Abstractive Text Summarization of Indonesian News Articles Using Long Short-Term Memory (LSTM)

I Gede Surya Mahardika[a1], Gusti Made Arya Sasmita[a2], I Nyoman Piarsa[b3]
[a]Department of Information Technology, Faculty of Engineering, Udayana University
Bukit Jimbaran, Bali, Indonesia-8036110
e-mail: [1]suryamahardika@student.unud.ac.id, [2]aryasasmita@unud.ac.id, [3]manpits@unud.ac.id

Abstrak

Era digital ditandai dengan jumlah artikel berita yang tersedia secara daring semakin meningkat, menyebabkan permasalahan informasi berlebih (information overload) bagi pembaca. Untuk mengatasi hal ini, penelitian ini mengembangkan sistem abstractive text summarization pada artikel berita bahasa Indonesia bermetode Long Short-Term Memory (LSTM) melalui tambahan word embedding FastText dan Attention Layer. Dataset yang digunakan berjumlah 105.588 data artikel berita, yang dikumpulkan melalui proses web scraping dari situs berita Detik.com. Model dikembangkan dalam arsitektur sequence-to-sequence (Seq2Seq) dan diuji dalam empat variasi, yaitu: LSTM Seq2Seq dasar (tanpa tambahan), LSTM dengan FastText embedding, LSTM dengan Attention Layer, dan LSTM dengan kombinasi FastText dan Attention Layer. Model terbaik yaitu, model LSTM dengan Attention Layer menggunakan distribusi data 80:20 dengan akurasi ROUGE-1 sebesar 0.5207, ROUGE-2 sebesar 0.4000 dan ROUGE-L sebesar 0.4970. Hasil ini menunjukkan bahwa model LSTM dengan Attention Layer memberikan performa yang lebih baik dalam menghasilkan ringkasan. Dengan demikian, penelitian ini memberikan kontribusi dalam pengembangan sistem abstractive text summarization berbahasa Indonesia.

Kata kunci: *Abstractive Text Summarization, LSTM, FastText, Attention Layer, ROUGE*

Abstract

The digital era is characterized by an increasing number of news articles available online, causing information overload problems for readers. To overcome this problem, this research develops an abstractive text summarization system on Indonesian news articles with the Long Short-Term Memory (LSTM) method with additional FastText word embedding and Attention Layer. The dataset used amounted to 105,588 news article data, which was collected through a web scraping process from the Detik.com news site. The model was developed in sequence-to-sequence architecture (Seq2Seq) and tested in four variations, namely: Basic Seq2Seq LSTM (no additions), LSTM with FastText embedding, LSTM with Attention Layer, and LSTM with FastText and Attention Layer combination. The best model is the LSTM model with Attention Layer using 80:20 data distribution with ROUGE-1 accuracy of 0.5207, ROUGE-2 of 0.4000 and ROUGE-L of 0.4970. The results show that the LSTM model with Attention Layer provides better performance in generating summaries. Thus, this research contributes to the development of abstractive text summarization system in Indonesian language.

Keywords : *Abstractive Text Summarization, LSTM, FastText, Attention Layer, ROUGE*

1. Introduction

Advancements in information technology has driven the rapid growth of digital content, including online news articles that are published daily in huge numbers. In Indonesia, with more than 200 million internet users, news sites such as Detik.com publish thousands of articles per day. This creates the phenomenon of information overload, where readers struggle to digest information quickly and efficiently[1]. As a result, users tend to need a concise and informative summary to understand the essence of a news story without having to read the entire text[2]. An approach that can be used to overcome this problem is text summarization, especially the abstractive summarization model. Unlike extractive summarization which only takes parts of

sentences from the original text, abstractive summarization produces new summaries with paraphrases and sentence structures that are more natural, resembling the way humans summarize[3]. This method, although more complex, is capable of producing more cohesive and meaningful summaries[4].

