

PERINGKASAN TEKS ABSTRAKTIF OTOMATIS MENGGUNAKAN MODEL SEQ2SEQ DENGAN LSTM ENCODER-DECODER DAN ATTENTION PADA DATA SET AMAZON FINE FOOD REVIEWS

Juliasmita Eka Rahayu, Dyah Paminta Rahayu, Sitta Alief Farihati*

Program Studi Matematika, Universitas Terbuka, Tangerang Selatan, Indonesia

*Penulis korespondensi: sitta@ecampus.ut.ac.id

ABSTRAK

Penelitian ini bertujuan untuk membangun model peringkasan teks otomatis dengan strategi abstraktif. Model dibangun menggunakan pendekatan sequence-to-sequence (seq2seq) dengan arsitektur *encoder* dan *decoder* berbasis *Long Short-Term Memory* (LSTM) yang efektif untuk menangani data teks yang memerlukan proses mengingat informasi penting dari awal urutan dan menghubungkannya pada bagian akhir urutan. Kemudian ditambahkan mekanisme *attention* untuk membantu *decoder* fokus pada bagian-bagian tertentu yang memerlukan perhatian. Model dilatih pada pasangan data teks ulasan dan teks ringkasan pada data set Amazon *Fine Food Reviews* kemudian dilatih menggunakan fungsi kerugian (*loss*) *categorical crossentropy* dan algoritma optimasi RMSProp. Dari pelatihan diperoleh akurasi sebesar 67.2% dan kerugian sebesar 1.6314. Metode evaluasi dilakukan dengan menggunakan metrik BLEU (*Bilingual Evaluation Understudy*) untuk membandingkan hasil prediksi keluaran model dengan referensi manusia. Dari hasil evaluasi, model mampu menghasilkan ringkasan dengan skor kumulatif BLEU-1 sebesar 0.428571, BLEU-2 sebesar 0.654654, BLEU-3 sebesar 0.756080, dan BLEU-4 sebesar 0.809107. Hasil metrik BLEU menunjukkan model dapat menghasilkan prediksi ringkasan yang cukup koheren dan relevan.

Kata kunci: *automatic teks summarization, bleu score, long short-term memory, natural language processing, sequence-to-sequence*

1 PENDAHULUAN

Berkembangnya era teknologi mendorong keberagaman aktivitas digital masyarakat. Mulai dari bertukar informasi, transaksi jual-beli, hingga aktivitas konsumsi layanan pada saat ini dapat dilakukan secara daring melalui sebuah *platform* digital. Untuk dapat bertahan dalam persaingan, *platform* digital seperti media sosial, *e-commerce*, hingga *platform* layanan memerlukan sebuah kolom ulasan sebagai bentuk penilaian terhadap suatu produk atau layanan yang disediakan. Menurut Febriana *et al.* (2023), ulasan dapat dijadikan sebagai alasan konsumen untuk lebih meyakini bahwa produk yang ditawarkan sesuai dengan yang diharapkan konsumen, baik ulasan positif atau ulasan negatif karena ulasan tersebut benar adanya diberikan oleh konsumen lain yang sudah berbelanja. Dengan demikian, ulasan dapat dijadikan sebagai sumber informasi dalam mengambil keputusan dan alat ukur untuk menilai apakah suatu produk layak atau memerlukan perbaikan.

Ulasan dengan teks yang panjang terkadang membuat produsen merasa bingung untuk membacanya. Oleh karenanya, dibuat sebuah alat peringkasan yang disebut dengan peringkasan teks otomatis atau *Automatic Text Summarization* (ATS). ATS merupakan salah satu teknik dari *Neural Language Processing* (NLP) yang merupakan proses otomatis untuk menghasilkan ringkasan dari teks panjang namun tidak menghilangkan makna utama teks tersebut. Tujuan dari ATS adalah untuk membuat ringkasan yang singkat, mudah dipahami

kalimatnya dan memiliki informasi penting yang ingin disampaikan dari teks original tersebut (Setyawan *et al.*, 2021).

ATS memiliki dua strategi utama yaitu *extractive* yang menyusun ringkasan dengan mengambil kalimat penting dalam sumber dan *abstractive* yang menghasilkan ringkasan berupa teks baru berdasarkan pemahaman dalam sumber namun tetap pada intinya (Luo *et al.*, 2024). Dari keduanya, strategi pendekatan *abstractive* menjadi lebih populer dibandingkan dengan *extractive*, karena pendekatan *abstractive* dapat menghasilkan sebuah ringkasan yang lebih alami layaknya hasil ringkasan manusia (Shakil *et al.*, 2024), namun dalam pengimplementasiannya membutuhkan teknik dan arsitektur yang lebih kompleks.

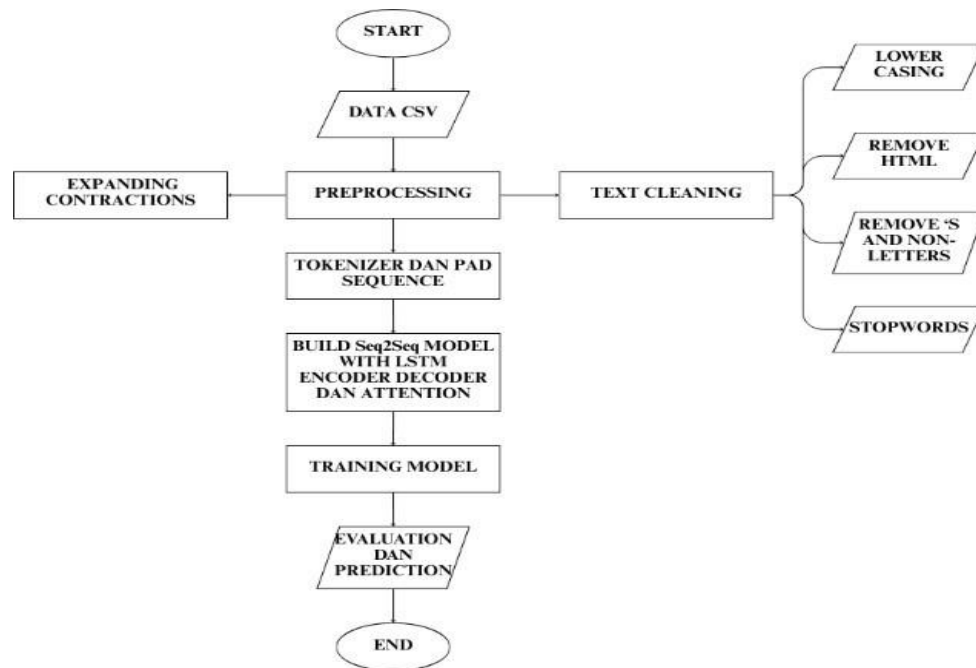
Salah satu metode yang digunakan dalam peringkasan teks abstraktif otomatis adalah model *Sequence-to-Sequence* (Seq2Seq) dengan menggunakan model *Recurrent Neural Networks* (RNN), khususnya arsitektur *Long Short-Term Memory* (LSTM). LSTM sangat efektif untuk menangani data teks yang memerlukan proses mengingat informasi penting dari awal urutan dan menghubungkannya pada bagian akhir urutan. Arsitektur ini dipandang lebih akurat menangani urutan temporal dan dependensi jarak jauh daripada RNN konvensional (Paaß & Giesselbach, 2023). Namun, sering kali LSTM kesulitan untuk mempertahankan fokus bagian-bagian penting saat panjang teks meningkat. Oleh karena itu, *attention* ditambahkan untuk membantu memilih informasi yang paling berguna selama proses *encoding* dan *decoding*.

