Transformer-based Cross-Lingual Summarization using Multilingual Word Embeddings for English - Bahasa Indonesia

Achmad F. Abka[1], Kurniawati Azizah[2], Wisnu Jatmiko[3]
National Research and Innovation Agency, Jakarta, Indonesia[1]
Faculty of Computer Science, Universitas Indonesia, Depok, Indonesia[1, 2, 3]

***Abstract*—Cross-lingual summarization (CLS) is a process of generating a summary in the target language from a source document in another language. CLS is a challenging task because it involves two different languages. Traditionally, CLS is carried out in a pipeline scheme that involves two steps: summarization and translation. This approach has a problem, it introduces error propagation. To address this problem, we present a novel end-to-end abstractive CLS without the explicit use of machine translation. The CLS architecture is based on Transformer which is proven to be able to perform text generation well. The CLS model is a jointly trained CLS task and monolingual summarization (MS) task. This is accomplished by adding a second decoder to handle the MS task, while the first decoder handles the CLS task. We also incorporated multilingual word embeddings (MWE) components into the architecture to further improve the performance of the CLS models. Both English and Bahasa Indonesia are represented by MWE whose embeddings have already been mapped into the same vector space. MWE helps to better map the relation between input and output that use different languages. Experiments show that the proposed model achieves improvement up to +0.2981 ROUGE-1, +0.2084 ROUGE-2, and +0.2771 ROUGE-L when compared to the pipeline baselines and up to +0.1288 ROUGE-1, +0.1185 ROUGE-2, and +0.1413 ROUGE-L when compared to the end-to-end baselines.**

Keywords—Cross-lingual summarization; multilingual word embeddings; transformer; automatic summarization

I. Introduction

Automatic summarization is a process of automatically producing a shorter version of an original while retaining contents and meanings that are considered essential. This shorter version is called a summary. The purpose of summarizing is to produce a summary that contains the main content of a document in less space [1]. Automatic summarization helps users get the main idea of a document without having to read the whole document thus saving time and effort compared to doing it manually [2]. In general, automatic summarization can be categorized into two based on the approach in summarizing: extractive summarization and abstractive summarization. The extractive approach produces a summary by copying words or sentences from source documents that are considered important. The abstractive approach produces a summary using its own words or sentences. In addition to text, summaries of images and videos can also be produced [3] [4] [5]. In the context of this work, automatic summarization is an activity that uses a machine (computer) to automatically summarize a document using a certain algorithm or method.

The study on automatic summarization was first reported by Luhn [6]. The system is based on bag-of-words. The frequency and relative position of a word in a sentence are the main features in determining how important the sentence is. Gradually, linguistic information such as word type and structure are utilized using natural language processing (NLP). The extractive approach is then widely used because of its simple approach without the need for extensive NLP [7] [8]. The success of the sequence-to-sequence (Seq2Seq) recurrent neural network (RNN) model has made the rapid development of abstractive approaches [9] [10]. Inspired by the attention-based neural machine translation (MT) model [11] [12], the RNN is used by adding the attention mechanism so that the model can focus on some parts of text while diminishing other parts at a certain time/context.

Cross-lingual summarization (CLS) is a task to produce a summary in the target language, from source documents in another language [13] [14]. Traditionally, CLS can be done by involving two processes: translation and monolingual summarization (MS). However, there is a problem when doing it with two processes in a pipeline scheme. The pipeline process introduces error propagation which adversely affects the final quality of the summary. This problem can be solved by conducting end-to-end model training [15] [16]. Another challenge in CLS research is the limited data or corpus that can be used to train the model. Research on end-to-end CLS is still relatively new, so resources related to it are still quite scarce. There is no dataset that is considered a standard that can be used as a benchmark. The researchers still tend to create their own dataset for the language pair domain that is of interest to them.

In this paper, we present a novel end-to-end abstractive CLS that solves the error propagation problem found in the pipeline scheme. We adapt transformer-based architecture and extended it by including multilingual word embeddings (MWE) [17] components. These components allow the model to be able to represent words in two different languages. This paper is an extended study of our previous work on the end-to-end abstractive CLS model [18]. The model is trained in the English domain for source documents and the Bahasa Indonesia domain for its summary using the dataset that we have constructed. Inspired by [19] and [20], we use a round-trip translation technique to generate CLS dataset from MS dataset. This technique is similar to that used by [15], but we apply it to the document and its summary, not just the summary. The resulting dataset is then evaluated using bilingual evaluation understudy (BLEU) [21]. BLEU is a method for evaluating the quality of text that has been machine translated from one language to another automatically. The CLS model can produce a Bahasa Indonesia summary from an English source document end-to-end without explicitly using a machine translator.

The main contributions of this work are:

- A novel framework for generating end-to-end abstractive cross-lingual summary by incorporating MWE components in the architecture. These components are used to represent words in both languages used in the source document and its summary.
- This works produces a new cross-lingual dataset. This dataset can be used for CLS research with the English domain as the source document and the Bahasa Indonesia domain as the summary.

The rest of this paper is organized as follows. In Section II, we review the related work in CLS. In Section III, we explain the proposed method. In Section IV, the details of the experimental setup and the evaluation metrics are presented. The experimental results are given in Section V. Finally, the conclusions of this research are found in Section VI.

II. Related Works

CLS is the process of generating a summary in the target language from source documents in another language. Unlike MS, CLS involves at least two different languages. Traditional approaches treat CLS as a pipeline scheme. The newer approach does it end-to-end to avoid error propagation that occurs in the pipeline process.

A. *Traditional Cross-Lingual Summarization*

CLS research has been conducted [22] [23] [24]. The approaches are generally divided into extraction-based and compression-based. Summarization is done in a pipeline manner which is divided into two steps: summarizing and translating. There are two patterns of utilization or use of machine translation. The first pattern is to first translate the source document using a machine translator and then the results of the translation are used to summarize. The second pattern summarizes the source document first and then translates the results using a machine translator into the target language. Another approach utilizes existing source documents in two languages to generate summaries in the target language [25] [26]. This target language is one of the two languages. In addition to the extraction process, a compression process is also carried out in the compression-based approach. Summarization begins with a selection process to obtain relevant content in the form of sentences or phrases (bilingual). The compression is carried out by removing parts of sentences or phrases that do not meet some criteria, such as information content, legibility, and grammar/structure [27] [28] [29].

