BAB II

TINJAUAN PUSTAKA DAN LANDASAN TEORI

2.1. Tinjauan Pustaka

Penelitian mengenai *text mining* dalam mengolah data buku maupun berita sudah banyak dilakukan, salah satunya dalam pembuatan kata-kata relevan dalam urutan tertentu berdasarkan kata-kata yang dimasukkan. Pengkajian dilakukan terhadap penelitian-penelitian terdahulu yang relevan dengan penelitian ini. Hal tersebut bermaksud untuk memperoleh sudut pandang peneliti terdahulu terkait data, metode, maupun hasil penelitian.

Penelitian yang dilakukan oleh Sivasurya Santhanam pada tahun 2020 yang berjudul "*Context Based Text Generation using LSTM Networks*" menjelaskan penggunaan algoritma LSTM dalam menghasilkan teks untuk satu set *input* kata yang diberikan bersamaan dengan vektor konteks. Dataset yang digunakan yaitu *domain vector space* berasal dari buku yang berjudul "*The Lord of The Ring*" dan *wiki vector space* yang merupakan kumpulan dari berbagai sumber wikipedia. Hasil evaluasi LSTM menggunakan sumber *wiki vector* dan *domain vector* mencapai performa maksimum pada *training* 20 *epochs* dengan nilai akurasi sekitar 67% hingga 85% [12].

Penelitian lainnya dilakukan oleh Sunanda Das, Sajal Basak Partha, dan Kazi Nasim Imtiaz Hasan, dengan judul "*Sentence Generation using LSTM Based Deep Learning*" pada tahun 2020 membahas mengenai pembuatan model yang dapat menghasilkan kalimat dengan tetap mempertahankan struktur tata bahasa yang tepat menggunakan arsitektur *Long Short-Term Memory*. Menggunakan perbandingan 80:20 untuk data *training* dan data *testing* yang diambil dari dataset "*News Summary*" pada akun *GitHub* sunyysai12345. Pada pelatihan model menggunakan 125-200

epochs, penelitian ini mendapatkan hasil akurasi yang stabil, yaitu sekitar 90% dan *loss* yang dihasilkan berbanding terbalik dengan hasil akurasi [11].

Shayak Chakraborty, Jayanta Banik, dan Shubham Addhya melakukan penelitian mengenai pembuatan model generasi berbasis karakter yang berjudul "*Study of Dependency on Number of LSTM Units for Character Based on Text Generation Models*". Model yang dibangun menggunakan arsitektur LSTM dan *Convolutional* LSTM tersebut memakai dua jenis dataset *self-complied* dari pemrograman bahasa C yang dikumpulkan dari publikasi *open source*. Pada dataset pertama mengandung 100.000 *characters* yang mengacu pada kode yang berisikan nama fungsi dan variable dan dataset kedua memiliki 80.000 *characters* yang mengacu pada token <*var*>. Model dilakukan pelatihan 20 *epochs.* Pada model LSTM untuk dataset pertama mendapatkan hasil akurasi tertinggi 86,3% dan *loss* 0,45 pada unit 96, sedangkan model *Convolutional* LSTM dengan akurasi 81,1% dan *loss* 0,12 pada unit 128. Pada model LSTM untuk dataset kedua mendapatkan hasil akurasi tertinggi 96,7% dan *loss* 0,12 pada unit 48, sedangkan model *Convolutional* LSTM dengan akurasi 96,6% dan *loss* 0,12 pada unit 256 [13].

Penelitian yang berjudul "*Generating Text Through Adversarial Training using Skip-Thought Vectors*" oleh Afroz Ahmad pada tahun 2019, membahas tentang mereproduksi gaya penulisan seorang penulis dalam teks yang dihasilkan model. Dataset yang digunakan berasal dari CMU-SE berisi 44,016 kalimat dengan kosakata sebanyak 3,122 kata. Terdapat 2 macam evaluasi yang dilakukan, berdasarkan akurasi dan penilaian oleh manusia. Model GAN dengan *embedding* Skip-Thought memberikan hasil akurasi lebih tinggi dibandingkan GloVe, namun tidak terlalu signifikan. Sampel kalimat yang dihasilkan model mendapatkan 39,03% suara dari hasil pemungutan suara. Sampel kalimat tersebut ditandai sebagai tulisan yang ditulis oleh penulis [14].

