BAB 2
LANDASAN TEORI

2.1 Ringkasan

Menurut Kamus Besar Bahasa Indonesia (KBBI), kata “ringkasan” berasal dari kata “ringkas” yang memiliki makna “singkat”, misalnya tentang perkataan atau cerita. Berdasarkan pengertian tersebut, di dalam konteks suatu peristiwa yang terjadi, dapat diartikan kembali bahwa, suatu ringkasan merupakan bentuk rangkaian peristiwa yang lebih singkat dari rangkaian peristiwa secara keseluruhan. Contoh ringkasan bisa dilakukan terhadap suatu artikel dengan tujuan untuk memperoleh informasi pokok yang terkandung dalam artikel tersebut.

2.2 Peringkasan Teks Otomatis

Pada peringkasan teks otomatis, sebuah ringkasan dapat didefinisikan sebagai sebuah teks yang dihasilkan dari satu atau lebih teks yang mengandung informasi penting dari teks aslinya, tidak lebih panjang dari setengah teks aslinya dan biasanya, jauh lebih pendek dari teks aslinya [11].

Peringkasan teks otomatis adalah sebuah proses menghasilkan suatu ringkasan yang fasih dan singkat dengan mempertahankan informasi kunci dari konten dan makna secara keseluruhan [1]. Ketika kita sebagai manusia meringkas sebuah potongan teks, kita biasanya membaca secara keseluruhan untuk mengambangkan pemahaman terhadap teks tersebut, kemudian menulis ringkasannya dengan menyoroti poin-poin utama [1]. Karena komputer tidak mempunyai pemahaman manusia dan kemampuan berbahasa, ini membuat peringkasan teks otomatis menjadi suatu pekerjaan yang sulit [1].

Terdapat dua metode peringkasan teks otomatis, yaitu abstraktif dan ekstraktif. Metode ekstraktif adalah sebuah metode untuk meringkas dokumen dengan cara memilih sebuah bagian dari teks atau kalimat di dalam dokumen [2]. Sedangkan, metode abstraktif adalah sebuah metode untuk meringkas dokumen dengan cara membuat kalimat-kalimat baru yang memiliki informasi yang sama dengan dokumen aslinya [2].

2.3 Berita

Berita merupakan informasi yang disajikan dalam bentuk lisan atau tulisan untuk menyampaikan suatu peristiwa yang terjadi—berita biasa disampaikan di televisi, radio, koran, dll. Menurut KBBI, kata "berita" mempunyai arti "cerita atau keterangan mengenai kejadian atau peristiwa yang hangat."

Terdapat unsur-unsur yang terkandung di dalam suatu berita, yaitu 5W + 1 yang merupakan *what* (apa), *where* (di mana), *when* (kapan), *who* (siapa), *why* (kenapa), dan *how* (bagaimana). Dalam jurnalistik, unsur 5W + 1H digunakan agar berita yang disampaikan kepada masyarakat dapat diterima dengan jelas. Jika suatu berita memenuhi unsur 5W + 1H, maka berita tersebut sudah memenuhi kaidah jurnalistik [12].

2.4 Text Preprocessing

Text preprocessing adalah proses mengubah teks tidak terstruktur ke dalam bentuk terstruktur sebelum memasuki tahap *training* [10]. Pada penelitian ini, akan dilakukan dua tahap *preprocessing*, yaitu *lower casing* dan *filtering*.

2.4.1 Lower Casing

Lower casing digunakan untuk mengubah semua huruf di dalam teks ke dalam bentuk *lower case* (huruf kecil). *Lower casing* bertujuan untuk membuat kata yang sama agar tidak dianggap berbeda, misalnya, kata "Buku" dan "buku", keduanya adalah kata yang sama namun memiliki perbedaan pada penggunaan huruf *b* dan *B*.

2.4.2 Filtering

Filtering digunakan untuk melakukan filter atau penyaringan terhadap teks untuk menghilangkan karakter-karakter yang tidak diinginkan. Misalnya, menghilangkan tanda pagar (#), garis miring (/), dll.

2.4.3 Tokenization

Tokenization adalah proses memotong deretan karakter ke dalam beberapa bagian, bagian-bagian tersebut disebut sebagai *token*. *Token* merupakan *instance*

(contoh) dari deretan karakter-karakter yang dikelompokkan secara bersamaan sebagai suatu unit semantik yang bermanfaat untuk pemrosesan [15].

