Received: June 21, 2024. Revised: July 15, 2024.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Indonesian News Stance Classification Based on Hybrid Bidirectional LSTM and Transformer Based Embedding

Esther Irawati Setiawan[1]* Willyanto Dharmawan[1] Kevin Jonathan Halim[1]
Joan Santoso[1] FX Ferdinandus[1] Kimiya Fujisawa[2] Mauridhi Hery Purnomo[3]

[1]*Institut Sains dan Teknologi Terpadu Surabaya, Indonesia*
[2]*Tokyo University of Technology, Japan*
[3]*Institut Teknologi Sepuluh Nopember, Indonesia*
*Corresponding author's Email: esther@istts.ac.id

Abstract: Stance classification is used to understand the relationship between sentences so that the model can recognize the attitude of a response to a topic, where the attitudes are classified into three, namely supporting (for), neutral (observing), and opposing (against). Furthermore, stance classification could aid the automatic fake news detection. This research is specially made for Indonesian news titles. The proposed model used to recognize these news attitudes is Bidirectional Long Short-Term Memory (Bi-LSTM). Thus, to obtain the word representation vector, the pre-trained Bidirectional Encoder Representations from Transformers (BERT) embedding model from indoBERT is used to process words in Indonesian. In Bi-LSTM, each word representation will be processed twice in a forward and backward direction sequentially, so to get a vector representation of the sentence from the input, the output is taken from the results of the representation process of the last word in the forward direction process and the representation process results of the first word in the backward direction. Then the results of the two outputs are combined to serve as a sentence representation. Based on the test results on the Indonesian news dataset, the model for stance classification task was able to achieve an F1 score with an average of 78.30%, with an F1 score label for (supportive) of 73.10%, label observing (neutral) of 89.57%, and label against (against) by 72.23%. The performance is on par with the results of experiments with several Large Language Models currently available.

Keywords: BERT embedding, Bi-LSTM, Indonesian news, Stance classification, Large language models.

1. Introduction

Fake news is a big problem, especially on how to check the validity of the data. Due to high number of data spread online in social media and internet, it is hard to check the validity of each news or information that spread in various social media. This makes a challenge on how to disseminate and get information very quickly to obtain the factual information [1]. One of major challenge is how to find the factual information since it is easy people to believe and spread unverified information or news. Although it does not pose a threat, new perceptions of the information received can spread and influence political and social conditions. Usually, the truth of information on social media can be checked by looking for related news from other sources [2, 3] to obtain the factual judgement of new are fake or not.

One of the problems is how to find the correct information that we receive from the information source. Most of the user find the truth source by finding the number of news that has the same opinion or judgement. This problem leads a problem how to automatically analyse large information that spread on the internet. Because the amount of data is growing rapidly and rapidly, it is very difficult to use human power to do this. The best solution is to use some machine learning approaches give an advantage on how to process the information to find the truth of the news offered.

The method offered to answer this problem is to use the stance classification proposed in this research. Utilizing stance classification for finding the

emerging information can easily identified [4]. Stance classification [5, 6] shows the position of support or response from public opinion to the news using classification techniques based on data or attitudes toward a particular target [7]. The proposed approach to stance classification analysis is to categorize related news into three classes, namely for, observing, and against. The meaning of for class means supporting, observing class means neutral, and against class means opposing [8]. If the information we have gets a lot of responses from other news, it means that the news we have is most likely true. If the information we have gets a lot of observing responses from other news, it means that the truth cannot be known. Then if the information we have gets a lot of against responses from other news, it means that most likely the news we have is wrong or a hoax [9, 10].

In this paper we propose a stance classification model [11, 12] by utilizing Transformer based encoding BERT and Bi LSTM as mentioned in [13, 14]. The main reason we chose this model are BERT Embedding is a word embedding method [15] used to represent text in vector form and Bi-LSTM is a machine learning method used to create and train models so that the model can capture the sequential word order that show how the data context so it can be applied to other applications [16].

Several works of stance classification did not incorporate the contextual embedding of each news. This context will help the model to understand on how the similar from each news that take as an input in the model. This problem was solved by utilizing the BERT model as the embedding methodology to obtain the contextual vector from each word.

Our works was divided into several section. Section 1 discuss about the introduction and research motivation of this study. Section 2 elaborates on the literature review from related state-of-the-art-research. Section 3 defines about the proposed methodology and section 4 consists of the experiments that successfully conducted by the authors. Finally, section 5 contains the conclusion and further works of this manuscript.

2. Literature review

Stance classification is an essential task in natural language processing that aims to ascertain an author's stance (neutral, supporting, or against) on a particular subject. This section examines a variety of methodologies that have been explored in prior research, including the techniques employed, their objectives, and their relevance to our work on stance classification with BERT and BiLSTM, with a particular focus on Indonesian language.

Nic [1] conducted a significant study that employed data and analysis on digital news consumption to comprehend trends in media engagement. Although this study offers valuable insights into news literacy and brand trust, it does not explicitly address stance classification; rather, it concentrates on consumption patterns. Our method is unique in that it directly addresses stance classification within Indonesian digital news.

In the context of social media credibility analysis, the proposed method in [2] employed BERT embeddings and machine learning to detect stances. Their method emphasizes the significance of robust embeddings in comprehending stance, a technique that we also implement, but in the context of Indonesian social media, and in conjunction with BiLSTM to improve performance.

Using image-text-concept features, [3] implemented multiview sentiment analysis in the context of Indonesian social media. Their multimodal approach illustrates the importance of integrating a variety of data types, which is distinct from our text-focused approach but serves to emphasize the potential of advanced feature integration. Our current proposed research concentrates on the classification of stances using text data.

Adversarial domain adaptation for stance detection was the primary focus of [4], which demonstrated the efficacy of transferring knowledge across domains. Although this contributes to our comprehension of domain adaptation, our methodology does not explicitly address domain shifts; rather, it seeks to enhance the accuracy of stance detection within a single domain.

The research framework in [5] addressed the importance of a real-time tweet classification framework, although they did not provide any specifics. In addition, [6] employed a convolutional neural network to detect tweet stances and verify the veracity of rumors, surpassing baseline classifiers across a variety of event data with robust F1 scores. Our research distinguishes itself by employing BiLSTM and BERT embeddings to more effectively capture sequential dependencies in Indonesian datasets.

Imron [7] implemented a combination of BERT, LSTM, and CNN in an additional investigation to implement aspect-based sentiment analysis of marketplace evaluations. This illustrates the practicality of hybrid models, which are consistent with our utilization of BiLSTM and BERT for stance classification. Our dataset, which is derived from

Table 1. Previous Approaches on Stance Classification

Author(s)	Techniques Used	Objective	Disadvatages
Karande et al. [2]	BERT embeddings, Machine Learning, Stance Detection	Stance detection for social media credibility analysis	High computational cost and potential for bias in training data
Xu et al. [4]	Adversarial Domain Adaptation, Stance Detection	Stance detection	Vulnerable to sophisticated adversarial attacks
Chen et al. [6]	Convolutional Neural Networks, Stance Detection, Rumor Verification	Stance detection and rumor verification	High dependency on labeled data and potential overfitting
Imron et al. [7]	BERT, LSTM, CNN, Aspect-Based Sentiment Analysis	Aspect-based sentiment analysis of marketplace product reviews	Requires extensive preprocessing and tuning
Kotonya & Toni [9]	Gradual Argumentation Evaluation, Stance Aggregation	Stance aggregation in fake news detection	May not handle contradictory evidence effectively
Alsafad [10]	Machine Learning, Stance Classification	Stance classification for fake news detection	Limited interpretability of machine learning models
Du et al. [11]	Neural Attention Networks, Stance Classification	Stance classification	Requires large annotated datasets for training
Bar-Haim et al. [12]	Stance Classification, Context-Dependent Claims Analysis	Stance classification of context-dependent claims	Difficulty in handling ambiguous contexts
Kochkina et al. [13]	Sequential Approach, Branch-LSTM, Rumor Stance Classification	Rumor stance classification	Complexity in capturing long-range dependencies
Setiawan et al. [14]	Sentence Embedding, LSTM, Stance Analysis	Indonesian stance analysis of healthcare news	Language-specific model limitations
Gao et al. [15]	Contrastive Learning, Sentence Embeddings	Sentence embedding learning	May require fine-tuning for specific tasks
Wang & Yang [16]	Attention-Based BiLSTM, Knowledge Distillation, BERT, Relation Classification	Relation classification	High computational cost and complexity in model training
Derczynski et al. [17]	Rumor Veracity Determination, Support Analysis	RumourEval: Determining rumour veracity and support	Handling diverse and noisy social media data
Gorrell et al.[18]	Rumor Veracity Determination, Support Analysis	RumourEval 2019: Determining Rumour Veracity and Support	Complexity in integrating multiple sources of information
Li et al. [19]	Content Analysis, User Credibility, Propagation Information	Rumor detection on social media	Dependency on the quality of user credibility data
Peshterliev et al. [20]	Elastic-net Linear Models, Text Classification, Named-Entity Recognition	Text classification and named-entity recognition	May not capture complex patterns in data
Dey et al. [21]	Subjectivity Analysis, Sentiment Polarity, Two-Phase Approach	Twitter stance detection	High sensitivity to subjectivity and sentiment variations
Küçük & Can [22]	Dataset Annotation, Named Entity Recognition, Stance Detection	Tweet dataset annotation	Dataset may not be representative of broader Twitter data
Heinisch [23]	Stance Classification, Argument Search	Stance classification in argument search	Handling complex argument structures
Ravichandiran [24]	BERT, NLP Models	Build and train state-of-the-art NLP models	High computational cost and resource requirements
Liu et al. [25]	Roberta, BERT Pretraining	BERT pre-training approach	Resource-intensive training process
Devlin et al. [26]	BERT, Bidirectional Transformers, Pre-training	Pre-training deep bidirectional transformers	High computational cost and potential biases in training data
Inoue [27]	Multi-Sample Dropout, Accelerated Training, Generalization	Accelerated training and better generalization	Increased training complexity
Putra et al. [28]	Multimodal Models, Open Models	Analyze stance on tweets related to COVID-19 vaccination, considering sentiment towards different aspects of the vaccine	Handling multi-task learning complexities

Indonesian social media and news, offers a distinctive testing environment for these methodologies.

The research in [8] underscored the significance of attention mechanisms by utilizing bidirectional GRU and multi-level attention for targeted aspect-based sentiment analysis. These components are also essential to our methodology, which employs multi-level attention mechanisms to improve classification accuracy.