Most of the research in this field is still dominated by English, while system development for Indonesian is still relatively limited, both in terms of datasets and exploration of methods used[5]. This limited linguistic resource magnifies the challenge of building a reliable summarization system for Indonesian. Most previous research also focuses on extractive methods, which tend to produce less natural summaries[6]. This research developed an abstractive text summarization system for Indonesian news articles is developed using Sequence-to-Sequence (Seq2Seq) architecture according to LSTM. To improve the model performance, the integration of FastText word embedding-which is able to better handle unknown words [7]. The model was tested in four configuration variations, and performance evaluation was conducted using the ROUGE metric, which is one of the evaluation standards in automatic summarization[9]. The research will contribute to the development of automatic text summarization technology in Indonesian, and become a practical solution in helping users access information more efficiently and relevantly in the midst of the rampant flow of digital information.

2. Research Method

The method used in abstractive text summarization on Indonesian news articles is data collection through scraping the Detik.com website which is stored in CSV format, followed by pre-processing stages such as text cleaning, lowercasing, tokenizing, padding sequence, and dividing data into training and validation data. After that, model training and testing are conducted, and the results are evaluated using the ROUGE metric.

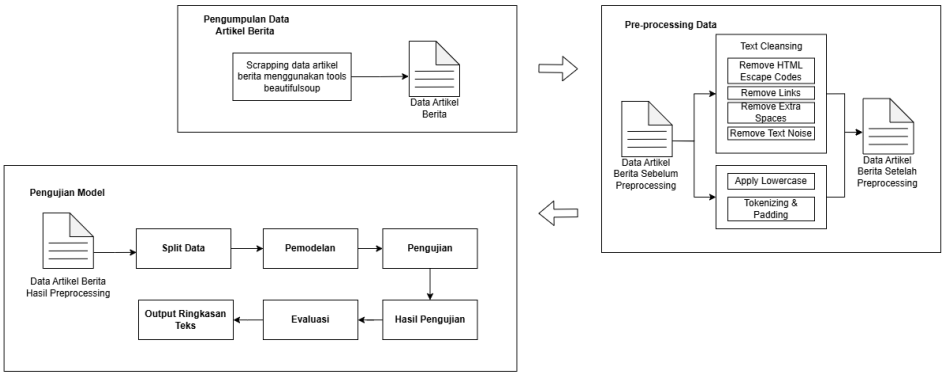

Figure 1. Overview of Research Methods

2.1. Data Collection

The data is Indonesian news article data. News data is obtained from the detik.com news website through a data collection method, namely a special python library for scrapping websites called Beautifulsoup. The news articles collected are based on the index page of detik.com, which is all news such as politics, law, sports, and others. The elements taken in the data scrapping process are “title”, “highlight”, “date”, ‘url’ and “content”.

2.2. Data Pre-processing

The pre-processing phase refines and prepares the data for use through various cleaning and transformation steps, namely text cleaning, apply lowercase, remove column date and url, and tokenizing.

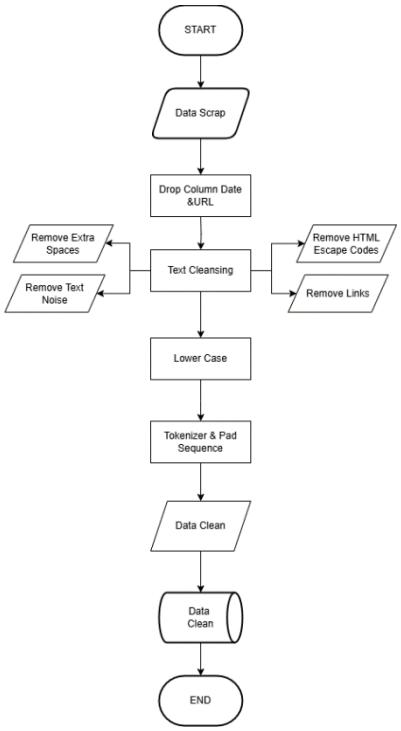

Figure 2. Pre-processing Data

The stages of data preprocess are as follows:

