

BAB II

TINJAUAN PUSTAKA

2.1 Penelitian Terkait

Pada saat ini ada berbagai penelitian yang membahas chatbot dengan metode pendekatan *Natural Language Processing* (NLP), diantaranya yaitu penelitian oleh Mukrodin dan Nurul Mega Sasmita pada tahun 2021. Yang membahas mengenai aplikasi chatbot yang dapat membantu pengguna dalam memperoleh informasi yang tepat dan akurat secara efisien di website obyek wisata Kabupaten Cilacap. Penelitian ini bertujuan untuk penggunaan aplikasi chatbot yang dapat melakukan percakapan seperti kepada manusia dan sistem akan menjawab otomatis dengan kata kunci yang telah disediakan. Penelitian ini menggunakan metode waterfall dalam melakukan teknik analisis data dalam pembuatan aplikasi chatbot pada website diantaranya yaitu *Communication, Planning, Modeling, Construction, Deployment*. Hasil penelitian ini menyimpulkan bahwa aplikasi ini sangat membantu user dalam proses pencarian informasi[5].

Yang kedua yaitu jurnal dari penelitian Jasen Aprian Putra dan Akhmad Budi pada tahun 2020. Yang membahas mengenai Aplikasi Chatbot Sebagai Media Pencarian Informasi dengan Menggunakan React dalam pembuatan buku pedoman tersebut ditujukan bagi mahasiswa dalam mencari informasi yang dibutuhkannya dalam memperoleh informasi dari buku pedoman di Institut Bisnis dan Informatika Kwik Kian Gie. Penelitian ini menggunakan metode waterfall dalam melakukan teknik analisis data dalam pembuatan perangkat lunak, diantaranya yaitu *Requirement Analysis, System design, implementation and testing, Operation dan Maintenance*. Hasil penelitian ini menyimpulkan dengan adanya aplikasi ini dapat memfasilitasi mahasiswa dalam bertanya tanpa harus melibatkan pihak lain, seperti BAAK (Bagian Akademik dan Administrasi Kemahasiswaan) atau Ketua Program Studi[6].

Yang ketiga yaitu jurnal dari penelitian Sri Mulyatun, Hastari Utama dan Ali Mustopa pada tahun 2021. Yang membahas Aplikasi chatbot sebagai media informasi bagi mahasiswa untuk memudahkan pencarian informasi akademik

dengan cara tanya jawab kepada customer service Virtual. Penelitian ini menggunakan metode *waterfall* dalam melakukan teknik analisis data dalam pendekatan *Natural Language Processing* Pada Aplikasi Chatbot Sebagai Alat Bantu *Customer Service*. Hasil penelitian ini menyimpulkan dengan adanya aplikasi ini *Customer Service* dapat dimudahkan dalam menanggapi pertanyaan pengunjung atau pelanggan yang bertanya tanpa harus menjawabnya secara manual[7].

Yang keempat yaitu jurnal dari Eka Yuniar dan Heri Purnomo pada tahun 2019. Yang membahas aplikasi Chatbot berbasis layanan yang didukung oleh sekumpulan aturan dan kecerdasan buatan, yang berinteraksi dengan pengguna melalui antarmuka obrolan. Chatbot ini dapat menjawab pertanyaan yang ditanyakan user selama 24 jam dan tidak terikat waktu. Konten ini dinamakan sebagai “ALITTA” Asisten *Virtual* dari BALITTAS sebagai Pusat Informasi di Balai Penelitian Tanaman Pemanis dan Serat di BALITTAS dirasa diperlukan sebagai salah satu pelayanan terhadap para peneliti baik dari internal maupun eksternal BALITTAS. Penelitian ini menggunakan metode Identifikasi dan Analisis kebutuhan desain, Mendeskripsikan desain arsitektur ALITTA, . Identifikasi *Knowledge base* ALITTA, Pembuatan *Inference Engine* ALITTA [8].

Yang ke lima yaitu jurnal dari Rionaldi Ali, Yuni Puspita Sari, dan Annisa Destaria Alawiyah pada tahun 2021. Yang membahas aplikasi Chatbot Dengan menggunakan metode *Semantic Search*, yang berinteraksi melalui percakapan dalam Aplikasi Darmajaya Academic Chatbot with *Semantic Search* didalamnya terdapat Chatbot yang memudahkan melakukan percakapan untuk memperoleh informasi. Namun, bedanya dengan percakapan internal yang terjadi pada manusia pada fitur Chatbot percakapan digantikan dengan mesin yang sudah diprogram untuk menjawab pertanyaan. Penelitian ini menggunakan metode identifikasi *Semantic Search*, dengan metodologi pengumpulan data, pembangunan perangkat lunak SDLC[9].

2.2 Kecerdasan Buatan

Kecerdasan Buatan (bahasa Inggris: *Artificial Intelligence* atau AI) didefinisikan sebagai kecerdasan yang ditunjukkan oleh suatu entitas buatan. Sistem seperti ini umumnya dianggap komputer. Kecerdasan diciptakan dan dimasukkan ke dalam suatu mesin (komputer) agar dapat melakukan pekerjaan seperti yang dapat dilakukan manusia. Beberapa macam bidang yang menggunakan kecerdasan buatan antara lain sistem pakar, permainan komputer (*games*), logika kabur, jaringan syaraf tiruan dan robotika[10]

Menurut Rich and Knight [1991]. Kecerdasan Buatan (AI) merupakan sebuah studi tentang bagaimana membuat komputer melakukan hal-hal yang pada saat ini dapat dilakukan lebih baik oleh manusia..

Dengan demikian, kecerdasan buatan adalah kecerdasan yang diterapkan pada mesin atau komputer agar mesin atau komputer dapat bertindak seperti manusia, seperti melakukan percakapan dengan manusia, menerjemahkan bahasa, membuat rekomendasi, mendeteksi penyakit tertentu, dan lainnya.[11].