In contrast to the CLS previously described which uses an extractive approach, Zhang, Zhou, and Zong [30] developed an abstractive CLS system. The system works by first translating the source document from English to Chinese using Google Translate. The next step is to extract bilingual concepts and fact pairs. Then the score of translation and salience is calculated from this set of pairs. Based on the scores, a set of pairs is selected to be used in the summary. This selection considers several criteria, including compatibility between concepts and facts, number of sentences, summary length, etc. Post-processing begins with determining which pairs of concepts and facts are included in a particular sentence. Then the last step is to put them in order. Abstractive summarization has the ability to paraphrase. In the system developed by [30], the abstractive process is carried out by combining pairs of concepts and facts into one sentence. This process is indeed a form of paraphrasing, but these pairs of concepts and facts are taken extractively and used as they are.

Similar to [30], our work also falls under the abstractive CLS category. The difference is that our approach is based on neural networks and summarization is done directly, rather than in two steps in a pipeline scheme. This straightforward CLS can avoid error propagation that occurs in the pipeline process.

B. *Neural Network-based Cross-Lingual Summarization*

In 2019, neural network-based CLS began to emerge. Ouyang, Song, and McKeown [31] use neural networks for CLS in pipeline schemes, on the translation side using Marian [32] and on the summarization side using pointer-generators [33]. The system summarizes Somali, Swahili, or Tagalog source documents into an English summary. Zhu et al. [15] are the first to report on end-to-end CLS. A CLS from English to Chinese was built using transformer architecture that has been proven to have good performance in text generation. The network is trained using a multi-task learning approach. Specifically, they combine CLS loss with MS loss. The CLS dataset used was built by modifying the CNN/Dailymail dataset. The dataset is translated into Chinese using a machine translation service. Duan et al. [16] see CLS as a zero-shot problem. They do not have a CLS dataset to train the model directly. The network is trained by adapting a paradigm on neural machine translation (NMT) called triangular NMT systems. A network called a student network is a CLS network that is trained to imitate the behavior of a teacher network which is an MS network. Ladhak et al. [34] proposed a benchmark dataset for abstractive CLS named WikiLingua. The data is taken from WikiHow in the form of document and summary pairs in 18 languages. Not all data is available in 18 languages, availability in each language varies. The dataset is tested for CLS with pipeline and end-to-end approach. The end-to-end approach uses mBART [35] which is fine-tuned using the source language document and the target language summary.

Our approach is inspired by [15]. However, we propose to add MWE components which we believe will improve the performance of the model. MWE represents the two languages in the CLS task in the same vector space. This facilitates the mapping of the relation between inputs and outputs during model training.

C. *Multilingual Word Embeddings*

Multilingual word embeddings (MWE) are word embeddings that represent words in various languages in one vector space [17]. Conneau et al. [36] developed MWE using an unsupervised approach. Supervised approaches generally require bilingual resources such as dictionaries or parallel corpus. They proposed an unsupervised way to map monolingual word embeddings without the need for bilingual data. Word embeddings for each language are initially trained independently. Then, each word in the two languages is mapped into the same vector space using deep adversarial networks [37]. Heinzerling & Strube [38] developed MWE for word segmentation (subword embeddings). Word segmentation is done by using the byte pair encoding (BPE) approach. BPE performs segmentation to words that rarely appear into several tokens. Embeddings are trained in a simple way. All articles from Wikipedia in various languages are combined, then used to train subword embeddings. Artetxe & Schwenk [39] also developed multilingual embeddings but specifically developed embeddings for sentences. These sentence embeddings were trained using a single bidirectional LSTM (BiLSTM) encoder.

In this work, we use MUSE from Conneau et al. [36]. MUSE is used because the embedding is word-based so it is expected to be able to cover most of the vocab. MWE based on word segmentation such as BPEmb from Heinzerling & Strube [38] may be better than MUSE because it can recognize tokens that rarely or never appear in the training data. However, it is not suitable for the case in this work because the embedding is trained using various language versions of the articles available on Wikipedia. The amount of data which is an aggregate of all articles in various language versions can also result in many words being segmented due to their small frequency, thus causing the size of vocab to be very large. In general, a vocab size that is too large can adversely affect the performance of NLP system.

III. Methods

CLS can be expressed as an input sentence in the source language $x = (x_1,\ x_2,\ x_3, \dots, x_m)$ and $x_i \in V_s$, where m is the input sentence length and V_s is the source vocab in the source language, with an output summary in the target language $y = (y_1,\ y_2,\ y_3, \dots, y_n)$ and $y_i \in V_t$, where n is the output summary length, $n \leq m$, and V_t is the target vocab in the target language.

A. *Transformer-based Cross-Lingual Summarization*

NLP systems that are built specifically for a particular language are generally trained using datasets in that language. This approach has a problem when applied to other languages. The system needs to be retrained with separate datasets in the other specific language. This is equivalent to building the system from scratch. Another way is to use machine translation. The system is combined with machine translation in a pipeline so that it can produce output in the desired language. MWE can be used to tackle this problem. Both languages are represented by embeddings that are mapped into the same vector space. This mapping makes the two languages seem to be the same language, at least at the word level. The CLS architecture proposed in this work is based on transformer architecture [40] by adding MWE components to handle cross-lingual problem. Transformer was chosen because it is state-of-the-art in many NLP topics including CLS and has proven to be able to perform text generation well [41] [42].

