15.18
Ruleset+Spell Correction [10]	71.67	12.45
SMT	88.14	5.63
Ruleset+SMT	$89.75^{\star\dagger\ddagger}$	$4.67^{\star\dagger\ddagger}$

Note: Significant differences of our method (ruleset+SMT) against raw data, ruleset+spell correction [10], and SMT are denoted by $\star$, †, and ‡, respectively.

TABLE VII. THE RESULTS OF TRANSLATION EXPERIMENT

Translation System	Model	BLEU	WER
Microsoft	Direct Translation	62.19	21.19
	MLF[10]	74.49	15.06
	MLF[10]+context	$75.36^{\star}$	$14.36^{\star}$
Google	Direct Translation	72.63	16.56
	MLF[10]	75.87	14.36
	MLF[10]+context	$77.77^{\star\dagger}$	$13.09^{\star\dagger}$

Note: Significant differences of our method (MLF[10]+context) against direct translation and MLF[10] are denoted by $\star$ and †, respectively.

TABLE VIII. THE RESULTS OF OVERALL CODE-MIXED NORMALISATION SYSTEM PIPELINE

Model	BLEU	WER
Raw Data	30.01	42.18
Barik et al.'s system [10]	48.86	28.18
Our System	$64.55^{\star\dagger}$	$18.65^{\star\dagger}$

Note: Significant differences of our method against raw data and Barik et al.'s system [10] are denoted by $\star$ and †, respectively.

Whilst the evaluation shows promising results, the CRF model in this research still has some difficulty classifying certain specific tokens (e.g. named entity tokens, such as a place or person) as either "id" or "en" when they should be labelled "un". The model also often misclassifies Indonesian loan words that are taken from English words such as "*stop*" ("stop"). Loan words present ambiguity that makes it difficult for the model to correctly label the token. Another challenge for the model is sub-word code-mixing, which happens when Indonesian affixes are added to English words – for example, "*gap-nya*" ("the gap"). In this case, the correct label is the language of the root word, but sub-word code-mixed tokens are often misclassified as Indonesian rather than English.

2) Results of Lexical Normalisation Module: The results for the lexical normalisation module are shown in Table VI. The new model outperformed the Barik et al.'s model by 18.08 on BLEU score and 7.78 on WER. Raw data exhibited the worst performance, with 47.79 BLEU score and 15.18 WER. This result shows that our model is able to perform normalisation properly. Table VI also shows that the ruleset slightly helps the SMT model when performing lexical normalisation, increasing BLEU score by 1.61 higher and reducing WER by 1.04.

The SMT model can properly normalise ambiguous slang tokens that can serve as both a formal word and a slang word. This was a problem in Barik et al.'s model that relied on a formal lexicon to differentiate between formal and slang words. For example, the Indonesian word "*aja*" can serve as slang for the formal word "*saja*" ("only"), but may also be the formal word "*aja*", an archaic way to refer to the daughter of a noble. The SMT model relies on the context of a word during the training process to normalise that word. Another problem in Barik et al.'s model is that the model was unable to properly normalise slang words that were very different from their formal versions. Their model normalised the slang word "*ga*" ("no") into the letter "g", but the actual formal version of this word is "*tidak*" ("no"). This occurred because Barik et al.'s model relied only on rules. However, our method, which uses SMT in addition to a ruleset, did not have this problem.

The problem with the SMT model is that it is fully supervised. Therefore, it is only able to normalise slang words that appear in the training phase and cannot process slang tokens that appear only during testing. This becomes a prominent issue in low-resource settings. However, the use of a ruleset enables our model to normalising some words that the SMT might have not encountered during training. This also explains why the combined model was able to achieve the best performances (see Table VI).

3) Results of Translation Module: Table VII presents the experimental results for the translation module. Google's MT system produced better translation results than Microsoft's on every translation configuration. The modified MLF produced more accurate translation results than the baselines, achieving a BLEU score 1.9 higher and a WER 1.27 lower than the MLF method on Google 's MT system. Direct translation had the worst performance, with a BLEU score of 5.14 and WER of 3.47 on Google's MT system. A similar result was observed when using Microsoft's MT system.