Pada penelitian Dipti Pawade, dkk yang berjudul "*Story Scrambler - Automatic Text Generation using Word Level RNN-LSTM*" membahas mengenai pembaruan cerita baru berdasarkan 2 masukan rangkaian cerita *output* yang dihasilkan. *Output* yang dihasilkan akan dievaluasi oleh 7 orang untuk memeriksa kebenaran tata bahasa, keterkaitan peristiwa, tingkat minat, dan keunikan dalam cerita yang dihasilkan. Hasil yang diperoleh dari dievaluasi oleh manusia dan akurasi adalah sebanyak 63% [15].

Sanidya Mangal, Poorva Joshi dan Rahul Modak melakukan penelitian mengenai pembuatan model generasi berbasis karakter yang berjudul "LSTM vs. GRU vs. *Bidirectional* RNN *for Script Generation*". Dilakukan pembangunan model 1 layer, 2 layer dan 4 layer yang dapat menghasilkan percakapan baru antar karakter serta skenario baru berdasarkan skrip sebelumnya. Model dilakukan pelatihan sebanyak 75 *epochs.* Model LSTM menghasilkan teks 100 karakter paling efisien dengan diikuti oleh GRU dan *Bidirectional* RNN, sementara *loss* terendah didapat oleh *Bidirectional* RNN, LSTM, dan GRU memiliki *loss* tertinggi. Model LSTM membutuhkan waktu paling sedikit untuk pembuatan teks, GRU membutuhkan waktu sedikit lebih lama dan RNN *Bidirectional* membutuhkan waktu tertinggi [16].

Penelitian yang berjudul "*MaskGAN: Better Text Generation Via Filling In The*" oleh William Fedus, et.al pada tahun 2018, membahas tentang model yang menghasilkan teks dengan mengambil sampel kata secara berurutan, dengan setiap kata dikondisikan pada kata sebelumnya. Terdapat 2 macam dataset yang digunakan, yaitu The Penn Treebank dan 100.000 ulasan film yang diambil dari *platform* IMDB. Model dibangun menggunakan arsitektur MaskGAN dan MaskMLE dan dilakukan *training* dengan masing-masing dataset yang berbeda. Output yang dihasilkan berupa 20 kata dan 40 kata. Sampel model MaskGAN yang terdiri dari 20 kata, pada kumpulan dataset IMDB secara signifikan lebih baik (tampak seperti manusia) daripada model MaskMLE dan SeqGAN [17].

Penelitian yang berjudul "*Long Text Generation via Adversarial Training*" oleh jiaxian Guo pada tahun 2018, membangun model LeakGAN, SeqGAN, dan RankGAN yang dapat menghasilkan teks panjang, menengah dan teks pendek pada masing-masing model. Terdapat 3 jenis dataset, yaitu EMNLP2017 WMT4 Dataset untuk teks panjang, COCO Image Captions untuk teks sedang, dan Chinese poems untuk teks pendek. Eksperimen ekstensif pada data sintetik dan berbagai tugas dunia nyata, dengan uji Turing dan skor BLUE menunjukkan bahwa LeakGAN sangat efektif dalam pembuatan teks panjang dan juga meningkatkan kinerja dalam skenario pembuatan teks pendek. Lebih penting lagi, tanpa pengawasan apa pun, LeakGAN akan dapat mempelajari struktur kalimat secara implisit [18].