Inspired by Zhu et al. [15], the architecture also added a second decoder with a single shared encoder. During training, the input of CLS decoder (decoder1) is a summary in Bahasa Indonesia, while the input of MS decoder (decoder2) is a summary in English. MS decoder is only used during training. So, in the training process, encoder-decoder1 is trained for CLS problems, while encoder-decoder2 is trained for MS problems. The losses from the two are combined and can be calculated as follows:

$$J(w) = -\frac{1}{N}\sum_{i=1}^{N}[y_i^a \log(\hat{y}_i^a) + (1 - y_i^a)\log(1 - \hat{y}_i^a)] + \\ -\frac{1}{N}\sum_{i=1}^{N}\left[y_i^b \log\left(\hat{y}_i^b\right) + \left(1 - y_i^b\right)\log\left(1 - \hat{y}_i^b\right)\right] \quad (1)$$

where w are parameters in the model, y_i^a and y_i^b are the correct labels for both tasks, and $\hat{y}_i^a$ and $\hat{y}_i^b$ are predictive labels for both tasks. The combined loss was used to update the weight. During test, the MS decoder was ignored. The architectural diagram can be seen in Fig. 1.

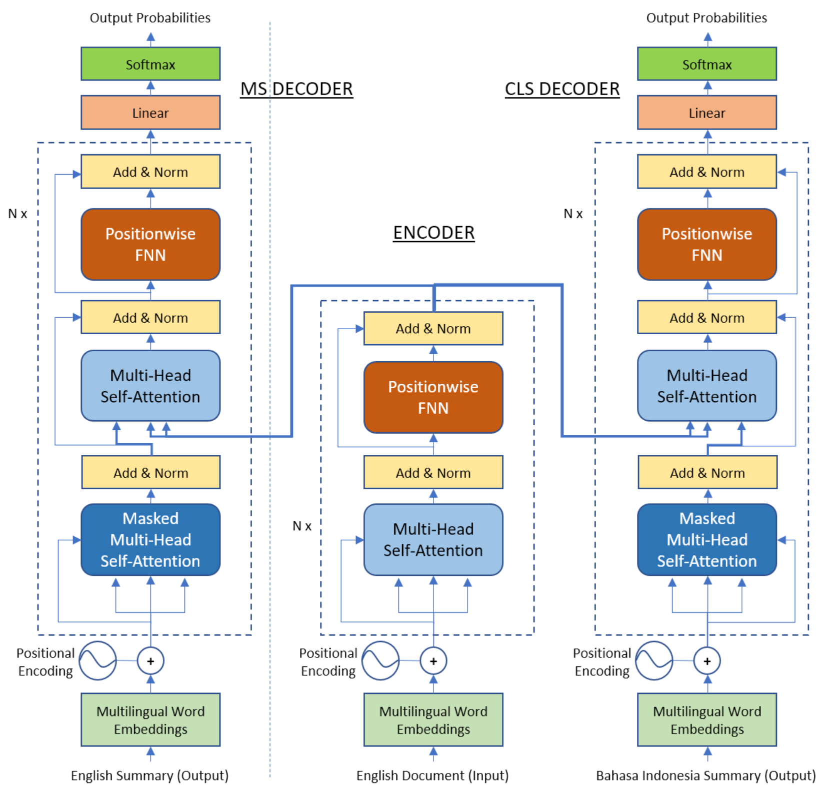

Fig. 1. CLS architecture

B. *Cross-Lingual Summarization Architecture Components*

1) Input and output: The input of the encoder is an English document $x = (x_1, x_2, x_3, \ldots, x_m)$. This document is tokenized using a subword tokenizer. Words that occur infrequently will be segmented into several tokens, while words that occur frequently (high frequency) are left as is. This sequence of tokens is then converted into a sequence of indexes according to the vocab. The output of CLS decoder is a Bahasa Indonesia summary and the output of MS decoder is an English summary. At the time of training, the tokens here are also an input that will be converted into a sequence of indexes according to the vocab. Vocab for Bahasa Indonesia is different from vocab for English. At the time of test, the CLS decoder generates a Bahasa Indonesia summary $y = (y_1, y_2, y_3, \ldots, y_n)$ while the MS decoder is ignored.

2) Multilingual word embeddings: The proposed CLS architecture incorporated pre-trained MWE from MUSE [36]. We use MUSE because it can cover most of the words in vocab while maintaining vocab cohesiveness. MWE Component maps the input English document $x = (x_1, x_2, x_3, \ldots, x_m)$ into a sequence of embeddings $Z = (Z_1, Z_2, Z_3, \ldots, Z_m)$ whose size varies with respect to the source sequence length.

3) Positional Encoding: This component provides position information to the tokens in the sequence by adding the "positional encodings" to the embeddings. The positional encodings are obtained using the following formula:

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d})$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d}) \quad (2)$$

where pos is the position $(0 \le pos < \frac{L}{2}$, where L is the length of the sequence), d is the embedding dimension, and i is used for mapping to indices of elements in positional encoding vector $(0 \le i < \frac{d}{2})$. Positional encoding is necessary because the proposed transformer-based architecture does not consider positional information or word order. Without positional encoding, this architecture is essentially a bag-of-words model.

4) Multi-head self-attention: Multi-head self-attention consists of several heads which correspond to scaled dot-product attention (self-attention). Called self-attention because it can generate its own value of query (Q), key (K), and value (V). These three values are abstractions that represent the input needed to calculate the attention weight. It is obtained using the following formula:

$$Q = ZW_Q$$
$$K = ZW_K$$
$$V = ZW_V \quad (3)$$

where Z is the input embeddings and W_Q, W_K, and W_V are learnable matrices. The scaled dot-product attention is then calculated using the scaling factor of $\frac{1}{\sqrt{d_k}}$.

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \quad (4)$$

where $d_k$is dimension of K. The output of each head is then concatenated to get the final values.

$$MultiHead(Q,K,V) = Concat(head_1, head_2, \ldots, head_h)W^O$$
$$where\ head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \quad (5)$$

where h is the number of heads and W^O, W_i^Q, W_i^K, and W_i^V are learnable matrices. Multi-head self-attention on the decoder is also called encoder-decoder attention because it receives input from the encoder and decoder. There is a mapping of the relation between input and output here. This component can be calculated in parallel because the attention weight here is independent of one another.

