BAB 2
TINJAUAN PUSTAKA

1.1 Ekstraksi Informasi

Ekstraksi Informasi adalah pencarian otomatis pada informasi yang terstruktur seperti entitas, hubungan antar entitas, dan atribut yang menggambarkan entitas dari sumber yang tidak terstruktur [1]. Pencarian dari suatu ekstraksi informasi merupakan sebuah sistem yang berfungsi untuk mencari data spesifik dalam Natural Language Text. Dalam sebuah template formulir atau tabel serta komponen – komponen teks yang ada diambil datanya untuk diekstraksi menjadi data yang memiliki kegunaan. Ekstraksi informasi berperan sebagai sistem yang akan mengenali data yang tidak terstruktur yang memiliki informasi yang belum dikategorikan dan belum memiliki arti yang spesifik. Tidak terstruktur bukan berarti secara data yang membingungkan atau tidak jelas dengan artian bahwa informasti yang terkandung didalamnya tidak dapat langsung diterjemahkan oleh komputer sehingga membutuhkan sistem yang dinamakan ekstraksi informasi. Ekstraksi informasi dapat digunakan untuk memproses dengan memberikan arti kepada informasi yang tidak terstruktur tersebut, seperti contoh pada data yang terkandung dalam teks, video, gambar dan audio. Pada proses ekstraksi informasi diperlukan suatu proses yang dinamakan preprocessing untuk mengidentifikasikan dan mencirikan suatu pola atau tanda dari isi informasi yang terkandung.

1.2 Kecerdasan Buatan

Kecerdasan buatan atau *Artificial Intelligence* merupakan suatu ilmu yang mempelajari bagaimana membuat komputer melakukan sesuatu pada suatu kejadian atau peristiwa yang mana orang melakukannya dengan baik. Kecerdasan buatan merupakan proses di mana peralatan mekanik dapat melaksanakan kejadian-kejadian dengan menggunakan pemikiran atau kecerdasan seperti manusia [10].

1.3 Text Pre-processing

Text pre-processing adalah bagian penting dari setiap sistem pemrosesan bahasa alami, karena karakter, kata, dan kalimat yang diidentifikasi pada tahap ini adalah unit dasar atau awal sebelum pemrosesan lebih lanjut [11]. Proses ini dilakukan karena data mengandung format yang berbeda-beda seperti angka, simbol, dan kata-kata yang tidak diperlukan dan dapat dihilangkan sehingga memudahkan proses selanjutnya.

1.3.1 Case Folding

Case folding adalah proses penyetaraan huruf untuk semua teks pada dokumen, yaitu dengan mengubah keseluruhan huruf menjadi bentuk yang sama, baik menjadi huruf kapital semua atau huruf kecil semua [12].

1.3.2 Filtering

Filtering adalah proses melakukan penyaringan dan menghilangkan simbol-simbol yang ada di dalam dokumen. Proses *filtering* dilakukan untuk mencegah terjadinya salah pemahaman pada komputer. Kata yang terdapat simbol, di depan, di belakang atau pun di antara huruf-hurufnya, akan membuat kata tersebut dianggap oleh komputer memiliki makna yang berbeda dari yang seharusnya [12].

1.3.3 Tokenization

Tujuan tahap akhir dari teks preprocessing ini adalah untuk proses tokenisasi. Tokenisasi adalah proses memecah aliran konten tekstual menjadi katakata, istilah, simbol, atau beberapa elemen bermakna lainnya yang disebut token [12].

1.4 Labelling

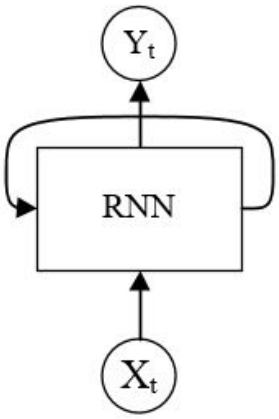
Tagging dalam teks *pre-processing* disini bisa disebut sebagai aplikasi kondisional yang digunakan untuk memberikan tag kondisional barang atau kata atau frasa tertentu dalam sebuah kalimat tergantung dari kata kunci yang terkandung dalam kalimat yang diidentifikasi oleh *sed command* [2].