SentencePiece adalah *tokenizer* yang bisa memecah kata menjadi beberapa *token*. SentencePiece bisa mengimplementasikan metode *unigram tokenization* yang memecah kata dengan memprioritaskan kandidat-kandidat susunan suku kata $x = (x_i, ..., x_M)$ yang memiliki nilai probabilitas paling tinggi.

$$P(x) = \prod_{i=1}^{M} p\,(x_i)$$

$$x_i \in V, \sum_{x_i \in V} p\,(x) = 1$$

Di mana *V* adalah sebuah *vocab* (kamus kata/sukukata) yang sebelumnya sudah ditentukan. Hasil segmentasi dengan nilai probabilitas paling tinggi x^* untuk *input* kalimat *X* adalah:

$$x^* = \underset{x \in S(X)}{argmax} P(x)$$

di mana *S(X)* adalah kumpulan kandidat segmentasi (susunan suku kata) yang dibangun dari kalimat *X*.

2.5 Word Embedding

Model *machine learning* (pembelajaran mesin) menerima vektor sebagai *input*. Ketika berurusan dengan teks, strategi pertama yang harus dilakukan adalah mengubah *string* menjadi angka—vektorisasi terhadap teks—sebelum memasukannya ke model [17]. *Word embedding* adalah vektor-vektor yang merepresentasikan kata-kata [18].

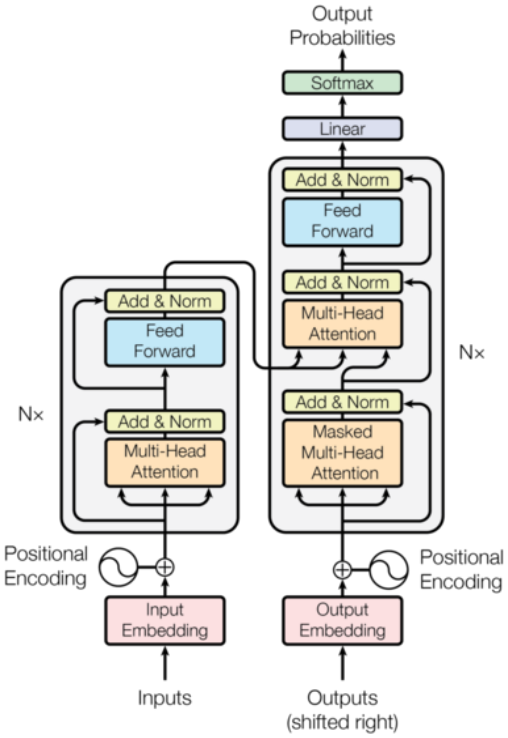
One-hot encoding adalah salah satu metode untuk membuat vektor-vektor yang merepresentasikan kata-kata. *One-hot encoding* yang bekerja dengan cara membuat vektor-vektor nol (0) dengan panjang yang sama sesuai dengan kosakata, kemudian menyimpan angka satu (1) di dalam indeks yang berkorespondensi

terhadap kata-kata [17]. *One-hot encoding* tidak efisien karena menghasilkan *sparse vector* di mana banyak dari indeks-indeksnya bernilai nol.

Sedangkan, *word embedding* adalah pendekatan yang lebih efisien karena, alih-alih menggunakan *sparse vector* seperti *one-hot encoding*, *word embedding* menggunakan *dense vector* dengan panjang vektor yang bisa ditentukan. *Word embedding* bekerja seolah-olah seperti *lookup table*.

2.6 Transformer

Transformer adalah model *deep learning* dengan arsitektur *encoder-decoder*. Transformer menawarkan mekanisme-mekanisme baru (*self-attention* dan *positional encoding*) yang dapat memberikan representasi waktu terhadap deretan input dan memberikan nilai fokus pada bagaimana kata-kata berhubungan satu sama lain [19].



Gambar 2.1 Arsitektur Transformer

Pada Gambar 2.1, bagian sebelah kiri merupakan *encoder* dan bagian sebelah kanan merupakan *decoder*, keduanya menerima vektor *embedding*.