5) Masked multi-head self-attention: This component has a similar function to the multi-head self-attention component. However, unlike multi-head self-attention, masked multi-head self-attention has a function to mask output that has not been seen/predicted. The model must predict the output based on the results of the previous output and must not look at the output that appears later. When the matrix operation is performed, the output that has not been seen/predicted will be masked/changed to zero so that it cannot be seen by the model.

6) Positionwise FNN: Positionwise FNN is a simple feedforward neural network that runs for all attention weights. Its main task is to convert the attention weight into an acceptable form for the next step, such as encoder layer, decoder layer, or linear layer. This component consists of two linear layers. The first layer uses ReLU activation.

$$FFN(x) = \max(0, xW_1 + b_1)\,W_2 + b_2 \quad (6)$$

7) Linear Layer and Softmax: Linear layer is a component in the form of a feedforward layer that acts as a classifier. This component is used to adjust the dimensions as needed. For example, to accommodate the number of words (classes) in the vocab. Finally, there is the softmax layer which converts the vector into a probability distribution.

IV. Experiments

This section describes the experiments. First, we describe the dataset which includes English documents and their summary, Bahasa Indonesia documents and their summary, and English-Bahasa Indonesia parallel corpus. Then, we explain the implementation and model variations including the baseline model. Finally, we discuss the evaluation metrics used to measure the performance of CLS models.

A. Dataset

Developing a CLS system is a challenging task because it involves two different languages, especially if the language is a low-resource language. The experiments conducted in this work require a cross-lingual dataset from English documents to Bahasa Indonesia summaries which is not readily available. Therefore, the dataset needs to be created first. Creating a dataset from scratch for deep learning-based models can take a lot of time and effort. The alternative is to create a new dataset by utilizing existing data. The closest problem to CLS is the MS problem. The best dataset as a basis to work with is a dataset of MS for Bahasa Indonesia because the output target of our CLS model is a summary in Bahasa Indonesia. We use IndoSum [43] as the basis for creating a new CLS dataset. IndoSum is an attempt to create a benchmark dataset for Bahasa Indonesia summarization. This dataset contains nearly 20000 news articles taken from online websites. This number is still relatively small when compared to the available English summarization dataset. Each article has an abstract summary that was created manually by 2 native Bahasa Indonesia speakers. The dataset has 6 categories: entertainment, inspiration, sports, show world, headlines, and technology. The dataset has been divided into 5-fold cross-validation and has been divided into training sets, development sets, and test sets. In this work, only the first fold was used. This dataset is written in JSON format.

Based on the need to train the CLS models, IndoSum's articles and summaries need to be translated into English. English articles are used as input for the CLS model. In addition, these English articles and their English summary are used to train the English MS model used in the pipeline approach. The strategy for creating the CLS dataset can be seen in Fig. 2. At first the original dataset of Bahasa Indonesia documents and summaries are translated into English using Google Translate (forward translation). The result of this English translation is then translated back into Bahasa Indonesia (back translation). This is done to ensure the quality of the CLS dataset. The back translation results were evaluated against the original dataset using BLEU [21]. The evaluation of the back translation can be seen in Table I. The evaluation is the cumulative BLEU score calculated using the NLTK. The results show that the quality of the cross-lingual dataset is quite good. As a comparison to get the intuition of the BLEU score, specific machine translators from English to Bahasa Indonesia have been reported to have BLEU scores of 25.3 [44] and 24.5 [45]. In the end, we obtained the Sum (Ina) dataset to train Bahasa Indonesia summarization, the Sum (Eng) dataset to train English summarization, and the CLS (Eng-Ina) dataset to train CLS as can be seen in Fig. 3. The total of data is 18774 document-summary pairs. The statistics of the CLS dataset are presented in Table II. To train machine translators from English to Bahasa Indonesia, the Pan Asia Networking Localization (PANL) parallel corpus dataset was used [45]. This data is divided into 15373 train data, 3845 validation data, and 4806 test data.

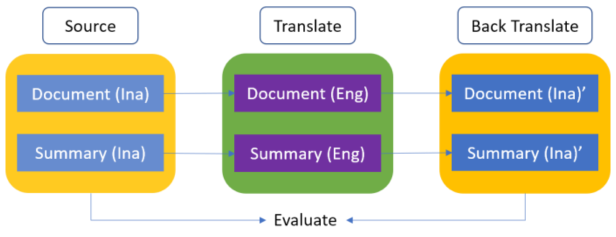

Fig. 2. CLS dataset creation strategy

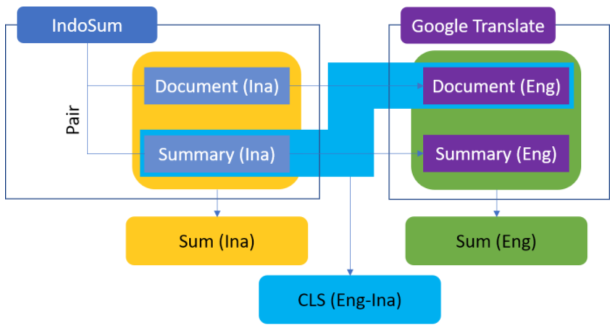

Fig. 3. CLS dataset diagram

TABLE I. CLS DATASET EVALUATION

Back Translation	BLEU-1	BLEU-2	BLEU-3	BLEU-4
Document	0.8056	0.7046	0.6251	0.5527
Summary	0.7991	0.7076	0.6327	0.5626

TABLE II. DATASET STATISTICS

CLS Dataset	Train	Val	Test
# Documents/Summaries	14262	750	3762
# Average Sentence in Document	20.08	19.89	19.92
# Average Sentence in Summary	4.69	4.65	4.69
# Average English Words in Document	377.50	377.86	374.41
# Average Bahasa Indonesia Words in Document	326.58	327.38	323.84
# Average English Words in Summary	75.35	75.18	75.40
# Average Bahasa Indonesia Words in Summary	64.39	64.22	64.37
# Average English Words in Sentence (Document)	16.80	16.96	16.79
# Average Bahasa Indonesia Words in Sentence (Document)	14.72	14.99	14.71
# Average English Words in Sentence (Summary)	14.54	14.63	14.55
# Average Bahasa Indonesia Words in Sentence (Summary)	12.45	12.57	12.44