This improvement shows that the use of a group of neighboring tokens or a language segment for translation process can produce more accurate translations than using individual token. The language segment provides sufficient context for the translation system to translate ambiguous words better than the original MLF used by Barik et al. However, our model still struggles to translate certain ambiguous words. For example, the English word "I" can be translated as either "*saya*" or "*aku*" in Indonesian; both have the same meaning, except that "*saya*" is often used in formal settings, whereas "*aku*" is often used in casual conversation.

B. Effectiveness of Overall Code-Mixed Normalisation System Pipeline

Table VIII shows the evaluation results for the code-mixed text normalisation system experiment. We used Google's MT system for the translation module, following the results of the previous experiment. The new system demonstrated better code-mixed text normalisation performance compared with the Barik et al.'s system, increasing BLEU score by 15.69% absolute improvement or 32.11% relative improvement and lowering WER by 9.53% absolute improvement or 33.82% relative improvement. This indicates the effectiveness of our modifications on Barik et al.'s modules. Both code-mixed text normalisation systems outperformed the evaluation results for the raw data baseline, which highlight the merit of normalizing the code-mixed text.

To understand the contribution of each module modified in this work with regard to improving the code-mixed normalisation system pipeline proposed by Barik et al., we conducted an additional experiment using each of our individual modules in Barik et al.'s system. The increases or decreases resulting

TABLE IX. THE CONTRIBUTIONS OF EACH MODULE

Model	△BLEU	%△BLEU
Language Identification	2.03	4.15%
Lexical Normalisation	14.08	28.82%
Translation	15.69	32.11%

TABLE X. THE EXAMPLE OF RESULTS GENERATED BY OUR CODE-MIXED NORMALISATION SYSTEM AND BARIK ET AL.'S SYSTEM

Input	Argh gimana cara save gif nya. [URL] (Argh how to save the gif [URL])
Output from Barik et al.'s system [10]	
Tokenisation	*Argh, gimana, cara, save, gif, nya, '.'*, [URL]
Language Identification	id, id, id, id, id, id, un, un
Lexical Normalisation	*karuan, kenapa, cara, edit, klik, ya, '.'*, [URL]
Translation	*karuan, kenapa, cara, edit, klik, ya, '.'*, [URL]
Output from our system	
Tokenisation	*Argh, gimana, cara, save, gif, nya, '.'*, [URL]
Language Identification	id, id, id, en, en, id, un, un
Lexical Normalisation	*argh, gimana, cara, save, gif, nya, '.'*, [URL]
Translation	*argh, gimana, cara, simpan, gif, nya, '.'*, [URL]
Gold standard	*argh, gimana, cara, menyimpan, gifnya, '.'*, [URL]

from this addition were then examined. For example, to see the contribution of our language identification module, we replaced Barik et al.'s language identification module with our module. The difference in BLEU scores (△ BLEU) was calculated for the scores of Barik et al.'s system using our modules versus the original system. A positive contribution is indicated by a positive △ BLEU score, whilst a negative contribution is indicated by a negative △ BLEU score. The results are summarized in Table IX.

As shown in Table IX, all of our modules positively contributed to the Barik et al.'s code-mixed normalisation system pipeline, since all △ BLEU scores are positive. The most important module is the translation module, which improved on Barik et al.'s system by 32.11%, followed by the lexical normalisation module (28.82% improvement) and the language identification module (4.15% improvement).

Table X compares the code-mixed text normalisation process for Barik et al.'s system and the new system implemented in this work. The former system detects the words "save" and "gif" as Indonesian words, whereas the latter properly detects them as English. Barik et al.'s system changes most tweets during lexical normalisation, leaving only "*cara*" ("method"), ".", and "[URL]" unchanged. The new system, however, did not perform any changes during lexical normalisation. Next, Barik et al.'s system does not perform any translation because all words are detected as Indonesian, whereas the new system translates "save gif" to "*simpan gif*". A comparison of both results indicates that the new system can produce results that are closer to the gold standard.