Kotonya and Toni [9] investigate the importance of stance detection in the identification of false news by employing a distinctive approach that applies gradual argumentation semantics to bipolar argumentation frameworks that are derived from stance detection.

The authors in [10] expand upon this theme by investigating the function of machine learning in the classification of stances in the context of the detection of false news. This theme is also relevant to our examination of Indonesian social media and news, as these studies underscore the importance of stance detection in misinformation contexts.

Du [11] proposed a new neural network-based model that integrates target-specific information into stance classification. Their attention mechanism is anticipated to identify the critical sections of the text that pertain to the target, which is consistent with our model's emphasis on context consideration and attention mechanisms in Indonesian datasets.

Bar-Haim [12] introduced the Claim Stance Classification task and presented the first benchmark dataset for this task. Their methodology incorporates an innovative algorithm for contrast detection that surpasses numerous baselines and yields promising outcomes. Our methodology is based on similar foundational concepts, but it seeks to enhance the approach by incorporating BERT and BiLSTM for the Indonesian language context.

Sequential approaches and sentence embeddings were investigated by [13, 14] for the classification of rumor and stance. These investigations demonstrate a variety of strategies that are used to inform our combined BERT and BiLSTM method. By emphasizing Indonesian datasets developed for this investigation, our methodologies are customized to the language's contextual and linguistic subtleties.

In order to optimize model performance, knowledge distillation and contrastive learning are implemented [15, 16]. Our pursuit of efficient model training and robust embeddings is motivated by the techniques from these studies, which are specifically applied to our Indonesian dataset.

Dale [29] and Cui [30] respectively addressed broader challenges in NLP and Chinese NLP, which inspires the context of our work within the broader language processing landscape. Nevertheless, our attention is focused on Indonesian data, which presents its own distinctive opportunities and challenges.

The significance of exhaustive datasets and veracity analysis was underscored by [17, 31], who focused on large-scale corpus pre-training and rumor veracity determination. We have developed an exhaustive Indonesian dataset for stance detection purposes, and this is pertinent to our objectives.

Gorrell [18] and Li [19] emphasized the ongoing challenges and advancements in rumor verification and stance analysis, building on the work of SemEval-2017. In the same vein, our research endeavors to improve stance classification techniques with a particular emphasis on the Indonesian context, thereby contributing to this discipline.

In their investigations of named entity recognition, Gorinski [32] and Peshterliev [20] examined methods for text classification and Electronic Health Record (EHR) data, respectively. These methodologies enhance our stance classification framework by providing information on preprocessing and feature extraction techniques that are pertinent to our Indonesian dataset.

Dey [21] developed a two-phase feature-driven model for Twitter stance detection that significantly outperformed the current state of the art. In addition to this, a research in [22] provided a Turkish tweet dataset that was annotated for named entity and stance information. This dataset demonstrates practical approaches to feature extraction and dataset construction that have an impact on our methodology.

Heinisch [23], Madry [33], and Ravichandiran [24] analyzed the role of stance classification in argument search, adversarial assault resistance, and the training of state-of-the-art NLP models. Our implementation, notably in the Indonesian context, considers these insights into model robustness and training strategies.

Liu [25] and Devlin [26] presented findings on BERT pretraining, which are fundamental to our utilization of BERT embeddings. Our research further improves the stance classification in Indonesian by incorporating BiLSTM into these embeddings.

Inoue [27], Lin [34], G.Team [35, 36] discussed the improvement of dropout techniques and multimodal models, which contribute to our comprehension of model design and training. Nevertheless, our primary objective is to develop a text-based stance classification system that is specifically designed for Indonesian data.

A research in [28] employed sentiment analysis and stance detection to investigate the sentiment

toward various aspects of the COVID-19 vaccine in tweets. Their methodology underscores the importance of rigorous evaluation in stance classification, which is essential to our investigation of Indonesian social media and news datasets.

Chang [37] assessed large language models (LLMs), providing vital insights for researchers and developers who are involved in the development of LLMs. This is consistent with our emphasis on the utilization of sophisticated NLP techniques, including BERT and BiLSTM, for the purpose of stance classification in the Indonesian context.

Zhuang et al. [38] presented a thorough analysis of various transfer learning methods, including inductive, transductive, and unsupervised approaches. Their findings demonstrated that transfer learning significantly improves model performance in scenarios with limited training data.

Our research suggests a comprehensive stance classification model that capitalizes on sequence modeling capabilities and robust embeddings by combining the advantages of these previous methods and addressing identified deficiencies.

In contrast to previous research, our approach incorporates BERT and BiLSTM to improve the accuracy of stance detection within Indonesian datasets that we have developed. Our strategy is designed to foster a more comprehensive comprehension of stance detection in digital communication and to attain performance on par with fine tuning LLMs.

2.1 Deep learning

Deep Learning is a machine learning approach that is based on the workings of the human brain or what is commonly called an artificial neural network. Models in deep learning have the ability to perform feature extraction from data automatically. So, using deep learning [33] can get better results.

2.2 Bidirectional LSTM

Bidirectional Long Short-Term Memory is the development of the LSTM model in which there are two LSTMs whose processes are opposite to each other, namely the direction of moving forward (forward) which processes from the first word to the last word and the direction of moving backward (backward) which processes from the last word to the last word. first. This model is very good for recognizing patterns in sentences because each word in the document is processed sequentially, and understands the word that is in the position before or after it.

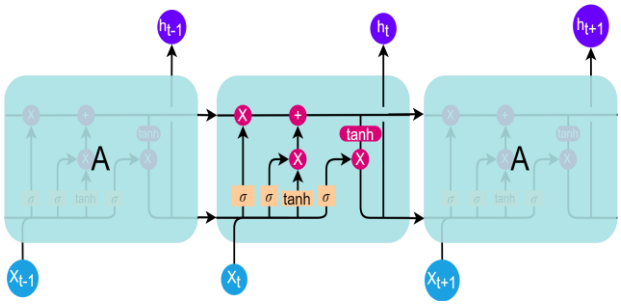

Figure. 1 LSTM Architecture

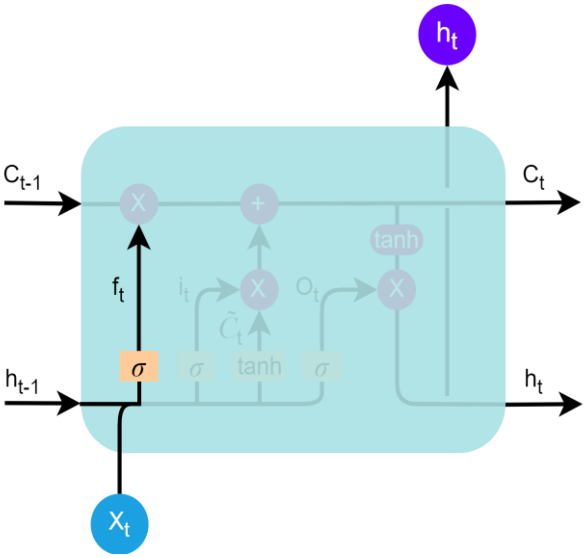

Figure. 2 Forget Gate

Figure 1 is the architecture of the LSTM, in the LSTM there are three gates whose job is to control information or data, namely input gate, forget gate, and output gate. The three gates referred to in the LSTM are layers with sigmoid functions, the output of the sigmoid function is a value of 0 or 1. It can be said that if the value generated from the sigmoid function is 0 then the data is not allowed to enter or is not included in the calculation, and vice versa. If the resulting value is 1, then the data is allowed to enter or can be included in the calculation.

Figure 2 is the forget gate section which has the purpose of erasing information or passing information to the cell state section.

$$f_t = \sigma(W_f * x_t + U_f * h_{t-1} + b_f) \quad (1)$$

In formula 1, it can be seen the formula of the forget gate using sigmoid activation. There are values of W and U which are the weight of the forget gate. Then there is also the value of b which is the bias of the forget gate. Then x is the input value in iteration t and h is the output data value in the previous iteration.

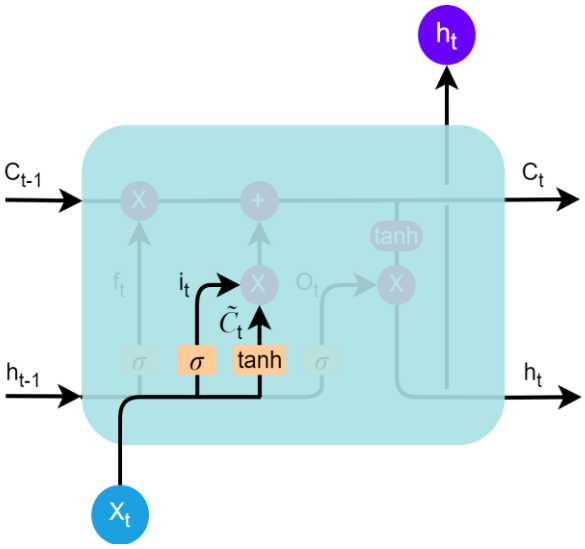

Figure. 3 Input Gate and Cell State Candidate

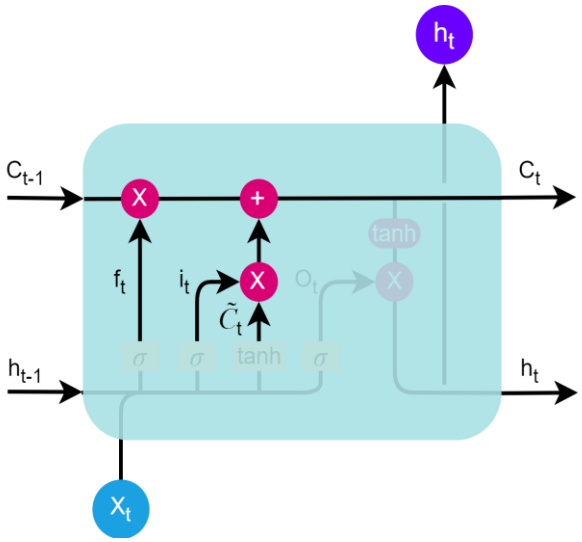

Figure. 4 Cell State

Figure 3 is the input gate and cell state candidate, where the input gate has the purpose of deleting information or forwarding information to the cell state candidate section. Then the cell state candidate is a vector that will later be added to the cell state. Then both the results of the gate input and the cell state candidate are combined with the multiplication mathematical operation which will later be used to update the cell state.

$$i_t = \sigma(W_i * x_t + U_i * h_{t-1} + b_i) \quad (2)$$

In formula 2 can be seen the formula of the input gate using sigmoid activation. There are values of W and U which are the weight of the input gate. Then there is also the value of b which is the bias of the input gate. Then x is the input value in iteration t and h is the output data value in the previous iteration.