B. *Implementation*

In this work, we train 10 CLS models. Two models are pipeline and eight models are end-to-end. All models use transformer as their basis. The model is implemented using Python programming language and TensorFlow library. TensorFlow is an open-source library for machine learning and artificial intelligence. TensorFlow is developed by Google Brain Team. TensorFlow can run on CPUs and GPUs and is available for Linux, macOS, Windows, Android, and iOS operating systems. TensorFlow is widely used for NLP and computer vision applications. Each model is accompanied by a subword tokenizer which is used to prepare the input to be submitted to the model. The vocab size for English is 26331 and the vocab size for Bahasa Indonesia is 27373. All models have the same hyperparameters as can be seen in Table III. Each model uses Adam's optimization [46] with a custom learning rate scheduler. Regarding randomness, all models are trained on the same seed value of 777, whether it is a seed for Python, NumPy, or TensorFlow. The same seed is applied to global conditions as well as to any operations involving randomness. The pre-processing carried out are: 1) converting all text to lowercase; 2) removing symbols, special characters, HTML tags, and emoticons; 3) performing Unicode normalization; and 4) brackets: (), [], {} and all characters in between are discarded. The 26 characters of the alphabet are preserved, digits/numbers are preserved, and some punctuation marks are preserved, such as period, comma, exclamation mark, and question mark.

TABLE III. CLS MODEL HYPERPARAMETERS

BUFFER_SIZE	15000
BATCH_SIZE	64
NUMBER_LAYERS	4
DIMENSION_MODEL	300
DIMENSION_FNN	512
NUMBER_HEADS	5
DROPOUT_RATE	0.1
POSITIONAL_INPUT	202
POSITIONAL_OUTPUT	122

C. *Model Variations*

In this work we build ten variations of the model which are grouped into three main groups: 1) pipeline CLS; 2) end-to-end CLS; and 3) end-to-end CLS with MWE components. We also conducted experiments by implementing two strategies of truncating the input, namely using only the head of the document and using the head and tail of the document. This input truncation strategy is inspired by Sun et al. [47] and Mutasodirin & Prasojo [48] who use this strategy for text classification problems. Both use the BERT model [42] but get different conclusions about this input truncation strategy, so this strategy cannot be generalized yet. In this work, the head strategy means using the first 200 tokens from the document, while the head-tail strategy means using the first 100 tokens from the document and concatenating them with the last 100 tokens from the document. So, the models in groups 2 and 3 are further divided into two groups: models that use the head of the document as input and models that use the head and tail of the document as input. Four out of 10 are baseline models: two pipeline models, PipeTS and PipeST, an end-to-end model that uses a vanilla transformer, VCLS, and an end-to-end model based on Zhu et al. [15], MCLS. The following is an explanation of each model variation:

1) PipeTS: Translation → Summarization. The model is built in a pipeline scheme, starting with the English translation process first and then continuing with the Bahasa Indonesia summarization process. The machine translation model is trained using the PANL dataset. The Bahasa Indonesia summarization model is trained using the Sum (Ina) dataset. Both the translation and the summarization models use a vanilla transformer.

2) *PipeST:* Summarization → Translation. The model is built in a pipeline scheme, starting with the English summarization process first and then continuing with the English translation process. The English summarization model is trained using the Sum (Eng) dataset. The machine translation model is the same as that used in PipeTS. Both the summarization and the translation models use a vanilla transformer.

3) *VCLS:* Head + CLS. The model is built end-to-end with head truncation strategy for the input. The model uses a vanilla transformer architecture but is trained using the CLS (Eng-Ina) dataset.

4) *MCLS:* Head + CLS-MS. The model is built end-to-end with head truncation strategy for the input. The model is based on Zhu et al. [15] CLS architecture. The model is trained with 2-task learning, which is jointly training CLS and MS using CLS (Eng-Ina) dataset and Sum (Eng) dataset.

5) *VCLS_T:* Head-Tail + CLS. The model is built end-to-end with head-tail truncation strategy for the input. The model uses a vanilla transformer architecture but is trained using the CLS (Eng-Ina) dataset.

6) *MCLS_T:* Head-Tail + CLS-MS. The model is built end-to-end with head-tail truncation strategy for the input. The model is based on Zhu et al. [15] CLS architecture. The model is trained with 2-task learning, which is jointly training CLS and MS using CLS (Eng-Ina) dataset and Sum (Eng) dataset.

7) *MWE_VCLS:* Head + CLS + MWE. The model is built end-to-end with head truncation strategy for the input. The model uses a vanilla transformer architecture but is trained using the CLS (Eng-Ina) dataset. This model is equipped with MWE components.

8) *MWE_MCLS:* Head + CLS-MS + MWE. The model is built end-to-end with head truncation strategy for the input. The model uses the proposed CLS architecture explained in section 3. The model is trained with 2-task learning, which is jointly training CLS and MS using CLS (Eng-Ina) dataset and Sum (Eng) dataset.

9) *MWE_VCLS_T:* Head-Tail + CLS + MWE. The model is built end-to-end with head-tail truncation strategy for the input. The model uses a vanilla transformer architecture but is trained using the CLS (Eng-Ina) dataset. This model is equipped with MWE components.

10)*MWE_MCLS_T:* Head-Tail + CLS-MS + MWE. The model is built end-to-end with head-tail truncation strategy for the input. The model uses the proposed CLS architecture explained in Section III. The model is trained with 2-task learning, which is jointly training CLS and MS using CLS (Eng-Ina) dataset and Sum (Eng) dataset.

D. Evaluation Metrics

This work uses a quantitative research method with the independent variable being the transformer-based CLS model and the dependent variable being the evaluation of the model's performance measured quantitatively. Recall-oriented understudy for gisting evaluation (ROUGE) is a suite of measurement metrics and a software package used to evaluate automatic summarization and machine translation [49] [50] [51]. ROUGE is a popular metric widely used to evaluate automatic summarization. The advantages of using ROUGE:

- ROUGE is easy and fast compared to evaluation by humans,
- ROUGE is used by “everyone” so it is easy to compare one research result with another.