Tabel 2.1 Penelitian Terdahulu

No	Penulis, Tahun	Judul	Masalah	Algoritma	Hasil	Perbedaan dengan Penelitian Ini
1.	Sivasurya Santhanam (2020) [12]	*Context Based Text Generation using LSTM Networks*	Menghasilkan teks untuk satu set *input* kata yang diberikan bersama dengan vektor konteks.	LSTM	Hasil evaluasi LSTM menggunakan sumber *wiki vector* dan *domain vector* mencapai performa maksimum pada *training* 20 *epochs* dengan akurasi 67% – 85%.	Dataset yang digunakan pada penelitian ini berupa teks novel Bahasa Indonesia, dan input berupa penggalan kata.
2.	Sunanda Das, et.al (2020) [11]	*Sentence Generation using LSTM Deep Learning*	Menghasilkan kalimat dengan tetap mempertahankan struktur tata bahasa yang tepat.	LSTM, *bidirectional* LSTM	Pelatihan model 200 *epochs*, mendapatkan hasil akurasi, yaitu sekitar 90% dan *loss* berbanding terbalik dengan hasil akurasi.	Penelitian ini menggunakan dataset berupa teks novel Bahasa Indonesia, dilakukan *training* model sebanyak 50 epochs dan tidak menggunakan *bidirectional*.

No	Penulis, Tahun	Judul	Masalah	Algoritma	Hasil	Perbedaan dengan Penelitian Ini
3.	Shayak Chakraborty, et.al (2020) [13]	*Study of Dependency on Number of LSTM Units for Character Based on Text Generation Models*	Menghasilkan data teks dengan memprediksi kemungkinan karakter berikutnya dalam teks.	LSTM dan *Convolutional* LSTM	Dengan pelatihan 20 *epochs*. Pada model LSTM dataset pertama akurasi tertinggi mencapai 86,3% dan *loss* 0,45 pada unit 96, sedangkan model ConvLSTM dengan akurasi 81,1% dan *loss* 0,12 pada unit 128. Pada model LSTM dataset kedua akurasi tertinggi didapat 96,7% dan *loss* 0,12 pada unit 48, sedangkan model ConvLSTM dengan akurasi 96,6% dan *loss* 0,12 pada unit 256.	Penelitian ini hanya menggunakan satu macam dataset, yaitu teks novel, dilakukan *training* model sebanyak 50 epochs dan tidak menggunakan arsitektur *Convolutional* LSTM.

No	Penulis, Tahun	Judul	Masalah	Algoritma	Hasil	Perbedaan dengan Penelitian Ini
4.	Afroz Ahmad (2019) [14]	*Generating Text Through Adversarial Training using Skip-Thought Vectors*	Mereproduksi gaya penulisan seorang penulis dalam teks yang dihasilkan model.	Generative Adversarial Network (GAN) dengan vektor Skip-Thought (STGAN)	Model GAN dengan *embedding* Skip-Thought memberikan hasil akurasi lebih tinggi dibandingkan GloVe, namun tidak terlalu signifikan. Sampel kalimat yang dihasilkan model mendapatkan 39,03% suara dari hasil pemungutan suara. Sampel kalimat tersebut ditandai sebagai tulisan yang ditulis oleh penulis.	Arsitektur yang digunakan menggunakan LSTM dan GRU. Dataset berupa teks novel. Penelitian tersebut menggunakan penilaian 14 responden, sedangkan penelitian ini menggunakan penilaian 40 responden.
5.	Dipti Pawade, et.al (2020) [15]	*Story Scrambler - Automatic*	Menghasilkan cerita baru berdasarkan	LSTM	Cerita baru yang terbentuk dari masukan 2 rangkaian cerita yang berbeda	Dataset yang digunakan hanya satu macam, input berupa penggalan kata, dan

No	Penulis, Tahun	Judul	Masalah	Algoritma	Hasil	Perbedaan dengan Penelitian Ini
		Text Generation using Word Level RNN-LSTM	serangkaian cerita yang dimasukkan.		dievaluasi oleh manusia (7 orang) dan diperoleh akurasi 63%.	menggunakan rentang filter 16-256. Penelitian ini menggunakan penilaian 40 responden.
6.	Sandhiya Mangal, et.al (2019) [16]	LSTM vs. GRU vs. *Bidirectional* RNN *for Script Generation*	Pembuatan model yang menghasilkan percakapan baru antar karakter serta skenario baru berdasarkan skrip sebelumnya	LSTM, GRU dan *Bidirectional RNN*	Model LSTM menghasilkan teks 100 karakter paling efisien dengan diikuti oleh GRU dan *Bidirectional* RNN, sementara untuk urutan loss terendah didapat oleh *Bidirectional* RNN, LSTM, dan GRU.	Arsitektur yang digunakan menggunakan LSTM dan GRU 1 layer dan 2 layer. Dataset berupa teks novel. Dilakukan *training* model sebanyak 50 epochs.