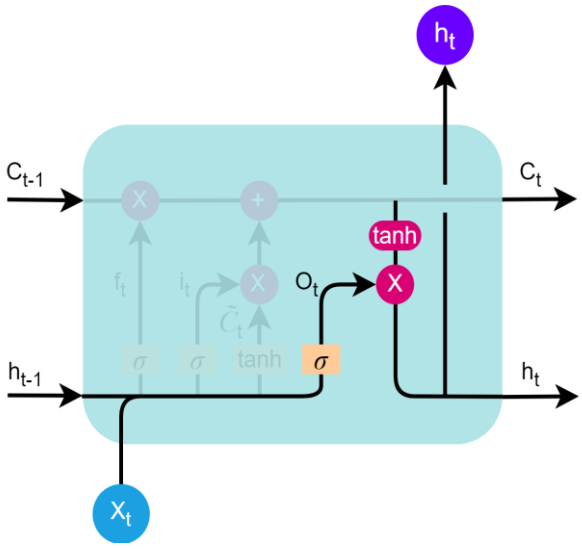

Figure. 5 Output Gate

$$\tilde{C}_t = tanh(W_{\tilde{C}} * x_t + U_{\tilde{C}} * h_{t-1} + b_{\tilde{C}}) \quad (3)$$

In formula 3, it can be seen the formula of the cell state candidate using tanh activation. There are W and U values which are the weight of the cell state candidate. Then there is also the value of b which is the bias of the cell state candidate. Then x is the input value in iteration t and h is the output data value in the previous iteration.

Figure 4 is part of the cell state, after getting the results from the forget gate, and the multiplication between the input gate and the cell state candidate, the cell state can be updated.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (4)$$

In formula 4 can be seen the formula of the cell state. There is a forget gate value in iteration t multiplied by the cell state value in the previous iteration. Then there is the input gate value at iteration t multiplied by the cell state candidate value at iteration t. Then the two results are added to get the cell state value.

Figure 5 is part of the output gate and the result of the LSTM unit. Furthermore, it is necessary to determine the results of the resulting output to be directed to the next unit, the resulting output will be based on the filtered cell state.

$$o_t = \sigma(W_o * x_t + U_o * h_{t-1} + b_o) \quad (5)$$

In formula 5 can be seen the formula of the output gate using sigmoid activation. There are values of W and U which are the weights of the output gate. Then there is also the value of b which is the bias of the gate output. Then x is the input value in iteration t and h is the output data value in the previous iteration.

$$h_t = o_t * tanh\,(C_t) \quad (6)$$

In formula 6, it can be seen that the formula for the unit results, first of all, enter the cell state through the tanh layer to change the value between -1 to 1, then perform the multiplication operation of the gate output result with the cell state.

2.3 BERT embedding

BERT Embedding[24, 25] is a contextual word embedding that is based on the transformers architectural model. Contextual here means that BERT can understand the context of sentences[26]. For example, there are the sentences "He was bitten by a Python snake" and "Python is his favorite programming language", the word "Python" here has two different meanings, the first sentence refers to the type of snake, and the second sentence refers to the programming language. In BERT the vector value of the word "Python" will have a different value.

In BERT, it is necessary to change the input into three embeddings first, namely token embedding, segment embedding, and position embedding. Token embedding is a tokenization process on input. Tokenization in BERT is done by adding a “[CLS]” token at the beginning of a sentence and a “[SEP]” token at the separator between sentences. Then To overcome the problem of out-of-vocabulary (OOV), BERT performs word splitting into sub-words on words that are not found in the vocabulary list. For example, there is the word "work" that is not found in the vocabulary list, then the word "work" will be split into "be" and "##kerja"(*##ing*), the "#" sign indicates that the token of this word is related to the previous token. Figure 6 shows the input from BERT.

Segment embedding is used to distinguish tokens from one sentence to another. Apart from the “[SEP]” token, additional markers need to be added to distinguish between two or more sentences. Position embedding is used to determine the order of tokens or words because in BERT the data is not processed sequentially, so position embedding is needed to arrange the output sequence according to the input.

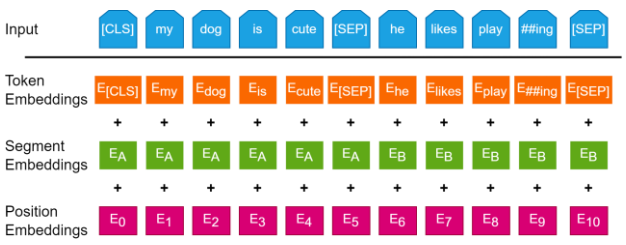

Figure. 6 BERT Embedding Input

3. Stance classification

3.1 System architecture

The implementation process of this research starts from collecting data from social media and Indonesian news on the internet. Then the data cleaning process is carried out on the data that has been collected. Then the process of taking additional features that may have an effect on increasing classification accuracy is carried out. Then the word vector formation process is carried out using BERT. Then a classifier model is created to perform stance classification. Figure 7 describes the flow of this research.

In Figure 8 you can see the process that the topic sentence and response sentence go through to get a label for the stance classification. The sentence is first tokenized, resulting in an array of words contained in the sentence plus tokens that can be processed by BERT embedding. Then the array of words or tokens is processed with BERT embedding, so that each word or token becomes a vector with dimensions according to the number of hidden states in BERT embedding. The resulting number of dimensions is usually 768.

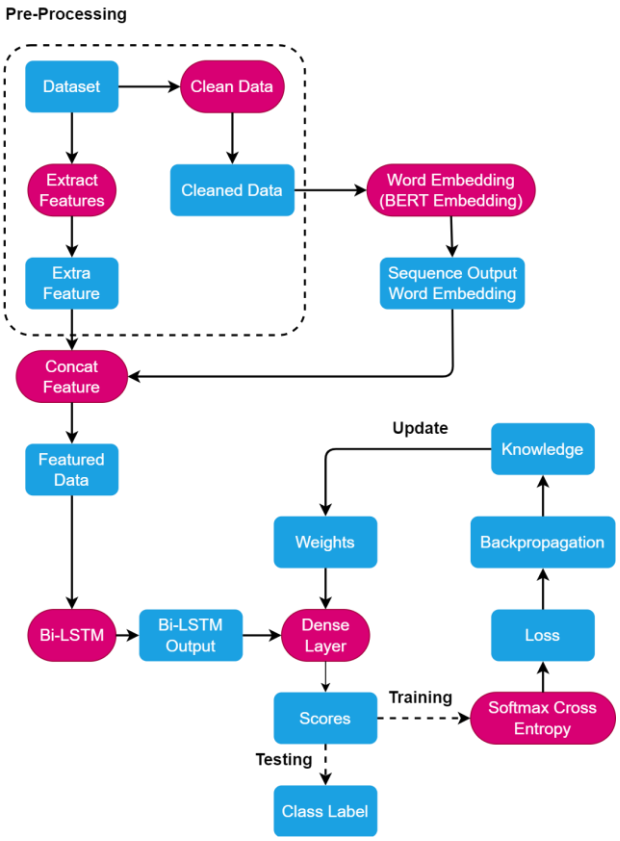

Figure. 7 System Architecture

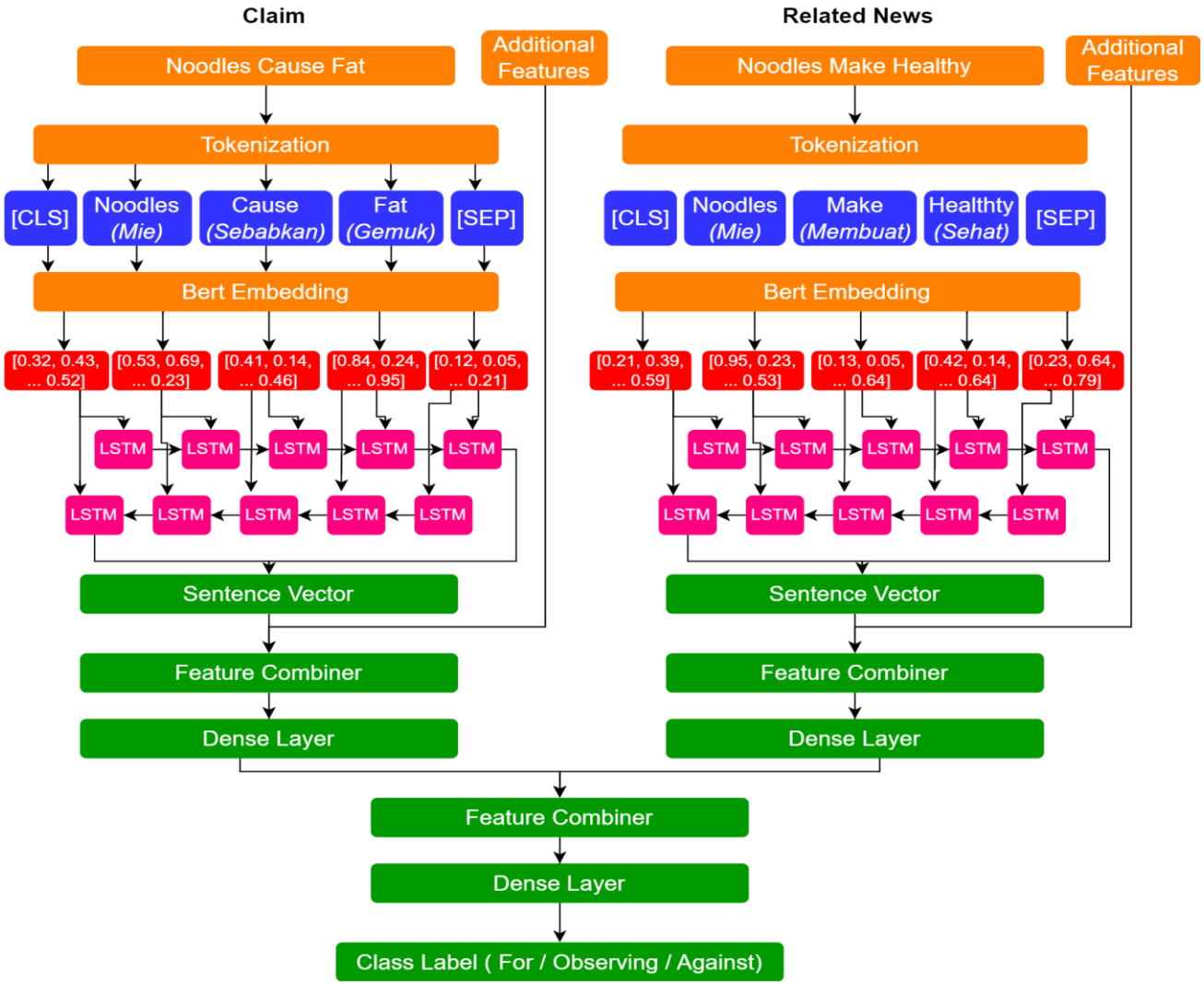

Figure. 8 Flow Pipeline

Then the token or word vector is processed by Bi-LSTM to obtain a sentence vector with dimensions according to the number of hidden states in Bi-LSTM. For example, the dimension of the sentence vector is 300, then it is combined with additional features. For example, the dimension of the additional features is 3, then the resulting dimension of the combination is 303. Then the combination of sentence vectors with additional features is processed by a dense layer. For example, if the dimensions produced by the dense layer are 64, then the two resulting dense layers of sentences that have been processed are combined.