ROUGE compares the summary generated by the system with a reference summary or a set of reference summaries. Here are some of the available comparison methods:

- ROUGE-N: Compares n-grams between system summary and reference summary.
- ROUGE-L: Compares the longest matching sequence (LCS) between system summary and reference summary.
- ROUGE-S: Compare skip-bigrams, whether there are any word pairs in a consecutive sentence, gaps are possible. For example, skip-bigram measures the overlap of word pairs that can have a maximum of two gaps between words. The sentence “I eat fried rice” then the skip-bigram is “I eat, I fried, I rice, eat fried, eat rice, fried rice.”

There are still other measurement metrics, for more information refer to Lin [49].

Bilingual evaluation understudy (BLEU) is a method for automatic evaluation of machine translation. The BLEU score is a measurement metric that assesses the degree of similarity between sentences produced by machine translation and reference sentences made by human translators. The more similar, the higher is the value. This method works by counting the number of n-grams in the produced sentence that matches the n-grams in the reference sentence. BLEU was proposed by papineni et al. [21].

V. Results and Discussion

We evaluated the models trained using the ROUGE metric described in the previous section. Specifically, we used ROUGE-1, ROUGE-2, and ROUGE-L. The discussion regarding the experimental results is divided into three parts. First, we discuss the comparison of the pipeline model and the end-to-end model. Second, we discuss the comparison of the end-to-end model with the MWE components and the end-to-end model without the MWE components. In the end, we discuss the strategy of input truncation.

A. Comparison between Pipeline CLS Model and End-to-End CLS Model

Traditionally, CLS is done in a pipeline which involves two steps: summarization and translation. There are two patterns of using a machine translator in a pipeline scheme. The first pattern does the translation on the source document first and then the results of the translation are used to generate the summary. The second pattern performs summarization on the source document first and then the summary is translated into the target language. To construct CLS for these two patterns, three constituent models are needed: English summarization model, Bahasa Indonesia summarization model, and English translation model. We trained a transformer-based English summarization model on Sum (Eng) dataset and a transformer-based Bahasa Indonesia summarization model on Sum (Ina) dataset. Then, we trained a transformer-based English translation model on PANL parallel corpus. The performance of these three models individually can be seen in Table IV and Table V.

We use the two patterns of the pipeline scheme as baseline models (PipeTS and PipeST). These two pipeline models were constructed using MS model and MT model which had previously been trained independently. Both are then tested using the CLS (Eng-Ina) dataset. The results are shown in Table VI. The pipeline approach introduces error propagation. To solve this, we trained several end-to-end CLS models (4 variations of VCLS models and 4 variations of MCLS models) on CLS (Eng-Ina) dataset. The performance of these end-to-end models can also be seen in Table VI. All the end-to-end models outperform the model that uses the pipeline scheme. The end-to-end model can achieve improvement from +0.1298 ROUGE-1, +0.0763 ROUGE-2, +0.1083 ROUGE-L up to +0.2981 ROUGE-1, +0.2084 ROUGE-2, +0.2771 ROUGE-L compared to the pipeline baselines. This verifies the purpose of performing end-to-end summarization. Unlike the pipeline method, the end-to-end model performs CLS in one direct step. This approach avoids the error propagation that occurs when doing it in two steps, namely in the pipeline scheme.

TABLE IV. PERFORMANCE OF OUR MONOLINGUAL SUMMARIZATION

Model	ROUGE-1	ROUGE-2	ROUGE-L
SumINA	0.6456	0.5580	0.6355
SumENG	0.6434	0.5123	0.6088

TABLE V. PERFORMANCE OF OUR MACHINE TRANSLATION

Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4
Tran	0.3683	0.2078	0.1191	0.0628

TABLE VI. ROUGE F1 SCORE

Type	Model	Variations	ROUGE-1	ROUGE-2	ROUGE-L
Pipeline	PipeTS	Translation → Summarization	0.1266	0.0142	0.1038
	PipeST	Summarization → Translation	0.1661	0.0278	0.1313
End-to-End	VCLS	Head + CLS	0.2959	0.1041	0.2396
	MCLS	Head + CLS-MS	0.4087	0.2112	0.3652
	VCLS_T	Head-Tail + CLS	0.2981	0.1093	0.2448
	MCLS_T	Head-Tail + CLS-MS	0.3809	0.1893	0.3385
End-to-End MWE	MWE_VCLS	Head + CLS + MWE	0.3617	0.1519	0.3038
	MWE_MCLS	**Head + CLS-MS + MWE**	**0.4247**	**0.2226**	**0.3809**
	MWE_VCLS_T	Head-Tail + CLS + MWE	0.3347	0.1311	0.2767
	MWE_MCLS_T	Head-Tail + CLS-MS + MWE	0.3886	0.1919	0.3438

B. *Comparison between End-to-End CLS Model with MWE and End-to-End CLS Model without MWE*

We also use two end-to-end models as baseline besides the two pipeline baselines which were discussed in the previous section. The first end-to-end baseline is an end-to-end CLS model built using a vanilla transformer (VCLS). The second end-to-end baseline is an end-to-end CLS model based on Zhu et al. [15] (MCLS). The model is trained with 2-task learning, which is jointly training CLS and MS using CLS (Eng-Ina) dataset and Sum (Eng) dataset.

CLS is a Seq2Seq problem, when given a text sequence in the source language it produces a shorter version of an original text sequence in the target language. In Seq2Seq problem, it is important to learn the relation (alignment) between the input sequence and the output sequence. The CLS model that cannot do a proper alignment mapping is unable to produce a correct and meaningful summary. We propose using MWE to help with this alignment. MWE can represent words in various languages in one vector space. Words that have similar meanings are in proximity. This pre-mapping of words is expected to facilitate the model in aligning word sequences between input and output. The VCLS and MCLS are then modified by adding MWE components (MWE_VCLS and MWE_MCLS).