1. Drop Column
   The drop column stage aims to remove columns that are not needed in abstractive text summarization research, such as the Date and URL columns.
2. Text Cleansing
   The text cleaning stage is a stage to clean and remove unnecessary parts in the text. In abstractive text summarization research, the text cleansing stage is used to remove HTML escape codes, links, extra spaces and text noises.
3. Lowercase
   The lowercase stage aims to convert all capital letters in the text into lowercase letters. This step is important to maintain data consistency, avoid duplication of words that are actually the same but differ in uppercase and lowercase writing, and improve accuracy in the tokenization process and model training.
4. Split Data
   The split data stage is to divide the dataset into several subsets. In abstractive text summarization research, the data is split with a ratio of 80:20.
5. Tokenizing
   Performed to convert text into numeric tokens. The tokenizer is prepared and trained using article data and summaries that have been cleaned in abstractive text summarization research. The tokenizing process in abstractive text summarization research is carried out using the TensorFlow library.

2.3. Model Training and Testing

Model training and testing is a process to determine which model or algorithm has better results and accuracy. In this abstractive text summarization research, the LSTM algorithm is used with four architectural variations. The first architecture uses the basic encoder-decoder model based on LSTM. The second architecture combines LSTM using FastText word represent and Attention Layer. The third architecture LSTM with FastText word represent only, while the fourth architecture uses LSTM with an Attention Layer only.

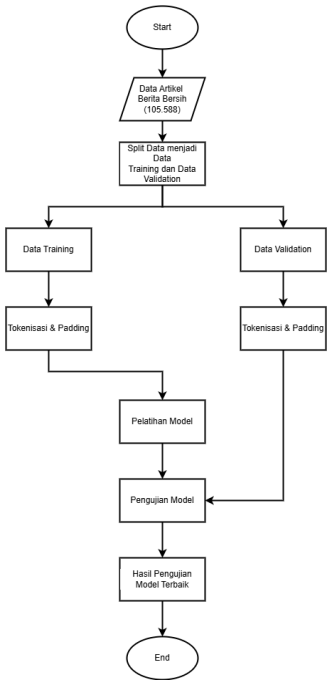

Figure 3. Model Training and Testing

The testing flow of the LSTM model above. A total of 105,588 cleaned news articles are for the training and validation data with an 80:20 scenario with a total of 84,470 training data and 21,118 validation data. Both data go through a tokenization process, with additional padding in the training and validation data. The model subsequently trained on the dataset and tested using four architecture variations: (1) Basic LSTM Seq2Seq, (2) LSTM + FastText + Attention, (3) LSTM + FastText, and (4) LSTM + Attention. Performance evaluation is performed using the ROUGE metric to determine the best model.

3. Literature Study

Research report writing theories are examined in the literature evaluation.

3.1. Abstractive Text Summarization

The natural language processing (NLP) technique in generating summaries through entire new sentence that are not directly taken from the original text[4]. According to Nalapati et al (2016), abstractive summarization is understanding the context and meaning of the text as a whole, then rearranging the information with new words that are more concise and clear. Abstractive summarization is different from extractive summarization, which only selects sentences or phrases that already exist.

3.2. Long Short-Term Memory (LSTM)

LSTM is an artificial neural network architecture specifically designed to overcome vanishing and exploding gradient problems that often arise in conventional RNN models[10]. In the Seq2Seq models, LSTM plays an important role as a key component that enables the model to process and generate sequential data, like in translation machine, text summarization, and dialog generation tasks[11].