The dimension of the vector obtained from the merger is 128. Then, from the merged vector, processing is carried out on a dense layer with dimension 3, namely according to the number of labels in the stance classification and softmax activation to obtain the probability value of the stance classification label. From this probability value, the highest value can be taken to become the predicted label. For example, the resulting vector obtained is [0.2, 0.5, 0.3] and index 0 represents the "For" label, index 1 represents the "Observing" label, and index 2 represents the "Against" label. So, with the vector results obtained, the label obtained is the label "Observing".

3.2 Dataset

Data collection is done manually on social media. The data sought is the title of the article or news, claims or responses, the number of likes and comments. The following in Table 2, is one example of a pair of article titles, claims, along with the stance of claims.

Each claim contains several article titles with at least one “for”, “observing”, and “against” stance each. There are two datasets used in this research, a dataset with a total of 3378 and a dataset of a total of 3941. Statistics from the dataset can be seen in Table 3 and Table 4.

Table 2. Claims and Stance Example

Claim	Article Title	Stance
Traditional Medicine Is Not Necessarily Safe (*Obat Tradisional Belum Tentu Aman*)	Herbal Medicines Can Also Be Dangerous, These Are the Characteristics (*Obat Herbal Juga Bisa Berbahaya, Seperti Ini Ciri-cirinya*)	For
	Are herbal medicines safe for users? (*Apakah Obat Herbal Aman Bagi Penggunanya?*)	Observing
	Traditional herbal medicines have no danger in their use (*Obat Herbal Tradisional Tidak Memiliki Bahaya Dalam Penggunaannya)*	Against

Table 3. Statistic Dataset 1

Stance	Amount
For	1126
Observing	1126
Against	1126
Total	3378

Table 4. Statistic Dataset 2

Stance	Amount
For	1391
Observing	1275
Against	1275
Total	3941

3.3 Data cleaning

Data cleaning is done by removing symbols in the claim text and article titles that can interfere with the classification results. In this process, the removal of punctuation marks such as periods or commas is not done, because removing punctuation marks can affect BERT in understanding the context of the sentence.

In Table 5, it can be seen that given input sentences that have symbols or punctuation marks produce output that does not have sentence symbols or punctuation marks. This can be done by swapping symbols or punctuation with empty text. In this method, we use the BERT method, which can be done with commas to understand the context better. Then, to delete data that has a null value, you can do this by providing a condition on the index that deletes the data if the condition is met.

Table 5. Statistic Dataset 1

Input	Output
Viral!! Liverpool's jersey beats MU. *(Viral!! Jersey Liverpool Bantai MU)*	Viral Liverpool jersey beats MU *(Viral Jersey Liverpool Bantai MU)*
[Hoax] Ice Water Causes Heart Disease *([Hoax] Air Es Sebabkan Sakit Jantung)*	Hoax Ice Water Causes Heart Disease *(Hoax Air Es Sebabkan Sakit Jantung)*
Hoax: Instant Noodles + Chocolate *(Hoax: Mie Instan + Coklat)*	Chocolate Instant Noodle Hoax *(Hoax Mie Instan Coklat)*

Table 6. Example of Stemming Data Input and Output

Input	Output
Farmers work plowing fields *(petani bekerja membajak sawah)*	Farmers work plowing rice fields (petani kerja bajak sawah)
Budi went to play in the park *(budi pergi bermain di taman)*	Budi went to play in the park (budi pergi main di taman)
mother went shopping at the market *(ibu pergi berbelanja ke pasar)*	mother went shopping at the market (ibu pergi belanja ke pasar)

Table 7. Example of Tokenization Input and Output

Input	Output
Viral Liverpools jersey beats MU. *(Viral Jersey Liverpool Bantai MU)*	[[CLS], Viral, Jersey, Liverpool, Bantai, MU, [SEP]]
Ice Water Hoax Causes Heart Disease *(Hoax Air Es Sebabkan Sakit Jantung)*	[[CLS], Hoax, Air, Es, Sebabkan, Sakit, Jantung, [SEP]]
Chocolate Instant Noodle Hoax *(Hoax Mie Instan Coklat)*	[[CLS], Hoax, Mie, Instan, Coklat, [SEP]]

In Table 6, the sentence input and sentence output results obtained after the stemming process is displayed, each word in the sentence becomes a base word. To carry out the stemming process for Indonesian, you can use the Python library from Sastrawi. In the literary library, you can import "StemmerFactory", then call the stem function with sentence input. Then the function will return the output results from the stemming process.

Table 7 displays the sentence input and output results from the tokenization process. In the output

results there are additional tokens "[CLS]" at the beginning and "[SEP]" at the end. In BERT embedding the token "[CLS]" is used to indicate the initial position of the given input, while the token "[SEP]" is used to separate sentences. If there are two or more sentences input, the token "[SEP]" is useful for indicating the end of each sentence.

Then, after the tokenization process has been carried out, you can pad the list of tokens. This causes the list of tokens to have the same list length and can be processed by the neural network. In BERT embedding you can add the token "[PAD]" at the end of the list to indicate that the token is padding. The following is an example of the input and output results from the padding process.

In Table 8 you can see the input tokens you have and the output results from the padding process. By carrying out the padding process, we get the same list length, in this case the length of the list is 8. Then, after the padding process is carried out, the token can be converted into an ID token. The ID token is a number to identify the token. This process can be done using the function provided by BERT embedding. The following is an example of the input and output results from the process of converting a token into an ID token.

In Table 9, the input token you have produces an output list of numbers from the process of converting it into an ID token. Then from this list of numbers it can be input into the pretrained model from BERT embedding to get a representation value for the word or token which can later be processed by the neural network. Before inputting the list of ID tokens into the pretrained model, you need to create a list of attention masks first. The attention mask list here aims to show which IDs need to be processed, the values of the attention masks are 0 and 1.

It can be said that if the ID token has a value of 0 then the value of the attention mask is 0. Then if the value of the ID token has a value not 0 then the value of the attention mask is 1. After creating the attention mask list you can input the two lists into the BERT pretrained model embedding. The following is an example of the input and output results from the process of getting a token or word representation.

In Table 10, it can be seen that the ID token and attention mask input produces an output representation of the word or sentence. The length of the list of representations obtained depends on the number of hidden layers in the pretrained model used. If the number of hidden layers used is 768 then the length of the representation list obtained is also 768. Then the output results obtained from BERT embedding are of two types.

Table 8. Example of Tokenization Input and Output

Input	**Output**
Viral Liverpool Jersey beats MU *(Viral Jersey Liverpool Bantai MU)*	[[CLS], Viral, Jersey, Liverpool, Bantai, MU, [SEP]]
Ice Water Hoax Causes Heart Disease *(Hoax Air Es Sebabkan Sakit Jantung)*	[[CLS], Hoax, Air, Es, Sebabkan, Sakit, Jantung, [SEP]]
Chocolate Instant Noodle Hoax *(Hoax Mie Instan Coklat)*	[[CLS], Hoax, Mie, Instan, Coklat, [SEP]]

Table 9. Example of Input and Output for Changing ID Tokens

Input	**Output**
[[CLS], Viral, Jersey, Liverpool, Bantai, MU, [SEP], [PAD]]	[2, 19946, 11690, 9603, 1345, 1643, 3, 0]
[[CLS], Hoax, Air, Es, Sebabkan, Sakit, Jantung, [SEP]]	[2, 18442, 514, 1660, 21193, 1252, 2937, 3]
[[CLS], Hoax, Mie, Instan, Coklat, [SEP], [PAD], [PAD]]	[2, 18442, 7703, 8189, 5747, 3, 0, 0]

Table 10. Example of Input and Output Getting Word Representation

Input	**Output**
Token ID : [2, 19946, 11690, 9603, 1345, 1643, 3, 0] Attention Mask : [1, 1, 1, 1, 1, 1, 1, 0]	[0.4812, 1.3011, 0.3649, … , 0.6199]
Token ID : [2, 18442, 514, 1660, 21193, 1252, 2937, 3] Attention Mask : [1, 1, 1, 1, 1, 1, 1, 1]	[0.5493, 0.6933, 0.3582, … , 0.3963]
Token ID : [2, 18442, 7703, 8189, 5747, 3, 0, 0] Attention Mask : [1, 1, 1, 1, 1, 1, 0, 0]	[0.1353, 0.8582, 0.5332, … , 0.5948]

The first output is called sequence output and the second output is called pooled output. The output sequence is the representation of each token or word entered. Pooled output is the result of the representation of all the tokens or words entered, it can be said to be the result of the representation of the sentence. The following is an example of sequence output and pooled output.