The experimental results can be seen in Table VI. We can find that the model that utilized MWE components beats the underlying baseline model that does not use it. This shows that the MWE helps to better map the relation (alignment) between source document input in English and its summary output in Bahasa Indonesia. This can happen because every word in both languages is already in the same vector space. The MWE_MCLS model can achieve maximum improvement up to +0.1288 ROUGE-1, +0.1185 ROUGE-2, and +0.1413 ROUGE-L when compared to the end-to-end baselines. This is the best-performing model in our experiments.

C. *Input Truncation Strategy*

Sun et al. [47] and Mutasodirin & Prasojo [48] use an input truncation strategy on text classification problems. The strategy is not only to take the head of the data as input but also to combine it with the tail of the data. Generally, only the head of the data is used because it is assumed that the core information is here. However, their experimental results did not reach the same conclusion, so this strategy cannot be generalized yet. We adapt this strategy to our CLS models which previously only took the head part of the document into a combination of the head and the tail. This adaptation produces models VCLS_T, MCLS_T, MWE_VCLS_T, and MWE_MCLS_T. The experimental results of these four models can be seen in Table VI. However, in general, the results have not been able to exceed the score obtained by the model that only uses the head of the document. The best score is obtained by MWE_MCLS_T. A decrease in performance of -0.0361 ROUGE-1, -0.0307 ROUGE-2, and -0.0371 ROUGE-L when compared to the best model uses the head truncation strategy (MWE_MCLS).

The IndoSum dataset that we use is constructed from online news articles. Upon further examination of this dataset, we found that most of the important information is at the beginning of the document. This information appears in the summary, while the end of the news article generally contains additional information or explanatory information that does not appear in the summary. Strategies that use a combination of head and tail from the document are proven to be unable to improve the performance of the models that originally uses head truncation strategy, it even can decrease the model performance due to including insignificant information in the summary.

VI. Conclusion

In this work, we present the end-to-end abstractive CLS for English documents to Bahasa Indonesia summary. The CLS architecture is based on transformers, modified by adding MWE components to address cross-lingual problems. The architecture also has a second decoder with a shared encoder. This second decoder is used only during training to carry out joint learning between CLS and MS. During the test, the second decoder is ignored. The model is also trained using two input truncation strategies: head and head-tail. The head truncation strategy cuts off and takes the head part of the document as input while the head-tail truncation strategy combines the head and the tail of the document as input. To train the model, we create a new CLS dataset from MS dataset by adapting the round-trip translation technique. The resulting CLS dataset is evaluated using BLEU to ensure its quality.

Based on the experimental results, it can be concluded that the use of MWE improves the performance of the CLS model, specifically for summarizing an English source document into a Bahasa Indonesia summary. The proposed model successfully outperformed the baseline model and improved its performance. The strategy of utilizing information at the end of the data failed in improving the performance of the model. Using only the head part of the data is still better. Furthermore, it can be concluded that the end-to-end model is better than the pipeline model. The use of machine translation is a weak point of the pipeline model because it introduces error propagation.

Acknowledgment

This work was funded by PUTI Q3 Universitas Indonesia 2020-2021 with number: NKB-4375/UN2.RST/HKP.05.00/2020 and supported as part of Visiting Professorship Project on Gaining International Accreditation and Enhancing Academic Reputation Program Universitas Indonesia 2020-2021.

References

[1] D. Radev, E. Hovy and K. McKeown, "Introduction to the special issue on summarization," Computational linguistics, vol. 28, p. 399–408, 2002.

[2] M. F. Mridha, A. A. Lima, K. Nur, S. C. Das, M. Hasan and M. M. Kabir, "A survey of automatic text summarization: Progress, process and challenges," IEEE Access, vol. 9, p. 156043–156070, 2021.

[3] Y.-F. Ma, L. Lu, H.-J. Zhang and M. Li, "A user attention model for video summarization," in Proceedings of the tenth ACM international conference on Multimedia, 2002.

[4] J. Xu and T.-C. Lu, "Seeing the big picture from microblogs: Harnessing social signals for visual event summarization," in Proceedings of the 20th International Conference on Intelligent User Interfaces, 2015.

[5] K. Zhang, W.-L. Chao, F. Sha and K. Grauman, "Video summarization with long short-term memory," in European conference on computer vision, 2016.

[6] H. P. Luhn, "The automatic creation of literature abstracts," IBM Journal of research and development, vol. 2, p. 159–165, 1958.

[7] C. D. Paice, "Constructing literature abstracts by computer: Techniques and prospects.," Inf. Process. Manage., vol. 26, p. 171–186, 1990.

[8] J. Kupiec, J. Pedersen and F. Chen, "A trainable document summarizer," in Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval, 1995.

[9] S. Chopra, M. Auli and A. M. Rush, "Abstractive sentence summarization with attentive recurrent neural networks," in Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2016.

[10] R. Nallapati, B. Zhou, C. Gulcehre, B. Xiang and others, "Abstractive text summarization using sequence-to-sequence rnns and beyond," arXiv preprint arXiv:1602.06023, 2016.

[11] D. Bahdanau, K. Cho and Y. Bengio, "Neural machine translation by jointly learning to align and translate," arXiv preprint arXiv:1409.0473, 2014.

[12] B. Sankaran, H. Mi, Y. Al-Onaizan and A. Ittycheriah, "Temporal attention model for neural machine translation," arXiv preprint arXiv:1608.02927, 2016.

[13] X. Wan, H. Li and J. Xiao, "Cross-language document summarization based on machine translation quality prediction," in Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, 2010.

[14] F. Boudin, S. Huet and J.-M. Torres-Moreno, "A graph-based approach to cross-language multi-document summarization," Polibits, p. 113–118, 2011.

[15] J. Zhu, Q. Wang, Y. Wang, Y. Zhou, J. Zhang, S. Wang and C. Zong, "NCLS: Neural cross-lingual summarization," arXiv preprint arXiv:1909.00156, 2019.

[16] X. Duan, M. Yin, M. Zhang, B. Chen and W. Luo, "Zero-shot cross-lingual abstractive sentence summarization through teaching generation and attention," in Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 2019.

[17] X. Chen and C. Cardie, "Unsupervised Multilingual Word Embeddings," in Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, 2018.