C. Effect of Code-Mixed Normalisation System on Emotion Classification

The result of emotion classification is presented in Table XI. The best overall result is achieved by using code-mixed normalisation in data preprocessing and the more modern approach with BERT. In terms of the results of using normalised code-mixed text, BERT is superior to Word2Vec by 16.56%

TABLE XI. THE RESULTS OF EMOTION CLASSIFICATION

Model	Preprocessing	Precision	Recall	F_1 Score	Accuracy
Word2Vec	TO	37.82	33.38	32.99	45.87
	SP	42.49	38.36	39.20	46.41
	CN	$53.15^{\star\dagger}$	$44.13^{\star}$	$45.44^{\star}$	$54.97^{\star\dagger}$
BERT	TO	45.85	43.98	43.92	50.33
	SP	47.82	45.21	45.92	50.34
	CN	$54.06^{\star\dagger}$	$51.44^{\star\dagger}$	$51.93^{\star\dagger}$	$56.84^{\star\dagger}$

Note: Significant differences of our emotion classification method using code-mixed normalisation (CN) as the preprocessing method against the methods that do not use code-mixed normalisation, but using tokenisation-only (TO) and simple preprocessing (SP) methods are denoted by $\star$ and †, respectively.

TABLE XII. THE CLASSIFICATION RESULTS FOR EACH EMOTION CLASS

Class	Precision	Recall	F_1 score	Accuracy
Love	68.75	30.56	42.31	94.86
Anger	59.46	68.75	63.77	74.32
Sadness	46.84	35.92	40.66	81.51
Joy	54.58	70.81	61.64	72.09
Fear	37.04	14.71	21.06	87.16

according to F_1 score. In both models, the best performance is achieved when code-mixed normalisation is applied as a preprocessing step before classification. This demonstrates the advantage of normalizing the code-mixed text before a main text processing task is conducted. Two factors could explain this result: lexical normalisation and translation.

The tokenisation-only and simple preprocessing approaches did not have a robust lexical normalisation step. This may result in some key emotion words or phrases involving some informal token becoming out of vocabulary (OOV) or the words being incorrectly represented by both classification models. The performance gap between using versus not using code-mixed normalisation indicates that the classic method is more affected than the BERT model. This is because the BERT model uses a sub-word tokeniser and is able to handle OOV tokens to some degree. In this case, code-mixed normalisation improved the F_1 score and accuracy of the Word2Vec model by 37.74% and 19.77%, respectively. The improvement obtained by the BERT model is slightly lower: 18.24% for F_1 score and 12.93% for accuracy.

Some key emotion words are also written in English. This leads to better performance on classification systems which applied code-mixed text normalisation in the preprocessing step. It is because they translate these words into Indonesian and therefore enables us to capture their meanings.

Table XII presents a breakdown of classification results for each emotion class. Since our dataset has uneven class distribution, F_1 score is more representative than accuracy because the values of false positives and false negatives do not have similar cost. The highest F_1 score was obtained by the class "anger", followed by the class "joy". This can be understood because these classes have the highest number of tweets in our dataset. Consequently, the classifier can successfully learn the characteristics of tweets expressing the emotions anger and joy. Therefore, the ratios of correctly predicted labels for these classes to the total predicted labels and to the total actual labels for these classes are high.

To better illustrate the number of correct and incorrect classifications for each class, a confusion matrix is displayed in Fig. 5. The confusion matrix shows that imbalance in

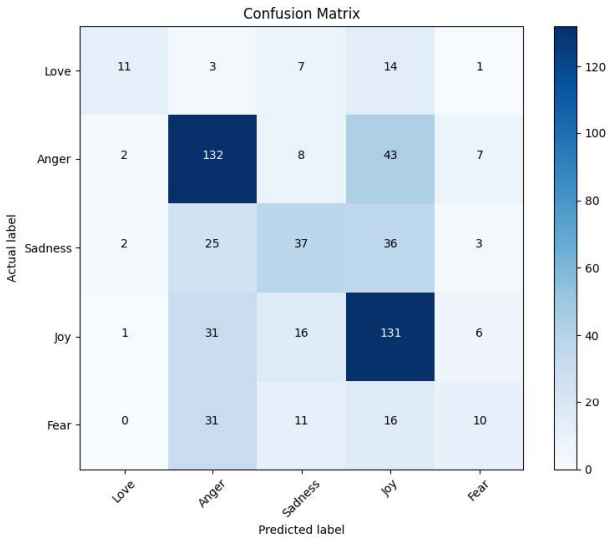

Fig. 5. Confusion Matrix of Emotion Classification Results.