2.3 Neural Network

Neural Network terdiri beberapa lapisan unit komputasi yang saling berhubungan yang disebut neuron. Jaringan menerima sinyal input dan menghitung output melalui serangkaian perhitungan matriks dan transformasi nonlinier. Setiap *Neural Network* terdiri dari lapisan input dan lapisan output, dan satu atau lebih lapisan output layer, masing-masing terdiri dari neuron. Koneksi antara neuron di lapisan yang berbeda membawa beban. Pelatihan jaringan mengacu pada pengaturan bobot sehingga output jaringan cocok dengan variabel target seakurat mungkin. Melatih jaringan saraf dengan menyesuaikan bobot koneksi setara dengan tugas memperkirakan model statistik dengan meminimalkan risiko empiris.

Dalam jaringan syaraf pada *Neural Network* diatur ke dalam kelompok unit yang disebut lapisan. Sebagian besar arsitektur jaringan saraf mengatur lapisan-lapisan ini dalam struktur rantai, dengan setiap lapisan menjadi fungsi dari lapisan yang mendahuluinya. Dalam struktur ini, lapisan pertama yaitu

$$h_t = g_h W_h^T x_t b_h$$

Dimana fungsi aktivasi lapisan tersembunyi g_h adalah fungsi dari matriks bobot tersembunyi W_h , vektor input x_t , dan offset b_h . Ini menghasilkan fungsi aktivasi tersembunyi h_t . Prediksinya adalah hasil penerapan fungsi aktivasi keluaran yang dibobotkan dengan jumlah aktivasi yang diterima pada layer output. Jadi, prediksi hanyalah fungsi dari input, bobot, dan bias.

$$h_t = g_h W_h^T x_t b_h$$

Bobot dan bias menentukan pemetaan dari x_t ke y_t dan akan dipelajari selama pelatihan. Aliran jaringan saraf memperlakukan urutan input sebagai data waktu-invarian dan dengan demikian sebagai agonistik pada fitur inheren yang melekat dari deret waktu. Pada saat waktu tertentu, t , untuk sebuah FNN mengambil input x_t yang memetakannya ke h_t aktif yang tersembunyi. Dalam proses ini diulang pada pengamatan berikutnya, $t + 1$, sedangkan kedua pemetaan tidak berhubungan[12].

2.4 Deep Learning

Deep learning merupakan bagian dari *machine learning* yang menggunakan *Deep Neural Networks* untuk menyelesaikan permasalahan pada domain *machine learning*. *Deep learning* menirukan cara berpikir manusia. *Deep learning* merupakan metode yang memanfaatkan *Artificial Neural Networks* yang berlapis lapis (*multi layer*). *Artificial Neural Networks* ini dibuat mirip dengan otak manusia, di mana neuron-neuron terkoneksi satu sama lain sehingga membentuk sebuah jaringan neuron yang sangat rumit. Jenis – jenis *algoritma deep learning* yaitu *convolutional neural networks* (CNN), *Long short term memory network* (LSTM), *reccurent neural network* (RNN), dan *self organizing maps* (SOM). CNN terdiri dari beberapa lapisan dan biasa digunakan untuk melakukan pemrosesan gambar dan deteksi suatu objek Contoh penerapan.

Pada bidang *deep learning* telah banyak berkembang sehingga lebih mudah digunakan karena banyaknya library dan application program interface (API). Library yang digunakan adalah Tensorflow, yang merupakan antarmuka untuk

mengimplementasikan machine learning atau algoritma machine learning dan untuk mengeksekusi perintah menggunakan informasi atau data milik suatu objek. Tensorflow memiliki fitur untuk menjalankan model pelatihan menggunakan model pelatihan CPU (*Central Processing Unit*) dan GPU (*Graphics Processing Unit*)[13].

2.5 Chatbot

Chatbot adalah program atau aplikasi yang dirancang untuk melakukan percakapan obrolan online dengan manusia. Chatbot menerima input melalui teks, ucapan-ke-teks, atau pemilihan opsi dan memberikan tanggapan dari serangkaian opsi yang telah ditentukan. Ini berbeda dari percakapan biasa, yang memungkinkan pengguna mengobrol dengan perwakilan langsung dengan cara yang sama. Meskipun chatbot dirancang untuk mensimulasikan percakapan manusia, sebenarnya tidak ada manusia di dalam percakapan tersebut. Chatbot dianggap sebagai bentuk kecerdasan buatan (AI) karena mereka "belajar" dengan menjadi lebih akurat saat mereka memperoleh lebih banyak data. Ada dua jenis utama chatbot: *rule-based* dan *contextual*.

Chatbot *rule-based* (juga disebut skrip) adalah bentuk chatbot yang paling sederhana, berfungsi sebagai pohon keputusan dasar dengan pernyataan if/then (jika orang tersebut mengatakan/memilih x, maka respons dengan y). Chatbot paling dasar dalam kategori ini bergantung sepenuhnya pada menu atau tombol, daripada mengizinkan pengguna untuk mengetik permintaan mereka. Pengguna memilih opsi, dan bot menjawab dengan tindakan atau respons yang telah ditentukan sebelumnya, tergantung pada pilihannya. Ini berfungsi seperti IVR telepon, di mana Anda memilih nomor untuk memberi sinyal apa yang ingin Anda lakukan dan diberikan opsi lebih lanjut sampai Anda diarahkan ke informasi atau orang yang benar.