2.6.1 Encoder dan Decoder

Encoder terdiri atas $N = 6$ *layer*, di mana setiap *layer* memiliki dua *sublayer*. *Sublayer* pertama adalah *multi-head attention* dan *sublayer* kedua merupakan *feed-forward network*. Setiap *sublayer* dilengkapi dengan *residual connection* dan *normalization layer*, $LayerNorm\big(x + Sublayer(x)\big)$, di mana *x* adalah vektor *embedding* dan $Sublayer(x)$ adalah fungsi yang diimplementasikan sebagai *sublayer*. *Embedding layer* dan *sublayer* memiliki keluaran dengan dimensi $d_{model} = 512$.

Decoder terdiri atas $N = 6$ *layer*. *Decoder* memiliki tiga *sublayer*, di mana dua di antaranya merupakan *sublayer* yang sama dengan *sublayer* yang dimiliki *encoder*, sedangkan satu *sublayer* lainnya merupakan *masked multi-head attention*.

2.6.2 Embedding

Transfomer menggunakan *embedding* untuk mengubah *token* input dan *token output* menjadi vektor dengan dimensi d_{model}. Matriks bobot pada *embedding layer* dikalikan dengan $\sqrt{d_{model}}$.

2.6.3 Positional Encoding

Karena Transformer tidak *recurrent*, agar model bisa mengetahui urutan kata-kata di dalam kalimat, maka harus ada informasi mengenai posisi dari token-token. Untuk melakukan hal tersebut, ditambahkan *positional encoding* kepada *input embedding* di bagian bawah *encoder* dan *decoder*. *Positional encoding* memiliki dimensi yang sama dengan *input embedding*, d_{model}, agar keduanya bisa dijumlahkan [3].

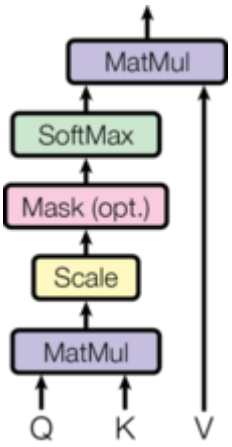
$$PE_{(pos,2i)} = sin\big(pos/10000^{2i/d_{model}}\big)$$

$$PE_{(pos,2i+1)} = cos\big(pos/10000^{2i/d_{model}}\big)$$

Di mana *pos* adalah posisi dan *i* adalah dimensi. Setiap dimensi dari *positional encoding* berkorespondensi terhadap sebuah sinusoid. Panjang gelombang membentuk deret geometri dari 2π sampai $10000 \cdot 2\pi$ [3].

2.6.4 Self-Attention

Sebuah *attention* dapat dideskripsikan sebagai pemetaan suatu *query* dan suatu pasangan *key-value* terhadap suatu *output*, di mana *query* dengan dimensi d_k, *key* dengan dimensi d_k, *value* dengan dimensi d_v, dan *output* merupakan vektor [3].



Gambar 2.2 Scaled Dot-Product Attention

Terdapat matriks weight W^Q, W^K, dan W^V yang akan digunakan untuk memproyeksikan setiap *input* vektor x_i menjadi representasi suatu *key*, *query*, atau *value* [19].

$$q_i = W^Q x_i; k_i = W^K x_i; v_i = W^V x_i;$$

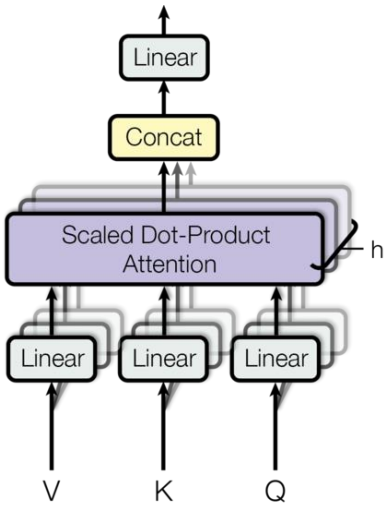
Input x dan vektor-vektor dari berbagai macam *layer* lainnya, semuanya mempunyai dimensi yang sama, yaitu $1 \times d$. Sehingga, dapat diasumsikan bahwa $W^Q \in R^{d \times d}$, $W^K \in R^{d \times d}$, dan $W^V \in R^{d \times d}$ [19].