Table 11. Example of Output Sequence Results

Word	Word Vector
I *(Saya)*	[0.34, 0.42, 0.33]
take *(Minum)*	[0.44, 0.53, 0.12]
medicine *(Obat)*	[0.65, 0.22, 0.55]

Table 12. Example of Pooled Output Results

Sentence	Sentence Vector
I take medicine *(Saya minum obat)*	[0.29, 0.32, 0.54]
I take medicine *(Budi mengikuti kelas)*	[0.45, 0.64, 0.22]
Adi goes to the mall *(Adi pergi ke mall)*	[0.54, 0.16, 0.46]

Table 13. Example of Normalized Input and Output

Sentence	Sentence Vector
I take medicine *(Saya minum obat)*	[0.29, 0.32, 0.54]
Budi attends class *(Budi mengikuti kelas)*	[0.45, 0.64, 0.22]
Adi goes to the mall *(Adi pergi ke mall)*	[0.54, 0.16, 0.46]

Table 14. Examples of Negation Word Input and Output

Input	Output
three year research from Italy confirms that e-cigarettes are not dangerous *(riset tiga tahun dari Italia pastikan rokok elektrik tidak bahaya)*	1
Our focus is on BPJS policies that are detrimental to society *(fokus kami soal kebijakan BPJS yang merugikan masyarakat)*	0
Benefits of drinking lemon water every morning for the body *(manfaat meminum air lemon setiap pagi untuk tubuh)*	0

In Table 11, an example of the output sequence results from tokens or words with a total of 3 hidden layers in BERT embedding is displayed. The greater the number of hidden layers of BERT embedding, the greater the word representation value obtained. If the input from BERT embedding is a sentence with 5 words, then the number of output results produced is 5-word vector outputs and 1 sentence vector output. Then the output result for the "[PAD]" token will have a value of 0.

In Table 12 is displayed an example of the pooled output results from sentences with 3 hidden layers in BERT embedding. The pooled output value is the output result of the "[CLS]" token. From the results of this word or sentence representation, the neural network will later process it to carry out the stance classification task. Then the neural network will produce an output label from the existing list of labels.

Extract features are the retrieval of additional features which are expected to help determine classification or increase the accuracy of the model to be created. From the existing dataset, the fields that could possibly be used are the number of likes and comments on the topic, and the number of likes and comments on the responses. Then you can normalize the values in the like and comment fields, so that the values do not have a large distance between them. In this case, normalization is carried out, so that the values for the number of likes and comments have a value range between 0 and 1. The following is an example of the input and output results from the normalization process.

In Table 13, it can be seen that the input list of numbers carried out by the normalization process produces an output with values in the list of numbers ranging from 0 to 1. Then another feature that can be added is a status that shows the sentence on the topic or response has the word negation. The way to determine whether a sentence in the topic or response has a negation word, it is necessary to make a list of words that are negation words. The list of negation words used are the words "not", "not", "don't", "not", "not yet", "hoax", and "hoax".

Then, after obtaining a list of negation words, fields can be added to accommodate the values from checking the topic sentence and response. If the topic sentence or response contains one of the words on the list of negation words, it can be given a value of 1. Then, if no negation word is found, it is given a value of 0. The following is an example of the input and output results from the process of checking negation words.

In Table 14, the input sentences and output results from the process of checking negation words from the list of negation words created is shown. Then, from the existing features, additional features can be combined with the main features (representation of topic sentences and responses). After combining the features, the process of dividing train data, validation data and test data can be carried out. Train data is data that will be used for the model training process, validation data is data that will be used to validate the model during the training process, and test data is data that is used to test the model when the training process is complete.

Table 15. Examples of Sequence Output Results

Word	Vector
Saya (*I)*	[0.34, 0.21, 0.46]
Minum (*Took)*	[0.23, 0.56, 0.77]
Obat (M*edicine)*	[0.31, 0.54, 0.55]
Batuk (C*ough)*	[0.32, 0.54, 0.15]

Table 16. Examples of Pooled Output Results

Word	Vector
Saya (*I)*	[0.34, 0.21, 0.46]
Minum (*Took)*	[0.23, 0.56, 0.77]
Obat (M*edicine)*	[0.31, 0.54, 0.55]
Batuk (C*ough)*	[0.32, 0.54, 0.15]

3.4 Additional feature retrieval

The addition of this feature aims to help the classifier model in classifying data. If the feature you want to add is in the form of text or string data, it is necessary to map the data into numbers. However, if the added feature is in the form of numbers, the data normalization can be done first. Another feature that can be added besides the main feature is the number of likes and comments.

3.5 Formation of word vector

The formation of this word vector aims to convert text data into numeric data so that it can be processed by the neural network. In this process, it is necessary to change the claim text and article title text into three embeddings first, so that they can be processed by BERT. In this research, Indonesian pre-trained BERT embedding from indoBERT is used. There are two outputs from BERT, namely sequence output and pooled output. Sequence output is the result of the embedding of each word contained in the sentence, while the pooled output is the result of the embedding of the sentence. Examples of the results of sequence output and pooled output can be seen in Table 15 and Table 16.

3.6 Model classifier

After getting the features that will be processed, then the process of making and training[27, 34] the model can be done. The parameter settings used are the pre-trained BERT layer, the Bi-LSTM layer with the input dimensions according to the embedding dimensions, namely 768, the linear layer with the input dimensions 1536, and the output dimension 3. The model is also formed with the Adam optimizer with a learning rate of 0.00003. The algorithm is displayed as Algorithm 1

Algorithm 1 Training Algorithm

```
FUNCTION train_epoch(model, data_loader, loss_fn, optimizer, scheduler, n_examples):
  SET model to training mode
  INITIALIZE empty list for losses
  INITIALIZE correct_predictions to 0
    FOR each batch in data_loader:
   GET input_ids from batch
   GET attention_mask from batch
        GET stance (labels) from batch
        PASS input_ids and attention_mask through model to get outputs
    GET predicted classes ( preds) from outputs
  CALCULATE loss using loss_fn with outputs and stance
    ADD to correct_predictions the number of correct preds
    APPEND loss value to losses list
   BACKPROPAGATE the loss
    CLIP gradients of model parameters to max_norm of 1.0
        PERFORM an optimization step
  UPDATE the learning rate with scheduler
   RESET gradients of optimizer
    PRINT "Correct Predictions: " + correct_predictions
RETURN correct_predictions divided by n_examples, mean of losses list
```

The train_epoch function is specifically built to manage a single epoch of training for a machine learning model. The model requires various parameters, including the model itself, a data loader to supply the training data, a loss function to evaluate prediction errors, an optimizer to update the model's parameters, a scheduler to modify the learning rate, and the number of examples in the training set.

The function initiates the training mode by using model.train(), which activates training-specific functionalities like dropout. Subsequently, it initializes a empty list to store loss values and a

counter to keep track of accurate predictions. Calculating the length of the data loader provides insight into the size of the dataset being processed during this epoch.

Subsequently, the function proceeds to iterate through each batch of data supplied by the data loader. For every batch of data, it retrieves the input_ids, attention_mask, and stance (the labels). The inputs are used as input to the model in order to produce output logits. The function subsequently identifies the predicting classes by selecting the highest value along the output dimension. The loss is computed by applying the given loss function, which involves comparing the model's outputs with the actual labels. The loss is added to the list of losses, and the accuracy is updated by comparing the anticipated classes with the actual labels.

After calculating the loss, the function carries out backpropagation to determine the gradients of the loss in relation to the parameters of the model. Subsequently, the gradients are cliped to a maximum norm of 1.0 in order to mitigate the issue of exploding gradients, which has the potential to disrupt the training process. The optimizer executes a process to modify the parameters of the model, while the scheduler adjusts the learning rate. The gradients of the optimizer are set to zero in order to be ready for the following iteration.

Upon completing the processing of all batches, the function outputs the total count of accurate predictions. Subsequently, it provides the accuracy, which is computed by dividing the number of right predictions by the total number of examples, as well as the average loss for the period. This technique guarantees that the model is trained on the complete dataset, with its parameters adjusted to minimize the prediction error, while also avoiding problems such as exploding gradients and dynamically adapting the learning rate.

3.7 Performance evaluation

Performance evaluation is carried out based on the F1 score of each stance label and the average F1 macro.

$$Precision = TP / (TP+FP) \quad (7)$$

Precision states how accurate the algorithm or model is to how many positive predictions are true positive (TP) from all positive predicted data (TP+FP).

$$Recall = TP / (TP+FN) \quad (8)$$

Recall states how many positive predictions are true positive (TP) from all truly positive data (TP+FN).

$$F1\ Score = 2(Precision\ .\ Recall\ )/(Precision+Recall) \quad (9)$$

F1 score is the balance value between precision and recall. In this research, the classification_report function from sklearn.metrics is used to get the performance of the model.

4. Experiments

This section presents the findings of various experiments conducted on different machine learning models. The analysis begins with the results from the first and second trials (Tables 17 & 18). Subsequently, this section examines how varying the batch size impacts the outcomes in both trials (Tables 19 & 20).

Next, the influence of stemming on the BERT model's performance in trials one and two is explored (Tables 21 & 22). This section then assesses the effectiveness of Fasttext on the model's performance in both trials (Tables 23 & 24).

Table 25 summarizes the test results obtained using different word embedding types. Table 26 presents the outcomes of employing CNN, LSTM, and Bi-LSTM classifier methods while utilizing Fasttext Cc.Id.300.Bin and Wiki.Id.Bin word embeddings.

The following tables (Tables 27-31) showcase the results of trials conducted with various models, including Gemini-1.0-pro-001 and RoBERTa, alongside comparisons to fine-tuning approaches implemented with PyTorch, Gemma, and Gemini.

Finally, Table 22 provides a comprehensive comparison of F1 scores achieved across all the methods explored in this section.

The distribution of the dataset used in all these trials is 56% train data, 14% validation data, and 30% test data. The first trial was carried out on the first dataset by changing the dropout value to see how much change in the accuracy results was obtained. test results can be seen in Table 7.

In Table 17, it can be seen that the accuracy results are quite good. Then a trial was carried out on the second dataset with a change in the dropout value as well. test results can be seen in Table 8.

In Table 8 it can be seen that the accuracy results are quite good and there is a slight increase in accuracy compared to Table 7. Then a trial was carried out on the first dataset by changing the batch size value to see how big the change in the accuracy results was, the test results can be seen in table 9.