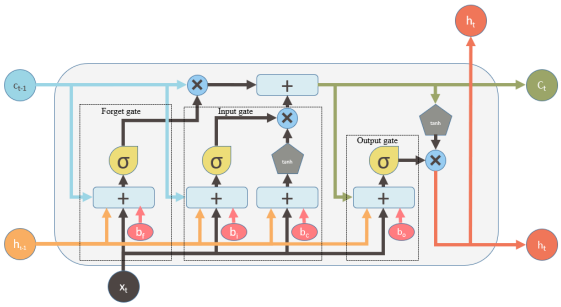

Figure 4. Arsitektur LSTM

The encoder part, the input layer receives sequential data with a predefined maximum length. Each token in this sequence is converted into a fixed-dimensional vector using the embedding layer. This embedding serves to reduce sparseness in the word representation and map words into a more meaningful low-dimensional space, with a vocabulary size of vocab_size + 3 that includes special tokens such as <unk>, <start>, and <end>. After embedding, this sequential data is processed by the LSTM layer which is configured to return the final state (hidden state and cell state) using the parameter return_state=True. The LSTM encoder produces outputs in the form of hidden vectors (h_t) and cell states (C_t), which are then used for decoder initialization. Mathematically, this process is described by the formula:

$$
\begin{aligned}
f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\
i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\
\tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \\
C_t &= f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \\
o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\
h_t &= o_t \cdot \tanh(C_t)
\end{aligned}
\tag{1}
$$

Description:

f_t : forget gate.

i_t : input gate.

$\tilde{C}_t$: candidate cell state.

C_t : new cell state.

o_t : output gate.

h_t : hidden state.

f_t is the forget gate decides how much of the previous cell state should be discarded. i_t is to controls the amount of new information to be added to the cell state, and o_t decides the quantity to create the current hidden state. All these mechanisms allow the LSTM to retain important information in the long run, while ignoring irrelevant information.

In the decoder section, a similar process is applied, but with some important differences. The decoder receives the target sequences to be embedded using an embedding layer with the same configuration as the encoder. The LSTM decoder layer is configured to return a complete sequence from each timestep, which allows the model to generate output

incrementally. The hidden state and cell state generated from the encoder are used as the initialization of the decoder through the parameter initial_state=encoder_states. This ensures that the decoder has a context that matches the input processed using encoder. After going through the LSTM layer, the output of the decoder is passed to the Dense layer which uses a softmax activation function in generating the probability distribution in all tokens in the vocabulary. This softmax function can be formulated as follows:

$$P(y_t \mid x) = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}} \quad (2)$$

z_i is the logit or score of the i-th token, and N is the total of tokens in the vocabulary. This probability distribution allows the model to predict the next token in the sequence according the previous token, thus forming the output sequence iteratively. This model is defined using the tf.keras function. A model that combines the encoder and decoder in one end-to-end architecture. The compilation process uses Adam's optimizer because of its ability to adaptively adjust the learning rate and accelerate model convergence. The sparse categorical cross-entropy loss function is formulated as:

$$L = -\frac{1}{N}\sum_{i=1}^{N} \log P(y_i \mid x_i) \quad (3)$$

y_i is the correct label for the i-th data, and $P(y_i \mid x_i)$ is the probability predicted by the model for that label. Using this architecture, the LSTM-based Seq2Seq model is able to handle various natural language processing tasks more effectively and produce output that is cohesive and appropriate to the input context.

3.3. FastText

FastText, developed by Facebook AI as part of Word2Vec, is an efficient word embedding technique for learning word representations and performing text classification tasks [12]. Instead of treating words as indivisible units, FastText breaks them down into sub-word n-grams. For example, the word 'apple' is represented by n-grams like 'ap', 'pp', and 'le', each with its own vector. The word's final embedding is the sum of the vectors of its constituent n-grams.

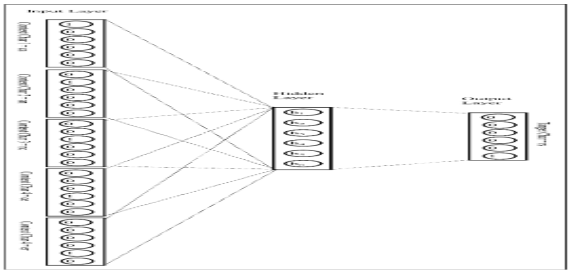

Figure 5. FastText Architecture

Source: Kadhim, 2024

It helps to capture the shorter words meaning and enables the embedding to better understand the prefixes and suffixes of words [7]. It allows FastText to capture morphological and semantic information by considering sub-word units, thereby handling out-of-vocabulary words and improving performance on various natural language processing tasks.