Dalam praktiknya, fungsi *attention* dihitung pada sekumpulan *query*, *key*, dan *value* secara bersamaan, dikemas dalam bentuk matriks *Q*, *K*, dan *V* [3]. Sehinga, fungsi *attention* didefinisikan sebagai:

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

2.6.5 Multi-Head Attention

Multi-head attention digunakan untuk melakukan *attention* sebanyak *h* kali secara paralel. Setiap *output* dari fungsi *attention* kemudian digabungkan dan diproyeksikan, menghasilkan nilai akhir [3].



Gambar 2.3 Multi-Head Attention

Multi-head attention memungkikan model untuk mengetahui informasi dari berbagai macam subruang pada posisi yang berbeda-beda [3]. *Multi-head attention* didefinisikan sebagai:

$$MultiHead(Q, K, V) = Concat(head_1, \ldots, head_h)$$

Di mana $head_i = Attention\left(QW_i^Q, KW_i^K, VW_i^V\right)$, $W_i^Q \in R^{d_{model} \times d_k}$, $W_i^K \in R^{d_{model} \times d_k}$, $W_i^V \in R^{d_{model} \times d_v}$ dan $W^O \in R^{hd_v \times d_{model}}$.

2.6.6 Feed-Forward Network

Encoder dan *decoder* memiliki *layer* yang berisi *feed-forward network*. *Layer* ini terdiri dari dua transformasi linear dengan fungsi aktivasi *ReLU* di antara keduanya [3].

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

Baik *input* maupun *output*, keduanya memiliki dimensi yang sama $d_{model} = 512$, dan *inner-layer* memiliki dimensi $d_{ff} = 2048$.

2.7 Text-to-Text Transfer Transformer

Text-to-Text Transfer Transformer (T5) adalah model Transformer yang sudah dilatih dengan teknik *transfer learning*. Model T5 dilatih menggunakan *dataset* Colosal Clean Crawled Corpus (C4) untuk melakukan tugas-tugas *natural language processing* (NLP)—peringkasan, tanya jawab, klasifikasi teks, dan lain-lain [20].

T5 menggunakan SentencePiece untuk meng-*encode* teks sebagai *token* [20]. SentencePiece adalah *tokenizer* yang tidak bergantung kepada bahasa tertentu (*language-independent*) dan memecah teks berdasarkan *subword* (suku kata) [21]. Ini memungkinkan kita untuk melakukan tugas-tugas NLP dalam berbagai macam bahasa, seperti mengimplementasikan peringkasan teks abstraktif dalam bahasa Indonesia.

2.8 ROUGE

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) merupakan satuan pengukuran untuk mengukur kesamaan antara ringkasan [14]. Satuan ini membandingkan sebuah ringkasan atau terjemahan yang dihasilkan secara otomatis dengan referensi ringkasan atau terjemahan yang dibuat oleh manusia.

ROUGE-N mengukur jumlah n-gram yang cocok antara teks yang dihasilkan oleh model dengan referensi yang dibuat oleh manusia. ROUGE-N didefinisikan sebagai berikut:

$$ROUGE\text{-}N = \frac{\sum_{S\in\{ReferenceSummaries\}} \sum_{gram_n \in S} Count_{match}\,(gram_n)}{\sum_{S\in\{ReferenceSummaries\}} \sum_{gram_n \in S} Count\,(gram_n)}$$

Di mana *n* adalah panjang n-gram, $gram_n$, dan $Count(gram_n)$ adalah jumlah maksimal n-gram yang terjadi secara bersamaan di dalam kandidat ringkasan dan referensi ringkasan [14]. Pengukuran ROUGE-N biasanya dilakukan dengan menggunakan, ROUGE-1 (*unigram*) untuk pencocokan setiap kata antara kandidat ringkasan dan referensi ringkasan dan, ROUGE-2 (*bigram*) untuk pencocokan setiap dua kata antara kandidat ringkasan dan referensi ringkasan.

Karena judul dan isi ringkasan bisa ditulis dengan berbeda, skor ROUGE sekitar 0.5 dianggap sebagai hasil yang baik [30]. Tabel 2.1 menunjukkan klasifikasi mutu ringkasan berdasarkan skor ROUGE-1 menurut [30].

Tabel 2.1 Klasifikasi Mutu Ringkasan

Mutu	ROUGE-1
A+	> 0.48
A	> 0.45
B	> 0.40
C	> 0.35
F	≤ 0.35

2.9 Python

Python adalah sebuah bahasa pemrograman tingkat tinggi (*high-level programming language*) yang terinterpretasi (*interpreted*) dan berorientasi objek (*object-oriented*) dengan semantik yang dinamis [13]. Python memiliki filosofi desain yang menekankan kode supaya mudah untuk dibaca (*readability*) dan menggunakan *significant indentation*.