1.5 Word Embedding

Word embedding adalah nama kolektif dari satu set model bahasa dan teknik pembelajaran fitur dalam pemrosesan bahasa alami dimana kata-kata dari kosa kata dipetakan menjadi vektor bilangan riil. Penelitian yang dilakukan (Yepes,2017) menunjukkan bahwa word embedding meningkatkan performa kinerja fitur dan memungkinkan juga digunakan pada pengklasifikasi RNN dengan basis LSTM [13].

1.6 Recurrent Neural Network

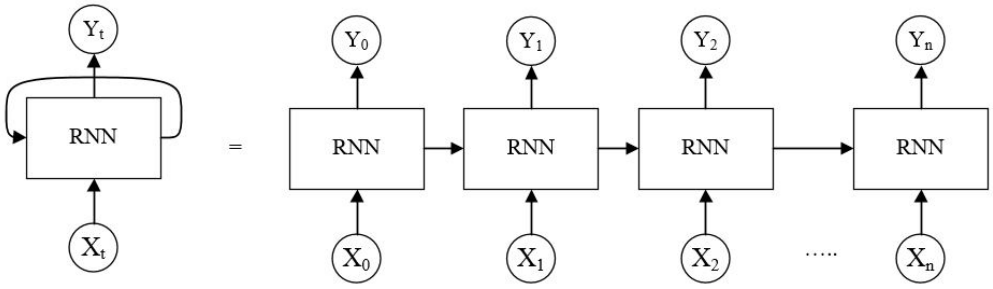
Algoritma *recurrent neural network* merupakan salah satu kategori dari deep learning. Recurrent Neural Network merupakan model yang meniru cara berpikir dalam pengambilan keputusan tersebut dimana RNN tidak membuang begitu saja informasi dari masa lalu dalam proses pembelajarannya yang secara otomatis informasi dari masa lalu tetap tersimpan. RNN memproses input secara sekuensial. Data sekuensial mempunyai karakteristik dimana sampel diproses dengan suatu urutan, dan suatu sampel dalam urutan mempunyai hubungan yang erat satu dengan yang lain [14]. Arsitektur Recurrent Neural Network dapat dilihat pada Gambar 2.1 Recurrent Neural Network.



Gambar 2.1 *Recurrent Neural Network*

Dimana Xt merupakan inputan terhadap t (waktu/urutan inputan berdasarkan waktu/inputan data ke-t) Yt merupakan hasil output. Dari gambar diatas bahwa

proses yang dilakukan pada RNN itu secara berulang-ulang sehingga data inputan sebelumnya tersimpan ke memori RNN. Unfolding Recurrent Neural Network dapat dilihat pada Gambar 2.2 Unfolding Recurrent Neural Network.



Gambar 2.2 ***Unfolding Recurrent Neural Network***

RNN merupakan bagian dari Neural network sehingga lapisan-lapisan dari RNN dibagi menjadi 3 bagian yaitu :

1. Input Layer

   Merupakan lapisan yang menerima input kemudian meneruskan ke neuron lain dalam jaringan.

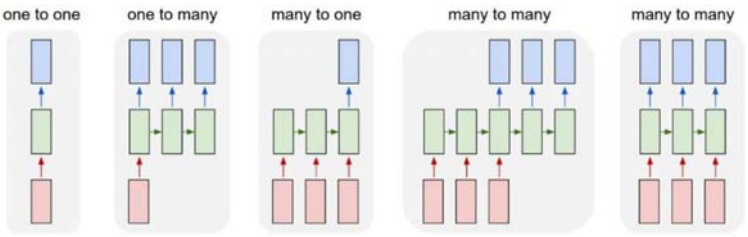
2. Hidden Layer

   Merupakan lapisan yang tersembunyi berfungsi untuk meningkatkan kemampuan jaringan dalam memecahkan masalah.