TABLE XIII. EXAMPLE OF EMOTION CLASSIFICATION RESULTS

Input: Not good, not good.. Maag gue selalu kambuh sejak kerja di sini.. Setres (Not good, not good.. My ulcer always relapses since working here.. stress) **Gold standard class:** Sad
Preprocessed input (TO): Not good , not good .. Maag gue selalu kambuh sejak kerja di sini .. Setres **Predicted class:** Joy
Preprocessed input (SP): good good maag gue kambuh kerja setres **Predicted class:** Fear
Preprocessed input (CN): tidak bagus , tidak bagus, . mag saya selalu kambuh sejak kerja di sini. setres **Predicted class:** Sad

Note: TO, SP, and CN stands for "Tokenisation-Only", "Simple Preprocessing", and "Code-mixed Normalisation", respectively.

the dataset affects the emotion classification results. This can be seen from the two classes with the highest frequency of occurrence in our dataset, i.e. anger and joy, which showed a low proportion of false negatives and a high proportion of false positives relative to the total number of instances for these classes. The opposite happened for the emotion labels with the lowest frequency in our dataset, i.e. love and fear. Compared to other labels, both of these labels obtained the highest proportion of false negatives and the lowest proportion of false positives. The true positive values for these classes were also much lower than the false negative values.

In Table XIII, we demonstrate how different preprocessing methods may affect the classification results. Word2Vec was used to implement the classification model. According to the human annotation label, the input tweet expresses a sad emotion. This emotion is indicated by terms such as "not good", "*selalu kambuh*" ("always relapses"), "*setres*" ("stress").

A classification method using tokenisation-only preprocessing misclassifies the tweet's emotion as joy, presumably because the model cannot properly detect the emotion contained in the phrase "not good". Because this phrase is not translated into Indonesian and it may appear infrequently in the data, the Word2vec model is unable to learn the semantics of the phrase well, which may lead to an incorrect classification result. A classification method using simple preprocessing also misclassifies the tweet's emotion as fear. This preprocessing method removes the term "not" from phrase "not good". This means that all tweets in the collection that contain the phrase "not [ADJECTIVE]" and "[ADJECTIVE]" will have similar preprocessing results for that phrase, eventhough they actually convey opposite emotions, since the term "not" indicates a contrast in meaning. We argue that this causes the model to be inaccurate in classifying the emotion. A classification method using a code-mixed text normalisation pipeline, on the other hand, can properly classify the tweet's emotion as sad. Translating the English phrase "not good" into the Indonesian phrases "*tidak bagus*" enables the Word2vec model to capture the semantics of the phrase since the translation phrase appears frequently in the collection.

VII. DISCUSSION AND FUTURE WORK

The number of code-mixed text used in this work is still relatively small when compared to the code-mixed text in other language pairs such as Bengali-English [45], Malayam-English [46], and Hindi-English [47]. Our dataset consists of 825 sentence pairs for our experiment on code-mixed normalisation and 584 sentences for our experiment on the effect of the code-mixed normalisation on emotion classification. Note that it is because a large dataset of Indonesian-English code-mixed text is still unavailable. The dataset from Barik et al. [10] that we use in this work is the only dataset of Indonesian-English code-mixed text that is available. Using this data also enables us to directly comparing our result with Barik et al. Therefore, further experiments using larger data may be useful to confirm the results reported in this work.

Our approach for code-mixed normalisation has some advantages compared to end-to-end deep learning approaches. Our approach does not require big data resources to achieve good results compared to deep learning methods. So, it is suitable for low-resource languages, such as Indonesian language. In addition, our approach is also more flexible in which users can easily add new rules or features in the individual modules of code-mixed normalisation system if needed.

The approach used in this task is aimed for code-mixed normalisation. However, each individual module in the pipeline can also be used separately for a specific task. For example, users can adopt the tokenisation module only if they just want to have more accurate tokenisation. Since our methods are mostly data driven (except the ruleset on lexical normalisation module), then with some tuning on the ruleset, we believe the individual modules as well as the overall system pipeline is applicable to other languages (or language pairs) as long as the data is available.