Python merupakan bahasa pemrograman yang bersifat *general-purpose* yang berarti, Python bisa digunakan untuk berbagai macam domain pengembangan perangkat lunak dan tidak ditujukan untuk masalah-masalah yang spesifik. Python sering digunakan untuk membangun situs web, otomatisasi tugas, dan analisis data.

2.10 Unified Modelling Language

Unified Modelling Language (UML) adalah standar bahasa pemodelan untuk perangkat lunak dan pengembangan sistem [22]. Model merupakan abstraksi dari sesuatu yang aslinya [22]. Ketika membuat model untuk suatu sistem, detail-

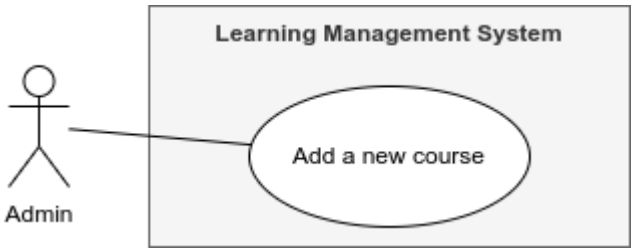
detail yang tidak relevan dihilangkan, sehingga model yang ditunjukkan merupakan gambaran umum yang merepresentasikan sistem aslinya.

2.10.1 Use Case Diagram

Use case adalah sebuah kasus (atau situasi) di mana suatu sistem digunakan untuk memenuhi satu atau lebih kebutuhan pengguna [22]. *Use case* menggambarkan bagian dari fungsionalitas yang disediakan oleh sistem [22]. *Use case diagram* berperan penting sebagai acuan mengenai fungsi atau fitur apa saja yang nantinya akan dimiliki oleh suatu sistem.

Komponen-komponen yang umumnya dipakai di dalam suatu use case diagram yaitu:

1. Aktor yang merupakan entitas yang berinteraksi dengan sistem.
2. *System boundary* yang menjadi lingkup pembatas sistem.
3. *Use case* yang berada di dalam *sistem boundary*.



Gambar 2.4 Contoh Use Case Diagram

Gambar 2.4 menunjukkan *use case diagram* untuk suatu *learning management system* di mana Admin bisa menambahkan kursus (*course*) di dalam sistem tersebut.

2.10.2 Activity Diagram

Activity diagram menggambarkan bagaimana suatu sistem mencapai tujuannya [22]. *Activity diagram* menunjukkan aksi-aksi yang *high-level* yang saling dihubungkan untuk merepresentasikan sebuah proses yang terjadi di dalam suatu sistem [22].

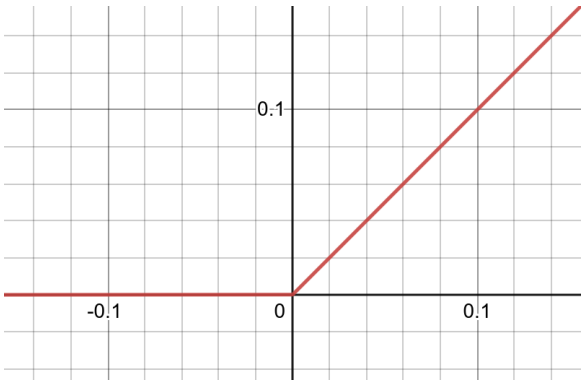
2.10.3 Class Diagram

Class diagram menggambarkan struktur dari sebuah sistem dengan cara menunjukkan kelas-kelas yang dimiliki oleh sistem tersebut. Suatu struktur sistem dibuat dari kumpulan objek [22]. Kelas menjabarkan perbedaan dari objek-objek yang bisa dimiliki oleh suatu sistem, dan *class diagram* (diagram kelas) memperlihatkan kelas-kelas tersebut beserta relasinya [22].