Table 17. BERT Dataset FIRST Trial Results

Method	Dropout	F1 For	F1 Observing	F1 Against	F1 Macro
Dense Layer	0.2	0.6765	0.8800	0.6423	0.7329
	0.3	0.6921	0.8846	0.6488	0.7418
	0.5	0.6706	0.8886	0.6322	0.7305
Bi-LSTM	0.2	0.6924	0.8995	0.6673	0.7531
	0.3	**0.7228**	0.8865	0.6945	**0.7746**
	0.5	0.6977	0.8917	0.6628	0.7507
CNN	0.2	0.6833	**0.9104**	**0.6950**	0.7629
	0.3	0.6758	0.8938	0.6498	0.7398
	0.5	0.6941	0.8959	0.6610	0.7503

Table 18. BERT Dataset SECOND Trial Results

Method	Dropout	F1 For	F1 Observing	F1 Against	F1 Macro
Dense Layer	0.2	0.6952	0.8626	0.6843	0.7474
	0.3	0.6951	0.8782	0.6595	0.7443
	0.5	0.7103	0.8784	0.6808	0.7565
Bi-LSTM	0.2	**0.7310**	0.8957	**0.7223**	**0.7830**
	0.3	0.7041	**0.9010**	0.7102	0.7718
	0.5	0.7150	0.8926	0.7137	0.7738
CNN	0.2	0.7125	0.8672	0.6863	0.7553
	0.3	0.7062	0.8926	0.6859	0.7616
	0.5	0.7107	0.8914	0.6902	0.7641

Table 19. BATCH SIZE BERT Dataset FIRST Trial Results

Method	BATCH SIZE	F1 FOR	F1 Observing	F1 Against	F1 Macro
Dense Layer	16	0.6978	0.8839	0.6389	0.7402
	32	0.6990	0.8909	0.6510	0.7470
	64	0.6893	0.9041	0.6680	0.7538
Bi-LSTM	16	0.6902	0.9057	**0.6862**	0.7607
	32	0.6911	0.9125	0.6636	0.7557
	64	**0.7153**	0.8981	0.6828	**0.7654**
CNN	16	0.6969	0.9112	0.6603	0.7561
	32	0.7004	0.8895	0.6709	0.7536
	64	0.6931	**0.9175**	0.6799	0.7635

In Table 19, it can be seen that the accuracy results obtained have a slight decrease compared to Table 7. Then a trial was carried out on the second dataset with changes in the batch size value as well, the test results can be seen in table 20.

Table 20. BATCH SIZE BERT Dataset SECOND Trial Results

Method	Batch Size	F1 For	F1 Observing	F1 Against	F1 Macro
Dense Layer	16	0.7040	0.8713	0.6911	0.7555
	32	0.7068	0.8963	0.6825	0.7619
	64	**0.7191**	0.8831	0.6701	0.7574
Bi-LSTM	16	0.6992	0.8993	0.6842	0.7609
	32	0.7109	**0.9012**	**0.7168**	**0.7763**
	64	0.7153	0.8910	0.7199	0.7754
CNN	16	0.7082	0.8964	0.7016	0.7687
	32	0.7133	0.8826	0.6964	0.7641
	64	0.7181	0.8908	0.7037	0.7709

Table 21. Stemming BERT Dataset FIRST Trial Results

Method	F1 FOR	F1 Observing	F1 Against	F1 Macro
Dense	0.6535	0.8353	0.6309	0.7066
Bi-LSTM	0.6400	0.8336	**0.6409**	0.7048
CNN	**0.6620**	**0.8524**	0.6535	**0.7226**

Table 22. Stemming BERT Dataset SECOND Trial Results

Method	F1 FOR	F1 Observing	F1 Against	F1 Macro
Dense	0.6348	0.7785	0.6265	0.6799
Bi-LSTM	0.6846	0.8355	0.6751	0.7317
CNN	0.6705	0.8362	0.6424	0.7164

In Table 20, it can be seen that the accuracy results obtained slightly decreased compared to Table 8. Then a trial was conducted on the first dataset by stemming with the literary python library on the claim text and the article title text, the test results can be seen in Table 11.

It can be seen in Table 21, there is a decrease in accuracy compared to Table 16 or Table 19 which does not stem the sentence. Then a trial was conducted on the second dataset by stemming the claim text and the article title text, the test results can be seen in Table 22.

It can be seen in Table 22, there is a decrease in accuracy compared to Table 18 or Table 20 which does not stem the sentence. Then a trial was carried

out using word embedding fastText to see a comparison of the accuracy obtained. In this trial, the Indonesian pre-trained cc.id.300.bin provided by fastText was used, the test results can be seen in Table 23.

As seen in Table 23, there is a significant decrease in accuracy compared to Table 17. Then a trial was conducted using the word embedding fastText on the second dataset, the test results can be seen in Table 24.

As seen in Table 24, there is an increase in accuracy when compared to Table 23. However, the comparison is still quite far when compared to Table 18. From all trials, it can be seen that the accuracy of F1 observing always has the highest value in all methods.

In Table 25 as described in the research paper 'Stance Classification Post Health in the Media Social With FastText Embedding and Deep Learning,' it can be seen that there is a difference in accuracy of 1.1% between the average F1 macro model with Word2Vec (52.7%) and the model with fastText (53.8%). The most accurate use of Word2Vec with LSTM only reached 55% and the most accurate use of fastText was obtained with CNN with F1 macro 55.4%.

As seen in Table 26, tested how well a classifier could perform using only basic summaries of sentences, instead of adding extra details. The CNN method achieved the best overall accuracy (with a score of 55.4%) when it used summaries from the 'cc.id.bin' dataset. Summaries from the 'wiki.id.bin' dataset also produced similar results.

Table 23. Fasttext BERT Dataset FIRST Trial Results

Method	F1 FOR	F1 Observing	F1 Against	F1 Macro
Dense	0.3134	0.5945	0.4575	0.4551
Bi-LSTM	0.4296	0.6436	0.5268	0.5333
CNN	0.4579	0.6323	0.5154	0.5352

Table 24. Fasttext BERT Dataset SECOND Trial Results

Method	F1 FOR	F1 Observing	F1 Against	F1 Macro
Dense	0.5327	0.6093	0.2712	0.4711
Bi-LSTM	0.5564	0.6928	0.4287	0.5593
CNN	0.5258	0.7217	0.4749	0.5741

Table 25. Test Results of Word Embedding Types

Approach	F1 FOR	F1 Observing	F1 Against	F1 Macro
Word2Vec + CNN	0.454	0.654	0.488	0.532
Word2Vec + LSTM	0.503	0.659	0.488	0.550
Word2Vec + BiLSTM	0.453	0.624	0.424	0.500
Average	**0.470**	**0.646**	**0.467**	**0.527**
fastText + CNN	**0.495**	**0.824**	0.345	**0.554**
fastText + LSTM	0.492	0.801	0.294	0.529
fastText + Bi-LSTM	0.441	0.817	**0.391**	0.530
Average	**0.476**	**0.814**	**0.343**	**0.538**

Table 26. Test Results Of CNN, LSTM, And Bi-LSTM Classifier Methods, With Fasttext Cc.Id.300.Bin And Wiki.Id.Bin

Classifier	F1 FOR	F1 Observing	F1 Against	F1 Macro
cc.id.300.bin				
CNN	**0.495**	**0.824**	0.345	**0.554**
LSTM	0.492	0.801	0.294	0.529
Bi-LSTM	0.441	0.817	**0.391**	0.530
wiki.id.bin				
CNN	**0.512**	0.807	0.345	**0.553**
LSTM	0.445	0.827	**0.350**	0.540
Bi-LSTM	0.496	**0.834**	0.315	0.547

Table 27 shows the results of a gemini-1.0-pro-001 model that sorts data into categories. It indicates how many items were classified correctly and breaks down performance for two different categories ("against" and "for"). The overall accuracy is 77%[35].

Table 28 evaluates the performance of a machine learning RoBERTa model on classification tasks. This information can be found on Hugging Face: https://huggingface.co/cahya/roberta-base-indonesian-522M. It analyzes the model's ability to categorize data points into predefined classes. For instance, it can classify sentiment as positive ("for") or negative ("against"). The included metrics, such as precision (37% for "against") and recall (78% for "observation"), assess how accurately the model assigns data points to the correct categories.

Table 27. Trials with gemini-1.0-pro-001

Class	Precision	Recall	F1 Score	Support
against	0.75	0.79	0.77	1116
for	0.84	0.53	0.65	1111
observing	0.71	0.93	0.81	1112
accuracy	-	-	0.75	3339
macro avg	0.77	0.75	0.74	3339
weighted avg	0.74	0.75	0.74	3339

Table 28. Trials with RoBERTa

Class	Precision	Recall	F1-Score	Support
against	0.37	0.46	0.41	181
for	0.55	0.48	0.51	257
observing	0.83	0.78	0.80	238
Accuracy	-	-	0.58	676
Macro Avg	0.58	0.57	0.58	676
Weighted Avg	0.60	0.58	0.59	676

Table 29. Comparison with Transfer Learning with Bert

Class	Precision	Recall	F1-Score	Support
against	0.63	0.56	0.59	225
for	0.63	0.69	0.66	226
observing	0.88	0.88	0.88	225
Accuracy	-	-	0.71	676
Macro Avg	0.71	0.71	0.71	676
Weighted Avg	0.71	0.71	0.71	676

Table 29 shows the effectiveness of a Transfer Learning with Bert on a classification task. It is based on Hugging Face: https://huggingface.co/cahya/bert-base-indonesian-522M. It leverages a pre-trained model and tailors it to a specific dataset of labeled examples, where each belongs to a predefined category. The table's metrics, like precision and recall, assess how accurately the model assigns these examples to their correct categories[38].

Table 30 evaluates the performance of a Gemma[36] model fine-tuned for a sentiment analysis task. With a precision of 85% for positive sentiment, the table indicates the model accurately identifies 85% of truly positive reviews. However, the model's recall for negative sentiment is 70%, meaning it misses 30% of negative reviews.

Table 31 evaluates the performance of a sentiment analysis model built with Gemini[37]. The model was fine-tuned, meaning it was adapted from a pre-trained model to classify reviews as positive or negative. For positive sentiment, the table shows a precision of 85%. This means out of every 100 reviews the model identifies as positive, 85 are truly positive. However, the recall for negative sentiment is 79%, indicating the model misses 21% of negative reviews.

Table 32 summarizes the F1 scores achieved by four different classification methods on a classification task. F1 score is a crucial metric for evaluating model performance, balancing both precision (correctly identifying positive cases) and recall (finding all actual positive cases).