3. Output Layer

   Merupakan lapisan yang meghasilkan output dari hasil pemrosesan.

Pada Gambar 2.2 dapat dilihat bahwa RNN akan memproses data input satu persatu secara sekuensial, hidden layer pun akan melempar data menuju ke hidden layer lainnya pada skala waktu selanjutnya. Begitu seterusnya secara sekuensial. Dibawah ini merupakan macam-macam pemrosesan pada RNN ditujukan pada gambar 2.3 Pemrosesan RNN.



Gambar 2.3 Pemrosesan RNN

Berikut adalah penjelasan dari gambar 2.3 sebagai berikut :

1. Satu ke Satu

   Pada jenis ini, model memproses satu input dan output yang dihasilkan juga satu.
2. Satu ke banyak

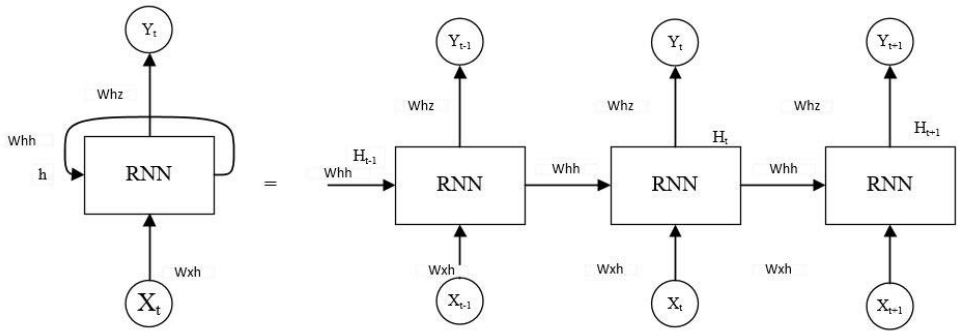
   Pada jenis pemrosesan ini, inputannya hanya satu namun dapat menghasilkan banyak output.
3. Banyak ke Satu

   Pada jenis pemrosesan ini, model memproses banyak input namun output yang dihasilkan hanya satu.
4. Banyak ke banyak

   Pada jenis ini, input dan ouputnya adalah data sekuensial yang jumlahnya banyak. pada jenis ini output yang dihasilkan tidak harus sama dengan jumlah input, dan model ini akan menghasilkan output hanya setelah semua input selesai diproses.

Proses pelatihan pada RNN sama seperti proses pelatihan pada *neural network* pada umumnya. Terdapat tiga langkah utama proses pelatihan pada RNN. Pertama, melakukan proses forward pass dan membuat prediksi. Pada proses ini, dilakukan perhitungan untuk setiap *hidden state* (h_t) berdasarkan setiap masukan (x_t) dan bobot yang telah ditentukan. Setelah ditemukan nilai dari *hidden state*, maka selanjutkan dilakukan perhitungan untuk keluaran atau hasil prediksinya (z_t). Langkah kedua, membandingkan hasil prediksi (z_t) dengan nilai keluaran yang sebenarnya atau disebut juga target, menggunakan *Loss Function*. *Loss Function* menghasilkan nilai kesalahan yang dapat menunjukkan apakah hasil prediksi sesuai

target atau bahkan jauh dari target sehingga dapat memberi kesimpulan seberapa baik atau buruk kinerja RNN tersebut. Langkah yang terakhir, dari nilai kesalahan yang dihasilkan oleh *Loss Function* selanjutnya dilakukan proses *Backpropagation Through Time* (BPTT) untuk menghitung gradien untuk setiap langkah waktu dalam jaringan. Proses BPTT dilakukan untuk mencari bobot-bobot dan juga bias yang lebih baik dari proses sebelumnya. Setelah proses BPTT selesai dilakukan, maka dilakukan pembaruan bobot dan juga bias dengan metode *Stochastic Gradient Descent* (SGD). Berikut adalah struktur RNN dengan varibel yang diperlukan untuk melakukan proses perhitungan, dapat diihat pada gambar 2.4 Struktur RNN.