No	Penulis, Tahun	Judul	Masalah	Algoritma	Hasil	Perbedaan dengan Penelitian Ini
7.	William Fedus, et.al (2018) [17]	*MaskGAN: Better Text Generation Via Filling In The*	Model yang menghasilkan teks dengan mengambil sampel kata secara urut, dengan melihat kata sebelumnya.	MaskGAN dan MaskMLE	Berdasarkan evaluasi manusia, sampel MaskGAN yang terdiri dari 20 kata, pada kumpulan dataset IMDB secara signifikan lebih baik daripada model MaskMLE.	Arsitektur yang digunakan menggunakan LSTM dan GRU dengan *training* 50 *epochs*. Dataset berupa teks novel. *Output* model berupa 6 kata lanjutan dari *input*.
8.	Jiaxian Gou, et.al (2017) [18]	*Long Text Generation via Adversarial Training*	Model yang menghasilkan teks panjang, menengah dan teks pendek.	LeakGAN, SeqGAN, dan RankGAN	Eksperimen ekstensif pada data sintetik dan berbagai tugas dunia nyata, dengan uji Turing dan skor BLUE menunjukkan bahwa LeakGAN sangat efektif	Arsitektur yang digunakan menggunakan LSTM dan GRU. Dataset yang digunakan hanya teks novel. Penelitian tersebut menggunakan penilaian 62

No	Penulis, Tahun	Judul	Masalah	Algoritma	Hasil	Perbedaan dengan Penelitian Ini
		with Leaked Information			dalam pembuatan teks panjang dan juga meningkatkan kinerja dalam skenario pembuatan teks pendek. Lebih penting lagi, tanpa pengawasan apa pun, LeakGAN akan dapat mempelajari struktur kalimat secara implisit.	responden, sedangkan penelitian ini menggunakan penilaian 40 responden.

Berdasarkan Tabel 2.1, terdapat penelitian yang serupa dengan penelitian ini. Penelitian sebelumnya yang menjadi literatur utama adalah penelitian nomor 2 oleh Sunanda Das, et.al. Meskipun memiliki tujuan yang sama, akan tetapi terdapat perbedaan pada dataset, jumlah *training epoch* dan arsitektur yang digunakan. Pada penelitian Sunanda dataset yang digunakan berupa "*News Summary*" yang diperoleh dari platform *GitHub* milik sunyysai12345, sedangkan penelitian yang akan dilakukan menggunakan dataset teks novel dari platform Z-Library dengan panjang 1.943.055 kata. Selain itu, penelitian tersebut menggunakan arsitektur LSTM dan *bidirectional* LSTM 200 *epochs*, sedangkan penelitian ini menggunakan arsitektur LSTM dan GRU dengan 50 *epochs*.

2.2. Landasan Teori

2.2.1. Novel

Menurut *Kamus Besar Bahasa Indonesia* (KBBI), novel diartikan sebagai karangan prosa yang panjang mengandung rangkaian cerita kehidupan seseorang dengan orang-orang disekelilingnya dengan menonjolkan watak dan sifat setiap pelaku [19]. Novel diambil dari bahasa Latin yaitu *novellus* lalu diturunkan menjadi *noveis* yang artinya baru. Kata 'baru' ini dikaitkan dengan kehadiran novel yang muncul belakangan dibandingkan cerita pendek dan roman [1].