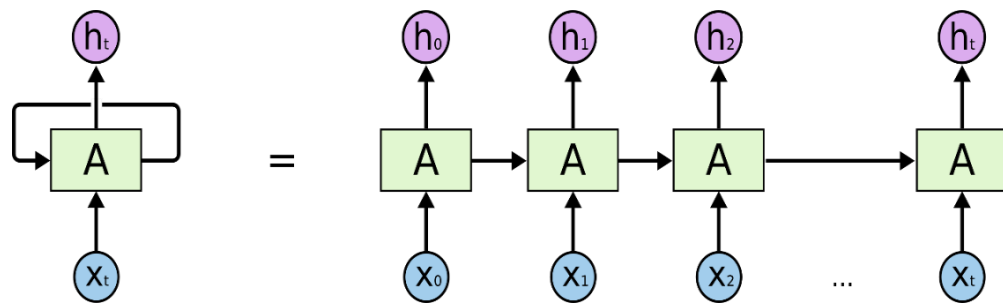
Chatbot kontekstual jauh lebih canggih daripada *rule-based*. Bot ini menggunakan *Natural Language Processing* (NLP) untuk memahami ucapan atau teks pengguna secara akurat. Disebut kontekstual karena dapat memahami konteks percakapan yang sebenarnya dan maksud di baliknya daripada hanya memindai teks untuk kata

kunci, mirip dengan cara asisten virtual seperti Siri atau Alexa berfungsi. Misalnya, jika Anda bertanya “Jam berapa sekarang di London?” itu bisa merespon dengan waktu yang tepat, dan jika Anda mengikutinya dengan bertanya, "Bagaimana dengan Paris?" itu dapat menyimpulkan bahwa Anda masih bertanya tentang waktu dan merespons dengan benar, meskipun waktu tidak termasuk dalam permintaan kedua[14].

2.6 *Reccurent Neural Network*

Reccurent Neural Network atau biasa disebut dengan RNN adalah metode Jaringan saraf tiruan yang berfungsi untuk melakukan proses data sequential atau berkesinambungan. Umumnya, dalam membuat suatu keputusan, manusia akan mempertimbangkan informasi dari sebelumnya atau informasi pada masa lalu. Seperti halnya manusia, informasi dari masa lalu disimpan dengan melakukan perulangan pada arsitekturnya yang akan menjadikan informasi dari masa sebelumnya tetap disimpan.

Dalam pemrosesan data sekuensial, RNN memasukkan loop umpan balik yaitu memberi output neuron sebelumnya sebagai input dari neuron setelahnya. Informasi dari *time step* sebelumnya disimpan sebagai hidden state dan dapat dijadikan input untuk layer selanjutnya. Jika dilihat dari urutan dari *time step* ke *time step*, RNN melakukan salinan jaringan saraf yang mirip rantai, masing-masing *time step* meneruskan informasi ke *time step* penerusnya. Dalam RNN, *training gradient descent* disebut dengan backpropagation melalui waktu, sebagai turunan dari error tidak hanya backpropagation melalui jaringan itu sendiri tapi juga kembali melalui waktu berulang. Struktur pada RNN dapat dilihat pada gambar sebagai berikut.



Gambar 2. 1 Struktur RNN

Tujuan dari training adalah menghitung turunan dari fungsi aktivasi dengan menggunakan aturan rantai sehingga gradien akan semakin mengecil. Dalam masalah RNN, terjadi masalah penghilangan gradien, diperkuat oleh pemrosesan data sekuensial dari jaringan. Terkadang, gradien menghilang tidak hanya melintasi layer tetapi juga melintasi time step. Akibatnya, RNN menghadapi kesulitan dalam menghadapi kesulitan dalam pemodelan dependensi jangka panjang. Perhatikan bahwa RNN dari aturan rantai dalam pelatihan jaringan saraf juga dapat menyebabkan masalah yang terkait erat dengan menghilangnya gradien. Masalah ini disebut ledakan *gradien* dan itu terjadi saat mengalikan bobot secara rekursif matriks dengan beberapa entri di atas satu di jalur mundur pelatihan jaringan.

2.7 Word Embedding

Word Embedding merupakan representasi kata yang dibuat dari teks mentah atau dokumen berdasarkan konteks kalimat. Representasi untuk setiap kata tersebut berupa vektor yang berisi bilangan riil. Salah satu tujuan penggunaan representasi vektor kata yaitu untuk menghitung kemiripan kata dari setiap nilai kata.

Berdasarkan Mkolov dkk, 2013a, arsitektur model untuk menentukan nilai vektor dari suatu kata ada dua, yaitu *continous bag-of-words model* (CBoW) dan *skip gram* (SG). Dari nilai vektor dapat dihitung kemiripan kata maupun kemiripan antara pasangan kata. Dua model vektor kata tersebut dapat dibangun dengan *feedforward neural network* sederhana dan dapat dilihat berdasarkan konteksnya. Perbedaanannya, CBoW dioptimalkan untuk melakukan prediksi kata dari kata – kata di sekitar atau

konteksnya, sedangkan SG lebih dioptimalkan untuk melakukan prediksi konteks dari kata sekarang dan melakukan prediksi kata-kata disekitarnya.

2.8 Long Short-Term Memory

Long short-term memory atau yang disingkat LSTM telah digunakan secara ekstensif dalam *Natural Language Processing* (NLP). Long Short Term Memory (LSTM) merupakan perpanjangan RNN yang membutuhkan input untuk disimpan dalam jangka waktu yang lama. LSTM memiliki memori tingkat lanjut, berbeda dengan memori internal dasar RNN yang mengalami titik hilang. Memori mungkin terjaga keamanannya di LSTM. Ini memiliki tiga gerbang yang disebut sebagai *forget gate*, *output gate* dan *input gate*,

Long Short-Term Memory (LSTM) merupakan salah satu jenis dari arsitektur RNN atau *Reccurent Neural Network* yang berguna untuk menghindari masalah dependensi jangka panjang. Dalam satu neuron, jaringan pada LSTM mempunyai empat lapisan yang saling berinteraksi, yaitu satu lapisan tanh dan tiga lapisan sigmoid. Fungsi aktivasi sigmoid adalah fungsi yang mengembalikan nilai dalam rentang nol dan satu. Sedangkan fungsi aktivasi tanh mengembalikan nilai dalam rentang negatif satu hingga satu.