Meskipun pendekatan dengan LSTM *encoder-decoder* dan *attention* cukup efektif untuk bidang peringkasan teks otomatis, namun tantangan tetap muncul ketika model digunakan pada data yang tidak berstruktur, memiliki panjang yang bervariasi, gaya bahasa yang informal, dan menggunakan kata-kata yang hiperbolis. Oleh karena itu, Nallapati *et al.* (2016) mengembangkan model Seq2Seq berbasis *encoder-decoder* RNN dengan *attention* untuk peringkasan abstraktif. Di sisi lain, S. & R. (2020) mengembangkan sistem peringkasan otomatis menggunakan model Seq2Seq berbasis LSTM *encoder-decoder* untuk menghasilkan ringkasan yang lebih natural dibanding pendekatan ekstraktif. Adapun Ivanedra dan Ivanedra & Mustikasari (2019) membahas implementasi RNN, khususnya LSTM, untuk membangun sistem peringkasan abstraktif berbahasa Indonesia.

Seperti halnya penelitian-penelitian tersebut, penelitian ini akan membahas pemanfaatan arsitektur LSTM dengan *attention* dalam sistem peringkasan teks otomatis (*ATS*). Namun, pada penelitian ini dilakukan pengoptimalan parameter *encoder-decoder* dengan mengimplementasikan arsitektur LSTM *encoder-decoder* untuk merangkum sekumpulan ulasan produk makanan dalam bahasa Inggris.

2 METODE

Penelitian ini menggunakan salah satu teknik dalam *neural language processing* (NLP) dengan pendekatan Seq2Seq. Model ini terdiri dari dua komponen yaitu *encoder* dan *decoder*. Dua komponen ini dibangun dengan jaringan LSTM. Kemudian ditambahkan mekanisme *attention* untuk memfokuskan *decoder* pada bagian-bagian penting dalam masukan saat menghasilkan ringkasan. Untuk mencapai tujuan penelitian, dilakukan tahapan-tahapan sebagai berikut: *data collecting*, *exploratory data analysis*, *preprocessing*, *tokenization*, *padding*, *modeling*, *training*, *evaluation*, dan *prediction*. Adapun hasil penelitian diperoleh dengan memanfaatkan pemrograman Python.

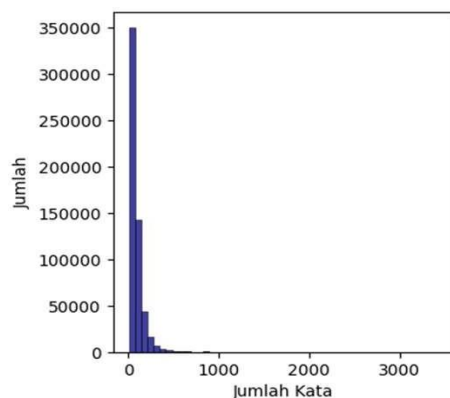


Gambar 1. Flowchart metodologi penelitian

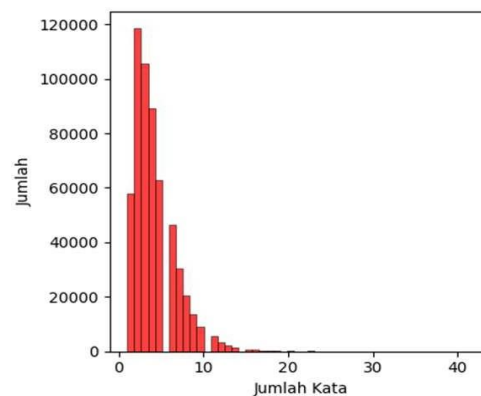
3 HASIL DAN PEMBAHASAN

Sesuai dengan metode yang digunakan, tahapan pertama yang dilakukan adalah *data collecting*. Dalam hal ini data set yang digunakan adalah kumpulan *review* makanan pada platform Amazon yang diunduh melalui Kaggle dengan nama *Amazon fine food review*. Data set ini berisi 568.454 ulasan dari 256.059 pengguna terhadap 74.258 produk yang dikumpulkan dalam periode Oktober 1999 sampai Oktober 2012. Data set ini tergolong data lama, namun hal ini tidak menjadi kendala, karena fokus penelitian ini pada arsitektur model untuk menghasilkan ringkasan.

Data set *Amazon fine food review* dipilih sebagai data untuk melatih model karena didasarkan pada distribusi panjang katanya yang tergolong pendek, sehingga pada saat pelatihan model lebih efisien dan cepat secara komputasi. Sebagian besar isi data set *Amazon fine food review* adalah ulasan yang terdiri dari jumlah kata di bawah seratus hingga dua ratus kata. Namun, ada juga sedikit ulasan dengan panjang di atas seribu kata. Selain itu ada beberapa kata yang dominan dalam dataset, seperti: “*gluten free*”, “*dog food*”, “*cat food*”, “*grocery store*”, “*highly recommend*”, “*much better*”, “*good*”, “*great product*”.