Gambar 2.4 Struktur RNN

Rumus-rumus untuk perhitungan yang terjadi dalam proses pelatihan RNN dapat dilihat pada rumus (2.1), (2.2), (2.3) dan (2.4), yaitu sebagai berikut [].

1. Perhitungan nilai *hidden state* untuk waktu t, digunakan rumus (2.1) dengan sebuah fungsi aktivasi tanh yang memiliki rumus (2.2).

$$h_t = tanh(W_{hh} \cdot h_{t-1} + W_{xh} \cdot x_t + b_h) \quad \textbf{(2.1)}$$

$$tanh(x) = \frac{sinh(x)}{cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{e^{2x} - 1}{e^{2x} + 1} \quad \textbf{(2.2)}$$

Keterangan :

a) h_t adalah nilai hidden state pada langkah waktu t. Ini adalah memori dari jaringan. Nilai h_t dihitung berdasarkan hidden state sebelumnya dan masukan pada langkah atau tahap yang sedang berlangsung.

b) Fungsi *tanh* adalah fungsi aktivasi non-linear, selain *tanh* dapat juga digunakan fungsi aktivasi non-linear yang lain seperti ReLU.
c) W_{hh} adalah bobot yang digunakan untuk langkah waktu sebelumnya, berupa nilai random antara 0 sampai dengan 1.
d) h_{t-1} adalah nilai hidden state pada langkah waktu sebelumnya, biasanya diinisialisasi ke semua nol.
e) W_{xh} adalah bobot yang digunakan untuk nilai masukan, berupa nilai random antara 0 sampai dengan 1.
f) x_t adalah nilai masukan.
g) b_h adalah nilai bias, berupa nilai random antara 0 sampai dengan 1.

2. Untuk menghitung hasil keluaran sebagai prediksi, maka digunakan rumus (2.3) dengan fungsi *softmax* untuk menghitung probabilitas setiap label yang ditebaknya. Kelebihan dari fungsi aktifasi ini adalah nilai keluarannya berupa nilai dengan rentang probabilitas 0 sampai 1 dan apabila setiap hasil fungsi *softmax* dijumlahkan maka bernilai 1.

$$z_t = softmax(W_{hz} \cdot h_t + b_h) = \frac{exp(W_{hz} \cdot h_t + b_h)}{\sum exp(W_{hz} \cdot h_t + b_h)} \quad \textbf{(2.3)}$$

Keterangan :

a) z_t adalah keluaran yang dihasilkan setelah proses pada langkah waktu t.
b) *softmax* adalah nilai eksponensial yang dinormalisasi.
c) W_{zh} adalah bobot yang digunakan untuk menghasilkan nilai keluaran, berupa nilai random antara 0 sampai dengan 1.
d) h_t adalah nilai hidden state pada langkah waktu t.
e) b_z adalah nilai bias, berupa nilai random antara 0 sampai dengan 1.

3. Untuk membandingkan hasil prediksi atau nilai keluaran yang telah dihasilkan dengan nilai target atau nilai yang sebenarnya, maka digunakan sebuah rumus *Cost Function* (2.4). Rumus *Cost Function* yang digunakan pada penelitian ini adalah *Mean Squared Error* (MSE). MSE adalah salah satu fungi untuk mengetahui seberapa besar nilai *error* yang dihasilkan oleh sebuah *machine learning* setelah melalui proses pelatihan.