We are aware there are more modern methods that utilize Neural Machine Translation (NMT) model for lexical normalisation [48], [49]. However, we decided not to use this method because the dataset used in this research is too small for NMT to be able to learn an accurate model for this complex task. This effect has been demonstrated before by Matoz, et. al. [28] that utilizes RNN encoder-decoder to perform lexical normalisation on English-Dutch and Wibowo et. al.[32] that utilizes GPT-2 to normalise Indonesian text. Both of these research showed that on low resource settings, SMT model still gives on par performance if not better than the NMT model because of the insufficient amount of training data.

There are several avenues through which future research might improve on the results of our proposed system. The SMT model in the lexical normalisation module could be improved by using an additional corpus for the language model. A large dataset could be built containing formal Indonesian-English code-mixed text to improve the accuracy of the lexical normalisation module. Another possible improvement that could be made to the lexical normalisation module involves the ruleset. The ruleset used in this work are still limited for normalising informal Indonesian words that contain character repetitions and reduplication shortening; and informal English words that contain contractions and code-mixed prefixes / suffixes. Therefore, some extra rules can be added to improve the accuracy of the lexical normalization module in our system. An example of additional rules could be derived from informal affixes that are common in Indonesian, such as the informal suffixes "*-ny*" or "*-x*", which can be converted to "*-nya*" and then prefix "*ng-*", which can be converted to "*meng-*".

Next, in this research, the effect of code-mixed normalisation was examined for an emotion classification task. Whilst the results showed a positive effect, this cannot be generalised to many other language processing tasks. Thus, similar analysis could be performed for other tasks – such as sentiment analysis, POS tagging and so on – to examine whether performing code-mixed normalisation can offer significant improvemenet for these tasks as well.

VIII. Conclusion

In this research, we improved a state-of-the-art code-mixed text normalisation system for Indonesian-English tweets. Specifically, we improved three modules of the original code-mixed normalisation system pipeline, including improving the feature set in the language identification module, combining an MT approach and a ruleset in the lexical normalisation module and adding some context in the translation module. Our experimental results show that our approach outperformed a state-of-the-art Indonesian-English code-mixed normalisation system. The new feature set in the language identification module showed an improvement of 4.26% in F_1 score. The use of an MT approach in the lexical normalisation module improved BLEU score by 25.22% and lowered WER by 62.49%. The addition of context to the translation process improved BLEU score by 2.5% and lowered WER by 8.84%. The overall effectiveness of the code-mixed text normalisation system was improved, with an increase of 32.11% in BLEU score and a decrease of 33.82% in WER.

This research also analysed the effect of code-mixed text normalisation process on emotion classification. Applying code-mixed normalisation process resulted in increased effectiveness of emotion classification systems. The systems that used code-mixed normalisation in the preprocessing step were more effective than those that did not. Compared with tokenisation-only preprocessing method, the code-mixed normalisation system achieved better evaluation results by up to 37.74% in F_1 score. The code-mixed normalisation system also outperformed a simple preprocessing method by up to 15.92% in F_1 score.

Acknowledgments

This research is supported by the PUTI (Publikasi Terindeks Internasional) Q3 grant with number NKB-4379/UN2.RST/HKP.05.00/2020 from Universitas Indonesia.

References

[1] S. Poplack and J. A. Walker, "Pieter muysken, bilingual speech: a typology of code-mixing. cambridge: Cambridge university press, 2000. pp. xvi+ 306." *Journal of Linguistics*, vol. 39, no. 3, pp. 678–683, 2003.

[2] K. Bali, J. Sharma, M. Choudhury, and Y. Vyas, ""i am borrowing ya mixing?" an analysis of english-hindi code mixing in facebook," in *Proceedings of the First Workshop on Computational Approaches to Code Switching*, 2014, pp. 116–126.

[3] L. S. Kia, X. Cheng, T. K. Yee, and C. W. Ling, "Code-mixing of english in the entertainment news of chinese newspapers in malaysia," *International journal of English linguistics*, vol. 1, no. 1, p. 3, 2011.

[4] K. Ariffin and M. Susanti Husin, "Code-switching and code-mixing of english and bahasa malaysia in content-based classrooms: Frequency and attitudes." *Linguistics Journal*, vol. 5, no. 1, 2011.