3.4. Attention Mechanism

The attention mechanism is a technique that enables models to prioritize relevant information while ignoring less important data. Originally developed to address challenges in neural machine translation, particularly for handling long sequences. [13]. It expanded to fields like image processing, speech recognition, and beyond. In NLP, attention is widely used for not only machine translation but also tasks such as text classification, generation, and answering

questions [14]. Sutskever et al. [15] and Cho et al. [16] Where the encoder processes input sequences of varying lengths, converting them into a consistent vector representation to facilitate the translation process, then the decoder generates a new sequence from the vector.

The encoder-decoder model struggles to capture all critical information from the source sentence, particularly in longer sentences. Cho et al. [16]The model's performance declines as sentence length increases. Bahdanau et al. [13]The limitation is that it incorporates an attention mechanism, which enables the model to move beyond the use of a fixed-dimensional vector for input sentences. In this enhanced framework, the decoder can refer to the entire input sequence at each step of decoding, prioritizing keywords to generate the next word in the target sequence, thus resulting in improved translation accuracy.

3.5. ROUGE

ROUGE is an evaluation method used to measure the quality of automated text summarization systems by comparing the results with human-made reference summaries (Ammar & Suyanto, 2020). In this research, ROUGE-1, ROUGE-2, and ROUGE-L are used, which evaluate performance based on Precision, Recall, and F1-Score. ROUGE-N specifically measures the n-gram similarity between the automatic summary and the reference.

$$ROUGE - N = \frac{\sum_{S\epsilon\{\text{ Reference Summaries }\}} \sum_{\text{gram }_n \epsilon S} \text{Count}_{\text{match}} \left(\text{gram}_n\right)}{\sum_{S\epsilon\{\text{ Reference Summaries}\}} \sum_{\text{gram }_n \epsilon S} \text{Count}\left(\text{gram}_n\right)} \quad (4)$$

ROUGE-L metric determines similarity by computing the Longest Common Subsequence (LCS), which looks at the longest matching word sequence in both summaries.

$$\begin{aligned} R_{lcs} &= \frac{LCS(X,Y)}{m} \\ P_{lcs} &= \frac{LCS(X,Y)}{n} \\ F_{lcs} &= \frac{(1+\beta^2)R_{lcs}P_{lcs}}{R_{lcs}+\beta^2 R_{lcs}} \end{aligned} \quad (5)$$

Formula 5 is used to calculate the ROUGE-L value, where LCS represents the length of the longest word sequence in common among the model text and the reference. Precision in ROUGE-L identifies the proportion of word sequences in the model result that match the main sequence in the reference, while Recall measures the extent to which word sequences in the reference are successfully represented by the model result.

4. Result and Discussion

The results of the research conducted by the author based on the research overview process are presented as follows.

4.1. Data Collection

The data needed in the research conducted is news article data on the Detik.com website. The data collected was 105,588 news articles. In addition, researchers have manually validated 10,000 news article data to ensure the quality of the dataset before being used in training and testing the model.

4.2. Data Pre-processing

The stage produces clean data. News article data has passed the stages in pre-processing such as text cleansing, lowercase and tokenizing. .