2.10.4 Sequence Diagram

Sequence diagram merupakan *interaction diagram* yang digunakan dalam UML. *Interaction diagram* merupakan model yang penting untuk menunjukkan interaksi-interaksi antara bagian-bagian yang membangun suatu sistem. *Sequence diagram* bertujuan untuk menunjukkan urutan dari interaksi-interaksi yang menjadi bagian dari suatu sistem [22]. *Sequence diagram* bisa menjelaskan suatu interaksi yang akan dipicu ketika suatu *use case* tertentu dieksekusi dan urutan dari interaksi-interaksi yang akan terjadi [22].

2.11 Rectified Linear Unit (ReLU)



Gambar 2.5 Grafik ReLU

Rectified Linear Unit (ReLU) merupakan salah satu fungsi aktivasi yang digunakan pada *deep learning*. ReLU menghasilkan 0 sebagai *output* ketika $x < 0$, dan menghasilkan sebuah linear dengan gradien atau kemiringan bernilai 1 ketika $x > 0$ [23].

$$f(x) = max(0, x)$$

2.12 Softmax

Softmax merupakan fungsi untuk merepresentasikan sebuah distribusi probabilitas dari sebuah variabel diskrit dengan *n* buah nilai-nilai yang mungkin [24]. Softmax biasanya digunakan sebagai *output* dari sebuah *classifier* untuk merepresentasikan distribusi probabilitas dari *n* buah kelas-kelas yang berbeda [24].

$$softmax(z)_i = \frac{exp(z_i)}{\sum_j exp\left(z_j\right)}$$

2.13 Layer Normalization

Layer normalization digunakan untuk melakukan normalisasi dari seluruh input terhadap *neuron-neuron* yang berada di dalam sebuah *layer*. *Layer normalization* menerapkan parameter γ dan β yang bisa dipelajari dengan μ adalah nilai rata-rata, σ adalah standar deviasi, dan ε adalah epsilon.

$$y = \frac{x - \mu}{\sigma + \epsilon}\gamma + \beta$$

2.14 Xavier Uniform

Mengisi input *tensor* dengan nilai berdasarkan *uniform distribution* $U(-a, a)$. fan_in adalah jumlah unit input *neuron* dan fan_out adalah jumlah unit *ouput*.

$$a = \sqrt{\frac{6}{\text{fan\_in} + \text{fan\_out}}}$$

2.15 Cross Entropy Loss

Cross entropy adalah metode yang digunakan untuk mengukur perbedaan antara dua distribusi probabilitas, misalnya *p* dan *q*, di mana *p* adalah probabilitas sesungguhnya (*one-hot*) dan *q* adalah probabilitas yang diperoleh.

$$Loss = -\sum_{i}^{N} p_i log(q_i)$$

2.16 Adam Optimizer

Adam adalah algoritma yang digunakan untuk melakukan optimasi pada *gradient descent*. Secara konsep, Adam menggabungkan dua teknik optimasi yang telah dikembangkan sebelumnya, *stochastic gradient descent* dengan momentum dan RMSProp. Diketahui *hyperparameter* $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, dan $\varepsilon = 10e{-}8$, algoritma Adam dijabarkan sebagai berikut:

$m_0 \leftarrow 0$ (inisialisasi)
$v_0 \leftarrow 0$ (inisialisasi)
$t \leftarrow 0$ (inisialisasi)
while θ_t not converged **do**
  $t \leftarrow t + 1$
  $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
  $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$
  $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$
  $\hat{m}_t \leftarrow m_t/(1 - \beta_1^t)$
  $\hat{v}_t \leftarrow v_t/(1 - \beta_2^t)$
  $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t/(\sqrt{\hat{v}_t} + \varepsilon)$
end while
return θ_t

Di mana:

1. α adalah *learning rate/step size*.
2. $\beta_1, \beta_2 \in [0, 1)$ adalah *decay rate*.
3. $f(\theta)$ adalah fungsi objektif *stochastic* dengan parameter θ.
4. t adalah *timestep*.
5. θ adalah parameter.
6. m, v adalah *moment vector*.

2.17 Penelitian Sebelumnya

Tabel 2.2 menunjukkan beberapa *literature review* yang dilakukan pada penelitian sebelumnya.