Table 30. Comparison with Gemma finetuning

Class	Precision	Recall	F1-Score	Support
against	0.85	0.70	0.77	225
for	0.69	0.92	0.79	225
observing	0.96	0.81	0.88	226
Accuracy	-	-	0.81	676
Macro Avg	0.83	0.81	0.81	676
Weighted Avg	0.83	0.81	0.81	676

Table 31. Comparison with Gemini Fine Tuning

Class	Precision	Recall	F1-Score	Support
against	0.90	0.79	0.84	225
for	0.87	0.83	0.85	225
observing	0.79	0.93	0.86	226
Accuracy	-	-	0.85	676
Macro Avg	0.86	0.85	0.85	676
Weighted Avg	0.86	0.85	0.85	676

Table 32. Comparison of F1 Scores for Different Methods

Method	F1 FOR	F1 Observing	F1 Against	F1 Macro
Dense Layer	0.6893	0.9041	0.6680	0.7538
Bi-LSTM	**0.7153**	0.8981	0.6828	**0.7654**
CNN	0.6931	**0.9175**	0.6799	0.7635
Gemini	0.65	0.81	**0.77**	0.74
RoBERTa	0.51	0.80	0.41	0.58
BERT	0.66	0.88	0.59	0.71
Gemma Fine Tuning	0.79	0.88	0.77	0.81
Gemini Fine Tuning	0.85	0.86	0.84	0.85

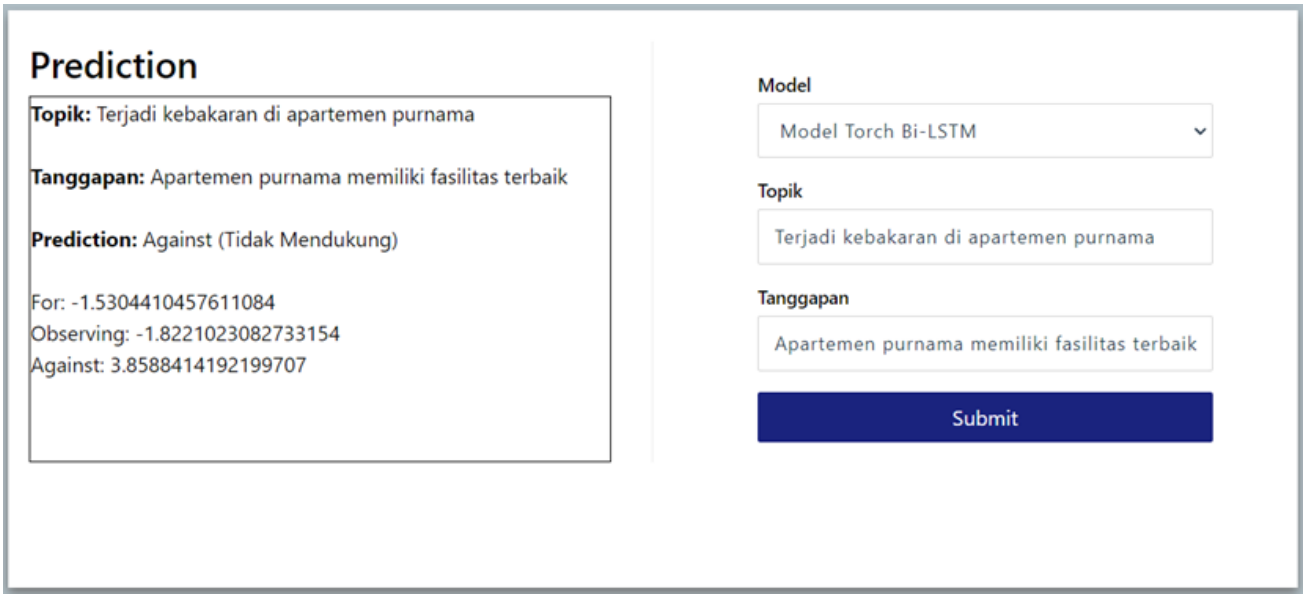

Figure. 9 Example of First Form Input and Output Results

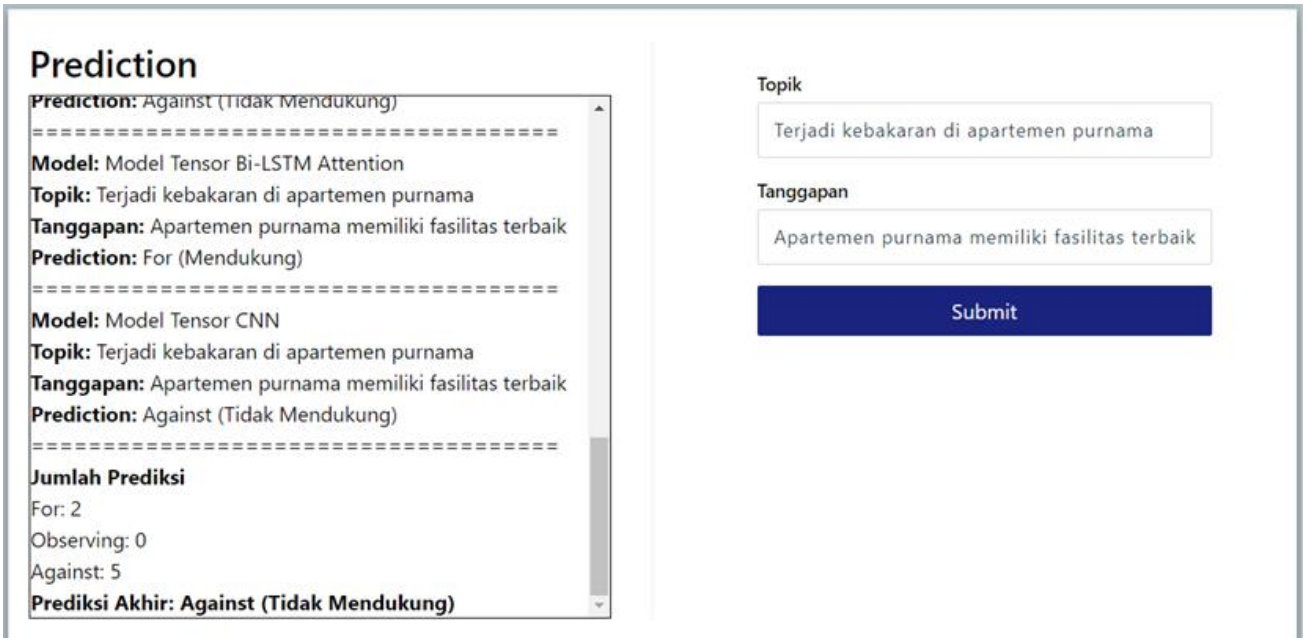

Figure. 10 Example of Application Form

Looking at the F1 scores for identifying positive cases, Bi-LSTM takes the lead at 0.7153, followed closely by Dense Layer at 0.6893. This indicates that Bi-LSTM might be slightly better at correctly classifying positive examples in this specific task. However, Dense Layer appears to excel in overall classification accuracy, as reflected by its F1 score of 0.7538 for observing all classifications.

In this study, we also developed an interface displaying the classification results for inference. The user could choose the model, claim topic, and related news and get the predictions as in Figure 9.

In Figure 10 you can see the appearance of the second form, on the left there is a text box for the output results from stance classification and on the right, there is text input for topics and responses. In this form, sentences will be processed by all existing models directly. Then the number is calculated and the highest number of predictions becomes the final prediction. The following is an example of input and output results from testing for stance classification with the second form.

In Figure 11 you can see an example of the input and output results from the third form. In this example, predictions are made by inputting the URL

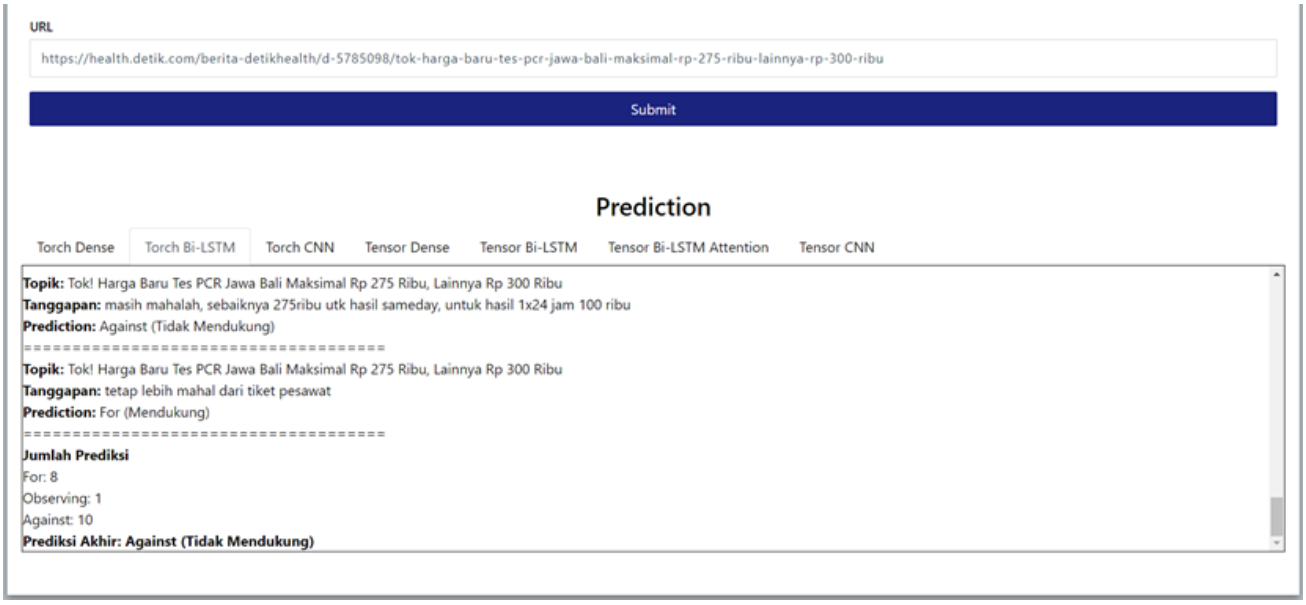

Figure. 11 Example of Third Form Input and Output Results

from "health.detik.com" regarding health news. Then predictions are made on the title of the news obtained using the comments on the news. In the output results section there are tabs for each model, so you can see the prediction results obtained by the model regarding the comments it has on this news. Then, from the number of existing classifications, the highest number of classifications is calculated to be used as the final prediction result. In this case, it can be said that it is not about false news or a hoax, but it can be said that many people do not support or disagree with the price change to the "PCR price".

5. Conclusion

This section will discuss some finding from this study. The proposed models by utilizing contextual embedding like BERT can enhance the model performance, achieving highest F1 Score of 77% compared to the previous studies as the baseline and on par with performance of publicly available LLMs such as Gemini and Gemma. Some of the problems rise in this study shows that shortcuts or encountering typos can cause problems for the model by not achieving the best performance. Furthermore, optimization such as stemming did not worked and cause the model failing to understand the contextual sentence meaning.