LSTM memiliki memori tingkat lanjut, seperti bertentangan dengan memori internal dasar RNN yang mengalami titik hilang. Memori itu mungkin terjaga keamanannya di LSTM. Sistem yang menerapkan LSTM dapat memproses, memprediksi, dan mengklasifikasikan informasi berdasarkan data deret waktu. Sesuai dengan konsepnya[15], Untuk keunggulan LSTM, LSTM dapat menyajikan informasi dari riwayat penyimpanan yang telah berlangsung cukup lama dan menghapus data-data lawas yang sudah tidak relevan lagi[16].

Komponen utama dari struktur LSTM adalah *forget gate*, *input gate*, *cell state*, dan *output gate*. Misalkan x adalah input sequence dan h menjadi *hidden state* yang dihasilkan dari LSTM layer, maka hubungan antara x dan h yaitu berdasarkan persamaan sebagai berikut.

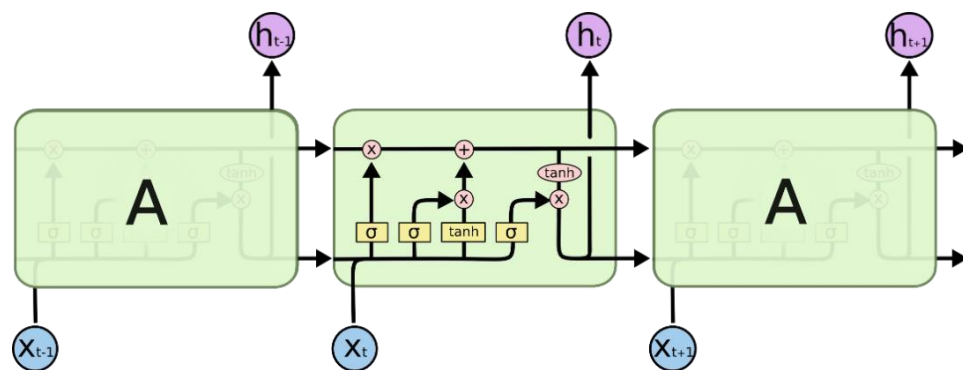
$$f_t = \sigma (W_f \cdot [h_{t-1}, x_t] + b_f) \quad (2.1)$$

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

$$C_t = f_t C_t + i_t \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \quad (2.3)$$

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

$$h_t = o_t \tanh(C_t) \quad (2.5)$$

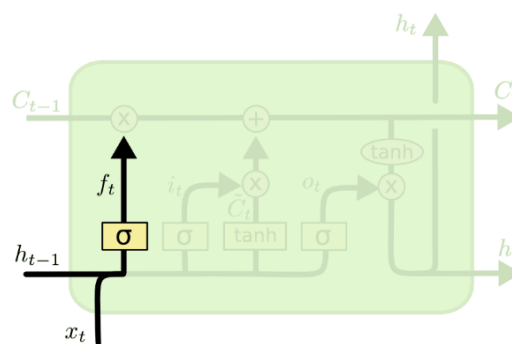


Gambar 2. 2 Jaringan LSTM

Keterangan berdasarkan Gambar 2.2 adalah sebagai berikut.

- Kotak berwarna kuning merupakan *neural network layer* yaitu jaringan saraf yang dipelajari.
- Anak panah tanpa cabang atau garis penggabungan adalah *vector transfer* yaitu garis yang membawa seluruh *vektor*, dari keluaran satu neuron ke masukan lainnya.

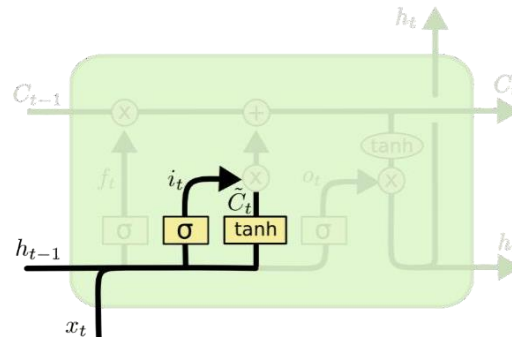
- Lingkaran berwarna merah muda adalah pointwise operation yaitu operasi yang dilakukan seperti penambahan *vektor*.
- Garis yang mengalami penggabungan adalah concatenate yaitu proses penggabungan yang menandakan rangkuman.
- Garis bercabang adalah Copy yaitu proses penyalinan nilai yang kemudian dibawa ke lokasi yang berbeda.
- Notasi x_t menyatakan masukan untuk setiap unit pada *time step* t dan nilainya berupa *vektor*.
- Notasi h_t menyatakan *hidden state* yang dihasilkan oleh setiap unit pada *time step* t dan nilainya berupa *vektor*.
- Notasi C_t menyatakan *cell state* yang dihasilkan oleh setiap unit pada *time step* t dan nilainya berupa *vektor*.



Gambar 2. 3 *Forget Gate* LSTM

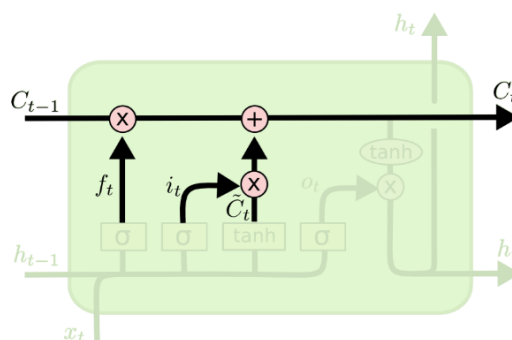
Persamaan gambar 2.3 adalah persamaan untuk *output forget gate*. Persamaan ini berfungsi untuk memutuskan apakah informasi akan disimpan atau dilupakan dari *cell state* C_{t-1} yang merupakan hasil perkalian dengan *cell state*. Keputusan ini dibuat oleh fungsi sigmoid. Fungsi *sigmoid* yang menerima input h_{t-1} dan x_t akan menghasilkan angka antara nol dan satu. Angka 1 artinya *cell state* akan disimpan.