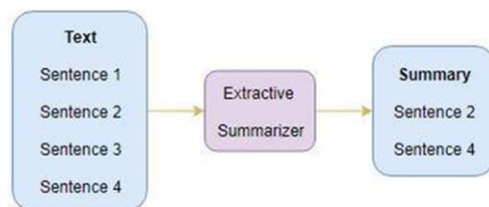
Gambar 2. Distribusi jumlah kata dalam ulasan



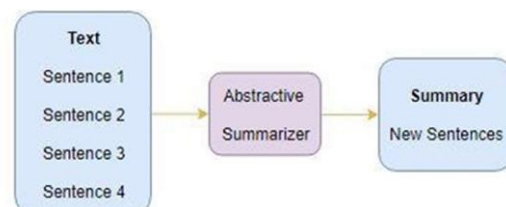
Gambar 3. Distribusi jumlah kata dalam ringkasan

Ulasan ditulis pengguna menggunakan bahasa Inggris, sehingga memudahkan proses *preprocessing* dengan bantuan pustaka *Natural Language Toolkit* (NLTK). Selain itu, ulasan ini bergaya bebas dan bervariasi dengan panjang yang tidak terstruktur secara eksplisit, sehingga sesuai dengan penelitian dalam bidang peringkasan teks otomatis khususnya teknik abstraktif. Data set diunduh dengan format CSV yang diolah dengan menggunakan bahasa pemrograman Python.

Selanjutnya dilakukan *exploratory data analysis* yaitu dengan melakukan peringkasan teks otomatis. Hal ini bertujuan untuk mengubah sejumlah besar data teks menjadi lebih ringkas, jelas dan mudah dipahami (Supriyono *et al.*, 2024). Peringkasan teks otomatis memiliki dua strategi, yaitu ekstraktif dan abstraktif. Peringkasan teks dengan strategi ekstraktif adalah memilih kalimat penting dari dalam sumber kemudian disusun jadi satu menjadi ringkasan, kalimat yang dipilih tetap dalam bentuk aslinya tanpa modifikasi apa pun (Tsai *et al.*, 2020). Adapun strategi abstraktif akan menghasilkan ringkasan berupa teks baru berdasarkan pemahaman dalam sumber namun tetap pada intinya (Luo *et al.*, 2024). Perbedaan keduanya dapat dilihat pada Gambar 4 dan 5.

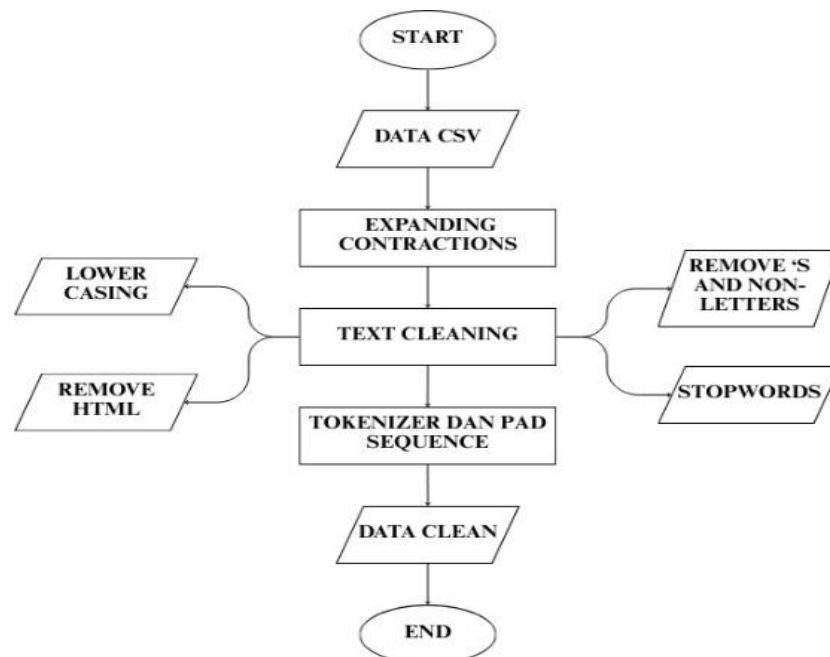


Gambar 4. Strategi Ekstraktif
(diadaptasi dari Akhter & Mehra
(2022))



Gambar 5. Strategi Abstraktif
(diadaptasi dari Akhter & Mehra
(2022))

Selanjutnya tahapan *preprocessing* yang bertujuan untuk meningkatkan kualitas data dengan mengeliminasi masalah-masalah yang ada pada data tersebut (Muflikhah *et al.*, 2023).



Gambar 6. Flowchart preprocessing

Dalam tahapan *preprocessing* dilakukan proses *expanding contractions* yaitu mengubah bentuk kata singkatan menjadi bentuk lengkapnya, seperti “*ain’t*” menjadi “*is not*”, “*aren’t*” menjadi “*are not*”, dan “*can’t*” menjadi “*cannot*”. Selanjutnya proses *cleaning text*, antara lain (1) melakukan *setting stopwords* dalam bahasa Inggris dari NLTK, (2) *lowercase* untuk mengubah huruf besar menjadi huruf kecil supaya tidak ada duplikasi, contohnya seperti “Jumbo” menjadi “jumbo” dan “Salted” menjadi “salted”, (3) menghapus *tag HTML*, (4) menghapus tanda baca, dan (5) menghapus kata yang pendek. Setelah tahapan *expanding contractions* dan *cleaning text* dijalankan, data yang awalnya berisi *noise* seperti *tag HTML* dan tanda baca menjadi lebih bersih, terstruktur dan bebas dari *noise* yang dapat mengganggu pemodelan.