Another experiment shows that negation in our experiment seems fails to generalize and classify by the models. It shows that the model performance was decreasing due to negative words in the dataset. Although our model incorporates transformer, it seems that our proposed models still struggled with long word sequences in the input.

However, from several experiments and performance comparison based on this study, BERT performed better than baseline methods for classify the stance in the experiments of this study. To improve future works, we recommend training a BERT model and use another model representation to address some multimodality data in stance classification by incorporating a larger dataset to train the model.

Notations

i_t	input gate
W_i	weight of the input gate
x_t	input value in iteration t
U_i	weight of the input gate
h_t	h is the output data value in the previous iteration t
b_i	bias of the input gate
$W_{\tilde{C}}$	weight of the cell state candidate
$U_{\tilde{C}}$	weight of the cell state candidate
$b_{\tilde{C}}$	bias of the cell state candidate
f_t	forget gate value in iteration t
W_f	weight of the forget gate
U_f	weight of the forget gate
b_f	bias of the forget gate
C_t	cell state
$\tilde{C}_t$	cell state candidate
o_t	gate output result
W_o	weights of the output gate
U_o	weights of the output gate

b_o	bias of the output gate
C_t	cell state
TP	true positive
FP	false positive
FN	false negative

Conflicts of Interest

The authors declare no conflict of interest.

Author Contributions

Conceptualization, Esther I. Setiawan, Willyanto Dharmawan; methodology, Esther I. Setiawan, Willyanto Dharmawan, Kimiya Fujisawa, Mauridhi Hery Purnomo; software, Willyanto Dharmawan and Kevin Jonathan; validation, Willyanto Dharmawan, Joan Santoso; formal analysis, Esther I. Setiawan and Kimiya Fujisawa; writing—original draft preparation, Esther I. Setiawan, Willyanto Dharmawan; writing—review and editing, Esther I. Setiawan, Joan Santoso, FX Ferdinandus and Kimiya Fujisawa ; visualization, Willyanto Dharmawan and Joan Santoso; supervision, Esther I. Setiawan and Kimiya Fujisawa; project administration, Esther I. Setiawan and Kimiya Fujisawa; funding acquisition, Esther I. Setiawan, FX. Ferdinandus, and Mauridhi Hery Purnomo.

Acknowledgments

This work was partially funded by Institut Sains dan Teknologi Terpadu Surabaya (ISTTS) under Institute for Research and Community Services or Lembaga Penelitian dan Pengabdian kepada Masyarakat (LPPM).

References

[1] N. Nic, R. Fletcher, A. Kalogeropoulos, D. A. L. Levy, and R. K. Nielsen, "Reuters institute digital news report 2018", *Reuters Institute for the Study of Journalism*, Vol. 39, 2018.

[2] H. Karande, R. Walambe, V. Benjamin, K. Kotecha, and T. S. Raghu, "Stance detection with BERT embeddings for credibility analysis of information on social media", *PeerJ Comput Sci*, Vol. 7, p. e467, 2021.

[3] E. I. Setiawan, H. Juwiantho, J. Santoso, S. Sumpeno, K. Fujisawa, and M. H. Purnomo, "Multiview Sentiment Analysis with Image-Text-Concept Features of Indonesian Social Media Posts", *International Journal of Intelligent Engineering & Systems*, Vol. 14, No. 2, 2021, doi: 10.22266/ijies2021.0430.47.

[4] B. Xu, M. Mohtarami, and J. Glass, "Adversarial domain adaptation for stance detection", *arXiv preprint arXiv:1902.02401*, 2019.

[5] I. Khan, S. K. Naqvi, M. Alam, and S. N. A. Rizvi, "An efficient framework for real-time tweet classification", *International Journal of Information Technology*, Vol. 9, pp. 215–221, 2017.

[6] Y.-C. Chen, Z.-Y. Liu, and H.-Y. Kao, "Ikm at semeval-2017 task 8: Convolutional neural networks for stance detection and rumor verification", In: *Proc. of the 11th international workshop on semantic evaluation (SemEval-2017*, pp. 465–469*)*, 2017.

[7] S. Imron, E. I. Setiawan, J. Santoso, M. H. Purnomo, and others, "Aspect Based Sentiment Analysis Marketplace Product Reviews Using BERT, LSTM, and CNN", *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, Vol. 7, No. 3, pp. 586–591, 2023.

[8] E. I. Setiawan, F. Ferry, J. Santoso, S. Sumpeno, K. Fujisawa, and M. H. Purnomo, "Bidirectional GRU for Targeted Aspect-Based Sentiment Analysis Based on Character-Enhanced Token-Embedding and Multi-Level Attention.", *International Journal of Intelligent Engineering & Systems*, Vol. 13, No. 5, 2020, doi: 10.22266/ijies2020.1031.35.

[9] N. Kotonya and F. Toni, "Gradual argumentation evaluation for stance aggregation in automated fake news detection", In: *Proc. of the 6th Workshop on Argument Mining*, pp. 156–166, 2019.

[10] M. Alsafad, "Stance Classification for Fake News Detection with Machine Learning", *The Eurasia Proceedings of Science Technology Engineering and Mathematics*, Vol. 22, pp. 191–198, 2023.

[11] J. Du, R. Xu, Y. He, and L. Gui, "Stance classification with target-specific neural attention networks", In: *Proc. of 26th International Joint Conference on Artificial Intelligence*, pp. 3988-3994, 2017.

[12] R. Bar-Haim, I. Bhattacharya, F. Dinuzzo, A. Saha, and N. Slonim, "Stance classification of context-dependent claims", In: *Proc. of the 15th*

Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers, pp. 251–261, 2017.

[13] E. Kochkina, M. Liakata, and I. Augenstein, "Turing at semeval-2017 task 8: Sequential approach to rumour stance classification with branch-lstm", *arXiv preprint arXiv:1704.07221*, 2017.

[14] E. Irawati Setiawan *et al.*, "Indonesian Stance Analysis of Healthcare News using Sentence Embedding Based on LSTM", *Jurnal Nasional Teknik Elektro dan Teknologi Informasi*, Vol. 9, No. 1, pp. 8–17, 2020, doi: 10.22146/jnteti.v9i1.115.

[15] T. Gao, X. Yao, and D. Chen, "Simcse: Simple contrastive learning of sentence embeddings", *arXiv preprint arXiv:2104.08821*, 2021.

[16] Z. Wang and B. Yang, "Attention-based bidirectional long short-term memory networks for relation classification using knowledge distillation from BERT", In: *Proc. of 2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, pp. 562–568, 2020.

[17] L. Derczynski, K. Bontcheva, M. Liakata, R. Procter, G. W. S. Hoi, and A. Zubiaga, "SemEval-2017 Task 8: RumourEval: Determining rumour veracity and support for rumours", *arXiv preprint arXiv:1704.05972*, 2017.

[18] G. Gorrell *et al.*, "SemEval-2019 Task 7: RumourEval 2019: Determining Rumour Veracity and Support for Rumours", In: *Proc. of the 13th International Workshop on Semantic Evaluation: NAACL HLT 2019*, pp. 845–854, 2019.

[19] Q. Li, Q. Zhang, and L. Si, "eventAI at SemEval-2019 task 7: Rumor detection on social media by exploiting content, user credibility and propagation information", In: *Proc. of the 13th international workshop on semantic evaluation*, 2019, pp. 855–859.

[20] S. Peshterliev, A. Hsieh, and I. Kiss, "F10-SGD: Fast Training of Elastic-net Linear Models for Text Classification and Named-entity Recognition", *arXiv preprint arXiv:1902.10649*, 2019.

[21] K. Dey, R. Shrivastava, and S. Kaushik, "Twitter stance detection—A subjectivity and sentiment polarity inspired two-phase approach", In: *Proc. of 2017 IEEE international conference on data mining workshops (ICDMW)*, pp. 365–372, 2017.

[22] D. Küçük and F. Can, "A tweet dataset annotated for named entity recognition and stance detection", *arXiv preprint arXiv:1901.04787*, 2019.

[23] P. Heinrich, "Stance Classification in Argument Search", *M.S. thesis, Dept. Computer Science, Univ. Paderborn*, 2019.

[24] S. Ravichandiran, *Getting Started with Google BERT: Build and train state-of-the-art natural language processing models using BERT*, Packt Publishing Ltd, 2021.

[25] Y. Liu *et al.*, "Roberta: A robustly optimized bert pretraining approach", *arXiv preprint arXiv:1907.11692*, 2019.

[26] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding", *arXiv preprint arXiv:1810.04805*, 2018.

[27] H. Inoue, "Multi-sample dropout for accelerated training and better generalization", *arXiv preprint arXiv:1905.09788*, 2019.

[28] C. B. P. Putra, D. Purwitasari, and A. B. Raharjo, "Stance Detection on Tweets with Multi-task Aspect-based Sentiment: A Case Study of COVID-19 Vaccination.", *International Journal of Intelligent Engineering & Systems*, Vol. 15, No. 5, 2022, doi: 10.22266/ijies2022.1031.45.

[29] R. Dale, "NLP in a post-truth world", *Nat Lang Eng*, Vol. 23, No. 2, pp. 319–324, 2017.

[30] Y. Cui, W. Che, T. Liu, B. Qin, S. Wang, and G. Hu, "Revisiting pre-trained models for Chinese natural language processing", *arXiv preprint arXiv:2004.13922*, 2020.

[31] L. Xu, X. Zhang, and Q. Dong, "CLUECorpus2020: A large-scale Chinese corpus for pre-training language model", *arXiv preprint arXiv:2003.01355*, 2020.

[32] P. J. Gorinski *et al.*, "Named entity recognition for electronic health records: a comparison of rule-based and machine learning approaches", *arXiv preprint arXiv:1903.03985*, 2019.

[33] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks", *arXiv preprint arXiv:1706.06083*, 2017.

[34] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection", In: *Proc. of the IEEE international conference on computer vision*, pp. 2980–2988, 2017.

[35] G. Team *et al.*, "Gemini: a family of highly capable multimodal models", *arXiv preprint arXiv:2312.11805*, 2023.

[36] G. Team *et al.*, "Gemma: Open models based on gemini research and technology", *arXiv preprint arXiv:2403.08295*, 2024.

[37] Y. Chang *et al.*, "A survey on evaluation of large language models", *ACM Trans Intell Syst Technol*, Vol. 15, No. 3, pp. 1–45, 2024.

[38] F. Zhuang *et al.*, "A comprehensive survey on transfer learning", *Proceedings of the IEEE*, Vol. 109, No. 1, pp. 43–76, 2020.