Table 1. *Pre-processing Data Results*

Sebelum: Satu unit rumah di Tegal Parang, Jakarta Selatan (Jaksel), terbakar. Pemadam kebakaran (Damkar) mengerahkan 19 unit mobil pemadam ke lokasi kejadian. Kebakaran itu dilaporkan terjadi pada Kamis (3/10/2024) pukul 17.58 WIB. Dinas Penanggulangan Kebakaran dan Penyelamatan (Gulkarmat) menyebutkan objek terbakar merupakan rumah tinggal. "Pengerahan Unit atau personel 19 unit, 60 personel," ujar Kadis Gulkarmat DKI Jakarta Satriadi Gunawan. Proses pemadaman sudah selesai sekitar 18.30 WIB. Petugas kemudian mendinginkan agar tidak terdapat api menyala. Satriadi belum menjelaskan apa penyebab kebakaran tersebut. Tak ada informasi korban akibat kebakaran itu.
Sesudah: satu unit rumah di tegal parang, jakarta selatan (jaksel), terbakar. pemadam kebakaran (damkar) mengerahkan 19 unit mobil pemadam ke lokasi kejadian. kebakaran itu dilaporkan terjadi pada kamis (3/10/2024) pukul 17.58 wib. Gulkarmat menyebutkan objek terbakar merupakan rumah tinggal. "pengerahan unit atau personel 19 unit, 60 personel," ujar kadis gulkarmat dki jakarta satriadi gunawan. proses pemadaman sudah selesai sekitar 18.30 wib. petugas kemudian mendinginkan agar tidak terdapat api menyala. satriadi belum menjelaskan apa penyebab kebakaran tersebut. tak ada informasi korban akibat kebakaran itu.

News article data starts from text data that still has noise, irrelevant sentences, after going through the text cleaning, lowercase, and tokenizing stages, the news article data becomes clean data and ready to used in training and testing stages of the model.

4.3. Model Training and Testing

Abstract text summarization research uses four models, namely LSTM Seq2Seq Basic, LSTM with FastText and Attention Layer, LSTM using FastText and LSTM with Attention Layer with each model using an 80:20 scenario to compare test results. The comparison of the training and testing results of the four abstractive text summarization models showed below.

Table 2. Comparison of Model Training and Testing Results

Model	**Skenario**	**ROUGE-1**	**ROUGE-2**	**ROUGE-L**
LSTM Seq2Seq Basic	80:20	0.2026	0.0738	0.1836
LSTM with *FastText* and *Attention Layer*	80:20	0.5101	0.3890	0.4869
LSTM with *FastText*	80:20	0.2052	0.0726	0.1836
LSTM with *Attention Layer*	80:20	0.5207	0.4000	0.4970

The comparison of four models in abstractive text summarization research. According to the ROUGE evaluation, the LSTM model using the Attention Layer provides the best

performance with a ROUGE-1 score is 0.5207, a ROUGE-2 score is 0.4000, and a ROUGE-L score of 0.4970, indicating a high degree of similarity of sentence structure with the reference summary. Meanwhile, the use of FastText embedding without Attention resulted in lower ROUGE scores. Overall, the addition of the Attention Layer was shown to contribute significantly to the improvement of summary quality, while the integration of FastText provided additional benefits, although not as great as Attention.

5. Conclusion

As a result, the application of the LSTM Seq2Seq method is proven to be able to perform abstractive text summarization on Indonesian news articles, although the summary quality to be improved. Experiments on four model variants show that the Attention Layer addition significantly develope the quality of the summary in terms of syntax and semantics. All stages, from data collection through scraping, preprocessing, training, to model testing were successfully implemented, proving that the end-to-end approach in this research is feasible as the basis for the automatic summarization development system in Indonesian. The best model is the LSTM model with Attention Layer using 80:20 data distribution with ROUGE-1 accuracy of 0.5207, ROUGE-2 of 0.4000 and ROUGE-L of 0.4970. Although the performance based on ROUGE evaluation is not optimal, the results of this study still show great potential for further development with the support of larger data and more complex models.

References

[1] E. R. Rahmi, E. Yumami, and N. Hidayasari, “Analisis Metode Pengembangan Sistem Informasi Berbasis Website: Systematic Literature Review,” *Remik*, vol. 7, no. 1, pp. 821–834, 2023, doi: 10.33395/remik.v7i1.12177.