Tabel 1. Perbedaan sebelum *preprocessing* dan setelah *preprocessing*

Sebelum <i>preprocessing</i>	Setelah <i>preprocessing</i>
Teks: <i>Product arrived labeled as Jumbo Salted Peanuts...the peanuts were actually small sized unsalted. Not sure if this was an error or if the vendor intended to represent the product as "Jumbo".</i>	Teks: <i>product arrived labeled jumbo salted peanuts peanuts actually small sized unsalted sure error vendor intended represent product jumbo</i>
Ringkasan: <i>Not as Advertised</i>	Ringkasan: <i>not as advertised</i>
Teks: <i>Great taffy at a great price. There was a wide assortment of yummy taffy. Delivery was very quick. If your a taffy lover, this is a deal.</i>	Teks: <i>great taffy great price wide assortment yummy taffy delivery quick taffy lover deal</i>
Ringkasan: <i>Great taffy</i>	Ringkasan: <i>great taffy</i>

Selanjutnya data yang telah bersih dilakukan analisis distribusi panjang kata yang dijadikan sebagai landasan untuk menentukan *hyperparameter* batas maksimum panjang teks. Batas maksimum ini digunakan sebagai tetapan untuk memastikan konsistensi panjang masukan dan keluaran pada saat pelatihan model. Selain itu, batas maksimum juga bertujuan untuk mengurangi beban komputasi dan mempercepat pelatihan model. Setelah dilakukan beberapa percobaan, diperoleh cakupan 63% dari data ulasan dengan panjang teks lebih dari sama dengan 35 kata, dan cakupan 94% dari data ringkasan dengan panjang teks lebih dari sama dengan 8 kata. Untuk membantu model mengetahui batas kalimat saat pelatihan sehingga *decoding* dapat menghasilkan ringkasan yang baik, maka ditambahkan *token* khusus di awalan dan di akhiran kata, yaitu *sostok* sebagai awalan (*start token*) dan *eostok* sebagai akhiran (*end token*).

Tokenization adalah proses memecah sebuah kalimat menjadi bagian-bagian kecil yang disebut token atau kata Nambiar *et al.* (2021). Dalam penelitian ini, proses *tokenization* menggunakan “Tokenizer” dari pustaka yang secara langsung dapat membangun *vocabulary* berdasarkan frekuensi kemunculan kata dalam data pelatihan. Jenis yang digunakan adalah

tokenization kata dengan tujuan memecah teks menjadi potongan kata. Misalkan suatu teks adalah $T = [w_1, w_2, \dots, w_n]$, maka hasil tokenisasinya Adalah sebagai berikut.

$$\text{Token}(T) = [i_1, i_2, \dots, i_n] \text{ dengan } i_j \in \mathbb{N} \quad (1)$$

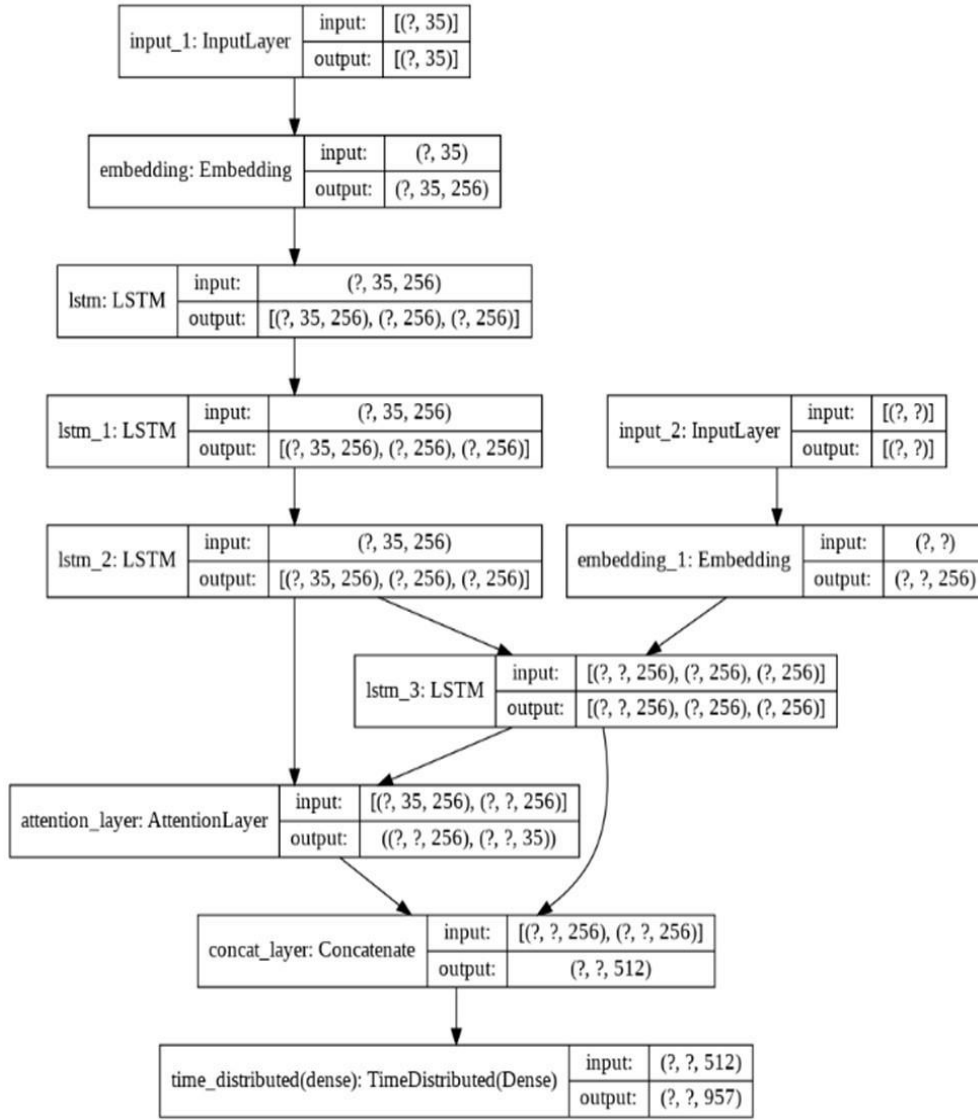
Selanjutnya, dilakukan analisis terhadap kata unik yang jarang muncul (*rare words*) dari ambang batas tertentu (*threshold*). Dalam studinya, Bystrov *et al.* (2023) mengemukakan bahwa menghapus kata-kata unik yang jarang muncul dapat mempercepat proses komputasi saat pelatihan model. Dalam penelitian ini, diambil *threshold* = 4 untuk teks ulasan dan *threshold* = 6 untuk teks ringkasan. Ini memiliki arti bahwa kata dalam korpus yang jarang muncul akan dipertahankan apabila muncul minimal sebanyak 4 kali pada teks ulasan, dan 6 kali pada teks ringkasan. Dengan *threshold* sebesar 4 diperoleh pengurangan sebanyak 65,61% kata unik yang jarang muncul namun hanya mengurangi sekitar 4,94% dari teks ulasan. Adapun pada teks ringkasan, dengan *threshold* sebesar 6 diperoleh pengurangan sebanyak 80,80% kata unik yang jarang muncul namun hanya mengurangi sekitar 9,10% dari teks ringkasan.