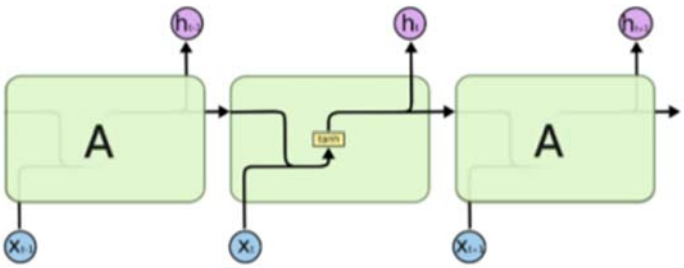
$$E = \frac{1}{n}\sum_{i=1}^{n}(y_i - z_i)^2 \quad \textbf{(2.4)}$$

Keterangan :

a) *E* adalah fungsi error atau fungsi cost MSE
b) *y* adalah nilai vektor target
c) *z* adalah nilai hasil keluaran dari perhitungan
d) *n* adalah jumlah kelas

1.7 Long Short Term Memory

Long Short Term Memory (LSTM) adalah salah satu variasi dari Recurrent Neural Network yang dibuat untuk menghindari masalah ketergantungan jangka Panjang pada Recurrent Nueral Network (RNN). LSTM dapat mengingat informasi jangka panjang [14]. Pada RNN perulangan jaringan hanya menggunakan satu layer sederhana, yaitu layer tanh seperti pada gambar 2.5 Layer *tanh* pada RNN.



Gambar 2.5 Layer* tanh *pada RNN

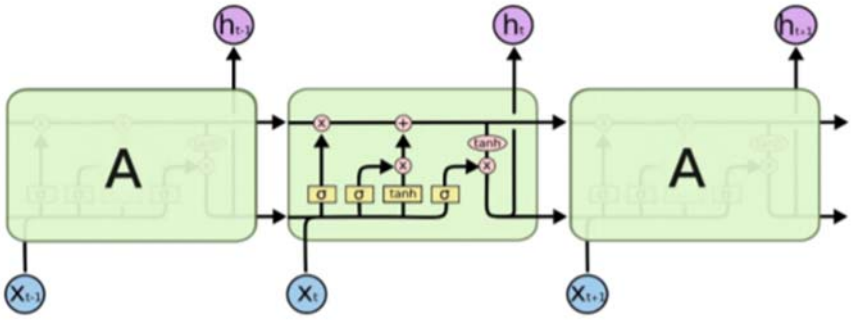
Persamaan tanh diuraikan pada persamaan 2.5.

$$tanh(x) = 2\sigma(2x) - 1 \quad \textbf{(2.5)}$$

Keterangan :

a. σ merupakan fungsi aktivasi sigmoid
b. x merupakan data input Sedangkan,

LSTM memiliki empat layer pada perulangan modelnya seperti pada gambar 2.6 Perulangan dengan empat layer LSTM.



Gambar 2.6 Perulangan dengan empat layer *LSTM*

Persamaan metode LSTM menurut Hochreiter & Schmidhber (1997) diuraikan pada persamaan 2.6

$$ft = \sigma(Wf.\,[ht{-}1, xt] + bf \quad \textbf{(2.6)}$$

$$it = \sigma(Wi.\,[ht{-}1, xt] + bi)$$

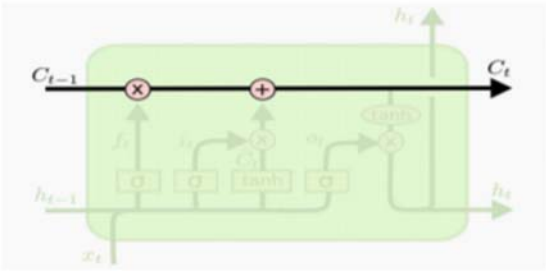
$$\bar{C} = tanh(WC.\,[ht{-}1, xt] + bC)$$

$$Ct = ft * Ct{-}1 + it * \bar{C})$$

$$ot = \sigma(Wo.\,[ht{-}1, xt] + bo)$$

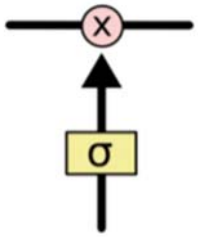
$$ht = ot * \tanh\,(Ct)$$

Kunci utama pada LSTM adalah cell state. Cell state adalah garis horizontal yang menghubungkan semua output layer pada LSTM seperti terlihat pada gambar 2.7 Cell State pada LSTM.