Angka 0 artinya *cell state* akan dilupakan. Proses ini digambarkan dalam gambar 2.3



Gambar 2. 4 Input Gate LSTM

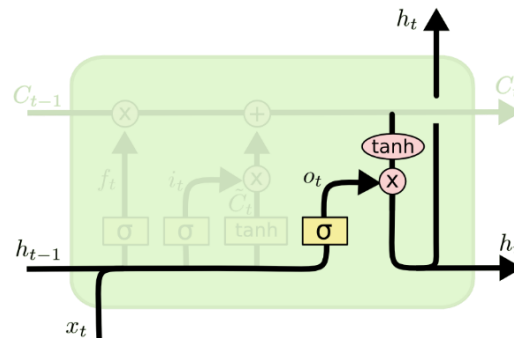
Persamaan 2.4 adalah persamaan gerbang input LSTM. Secara sederhana, persamaan ini menciptakan kandidat baru untuk *cell state*. Input gate memiliki dua bagian. Pertama yaitu *sigmoid* yang akan menentukan nilai yang mana yang akan di perbarui. Selanjutnya yaitu fungsi aktivasi *tanh* yang menciptakan kandidat baru yang ditambahkan dengan *cell state*. Pada langkah selanjutnya akan digabungkan untuk membuat pembaruan *cell state*. Proses ini dapat dilihat pada gambar 2.5.



Gambar 2. 5 Cell State LSTM

Cell state kemudian diperbarui berdasarkan persamaan 2.5 Sederhananya, *cell state* akan menggabungkan *cell state* hasil dari *forget gate* dan kandidat baru hasil dari *input gate*. *Cell state* hasil dari *forget gate* yang memutuskan apakah akan dilupakan atau diteruskan didapat dengan mengalikan f_t dengan *cell state*

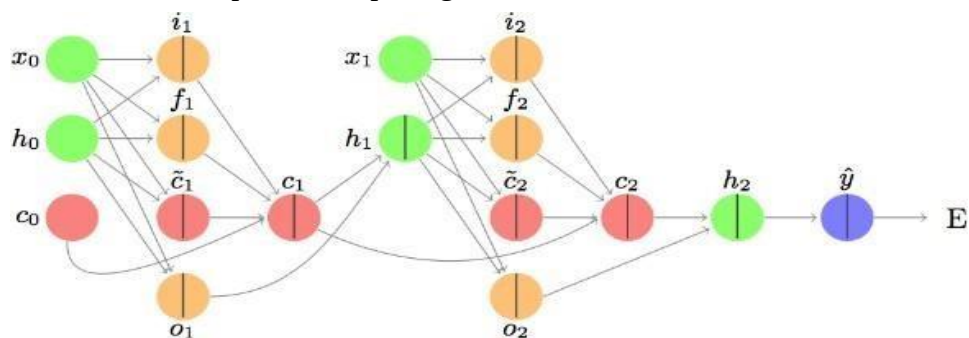
sebelumnya. Calon kandidat baru *cell state* didapat dengan mengalikan i_t dengan C_t . Proses ini dapat dilihat pada gambar 2.5



Gambar 2. 6 Output Gate LSTM

Setelah *cell state* diperbarui, *output gate* dapat dihitung menggunakan persamaan pada Gambar 2.6 untuk menghasilkan *hidden state*. *Hidden state* didasarkan pada *cell state* namun di saring kembali. Pertama, fungsi aktivasi sigmoid akan memutuskan bagian mana dari *cell state* yang akan dijadikan *output*. Kemudian *cell state* dimasukkan dalam fungsi aktivasi tanh yang menghasilkan nilai antara -1 dan 1 kemudian dikalikan dengan hasil dari sigmoid yang akan menghasilkan output.

fungsi aktivasi tanh yang menghasilkan nilai antara -1 dan 1 kemudian dikalikan dengan hasil dari *sigmoid* yang akan menghasilkan *output*. Gambar lain untuk lebih memahami LSTM dapat dilihat pada gambar 2.7[17].



Gambar 2. 7 Ilustrasi LSTM dengan Dua Cell

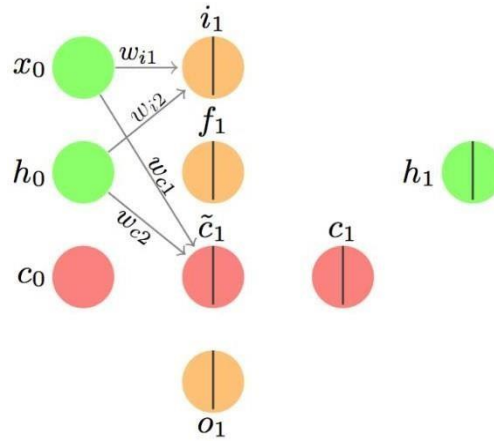
Contoh perhitungan LSTM adalah sebagai berikut.

Misalkan input *sequence* $X_0 = 0.1$, $X_1 = 0.2$, bobot, dan bias adalah sebagai berikut.

$$W = \begin{bmatrix} W_{i1} & W_{i2} & b_i \\ W_{c1} & W_{c2} & b_c \\ W_{f1} & W_{f2} & b_f \\ W_{o1} & W_{o2} & b_o \\ W_y & 0 & b_y \end{bmatrix} = \begin{bmatrix} 0.5 & 0.25 & 0.01 \\ 0.3 & 0.4 & 0.05 \\ 0.03 & 0.06 & 0.002 \\ 0.02 & 0.04 & 0.001 \\ 0.6 & 0 & 0.025 \end{bmatrix}$$

Maka perhitungan pada cell pertama adalah sebagai berikut.

Input gate akan melakukan perhitungan sebagai berikut.