Setelah dilakukan tokenisasi, diperoleh panjang *sequence* yang berbeda-beda. Oleh karena itu diperlukan *padding* untuk menyamakan panjang masukan sebelum diproses oleh model. ***Padding*** dilakukan dengan menambahkan token khusus seperti nol di awalan yang disebut *pre-padding* dan di akhiran yang disebut *post-padding*. Hal ini dapat digambarkan dengan:

$$X_{pad}, Y_{pad} = \begin{cases} [0, 0, \dots, 0, i_1, i_2, \dots, i_n] & \text{jika } n < L \\ [i_1, i_2, \dots, i_L] & \text{jika } n \geq L \end{cases} \quad (2)$$

dengan 0 adalah *padding* yang disisipkan apabila $n < L$; $i_1, i_2, \dots, i_n$ adalah token hasil dari *tokenization*, dan L adalah panjang maksimum *sequence*.

Tahapan berikutnya adalah ***modeling***. Dalam penelitian ini, model dibangun dengan Seq2Seq model yang terdiri dari dua bagian utama yaitu *Encoder* dan *Decoder* yang dibangun dengan jaringan LSTM serta ditambahkan mekanisme *attention*. Menurut Niu *et al.* (2021), model *Encoder-Decoder* adalah suatu kerangka berbasis jaringan saraf yang dirancang untuk menangani pemetaan antara masukan dan keluaran yang memiliki struktur kompleks. Dalam bukunya, Paaß & Giesselbach (2023) menyatakan bahwa tantangan utama dalam Seq2Seq model adalah *decoder* yang harus memperhatikan penyematan token *encoder*. Dengan begitu, *Attention* ditambahkan untuk mengatasi permasalahan dalam pemrosesan teks yang memerlukan perhatian pada bagian-bagian tertentu. Arsitektur yang digunakan dibangun oleh beberapa komponen, yaitu *embedding layer*, *encoder LSTM* bertingkat, *decoder LSTM*, lapisan *attention* oleh Bahdanau, dan *TimeDistributed Dense layer* sebagai keluaran. Skrip Seq2Seq model dapat dilihat pada gambar sebagai berikut.



Gambar 18. Arsitektur model

Model diawali dengan dua *InputLayer* untuk *encoder* dan *decoder*. *Encoder* berfungsi untuk mengubah kalimat masukan dan mempertahankan konteks dan makna aslinya dengan memanfaatkan lapisan LSTM yang menyimpan informasi dalam bentuk *hidden state* (h_t) dan *cell state* (C_t) sebagai umpan balik internal (Chaurasia *et al.*, 2023). Dalam penelitian ini, *encoder* menerima masukan yang berbentuk urutan token dengan panjang 35 token dan mengubahnya menjadi vektor menggunakan *embedding layer* yang berukuran (35, 256). Vektor ini kemudian diteruskan ke tiga buah lapisan LSTM bertingkat (*lstm*, *lstm_1*, *lstm_2*) sebagai *encoder*. Dalam bukunya, Yudistira *et al.* (2023) merumuskan bahwa model LSTM bekerja pada gerbang-gerbang dengan proses di setiap waktu didefinisikan sebagai:

$$\begin{aligned}
 f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\
 i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\
 \tilde{C}_t &= \tanh(W_{\tilde{C}} \cdot [h_{t-1}, x_t] + b_{\tilde{C}}) \\
 C_t &= f_t \odot C_t + i_t \tilde{C}_t \\
 o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\
 h_t &= o_t \odot \tanh(C_t)
 \end{aligned} \tag{3}$$

dengan f_t adalah *forget gate*; i_t adalah *input gate*; $\tilde{C}_t$ adalah *candidate cell state*; C_t adalah *new cell stat*; o_t adalah *output gate*; h_t adalah *hidden state*; σ adalah fungsi aktivasi sigmoid; x_t adalah masukan saat t ; W dan b adalah bobot dan bias.

Dalam penelitian ini, model menggunakan *encoder unidirectional* LSTM berupa 3 LSTM bertingkat satu arah, sehingga keluaran dari setiap layer akan menjadi masukan untuk layer berikutnya. Dengan mengambil formula bentuk *hidden state* (h_t) dan *cell state* (C_t) dari buku Yudistira *et al.* (2023), maka secara matematis untuk layer pertama (lstm) dirumuskan dengan:

$$h_t^{(1)}, c_t^{(1)} = LSTM_1(h_{t-1}^{(1)}, c_{t-1}^{(1)}) \quad (4)$$

dan untuk layer ke dua (lstm_1) dirumuskan dengan:

$$h_t^{(2)}, c_t^{(2)} = LSTM_2(h_t^{(1)}, h_{t-1}^{(2)}, c_{t-1}^{(2)}) \quad (5)$$

Adapun untuk *layer* ketiga (lstm_2) yang merupakan keluaran akhir *encoder* dirumuskan dengan:

$$h_t^{(3)}, c_t^{(3)} = LSTM_2(h_t^{(2)}, h_{t-1}^{(3)}, c_{t-1}^{(3)}). \quad (6)$$