2.2.2. *Natural Language Program* (NLP)

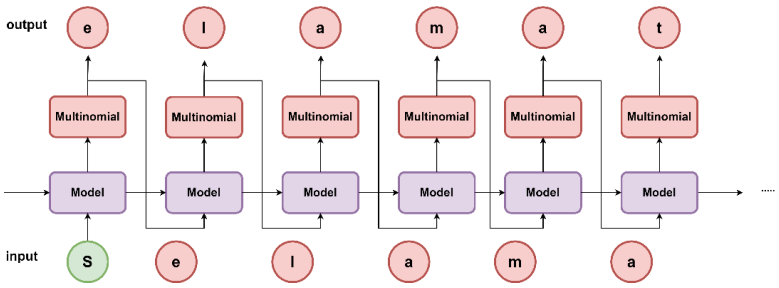
Natural Language Program atau NLP merupakan cabang dari bidang rekayasa perangkat lunak dan *Artificial Intelligent* (AI) yang mengelola bahasa manusia agar dipahami oleh mesin [20]. NLP terdiri dari rangkaian teknik komputasi untuk menganalisis dan merepresentasikan bahasa berupa teks secara otomatis dengan tujuan untuk mencapai pemahaman bahasa manusia dalam berbagai tugas atau aplikasi [21]. Model NLP melakukan pembelajaran vektor fitur

dengan panjang tetap yang mengkodekan sifat semantik dan sintaksis kalimat. Salah satu pendekatan yang populer untuk melatih model kalimat adalah *framework encoder-decoder* menggunakan *Recurrent Neural Network* (RNN) [20].

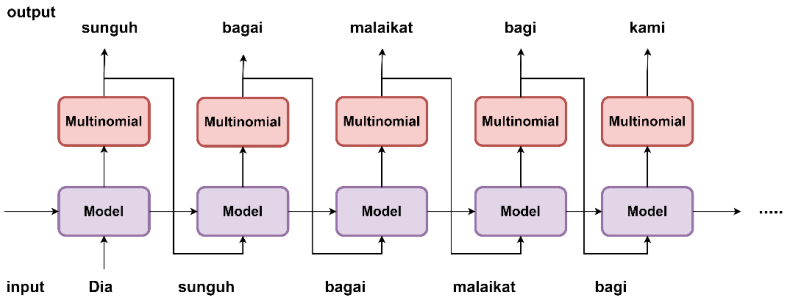
2.2.3. *Recurrent Neural Network* (RNN)

Recurrent Neural Network (RNN) merupakan model yang menggunakan konteks dengan ukuran yang tidak terbatas dan memiliki siklus informasi yang dapat bertahan dengan waktu yang lama. Model ini memberikan generalisasi lebih lanjut di dalam *neuron* dari koneksi berulang yang diasumsikan untuk merepresentasikan memori jangka pendek. Untuk memperkecil jaringan dan mempercepat pelatihan serta pengujian agar mendapatkan akurasi yang lebih baik dapat dilakukan modifikasi pada arsitektur model RNN.

Layer atau lapisan yang dimiliki arsitektur RNN sedikit, namun rumit. Hal tersebut dikarenakan adanya aliran koneksi mundur, sehingga RNN sesuai untuk data sekuens, *timeseries*, suara, dan teks (sekuens kalimat, kata, subkata, atau huruf) [22]. Algoritma RNN mencapai hasil yang mengesankan dalam pembuatan kalimat [23], [24]. Gambar 2.1 dan Gambar 2.2 merupakan gambaran dari proses pemodelan dalam pembuatan teks otomatis menggunakan algoritma RNN.



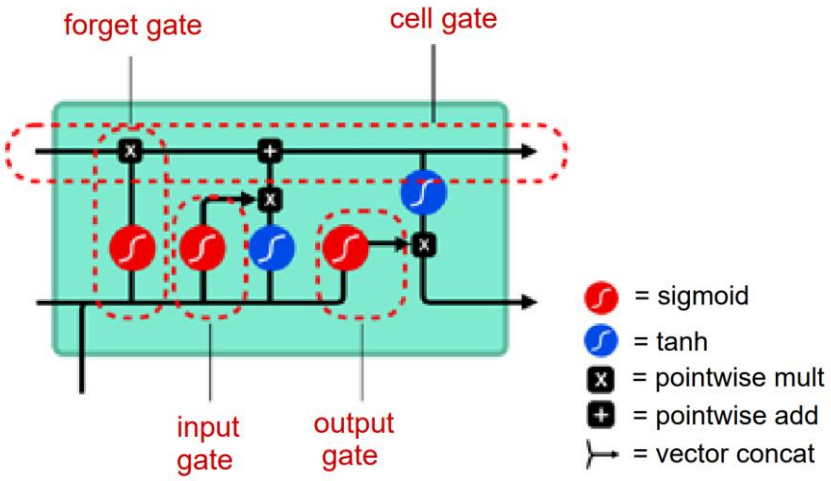
Gambar 2.1 *Generate Text* Berdasarkan Karakter