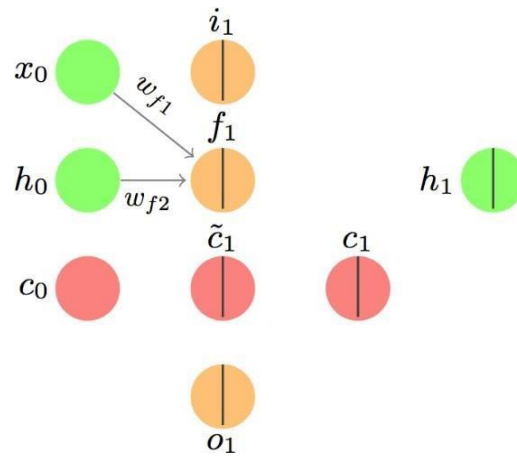
Gambar 2. 8 Perhitungan *Input Gate*

$$\begin{aligned}
 i_1 &= \sigma(W_{i1}X_1 + W_{i2}h_0 + b_i) \\
 &= \sigma(0.5 \times 0.1 + 0.25 \times 0 + 0.01) \\
 &= \sigma(0.06) \\
 &= \frac{1}{1 + e^{-0.06}} \\
 &= 0.515
 \end{aligned}$$

Input kedua yaitu terkait dengan sel memori yang akan membuat suatu *cell state* baru.

$$\begin{aligned}
 \tilde{c}_1 &= \tanh(W_{c1}X_1 + W_{c2}h_0 + b_c) \\
 &= \tanh(0.3 \times 0.1 + 0.4 \times 0 + 0.05) \\
 &= \tanh(0.08) \\
 &= e^{(0.08)} - e^{(-0.08)} \\
 &= e^{(0.08)} + e^{(-0.08)} \\
 &= 0.0798
 \end{aligned}$$

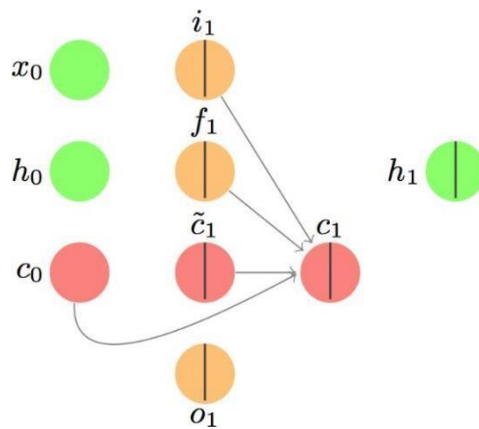
Forget gate akan melakukan perhitungan sebagai berikut.



Gambar 2. 9 Perhitungan *Forget Gate*

$$\begin{aligned}
 f_1 &= \sigma(W_{f1}X_1 + W_{f2}h_0 + b_f) \\
 &= \sigma(0.03 \times 0.1 + 0.06 \times 0 + 0.02) \\
 &= \sigma(0.05) \\
 &= \frac{1}{1 + e^{-0.05}} \\
 &= 0.5012
 \end{aligned}$$

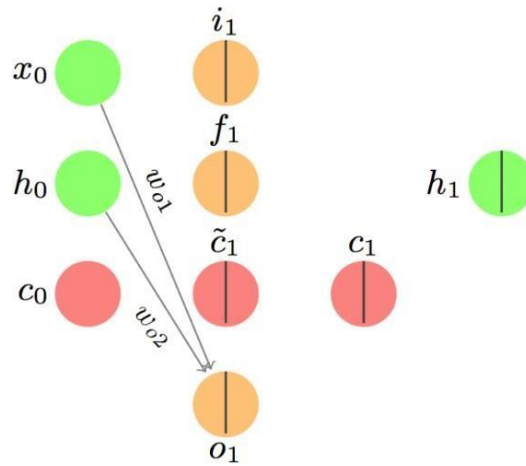
Memory cell akan melakukan perhitungan sebagai berikut.



Gambar 2. 10 Perhitungan *Cell State*

$$\begin{aligned}
 C_1 &= i_t \times C_0 + f_1 \times \tilde{c}_1 \\
 &= 1 \times 0.0798 + 1 \times 0 \\
 &= 0.0798
 \end{aligned}$$

Output gate akan melakukan perhitungan sebagai berikut.



Gambar 2. 11 Perhitungan *Output Gate*

$$\begin{aligned}
 O_1 &= \sigma(W_{o1}X_1 + W_{o2}h_0 + b_o) \\
 &= \sigma(0.02 \times 0.1 + 0.04 \times 0 + 0.001) \\
 &= \sigma(0.003) \\
 &= \frac{1}{1 + e^{-0.003}} \\
 &= 0.5007
 \end{aligned}$$

Hidden state akan melakukan perhitungan sebagai berikut.

$$\begin{aligned}
 h_1 &= O_1 \times \tanh(C_1) \\
 &= 1 \times 0.0796 \\
 &= 0.0796
 \end{aligned}$$

Selanjutnya yaitu perhitungan pada *cell* kedua yaitu sebagai berikut.

Input gate akan melakukan perhitungan sebagai berikut.

$$\begin{aligned} i_2 &= \sigma(W_{i1}X_2 + W_{i2}h_1 + b_i) \\ &= 0.52875 \end{aligned}$$

Input kedua yaitu terkait dengan sel memori yang akan membuat suatu *cell* state baru.

$$\begin{aligned} \tilde{c}_2 &= \tanh(W_{c1}X_2 + W_{c2}h_1 + b_c) \\ &= 0.11768 \end{aligned}$$

Forget gate akan melakukan perhitungan sebagai berikut.

$$\begin{aligned} f_2 &= \sigma(W_{f1}X_2 + W_{f2}h_1 + b_f) \\ &= 0.50231 \end{aligned}$$

Memory cell akan melakukan perhitungan sebagai berikut.

$$\begin{aligned} C_2 &= i_2 \times \tilde{c}_2 + f_2 \times C_1 \\ &= 0.61999 \end{aligned}$$

Output gate akan melakukan perhitungan sebagai berikut.

$$\begin{aligned} O_2 &= \sigma(W_{o1}X_2 + W_{o2}h_1 + b_o) \\ &= 0.50145 \end{aligned}$$

Hidden state akan melakukan perhitungan sebagai berikut.

$$\begin{aligned} h_2 &= O_2 \times \tanh(C_2) \\ &= 0.5511 \end{aligned}$$

Perhitungan akhir untuk mencari prediksi adalah sebagai berikut.

$$\begin{aligned} \hat{y} &= W_y h_2 + b_y \\ \hat{y} &= 0.6 \times 0.511 + 0.025 \\ \hat{y} &= 0.335566 \end{aligned}$$

2.9 Dense Layer

Dalam setiap jaringan saraf, *dense layer* adalah lapisan yang sangat terhubung dengan lapisan sebelumnya yang berarti neuron dari lapisan terhubung ke setiap neuron dari lapisan sebelumnya. Lapisan ini merupakan lapisan yang paling umum digunakan dalam jaringan syaraf tiruan.