Selanjutnya, informasi dari masukan yang telah disaring oleh *encoder* diteruskan ke *decoder* LSTM (lstm_3). *Decoder* bertugas memprediksi kalimat keluaran secara bertahap kata demi kata (Chaurasia *et al.*, 2023). *Decoder* menerima urutan target yang kemudian diubah menjadi vektor-vektor melalui lapisan *embedding* yang sama dengan *encoder*. Dengan kata lain *decoder* tidak mulai dari nol, melainkan menggunakan *initial state* dari *encoder layer* ketiga (lstm_2) dengan $s_0 = h_{T_x}^{(3)}, c_0 = c_{T_x}^{(3)}$. Dengan demikian, saat *decoder* pada waktu t dalam matematika dirumuskan dengan:

$$s_t, c_t = LSTM(y_{t-1}, s_{t-1}, c_{t-1}) \quad (7)$$

Karena model untuk peringkasan teks otomatis yang dibangun dengan Seq2Seq model tergolong lemah, maka diperlukannya *attention*. Dalam hal ini, konsep *attention* memungkinkan model *Neural Network* untuk melihat kembali bagian-bagian input yang relevan saat menghasilkan setiap token *output* sehingga meningkatkan akurasi penerjemahan maupun ringkasan (Bahdanau *et al.*, 2014). Untuk meningkatkan akurasi pemodelan, ditambahkan mekanisme *attention* sesuai dengan penelitian Bahdanau *et al.* (2014).

Dalam mekanisme *attention* ini, keluaran *encoder* (h_i) diberikan bobot berdasarkan seberapa relevan status tersembunyi terakhir *decoder* (s_{t-1}) kemudian digabungkan menjadi vektor (Bahdanau *et al.*, 2014). Untuk menentukan kata keluaran pada langkah t , sistem menghitung *attention weights* (α_{ti}) untuk setiap keluaran *encoder* (h_i) berdasarkan seberapa *score* kecocokan (e_{ti}) vektor h_i dengan status tersembunyi *decoder* s_{t-1} . Dalam penelitian Bahdanau *et al.* (2014), *score* kecocokan secara matematis ditulis:

$$e_{ti} = score(s_{t-1}, h_i) = v_a^T \tanh W_a s_{t-1} + U_a h_i \quad (8)$$

dengan W_a , U_a , dan v_a adalah bobot pelatihan.

Attention weights dalam Bahdanau *et al.* (2014) dirumuskan dengan:

$$a_{ti} = \frac{\exp(e_{ti})}{\sum_{k=1}^T \exp(e_{tk})} \quad (9)$$

Rumus ini menggunakan fungsi *softmax* yang dapat mengubah skor e_{ti} menjadi nilai probabilitas. Adapun konteks vektor yang digunakan oleh *decoder* adalah:

$$c_t = \sum_{i=1}^T \alpha_{ti} h_i \quad (10)$$

Konteks vektor c_t tersebut adalah hasil gabungan dari semua keluaran dari *encoder* yang telah diberikan bobot α_{ti} . Hal ini menandakan seberapa besar perhatian yang diberikan model terhadap setiap elemen dalam masukan saat menghasilkan keluaran pada waktu ke- t sehingga c_t bertugas sebagai pembantu *decoder* untuk menghasilkan keluaran (Bahdanau *et al.*, 2014).

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 35)]	0	
embedding (Embedding)	(None, 35, 256)	1312768	input_1[0][0]
lstm (LSTM)	[(None, 35, 256), (N 525312		embedding[0][0]
input_2 (InputLayer)	[(None, None)]	0	
lstm_1 (LSTM)	[(None, 35, 256), (N 525312		lstm[0][0]
embedding_1 (Embedding)	(None, None, 256)	244992	input_2[0][0]
lstm_2 (LSTM)	[(None, 35, 256), (N 525312		lstm_1[0][0]
lstm_3 (LSTM)	[(None, None, 256), 525312		embedding_1[0][0] lstm_2[0][1] lstm_2[0][2]
attention_layer (AttentionLayer ((None, None, 256),	131328		lstm_2[0][0] lstm_3[0][0]
concat_layer (Concatenate)	(None, None, 512)	0	lstm_3[0][0] attention_layer[0][0]
time_distributed (TimeDistribut	(None, None, 957)	490941	concat_layer[0][0]
Total params: 4,281,277			
Trainable params: 4,281,277			
Non-trainable params: 0			

Gambar 19. Ringkasan model

Setelah arsitektur model dirancang, dilakukan proses **training** untuk mengoptimalkan parameter-parameter model. Untuk menghasilkan ringkasan yang akurat, model membutuhkan fungsi objektif dengan probabilitas bersyarat keluaran serta fungsi kerugian (*loss*) yang dapat membantu proses pelatihan yang disebut dengan *likelihood function*. Dalam Bahdanau *et al.* (2014), hal ini diformulasikan sebagai:

$$P(y) = \prod_{t=1}^T P(y_t | y_1, y_2, \dots, y_{t-1}, c) \quad (11)$$

maka,

$$P(y) = \prod_{t=1}^T P(y_t | y_{<t}, c) \quad (12)$$

$$\log P(y) = \log \prod_{t=1}^T P(y_t | y_{<t}, c) \quad (13)$$

$$\log P(y) = \prod_{t=1}^T \log P(y_t | y_{<t}, c) \quad (14)$$

dengan:

y_t : token ke- t dari keluaran,

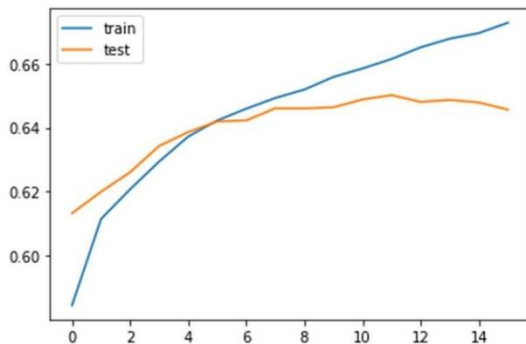
$y_{<t}$: semua token sebelum waktu t ,

c : hasil dari *encoder + attention*.