[5] E. Syam *et al.*, "Code switching by an indonesian muslim preacher," in *7th International Conference on English Language and Teaching (ICOELT 2019)*. Atlantis Press, 2020, pp. 18–22.

[6] D. Rusydah, "Bahasa anak jaksel: A sociolinguistics phenomena," *LITERA KULTURA*, vol. 8, no. 1, 2020.

[7] M. Dhar, V. Kumar, and M. Shrivastava, "Enabling code-mixed translation: Parallel corpus creation and mt augmentation approach," in *Proceedings of the First Workshop on Linguistic Resources for Natural Language Processing*, 2018, pp. 131–140.

[8] Q. Zhang, H. Chen, and X. Huang, "Chinese-english mixed text normalization," in *Proceedings of the 7th ACM international conference on Web search and data mining*, 2014, pp. 433–442.

[9] M. A. Menacer, D. Langlois, D. Jouvet, D. Fohr, O. Mella, and K. Smaïli, "Machine translation on a parallel code-switched corpus," in *Canadian Conference on Artificial Intelligence*. Springer, 2019, pp. 426–432.

[10] A. M. Barik, R. Mahendra, and M. Adriani, "Normalization of indonesian-english code-mixed twitter data," in *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, 2019, pp. 417–424.

[11] R. van der Goot and Ö. Çetinoğlu, "Lexical normalization for code-switched data and its effect on pos tagging," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, 2021, pp. 2352–2365.

[12] R. Singh, N. Choudhary, and M. Shrivastava, "Automatic normalization of word variations in code-mixed social media text," *arXiv preprint arXiv:1804.00804*, 2018.

[13] D. Patel and R. Parikh, "Language identification and translation of english and gujarati code-mixed data," in *2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)*. IEEE, 2020, pp. 1–4.

[14] C. Myers-Scotton, *Duelling languages: Grammatical structure in codeswitching*. Oxford University Press, 1997.

[15] S. K. Mahata, S. Mandal, D. Das, and S. Bandyopadhyay, "Code-mixed to monolingual translation framework," in *Proceedings of the 11th Forum for Information Retrieval Evaluation*, 2019, pp. 30–35.

[16] J. Gao, Y. Fu, Y.-G. Jiang, and X. Xue, "Frame-transformer emotion classification network," in *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*, 2017, pp. 78–83.

[17] M. Grimm and K. Kroschel, "Rule-based emotion classification using acoustic features," in *in Proc. Int. Conf. on Telemedicine and Multimedia Communication*. Citeseer, 2005.

[18] M. S. Saputri, R. Mahendra, and M. Adriani, "Emotion classification on indonesian twitter dataset," in *2018 International Conference on Asian Language Processing (IALP)*. IEEE, 2018, pp. 90–95.

[19] Y.-P. Lin, C.-H. Wang, T.-P. Jung, T.-L. Wu, S.-K. Jeng, J.-R. Duann, and J.-H. Chen, "Eeg-based emotion recognition in music listening," *IEEE Transactions on Biomedical Engineering*, vol. 57, no. 7, pp. 1798–1806, 2010.

[20] P. Ekman, "An argument for basic emotions," *Cognition & emotion*, vol. 6, no. 3-4, pp. 169–200, 1992.

[21] P. Shaver, J. Schwartz, D. Kirson, and C. O'connor, "Emotion knowledge: further exploration of a prototype approach." *Journal of personality and social psychology*, vol. 52, no. 6, p. 1061, 1987.

[22] R. Plutchik, "A general psychoevolutionary theory of emotion," in *Theories of emotion*. Elsevier, 1980, pp. 3–33.

[23] J. Li, Y. Xu, H. Xiong, and Y. Wang, "Chinese text emotion classification based on emotion dictionary," in *2010 IEEE 2nd Symposium on Web Society*. IEEE, 2010, pp. 170–174.

[24] S. M. Mohammad and P. D. Turney, "Crowdsourcing a word–emotion association lexicon," *Computational intelligence*, vol. 29, no. 3, pp. 436–465, 2013.

[25] A. Agrawal and A. An, "Unsupervised emotion detection from text using semantic and syntactic relations," in *2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, vol. 1. IEEE, 2012, pp. 346–353.