Neuron dense layer dalam sebuah model menerima output dari setiap neuron lapisan sebelumnya, di mana *neuron dense layer* melakukan perkalian matriks-vektor. Perkalian vektor matriks adalah prosedur di mana vektor baris dari output dari lapisan sebelumnya sama dengan vektor kolom dari lapisan padat. Aturan umum perkalian matriks-vektor adalah bahwa vektor baris harus memiliki banyak kolom seperti vektor kolom. *Dense layer* sebagai output digunakan untuk melakukan prediksi berdasarkan inputan pada layer sebelumnya.

2.10 Text Mining

Text mining merupakan proses menambang data yang berupa teks dimana sumber data biasanya didapatkan dari dokumen dan tujuannya adalah mencari kata-kata yang dapat mewakili isi dari dokumen sehingga dapat dilakukan analisa keterhubungan antar dokumen. Tujuan dari *text mining* adalah mengekstrak informasi yang berguna dari sumber data. Jadi, sumber data yang digunakan pada *text mining* adalah sekumpulan dokumen yang memiliki format yang tidak terstruktur melalui identifikasi dan eksplorasi pola yang menarik. Adapun tugas khusus *text mining* antara lain, pengkategorisasian teks (*text categorization*) dan pengelompokan teks (*text clustering*). Tahapan *text mining* terdiri dari teks, Pengolahan teks (*tokenisasi*) adalah memecah kalimat menjadi kata per kata, perubahan huruf besar ke huruf kecil (*kapitalisasi*) dan menghilangkan tanda baca, Perubahan teks (*stemming*) adalah perubahan kata berimbuhan menjadi kata dasar, pemilahan teks (*filtering*) adalah melakukan perhitungan dan pengelompokkan kata per kata, Data Mining (*Pattern Discovery*) adalah proses pencarian pengetahuan atau pola yang menarik/bernilai, Evaluasi adalah penafsiran pola yang ditemukan[18].

2.11 *Indexing word*

Karena bahasa adalah bagian *integral* dari kehidupan kita dan masyarakat kita, kitasecara alami dikelilingi oleh banyak teks. Teks tersedia bagi kita dalam bentuk buku, artikel berita, artikel Wikipedia, tweet dan dalam banyak bentuk lainnya dandari banyak sumber yang berbeda.

Jumlah teks yang sangat besar ini mungkin merupakan jumlah data terbesar yang tersedia untuk kami. Banyak wawasan dapat diperoleh dari kata-kata tertulis dan informasi yang diekstraksi dapat menghasilkan hasil yang sangat berguna yang dapat digunakan untuk berbagai aplikasi. Namun, semua data ini memiliki satu kekurangan kecil, yaitu, teks adalah bentuk data yang paling tidak terstruktur yangtersedia bagi kita. Ini adalah bahasa murni tanpa implikasi matematis apa pun. Sayangnya, semua algoritma pembelajaran mesin dan pembelajaran mendalam kami bekerja pada angka dan bukan pada teks. Oleh karena itu, kita perlu mengubahnya menjadi data angka yang dapat kita masukkan ke algoritma sehinggamereka dapat menghasilkan model .

Salah satu metode yang digunakan untuk mengubah *text* menjadi *integer* adalah menggunakan submodule text dari sklearn. *Feature_extraction.Count Vectorizer* adalah kelas dalam modul *sklearn.feature_extraction.text*. Kelas ini berguna untukmembangun kosa kata. Selain itu, menghasilkan representasi *numerik* dari teks yang kita butuhkan, yaitu *vektor Numpy*.

2.12 *Padding*

Semua *neural network* membutuhkan sebuah inputan yang berukuran dan berbentuk sama. Dalam pemrosesan teks sebagai input untuk model seperti dalam LSTM, tidak semua kalimat memiliki panjang yang sama. Secara alami, sebuah kalimat bisa panjang dan bisa pendek. Oleh karena itu kita perlu mengubah input menjadi teks dengan ukuran yang sama. Mengubah ukuran teks menjadi ukuran yang sama disebut dengan padding.

Kita membutuhkan pading karena setiap kalimat dalam teks tidak mempunyai jumlah kata yang sama. Kita dapat mendefinisikan jumlah maksimal kata untuk setiap kalimat. Jika kalimat lebih panjang dari maksimal yang telah ditentukan,

maka sisa kata akan dihapus. Jika kalimat lebih pendek maka kalimat akan diberi penambahan angka 0 di awal atau di akhir kalimat.

2.13 *One Hot Encoding*

One hot encoding adalah salah satu teknik untuk menyeragamkan data kategorik dan numerik. Cara kerja *one hot encoding* adalah dengan membuat sebuah array 1 dimensi dengan panjang sebanyak jenis fitur yang ada dan mempunyai isi biner antara 0 dan 1. *One hot encoding* dapat merepresentasikan data bertipe kategorik menjadi lebih ekspresi. Hal ini berlaku pada penelitian ini karena penelitian ini bekerja dengan jenis klasifikasi urutan teks menggunakan metode *deep learning Long Short-Term Memory (LSTM)*.