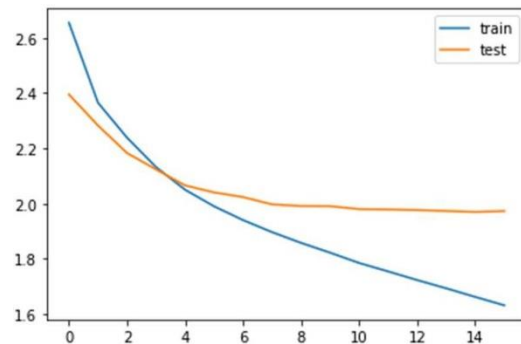
Dalam penelitian ini, fungsi kerugian di atas diimplementasikan ke dalam fungsi *loss* pada Keras yaitu *sparse_categorical_crossentropy*. Seperti yang dijelaskan oleh Goodfellow *et al.* (2016), maksimum likelihood menjadi peminimalan *negative log-likelihood* (NLL), atau secara setara, peminimalan *cross-entropy*. Dengan begitu, fungsi objektif *log-likelihood* di atas diubah menjadi bentuk NLL sehingga bentuk formulanya menjadi:

$$\mathcal{L} = - \sum_{t=1}^T \log P(y_t | y_{<t}, c) \quad (15)$$

Sedangkan untuk algoritma optimasi dipilih menggunakan RMSProp yang efektif untuk RNN seperti LSTM (Matrenin *et al.*, 2020). Model ini mendapatkan nilai akurasi (*accuracy*) sebesar sekitar 67.27% pada data *training* di *epoch* ke-16, dan akurasi sebesar sekitar 64.56% pada data validasi di *epoch* yang sama. Untuk fungsi kerugian (*loss*) mendapatkan nilai sebesar sekitar 1.6314 pada data *training* dan sekitar 1.9727 pada data validasi di *epoch* ke-16. Skrip kompilasi model ini dapat dilihat gambar sebagai berikut.



Gambar 23. Grafik akurasi



Gambar 24. Grafik kerugian

Metode **evaluasi** dilakukan dengan menggunakan metrik BLEU (*Bilingual Evaluation Understudy*) untuk membandingkan hasil prediksi keluaran model dengan referensi manusia. BLEU *score* menghitung kesamaan antara n -gram pada kalimat hasil prediksi (*candidate*) dengan kalimat referensi (*reference*) (Papineni *et al.*, 2002).

Misalkan c adalah panjang dari kalimat hasil prediksi (*candidate*) dan r adalah panjang efektif dari kalimat referensi (*reference*). Dalam Papineni *et al.* (2002), untuk menghitung penalti singkat (*brevity penalty*), yang dinotasikan dengan BP, digunakan rumus:

$$BP = \begin{cases} 1 & \text{jika } c > r \\ e^{(1-\frac{r}{c})} & \text{jika } c \leq r \end{cases} \quad (16)$$

Skor BLEU dihitung dengan mengalikan penalti singkat ini dengan rata-rata geometrik dari presisi n -gram, yaitu:

$$BLEU = BP \times \exp \left(\sum_{n=1}^N w_n \log p_n \right) \quad (17)$$

di mana p_n adalah presisi untuk n -gram ke- n , dan w_n adalah bobot untuk n -gram tersebut. Dalam implementasi standar, digunakan $N = 4$ dan bobot yang seragam, yaitu $w_n = \frac{1}{N}$ sehingga persamaan BLEU menjadi:

$$\log BLEU = \min \left(1 - \frac{r}{c}, 0 \right) + \sum_{n=1}^N w_n \log p_n \quad (18)$$

Selanjutnya, ditentukan BLEU *score* dengan menggunakan fungsi `sentence_bleu()` dari pustaka NLTK dengan pengaturan bobot yang merepresentasikan bobot kumulatif untuk n -gram dari 1 hingga 4. Untuk menghitung BLEU-1, hanya mempertimbangkan kemunculan 1-gram dengan bobot penuh pada $w_1 = 1$ dan sisanya nol. Untuk BLEU-2, digunakan bobot rata untuk 1-gram dan 2-gram (kumulatif) dengan $w_1 = 0,5$ dan $w_2 = 0,5$. Kemudian untuk BLEU-3 digunakan total bobot = 1 (dibagi rata ketiga n -gram), sedangkan untuk BLEU-4 digunakan bobot kumulatif secara langsung. Secara matematis, Papineni *et al.* (2002) menjabarkan formula BLEU sebagai berikut:

$$BLEU_1 = BP \cdot \exp(\log p_1) = BP \cdot \exp p_1 \quad (19)$$

$$BLEU_2 = BP \cdot \exp\left(\frac{1}{2}(\log p_1 + \log p_2)\right) \quad (20)$$

$$BLEU_3 = BP \cdot \exp\left(\frac{1}{3}(\log p_1 + \log p_2 + \log p_3)\right) \quad (21)$$

$$BLEU_4 = BP \cdot \exp\left(\frac{1}{4}(\log p_1 + \log p_2 + \log p_3 + \log p_4)\right) \quad (22)$$

Berikut ini tabel hasil perhitungan BLEU *score* kumulatif pada tahapan evaluasi. Adapun skrip BLEU *score* kumulatif dapat dilihat pada tabel sebagai berikut.

Tabel 2. Tabel hasil metode evaluasi BLEU *score* kumulatif

Jenis	Score
BLEU-1	0.428571
BLEU-2	0.654654
BLEU-3	0.756080
BLEU-4	0.809107

Dalam Mastropaolo *et al.* (2023) dinyatakan bahwa jika nilai BLEU berada pada rentang nol hingga satu, di mana nilai semakin mendekati satu, maka menunjukkan bahwa ringkasan semakin mendekati referensi. Dengan kata lain, saat nilai BLEU mendekati satu, maka ringkasan dianggap koheren dan relevan. Sesuai dengan pernyataan tersebut, dalam penelitian ini, *score* BLEU-1 hingga BLEU-4 terus meningkat dengan BLEU-4 sebesar 0.809107. Hal ini menunjukkan bahwa model mampu menghasilkan ringkasan yang koheren dan relevan.

Tahapan akhir adalah ***prediction*** yang hasilnya ditunjukkan pada tabel berikut ini.