Gambar 2.7 Cell State pada LSTM

LSTM memiliki kemampuan untuk menambah dan menghapus informasi dari cell state. Kemampuan ini disebut dengan gates. Gates sebagai pengatur apakah informasi akan diteruskan atau diberhentikan. Gates terdiri dari sigmoid layer dan pointwise multiplication operation seperti yang terlihat pada Gambar 2.8 Sigmoid Layer pada LSTM.



Gambar 2.8 Sigmoid Layer pada LSTM

Output dari sigmoid layer adalah angka 1 atau 0 yang menunjukkan apakah informasi tersebut akan diteruskan atau diberhentikan. Angka 0 menunjukkan bahwa tidak ada informasi yang akan diteruskan, sedangkan angka 1 menunjukkan bahwa semua informasi akan diteruskan. Persamaan sigmoid diuraikan pada persamaan 2.7.

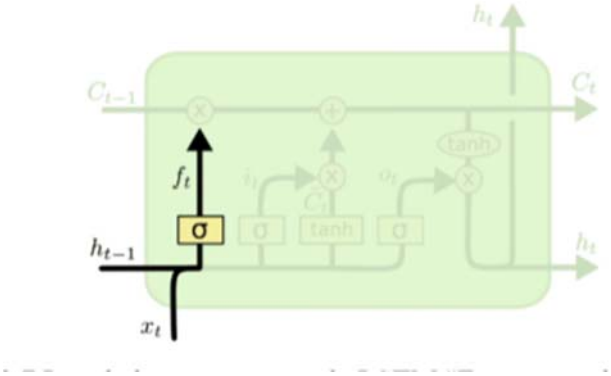
$$(x) = 1/(1 + \epsilon - x) \quad \textbf{(2.7)}$$

Keterangan :

a) x merupakan data input
b) ϵ merupakan konstanta matematika (2,71828 18284 59045 23536 02874 71352)

LSTM memiliki 3 jenis gates diantaranya adalah forget gate, input gate, dan output gate. Forget gate adalah gate yang memutuskan informasi mana yang akan dihapus dari cell. Input gate adalah gate yang memutuskan nilai dari input untuk di diperbarui pada state memori. Output gate adalah gate yang memutuskan apa yang akan dihasilkan output sesuai dengan input dan memori pada cell. Langkah - langkah panduan jalannya metode LSTM:

Langkah pertama adalah LSTM memutuskan informasi apa yang akan dihapus dari cell state. Keputusan ini dibuat oleh sigmoid layer yang bernama "forget gate layer". Forget gate layer akan memproses ht-1 dan xt sebagai input, dan menghasilkan output berupa angka 0 atau 1 pada cell state Ct-1 seperti yang terlihat pada Gambar 2.9 Forget Layer Gate LSTM.



Gambar 2.9 Forget Layer Gate LSTM

Persamaan Forget Gate dapat dilihat pada Persamaan 2.8

$$f_0 = \sigma\left(W_f \cdot x_0 + U_f \cdot out_{-1} + b_f\right) \quad \textbf{(2.8)}$$

Keterangan :

a) ft merupakan forget gate
b) σ merupakan fungsi sigmoid
c) Wf merupakan nilai weight untuk forget gate
d) h_{t-1} merupakan nilai output sebelum orde ke t
e) x_t merupakan nilai input pada orde ke t
f) b_f merupakan nilai bias pada forget gate

Nilai weight diuraikan pada persamaan 2.9.