Gambar 2.2 *Generate Text* Berdasarkan Kata

2.2.4. *Long Short-Term Memory* (LSTM)

Long Short-Term Memory atau LSTM merupakan arsitektur khusus dari RNN yang mampu mengatasi masalah ketergantungan penyimpanan informasi dalam jangka waktu yang lama [25]. Setiap *gate* pada LSTM membantu memprediksi urutan yang lebih baik dari data yang ada berdasarkan token kontekstual [23].



Gambar 2.3 Arsitektur LSTM [26]

Seperti yang ditunjukkan pada Gambar 2.3 di atas, LSTM memiliki satu *memory cell* dengan tiga unit *gate*, diantaranya *input gate*, *forget gate*, *output gate*. *Input gate* berfungsi untuk memutuskan perlu penambahan dan pembaruan *input* saat itu ke dalam memori *cell state* atau tidak. *Forget gate* bertugas untuk menentukan penghapusan memori dari *cell state* diperlukan atau tidak. *Output gate* bertugas menentukan besar pengaruh memori *cell*

state terhadap hasil prediksi yang dikeluarkan [11], [26], [27]. Perhitungan yang terlibat untuk setiap *update* pada LSTM, sebagai berikut :

$$\widetilde{C_t} = \tanh\left(W_f[a_{t-1}, x_t] + b_c\right) \quad \dots (2.1)$$

$$i_t = \sigma(W_i[a_{t-1}, x_t] + b_i) \quad \dots (2.2)$$

$$f_t = \sigma\left(W_f[a_{t-1}, x_t] + b_f\right) \quad \dots (2.3)$$

$$o_t = \sigma(W_o[a_{t-1}, x_t] + b_o) \quad \dots (2.4)$$

$$C_t = (i_t * \widetilde{C_t}) + (f_t * C_{t-1}) \quad \dots (2.5)$$

$$a_t = o_t * C_t \quad \dots (2.6)$$

Keterangan :

$\widetilde{C_t}$ = Informasi penting di dalam *cell state.*

a_t = *output* dari *current unit*

a_{t-1} = *output unit* dari informasi penting

i_t, f_t, o_t = *input gate, forget* gate *dan output* gate

t = *time step*

$W_?$ = *weight* untuk masing-masing *gate*

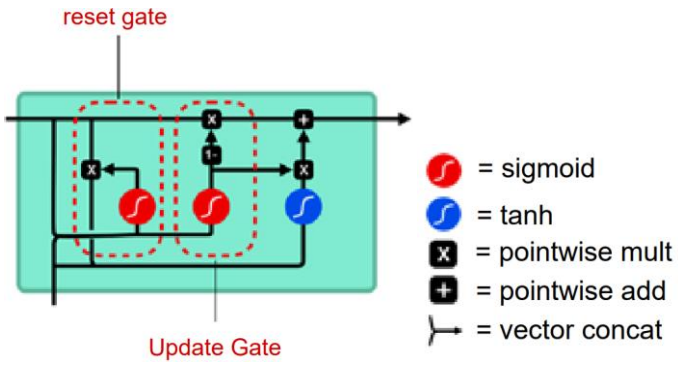
$b_?$ = *bias* untuk setiap operasi pada *weight*

Persamaan (2.1) untuk menentukan informasi penting di dalam *cell state.* Persamaan (2.2) merupakan perhitungan dari *input gate*. Persamaan (2.3) untuk perhitungan *forget gate.* Persamaan (2.4) dan (2.6) untuk mencapai keluaran sel (o*utput gate).* Persamaan (2.5) mencapai pembaruan terakhir dari keadaan sel saat ini [28]. Dilihat dari banyaknya perhitungan yang dilibatkan, *training* pada LSTM akan memakan waktu lebih lama [26].