Tabel 3. Tabel perbandingan ringkasan hasil prediksi model dengan teks ringkasan asli

No	Teks ulasan	Teks ringkasan asli	Hasil prediksi
1	<i>thing buy lot stores much say love loves bought numerous times</i>	<i>great product great price</i>	<i>great product</i>
2	<i>tasty satisfying meat protien sticks perfect meal snack addition protien diet pepper flavor adds something missing regular flavor variety snacks recommended</i>	<i>meat of</i>	<i>delicious</i>
3	<i>chips great delicious healthier regular deli store chips first purchased supermarket found delicious found amazon cheaper ordered variety flavor flavors liking con addicting warned purchased amazon</i>	<i>delicious</i>	<i>great chips</i>
4	<i>always happy newman products coffee exception found priced love fact organic</i>	<i>coffee</i>	<i>great coffee</i>

4 SIMPULAN

Model berhasil mendapatkan akurasi sebesar 67,2% dengan menggunakan *unidirectional* LSTM yaitu 3 LSTM bertingkat satu arah pada *encoder*, dan pada *decoder* menggunakan *unidirectional* LSTM satu lapis yang menggunakan initial state dari *encoder* (state_h, state_c) serta menambahkan mekanisme *attention*. Terkait metrik BLEU, model berhasil mendapatkan skor kumulatif BLEU-1 sebesar 0.428571, BLEU-2 sebesar 0.654654, BLEU-3 sebesar 0.756080, dan BLEU-4 sebesar 0.809107.

DAFTAR PUSTAKA

- Akhter, B., & Mehra, M. (2022). A study of implementation of deep learning techniques for text summarization. *International Journal of Innovative Research in Engineering and Management*, 9(2), 18–28.
- Bahdanau, D., Cho, K., & Bengio, Y. (2014). *Neural Machine Translation by Jointly Learning to Align and Translate*.
- Bystrov, V., Naboka-Krell, V., Staszewska-Bystrova, A., & Winker, P. (2023). Analysing the Impact of Removing Infrequent Words on Topic Quality in LDA Models. *ArXiv Preprint*, 1–21.
- Chaurasia, S., Dasgupta, D., & Regunathan, R. (2023). T5LSTM-RNN based Text Summarization Model for Behavioral Biology Literature. *Procedia Computer Science*, 218, 585–593. <https://doi.org/10.1016/j.procs.2023.01.040>
- Febriana, A., Nur, Y., & Mariah, M. (2023). Pengaruh ulasan produk, kemudahan, dan keamanan terhadap keputusan pembelian produk secara online pada Shopee di Kota Makassar. *Jurnal Malomo : Manajemen Dan Akuntansi*, 1(3), 281–292.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. The MIT Press.
- Ivanedra, K., & Mustikasari, M. (2019). Implementasi Metode Recurrent Neural Network pada Text Summarization dengan Teknik Abstraktif. *Jurnal Teknologi Informasi Dan Ilmu Komputer*, 6(4), 377–382. <https://doi.org/10.25126/jtiik.2019641067>
- Luo, M., Xue, B., & Niu, B. (2024). A comprehensive survey for automatic text summarization: Techniques, approaches and perspectives. *Neurocomputing*, 603, 128280. <https://doi.org/10.1016/j.neucom.2024.128280>

- Mastropaolo, A., Ciniselli, M., Di Penta, M., & Bavota, G. (2023). *Evaluating Code Summarization Techniques: A New Metric and an Empirical Characterization*.
- Matrenin, P. V., Manusov, V. Z., Khalyasmaa, A. I., Antonenkov, D. V., Eroshenko, S. A., & Butusov, D. N. (2020). Improving Accuracy and Generalization Performance of Small-Size Recurrent Neural Networks Applied to Short-Term Load Forecasting. *Mathematics*, 8(12), 2169. <https://doi.org/10.3390/math8122169>
- Muflikhah, L., Mahmudy, W. F., & Kurnianingtyas, D. (2023). *Machine learning: Cetakan Pertama*. UB Press.
- Nallapati, R., Zhou, B., dos Santos, C., Gulcehre, C., & Xiang, B. (2016). Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond. *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, 280–290. <https://doi.org/10.18653/v1/K16-1028>
- Nambiar, S. K., Peter S, D., & Idicula, S. M. (2021). Attention based Abstractive Summarization of Malayalam Document. *Procedia Computer Science*, 189, 250–257. <https://doi.org/10.1016/j.procs.2021.05.088>
- Niu, Z., Zhong, G., & Yu, H. (2021). A review on the attention mechanism of deep learning. *Neurocomputing*, 452, 48–62. <https://doi.org/10.1016/j.neucom.2021.03.091>
- Paaß, G., & Giesselbach, S. (2023). *Foundation Models for Natural Language Processing -- Pre-trained Language Models Integrating Media*.
- Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). BLEU: A method for automatic evaluation of machine translation. *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02*, 311–318. <https://doi.org/10.3115/1073083.1073135>
- S., K., & R., V. (2020). Abstractive Text Summarization using Seq2seq Model. *International Journal of Computer Applications*, 176(33), 24–26. <https://doi.org/10.5120/ijca2020920401>
- Setyawan, C., Benarkah, N., & Prasetyo, V. R. (2021). Automatic Text Summarization Berdasarkan Pendekatan Statistika pada Dokumen Berbahasa Indonesia. *KELUWIH: Jurnal Sains Dan Teknologi*, 2(1), 9–15. <https://doi.org/10.24123/saintek.v2i1.4045>
- Shakil, H., Farooq, A., & Kalita, J. (2024). Abstractive text summarization: State of the art, challenges, and improvements. *Neurocomputing*, 603, 128255. <https://doi.org/10.1016/j.neucom.2024.128255>
- Supriyono, S., Wibawa, A. P., Suyono, S., & Kurniawan, F. (2024). A survey of text summarization: Techniques, evaluation and challenges. *Natural Language Processing Journal*, 7, 100070. <https://doi.org/10.1016/j.nlp.2024.100070>
- Tsai, C.-F., Chen, K., Hu, Y.-H., & Chen, W.-K. (2020). Improving text summarization of online hotel reviews with review helpfulness and sentiment. *Tourism Management*, 80, 104122. <https://doi.org/10.1016/j.tourman.2020.104122>
- Yudistira, N., Alfiansih, L. M. D., Andriyani, N. I., Essayem, W., Maulida, N., Maghfiroh, N. A., & Nurdian, I. W. (2023). *Prediksi deret waktu menggunakan deep learning*. UB Press.