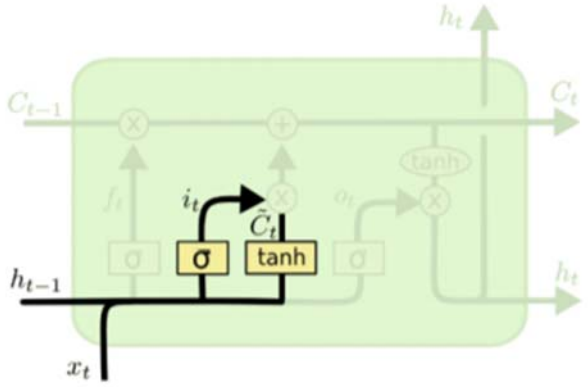
$$W = (-1\ \sqrt{n}\ ,\ 1\ \sqrt{n}) \quad \textbf{(2.9)}$$

Keterangan :

a) W merupakan bobot/weight

b) *n* merupakan jumlah data

Langkah kedua adalah memutuskan informasi apa yang akan disimpan di cell state. Untuk langkah ini terdapat dua bagian. Bagian pertama, sigmoid layer yang bernama "input gate layer" memutuskan nilai mana yang akan diperbaruhi. Selanjutnya, tanh layer membuat satu kandidat dengan nilai baru, C ~ t, yang dapat ditambahkan ke cell state. Tahap selanjutnya adalah output dari input gate layer dan tanh layer akan digabungkan untuk memperbaruhi cell state. Langkah kedua digambarkan pada Gambar 2.10 Input Gate Layer & tanh Layer.



Gambar 2.10 Input Gate Layer & tanh Layer

Persamaan Input gate diuraikan menjadi Persamaan 2.10

$$it = \sigma(Wi.\ [ht{-}1,\ xt] + bi) \tag{2.10}$$

Keterangan :

a) it merupakan input gate
b) σ merupakan fungsi sigmoid
c) Wi merupakan nilai weight untuk input gate
d) h_{t-1} merupakan nilai output sebelum orde ke t
e) x_t merupakan nilai input pada orde ke t
f) bi merupakan nilai bias pada input gate

Persamaaan Kandidat baru diuraikan pada persamaan 2.11.

$$\bar{C} = tanh(WC.\,[ht{-}1, xt] + bC) \quad \textbf{(2.11)}$$

Keterangan :

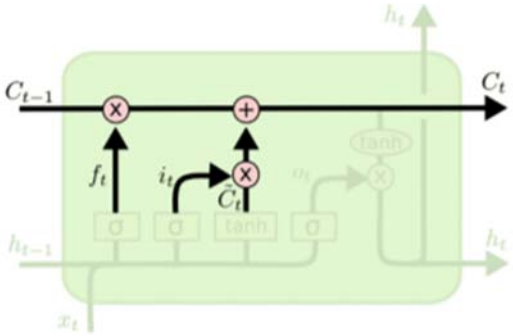
a) $\bar{C}$ = nilai baru yang dapat ditambahkan ke cell state
b) $tanh$ = fungsi tanh
c) WC = nilai weight untuk cell state
d) $ht{-}1$ = nilai output sebelum orde ke t
e) xt = nilai input pada orde ke t
f) bf = nilai bias untuk cell state

Langkah ketiga adalah memperbaruhi cell state yang lama, Ct-1, menjadi cell state baru, Ct seperti digambarkan pada gambar 2.9. Dengan mengkalikan state lama dengan ft, untuk menghapus informasi yang sudah ditentukan sebelumnya pada langkah forget gate layer. Selanjutnya, ditambahkan dengan it * C ~ t , yang merupakan nilai baru dan digunakan untuk memperbaruhi state. Persamaan cell state diuraikan pada persamaan 2.12.