Tabel 2.2 Literature Review

***Review* ke-1**

Judul	Implementasi Metode Recurrent Neural Network Pada Text Summarizatio dengan Teknik Abstraktif
Penulis	Kasyfi Ivanedra, Metty Mustikasari
Review	*Dataset* yang digunakan adalah artikel berita sebanyak 4.515 buah artikel. Hasil pengujian menggunakan teknik *non-stemming* menghasilkan *recall* sebesar 41%, *precision* sebesar 81%, dan *F-measure* sebesar 54,27%. Sedangkan, pengujian menggunakan teknik *stemming* menghasilkan *recall* sebesar 44%, *precision* sebesar 88%, dan *F-measure* sebesar 58,20%.
***Review* ke-2**	
Judul	Peringkas Teks Otomatis Bahasa Indonesia secara Abstraktif Menggunakan Metode Long Short-Term Memory
Penulis	M. Alfhi Saputra, W. Fawwaz Al Maki
Review	*Dataset* yang digunakan adalah kumpulan artikel berita media daring Bahasa Indonesia. Pengujian dilakukan dengan tiga skenario. Skenario 1 menggunakan teknik *stemming* dan *stopword removal*, menghasilkan skor ROUGE-1 sebesar 0,08612. Skenario 2 menggunakan teknik *stemming* tanpa *stopword removal*, menghasilkan skor ROUGE-1 sebesar 0,13846. Dan, skenario 3 tidak menggunakan teknik *stemming* ataupun *stopword removal*, menghasilkan skor ROUGE-1 sebesar 0,13825. Berdasarkan data tersebut, *stemming* sedikit mempengaruhi performa, sedangkan *stopword removal* sangat mempengaruhi performa.
***Review* ke-3**	
Judul	Indonesian Abstractive Text Summarization Using Bidirectional

	Gated Recurrent Unit
Penulis	Rike Adelia, Suyanto Suyanto, Untari Novia Wisesty
Review	*Dataset* menggunakan 500 dari 1000 dokumen jurnal Indonesia. Pengujian dilakukan pada dua skenario. Skenario 1 menggunakan model dengan 128 *hidden unit* menghasilkan ROUGE-1 sebesar 0.11975 dan ROUGE-2 sebesar 0.01199, sedangkan skenario 2 menggunakan model dengan 64 *hidden unit* menghasilkan ROUGE-1 sebesar 0.06745 dan ROUGE-2 sebesar 0.00550.
***Review* ke-4**	
Judul	Attention Is All You Need
Penulis	A. Vaswani dkk.
Review	Transformer adalah arsitektur yang menggunakan mekanisme *attention* untuk memperoleh hubungan antar kata di dalam kalimat. Penelitian ini mengimplementasikan model Transformer pada *dataset* WMT 2014 Englishto-German dan memperoleh skor BLEU sebesar 28,4 sedangkan, pada *dataset* WMT 2014 English-to-French, skor BLEU yang dihasilkan adalah sebesar 41,8 yang berhasil melebihi ConvS2S Ensemble yang memiliki skor BLEU sebesar 41,29.
***Review* ke-5**	
Judul	Automated News Summarization Using Transformers
Penulis	A. Gupta, D. Chugh, Anjum, R. Katarya
Review	Mengimplementasikan *pre-trained model* yang berbasis arsitektur Transformer untuk melakukan peringkasan teks abstraktif pada

	dataset berita BBC sebanyak 2225 dokumen. Pada hasil pengujian dalam ROUGE-1, model Pipeline memperoleh skor sebesar 0.47, BART memperoleh skor sebesar 0.40, T5 memperoleh skor sebesar 0.47, dan PEGASUS memperoleh skor sebesar 0.42.
***Review* ke-6**	
Judul	Abstractive Summarization: A Survey of the State of the Art
Penulis	H. Lin dan V. Ng
Review	Melakukan survei mengenai peringkasan teks abstraktif. Pada survei tersebut dijelaskan terkait *evaluation metrics* yang sering dipakai pada peringkasan abstraktif—BLEU, ROGUE, dll. Dijelaskan juga tentang Encoder-Decoder *framework* yang termasuk ke dalam model Seq2Seq yang sering dipakai pada peringkasan abstraktif. *Encoder* mengubah bentuk sumber kalimat menjadi sebuah kumpulan vektor berukuran tetap, masing-masing vektor merepresentasikan kata dan konteksnya. *Decoder* menampilkan hasil ringkasan berdasarkan vektor dari *encoder*. Arsitektur ini dilatih menggunakan data yang berisi pasangan-pasangan *document-summary* untuk memaksimalkan hasil ringkasan.