[26] A. Das and B. Gambäck, "Identifying languages at the word level in code-mixed indian social media text," in *Proceedings of the 11th International Conference on Natural Language Processing*, 2014, pp. 378–387.

[27] J. R. Landis and G. G. Koch, "The measurement of observer agreement for categorical data," *biometrics*, pp. 159–174, 1977.

[28] C. M. Veliz, O. De Clercq, and V. Hoste, "Comparing mt approaches for text normalization," in *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, 2019, pp. 740–749.

[29] A. Kurnia and E. Yulianti, "Statistical machine translation approach for lexical normalization on indonesian text," in *2020 International Conference on Asian Language Processing (IALP)*. IEEE, 2020, pp. 288–293.

[30] E. Yulianti, I. Budi, A. N. Hidayanto, H. M. Manurung, and M. Adriani, "Developing indonesian-english hybrid machine translation system," in *2011 International Conference on Advanced Computer Science and Information Systems*. IEEE, 2011, pp. 265–270.

[31] A. Aw, M. Zhang, J. Xiao, and J. Su, "A phrase-based statistical model for sms text normalization," in *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, 2006, pp. 33–40.

[32] H. A. Wibowo, T. A. Prawiro, M. Ihsan, A. F. Aji, R. E. Prasojo, R. Mahendra, and S. Fitriany, "Semi-supervised low-resource style transfer of indonesian informal to formal language with iterative forward-translation," in *2020 International Conference on Asian Language Processing (IALP)*. IEEE, 2020, pp. 310–315.

[33] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens *et al.*, "Moses: Open source toolkit for statistical machine translation," in *Proceedings of the 45th annual meeting of the association for computational linguistics companion volume proceedings of the demo and poster sessions*, 2007, pp. 177–180.

[34] F. J. Och and H. Ney, "A systematic comparison of various statistical alignment models," *Computational linguistics*, vol. 29, no. 1, pp. 19–51, 2003.

[35] K. Heafield, "Kenlm: Faster and smaller language model queries," in *Proceedings of the sixth workshop on statistical machine translation*, 2011, pp. 187–197.

[36] C. Cherry and G. Foster, "Batch tuning strategies for statistical machine translation," in *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2012, pp. 427–436.

[37] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[38] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 4171–4186.

[39] B. Wilie, K. Vincentio, G. I. Winata, S. Cahyawijaya, X. Li, Z. Y. Lim, S. Soleman, R. Mahendra, P. Fung, S. Bahar *et al.*, "Indonlu: Benchmark and resources for evaluating indonesian natural language understanding," in *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, 2020, pp. 843–857.

[40] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," *Advances in neural information processing systems*, vol. 28, pp. 649–657, 2015.

[41] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[42] D. Contractor, T. A. Faruquie, and L. V. Subramaniam, "Unsupervised cleansing of noisy text," in *Coling 2010: Posters*, 2010, pp. 189–196.

[43] B. Han and T. Baldwin, "Lexical normalisation of short text messages: Makn sens a# twitter," in *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, 2011, pp. 368–378.

[44] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 2002, pp. 311–318.

[45] S. Mandal, S. K. Mahata, and D. Das, "Preparing bengali-english code-mixed corpus for sentiment analysis of indian languages," in *The 13th Workshop on Asian Language Resources*, 2018, p. 57.

[46] B. R. Chakravarthi, N. Jose, S. Suryawanshi, E. Sherly, and J. P. McCrae, "A sentiment analysis dataset for code-mixed malayalam-english," *arXiv preprint arXiv:2006.00210*, 2020.

[47] P. Makhija, A. Srivastava, and A. Gupta, "hinglishnorm-a corpus of hindi-english code mixed sentences for text normalization," in *Proceedings of the 28th International Conference on Computational Linguistics: Industry Track*, 2020, pp. 136–145.

[48] M. Lusetti, T. Ruzsics, A. Göhring, T. Samardzic, and E. Stark, "Encoder-decoder methods for text normalization," in *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2018)*, 2018, pp. 18–28.

[49] B. Muller, B. Sagot, and D. Seddah, "Enhancing bert for lexical normalization," in *The 5th Workshop on Noisy User-generated Text (W-NUT)*, 2019.