2.2.5. *Gated Recurrent Unit* (GRU)

Terdapat arsitektur algoritma RNN selain LSTM, yaitu *Gated Recurrent Unit* (GRU). GRU merupakan modifikasi arsitektur LSTM dengan jaringan *gate* yang menghasilkan sinyal pengontrol input saat ini, memori sebelumnya untuk memperbarui aktivasi saat

ini dan status jaringan saat ini [20]. GRU telah banyak digunakan dalam arsitektur *generative* untuk menghasilkan teks [16], [24].



Gambar 2.4 Arsitektur GRU [26]

Berbeda dengan arsitektur LSTM, GRU hanya memiliki 2 *gate*, seperti yang ditunjukkan pada Gambar 2.2 di atas. Pada GRU, *Hidden state* digunakan untuk menyimpan informasi. *Reset gate* digunakan untuk menentukan informasi perlu dilupakan atau tidak. *Update gate* digunakan untuk mengingat informasi [26]. Perhitungan yang terlibat untuk setiap *update* pada GRU, sebagai berikut :

$$\widetilde{C_t} = \tanh\left(W_f[r_t * a_{t-1}, x_t] + b_c\right) \quad (2.7)$$

$$u_t = \sigma(W_u[C_{t-1}, x_t] + b_u) \quad (2.8)$$

$$r_t = \sigma\left(W_f[C_{t-1}, x_t] + b_r\right) \quad (2.9)$$

$$C_t = (u_t * \widetilde{C_t}) + ((1 - u_t) * C_{t-1}) \quad (2.10)$$

$$a_t = C_t \quad (2.11)$$

Keterangan :

$\widetilde{C_t}$ = Informasi penting di dalam *cell state.*

a_t = *output* dari *current unit*

a_{t-1} = *output unit* dari informasi penting

u_t, r_t = *update gate dan reset* gate

t = *time step*

$W_?$ = *weight* untuk masing-masing *gate*

$b_?$ = *bias* untuk setiap operasi pada *weight*

Persamaan (2.7) merupakan perhitungan keseluruhan dari GRU. Persamaan (2.8) untuk perhitungan *update gate*. Persamaan (2.9) untuk perhitungan *reset gate.* Persamaan (2.10) mencapai pembaruan terakhir dari keadaan sel saat ini. Persamaan (2.11) untuk mencapai keluaran sel [28]. Dilihat dari banyaknya perhitungan yang dilibatkan, proses pelatihan LSTM akan memakan waktu lebih lama [26]. Secara teori, pelatihan GRU lebih cepat daripada LSTM, karena perhitungaan yang dilibatkan lebih sedikit.

2.2.6. Loss

Kinerja model diukur dari optimalnya nilai suatu fungsi seperti *loss* dan *error*. *Loss* adalah ukuran seberapa dekat atau berbeda model yang dihasilkan dengan data asli. *Loss* berfungsi untuk menghitung seberapa baik mesin belajar. Tinggi rendahnya nilai dari *loss* tergantung dengan parameter pembelajaran yang digunakan [29]. Terdapat berbagai macam perhitungan *loos,* namun penelitian ini menggunakan *Sparse Categorical Cross Entropy* (SCCE)[30]. Berikut sistematis perhitungan *loss* SCCE[30], [31]:

$$L(\theta) = -\sum_{i=1}^{k} y_i \log(\hat{y}_i) \qquad (2.12)$$

Keterangan :

$L(\theta)$ = *loss*

k = jumlah token dalam kamus teks (kategori/kelas)

y_i = label untuk kelas yang benar

$\hat{y}_i$ = prediksi yang dihasilkan oleh model

2.2.7. Validation Loss

Validasi *loss* merujuk pada nilai *loss* yang dihitung pada data validasi yang tidak digunakan selama pelatihan model. Data validasi digunakan untuk mengukur kinerja model yang tidak terpengaruh oleh pelatihan, sehingga validasi *loss* memberikan gambaran tentang

seberapa baik model tersebut dapat melakukan generalisasi pada data baru yang belum pernah dilihat sebelumnya. Validasi *loss* bertujuan untuk memantau apakah model mengalami *overfitting* atau *underfitting*. *Overfitting* merupakan keadaan ketika model terlalu terbiasa dengan data pelatihan dan tidak bisa generalisasi dengan baik pada data baru. *Underfitting* merupakan keadaan ketika model terlalu sederhana untuk menangkap pola dalam data [32]