[18] A. F. Abka, M. Pratama and W. Jatmiko, "Cross-Lingual Summarization: English - Bahasa Indonesia," in 2021 6th International Workshop on Big Data and Information Security (IWBIS), 2021.

[19] R. Sennrich, B. Haddow and A. Birch, "Improving Neural Machine Translation Models with Monolingual Data," in Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2016.

[20] G. Lample, A. Conneau, L. Denoyer and M. Ranzato, "Unsupervised Machine Translation Using Monolingual Corpora Only," in International Conference on Learning Representations, 2018.

[21] 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.

[22] W. Ogden, J. Cowie, M. Davis, E. Ludovik, H. Molina-Salgado and H. Shin, "Getting information from documents you cannot read: An interactive cross-language text retrieval and summarization system," in Joint ACM DL/SIGIR workshop on multilingual information discovery and access, 1999.

[23] H. Saggion, D. R. Radev, S. Teufel, W. Lam and S. M. Strassel, "Developing Infrastructure for the Evaluation of Single and Multi-document Summarization Systems in a Cross-lingual Environment.," in LREC, 2002.

[24] L. Yu and F. Ren, "A study on cross-language text summarization using supervised methods," in 2009 international conference on natural language processing and knowledge engineering, 2009.

[25] X. Wan, "Using bilingual information for cross-language document summarization," in Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1, 2011.

[26] X. Wan, F. Luo, X. Sun, S. Huang and J.-g. Yao, "Cross-language document summarization via extraction and ranking of multiple summaries," Knowledge and Information Systems, vol. 58, p. 481–499, 2019.

[27] J.-g. Yao, X. Wan and J. Xiao, "Phrase-based compressive cross-language summarization," in Proceedings of the 2015 conference on empirical methods in natural language processing, 2015.

[28] E. L. Pontes, S. Huet and J.-M. Torres-Moreno, "A Multilingual Study of Compressive Cross-Language Text Summarization," in Mexican International Conference on Artificial Intelligence, 2018.

[29] E. L. Pontes, S. Huet, J.-M. Torres-Moreno and A. C. Linhares, "Cross-language text summarization using sentence and multi-sentence compression," in International Conference on Applications of Natural Language to Information Systems, 2018.

[30] J. Zhang, Y. Zhou and C. Zong, "Abstractive cross-language summarization via translation model enhanced predicate argument structure fusing," IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 24, p. 1842–1853, 2016.

[31] J. Ouyang, B. Song and K. McKeown, "A robust abstractive system for cross-lingual summarization," 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.

[32] M. Junczys-Dowmunt, R. Grundkiewicz, T. Dwojak, H. Hoang, K. Heafield, T. Neckermann, F. Seide, U. Germann, A. Fikri Aji, N. Bogoychev, A. F. T. Martins and A. Birch, "Marian: Fast Neural Machine Translation in C++," in Proceedings of ACL 2018, System Demonstrations, Melbourne, 2018.

[33] A. See, P. J. Liu and C. D. Manning, "Get to the point: Summarization with pointer-generator networks," arXiv preprint arXiv:1704.04368, 2017.

[34] F. Ladhak, E. Durmus, C. Cardie and K. McKeown, "WikiLingua: A new benchmark dataset for cross-lingual abstractive summarization," arXiv preprint arXiv:2010.03093, 2020.

[35] Y. Liu, J. Gu, N. Goyal, X. Li, S. Edunov, M. Ghazvininejad, M. Lewis and L. Zettlemoyer, "Multilingual denoising pre-training for neural machine translation," Transactions of the Association for Computational Linguistics, vol. 8, p. 726–742, 2020.

[36] A. Conneau, G. Lample, M. Ranzato, L. Denoyer and H. Jégou, "Word Translation Without Parallel Data," arXiv preprint arXiv:1710.04087, 2017.

[37] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, "Generative adversarial nets," Advances in neural information processing systems, vol. 27, 2014.

[38] B. Heinzerling and M. Strube, "BPEmb: Tokenization-free Pre-trained Subword Embeddings in 275 Languages," in Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018), Miyazaki, 2018.

[39] M. Artetxe and H. Schwenk, "Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond," Transactions of the Association for Computational Linguistics, vol. 7, p. 597–610, 2019.

[40] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser and I. Polosukhin, "Attention is all you need," in Advances in neural information processing systems, 2017.

[41] S. Edunov, M. Ott, M. Auli and D. Grangier, "Understanding back-translation at scale," arXiv preprint arXiv:1808.09381, 2018.

[42] 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), Minneapolis, 2019.

[43] K. Kurniawan and S. Louvan, "Indosum: A new benchmark dataset for indonesian text summarization," in 2018 International Conference on Asian Language Processing (IALP), 2018.

[44] T. W. Guntara, A. F. Aji and R. E. Prasojo, "Benchmarking multidomain english-indonesian machine translation," in Proceedings of the 13th Workshop on Building and Using Comparable Corpora, 2020.

[45] A. Hermanto, T. B. Adji and N. A. Setiawan, "Recurrent neural network language model for English-Indonesian Machine Translation: Experimental study," in 2015 International conference on science in information technology (ICSITech), 2015.

[46] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in Proceedingsof the International Conference on Learning Representations (ICLR), 2015.

[47] C. Sun, X. Qiu, Y. Xu and X. Huang, "How to fine-tune bert for text classification?," in China national conference on Chinese computational linguistics, 2019.

[48] M. A. Mutasodirin and R. E. Prasojo, "Investigating Text Shortening Strategy in BERT: Truncation vs Summarization," in 2021 International Conference on Advanced Computer Science and Information Systems (ICACSIS), 2021.

[49] C.-Y. Lin, "ROUGE: A Package for Automatic Evaluation of Summaries," in Text Summarization Branches Out, Barcelona, 2004.

[50] C.-Y. Lin, G. Cao, J. Gao and J.-Y. Nie, "An information-theoretic approach to automatic evaluation of summaries," in Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics, 2006.

[51] K. Ganesan, ROUGE 2.0: Updated and Improved Measures for Evaluation of Summarization Tasks, 2018.