One Hot Encoding adalah bagian yang sangat penting dari proses rekayasa fitur dalam pelatihan untuk teknik pembelajaran. Misalnya, variabel seperti warna dan labelnya adalah "merah", "hijau", dan "biru", label dapat dikodekan sebagai vektor biner tiga elemen sebagai Merah: [1, 0, 0], Hijau: [0, 1, 0], dan Biru: [0, 0, 1]. Data kategoris saat memproses, harus dikonversi ke bentuk numerik. *One Hot Encoding* umumnya diterapkan pada representasi integer dari data. Di sini variabel yang dikodekan integer dihapus dan variabel biner baru ditambahkan untuk setiap nilai integer unik. Selama proses, dibutuhkan kolom yang memiliki data kategorikal, yang telah dikodekan label dan kemudian membagi kolom tersebut menjadi beberapa kolom.

2.14 *Optimizer*

Optimizers memiliki beberapa jenis untuk model *deep learning*, diantaranya yaitu SGD, Adam, RMSProp, dan banyak lagi. Penelitian ini menerapkan optimizers Adam untuk pelatihan data. *Optimizers* Adam dapat berguna dalam mengontrol masalah *sparse gradient*. Hal ini yaitu perluasan untuk turunan *gradien stokastik* yang pada saat ini lebih luas dalam aplikasi *deep learning* seperti *natural language processing*. Adam melakukan proses perubahan bobot menggunakan *gradient descent*, yaitu mengubah *learning rate* yang telah ditetapkan secara dinamis sehingga dapat terhindar dari *gradient descent* yang terjebak pada minimum lokal.

Perbandingan performa dengan berbagai jenis optimasi *gradien descent* yang terbaik adalah Adam. Adam lebih mudah dalam implementasi, memori yang dibutuhkan relatif kecil, efisien jika dihitung dalam komputasi. Algoritma adam menghitung rata-rata bergerak eksponensial dari gradien dan gradien kuadrat.

2.15 User Validation

User validation adalah metode validasi dimana user dapat mencoba langsung sistem yang telah dibangun dan kemudian user dapat langsung menilai apakah output yang diberikan oleh sistem sudah sesuai dengan kebutuhan dari user. Akurasi sistem dapat dihitung dengan cara membagi jumlah validasi benar dengan total pertanyaan data uji kemudian dikali dengan 100%.

2.16 Akurasi

Pada penelitian ini menggunakan perhitungan akurasi untuk mengetahui tingkat akurasi dari klasifikasi RNN. Untuk menghitung akurasi dari RNN dilakukan dengan perhitungan rumus berikut.

$$\text{Akurasi (\%)} = \frac{\text{jumlah yang diklasifikasi dengan benar}}{\text{jumlah sampel data uji}} \times 100$$

2.17 Pemodelan Sistem

Permodelan sistem merupakan metode yang memungkinkan untuk memodelkan sistem pada alur pengembangan chatbot. Adapun permodelan yang digunakan pada penelitian tugas akhir ini.

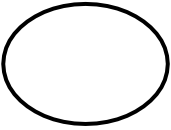
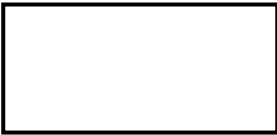
2.17.1 Diagram Konteks

Context Diagram adalah gambaran umum tentang suatu sistem yang terdapat didalam suatu organisasi yang memperlihatkan batasan (*boundary*) sistem, adanya interaksi antara eksternal entity dengan suatu sistem dan informasi secara umum mengalir diantara entity dan sistem. *Context Diagram* merupakan alat bantu yang digunakan dalam menganalisa sistem yang akan dikembangkan. Simbol-simbol yang digunakan di dalam *Context Diagram* hampir sama dengan

simbol-simbol yang ada pada DFD, hanya saja pada *Context Diagram* tidak terdapat simbol file [20].

Tabel 2.1 berikut ini adalah Simbol-simbol yang digunakan dalam Diagram Konteks :

Tabel 2. 1 Simbol Diagram Konteks

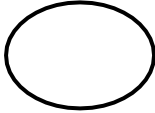
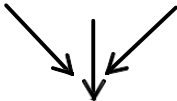
No	Simbol	Keterangan
1	Proses 	Menggambarkan suatu proses atau sistem yang akan dibangun
2	Proses 	Proses dapat digambarkan dengan symbol lingkaran atau persegi panjang dengan sisi-sisi tumpul
3	Entitas/entity 	Menggambarkan entitas atau pengguna dari sistem aplikasi
4	Aliran data (data flow) 	Aliran data yang masuk dan keluar dari sistem.

2.17.2 Data Flow Diagram (DFD)

DFD merupakan gambaran sistem secara logika yang tidak tergantung pada perangkat keras, lunak, struktur data dan organisasi file. Keuntungan dari DFD adalah untuk memudahkan pemakai yang kurang menguasai bidang komputer untuk mengerti sistem yang akan dikerjakan atau dikembangkan [20].

Tabel 2.2 berikut ini adalah Simbol-simbol yang digunakan dalam Data flow Diagram :

Tabel 2. 2 Simbol Data *Flow* Diagram

No	Simbol	Keterangan
1		Proses atau fungsi atau prosedur, pada pemodelan perangkat lunak yang akan diimplementasikan dengan pemrograman terstruktur, maka pemodelan notasi inilah yang seharusnya menjadi fungsi atau prosedur didalam kode program.
2		File atau basis data atau penyimpanan (storage) pada pemodelan perangkat lunak yang akan diimplementasikan dengan pemrograman terstruktur
3		Entitas Luar (<i>external entity</i>) atau masukan (<i>input</i>) atau keluaran (<i>output</i>) atau orang yang memakai/berinteraksi dengan perangkat lunak yang dimodelkan atau sistem lain yang terkait dengan aliran data dari sistem yang dimodelkan.
4		Aliran data merupakan data yang dikirim antar proses, dari penyimpanan keproses, atau dari proses ke masukan (<i>input</i>)

		atau keluaran (<i>output</i>).
--	--	----------------------------------