[2] H. Shakil, A. Farooq, and J. Kalita, “Abstractive text summarization: State of the art, challenges, and improvements,” *Neurocomputing*, vol. 603, 2024, doi: 10.1016/j.neucom.2024.128255.

[3] R. Adelia, S. Suyanto, and U. N. Wisesty, “Indonesian abstractive text summarization using bidirectional gated recurrent unit,” *Procedia Comput. Sci.*, vol. 157, pp. 581–588, 2019, doi: 10.1016/j.procs.2019.09.017.

[4] O. B. Mercan, S. N. Cavsak, A. Deliahmetoglu, and S. Tanberk, “Abstractive Text Summarization for Resumes With Cutting Edge NLP Transformers and LSTM,” 2023. doi: 10.1109/ASYU58738.2023.10296563.

[5] A. Bahari and K. E. Dewi, “Peringkasan Teks Otomatis Abstraktif Menggunakan Transformer Pada Teks Bahasa Indonesia,” *Komputa J. Ilm. Komput. dan Inform.*, vol. 13, no. 1, pp. 83–91, 2024, doi: 10.34010/komputa.v13i1.11197.

[6] I. M. Karo Karo, S. Dewi, and A. Perdana, “Implementasi Text Summarization Pada Review Aplikasi Digital Library System Menggunakan Metode Maximum Marginal Relevance,” *JEKIN - J. Tek. Inform.*, vol. 4, no. 1, pp. 25–31, 2024, doi: 10.58794/jekin.v4i1.671.

[7] A. Nurdin, B. Anggo Seno Aji, A. Bustamin, and Z. Abidin, “Perbandingan Kinerja Word Embedding Word2Vec, Glove, Dan Fasttext Pada Klasifikasi Teks,” *J. Tekno Kompak*, vol. 14, no. 2, p. 74, 2020, doi: 10.33365/jtk.v14i2.732.

[8] M. Varaprasad Rao, K. Chakma, A. Jamatia, and D. Rudrapal, “An innovative Telugu text summarization framework using the pointer network and optimized attention layer,” *Multimed. Tools Appl.*, pp. 84539–84564, 2024, doi: 10.1007/s11042-024-19187-8.

[9] A. N. Ammar and S. Suyanto, “Peringkasan Teks Ekstraktif Menggunakan Binary Firefly Algorithm,” *Indones. J. Comput.*, vol. 5, no. 2, pp. 31–42, 2020, doi: 10.21108/indojc.2020.5.2.440.

[10] R. Gangundi and R. Sridhar, “IWM-LSTM encoder for abstractive text summarization,” *Multimed. Tools Appl.*, 2024, doi: 10.1007/s11042-024-19091-1.

[11] T. Shi, Y. Keneshloo, N. Ramakrishnan, and C. K. Reddy, “Neural Abstractive Text Summarization with Sequence-to-Sequence Models,” *ACM/IMS Trans. Data Sci.*, vol. 2, no. 1, pp. 1–37, 2021, doi: 10.1145/3419106.

[12] F. N. Puteri, Y. Sibaroni, and F. F., “Hate Speech Detection in Indonesia Twitter Comments Using Convolutional Neural Network (CNN) and FastText Word Embedding,” *J. Media Inform. Budidarma*, vol. 7, no. 3, p. 1154, 2023, doi: 10.30865/mib.v7i3.6401.

[13] D. Bahdanau, K. H. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, pp. 1–15, 2015.

[14] A. Fan, M. Lewis, and Y. Dauphin, “Hierarchical neural story generation,” *ACL 2018 - 56th Annu. Meet. Assoc. Comput. Linguist. Proc. Conf. (Long Pap.*, vol. 1, pp. 889–898, 2018, doi: 10.18653/v1/p18-1082.

[15] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” *Adv. Neural Inf. Process. Syst.*, vol. 4, no. January, pp. 3104–3112, 2014.

[16] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, “Natural language processing (almost) from scratch,” *J. Mach. Learn. Res.*, vol. 12, pp. 2493–2537, 2011.