2.2.8. Bias Loss

Bias loss merupakan nilai dari perbedaan *train loss* dan *validation loss*. Semakin rendah nilai *bias loss*, maka semakin baik presisi model yang dihasilkan. Namun, pada penelitian ini hasil perhitungan *bias loss* dimutlakkan, sehingga tidal ada nilai negative pada hasil *bias loss* [11].

$$bias\ loss = loss - validation\ loss \tag{2.13}$$

2.2.9. Evaluasi Hasil Secara Manusia

Evaluasi hasil *generate text* secara manusia dilakukan dengan perhitungan kata yang relevan dengan kata masukan pada masing-masing hasil *generate*. Setelah mendapatkan jumlah kata yang relevan dari satu model, dilakukan perhitungan nilai rata-rata. Untuk melihat jauh dekatnya persebaran dari suatu data menggunakan rumus standar deviasi dari. Berikut merupakan sistematis perhitungan rata-rata dan standar deviasi[33]:

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} \tag{2.14}$$

$$std(x) = \sqrt{\frac{\sum_{i=1}^{n} (x_i \cdot \bar{x})^2}{n-1}} \tag{2.15}$$

Keterangan :

$\bar{x}$ = *nilai rata-rata variabel*

$std(x)$ = *loss.*

x_i = nilai valiabel

n = jumlah atau kuantitas dari suatu objek

Ditulis dalam bukunya, Gay dan Diehl mengatakan bahwa semakin banyak sampel yang diambil, semakin *representative* dan tergenelisir hasilnya. Ukuran sampel yang diambil akan sangat bergantung pada jenis penelitiannya. Minimum sampel penelitian korelasional adalah 30 subjek[34]. Oleh karena itu, evaluasi penelitian ini menggunakan responden yang difokuskan kepada guru-guru Sekolah Dasar dalam pengisian kuesioner *google form* sebanyak 40 orang. Penentuan ukuran sampel suatu populasi yang diasumsikan terdistribusi normal dapat dihitung besar *margin error* dengan teorema slovin yang terdapat pada persamaan 2.16[35].

$$n = \frac{N}{1+N\varepsilon^2} \qquad (2.16)$$

Keterangan :

S = jumlah sampel

N = jumlah populasi

Ɛ = *margin error*

Hasil kuesioner tersebut akan dievaluasi rentang kepuasan dengan menggunakan distribusi normal 68-95-99.7 *rule* atau disebut juga aturan empiris[36]. Aturan empiris merupakan aturan pengelompokan sebaran data berdasarkan nilai rata-rata dan standar deviasi[37]. Perhitungan ini mengacu pada nilai *mean* (μ) dan standar deviasi (σ). Secara khusus, aturan empiris memprediksi bahwa dalam distribusi normal[36]–[38] :

1. 68% pengamatan termasuk dalam standar deviasi pertama

$$(\mu - 1\sigma) \leq X \leq (\mu + 1\sigma) \qquad (2.17)$$

2. 95% pengamatan termasuk dalam standar deviasi kedua

$$(\mu - 2\sigma) \leq X \leq (\mu + 2\sigma) \qquad (2.18)$$

3. 99.7% pengamatan termasuk dalam standar deviasi ketiga

$$(\mu - 3\sigma) \leq X \leq (\mu + 3\sigma) \qquad (2.19)$$

2.2.10. Python

Python adalah bahasa pemrograman yang ditujukan untuk *general purpose programming*, bersifat *open source* dan termasuk dalam kategori *high-level programming language*. Bahasa *Python* merupakan bahasa yang paling populer digunakan untuk bidang *Deep Learning, Machine Learning,* dan *Artificial Intelligent*. Tersedianya beragam *library* (modul-modul siap pakai) dapat mendukung untuk menyelesaikan permasalahan pada bidang tersebut. *Library* tersebut diantaranya, *Matplotlib, Numpy, Pandas, Scikit-Learn*, dan lainnya [29].