$$Ct = ft * Ct{-}1 + it * \bar{C}) \quad \textbf{(2.12)}$$

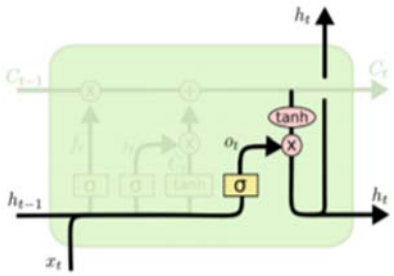
Keterangan :

a) Ct merupakan Cell state
b) ft meruupakan forget gate
c) $Ct{-}1$ merupakan Cell state sebelum orde ke t
d) it merupakan input gate
e) $\bar{C}$ merupakan nilai baru yang dapat ditambahkan ke cell state



Gambar 2.11 Membuat Cell State Baru

Langkah keempat adalah langkah terakhir dalam metode LSTM yang bertujuan untuk memutuskan hasil output digambarkan pada gambar 2.11. Output harus sesuai dengan cell state yang telah diproses terlebih dahulu. Pertama sigmoid layer memutuskan bagian dari cell state yang menjadi output. Selanjutnya, output dari cell state dimasukkan kedalam tanh layer (untuk mengganti nilai menjadi diantara -1 dan 1) dan dikalikan dengan sigmoid gate, agar output yang dihasilkan sesuai dengan apa yang kita putuskan sebelumnya.



Gambar 2.12 Menentukan Output

Persamaan output gate diuraikan pada Persamaan 2.13.

$$ot = \sigma(Wo.\,[ht{-}1, xt] + bo) \qquad \textbf{(2.13)}$$

Keterangan :

a) ot merupakan output gate
b) σ merupakan fungsi sigmoid
c) Wo merupakan nilai weight untuk output gate
d) $ht-1$ merupakan nilai output sebelum orde ke t
e) xt merupakan nilai input pada orde ke t
f) bo merupakan nilai bias pada output gate

Persamaan nilai output orde t diuraikan pada persamaan 2.14.

$$ht = ot * \tanh(Ct) \quad \textbf{(2.14)}$$

Keterangan :

a) ht merupakan nilai output orde t
b) ot merupakan output gate
c) $tanh$ merupakan fungsi tanh
d) Ct merupakan Cell state

Dalam proses *Backward Propagation*, terdapat beberapa komponen antara lain :

$$\delta out_t = \Delta_t + \Delta out_t$$
$$\delta state_t = \delta out \odot a_t \odot \left(1 - \tanh^2(state_t)\right) + \delta state_{t+1} \odot f_{t+1}$$
$$\delta a_t = \delta state_t \odot i_t \odot (1 - a_t^2)$$
$$\delta i_t = \delta state_t \odot a_t \odot i_t \odot (1 - i_t)$$
$$\delta f_t = \delta state_t \odot state_{t-1} \odot f_t \odot (1 - f_t)$$
$$\delta o_t = \delta out_t \odot \tanh(state_t) \odot o_t \odot (1 - o_t)$$
$$\delta x_t = W^T \cdot \delta gates_t$$
$$\Delta out_{t-1} = U^T \cdot \delta gates_t$$

1.8 Database

Database adalah suatu kumpulan file yang terkait atau tabel yang berisi data. *Database* adalah kumpulan logis terorganisir data terkait dirancang dan

dibangun untuk tujuan tertentu, teknologi untuk menarik bersama-sama fakta-fakta yang memungkinkan untuk dipilih dan dicampurkan untuk mencocokkan data. Data dalam *database* memiliki beberapa makna yang melekat. Dengan kata lain, berbagai macam data tidak benar disebut *database*. Sebuah *database* dapat dari berbagai ukuran dan tingkat kerumitan, dan itu dapat dipertahankan secara manual atau dengan perangkat lunak pada komputer [14].

1.9 Entity Relationship Diagram (ERD)

ERD merupakan suatu model jaringan yang menggunakan susunan data yang disimpan pada sistem secara abstrak. ERD juga menggambarkan hubungan antara satu entitas yang memiliki sejumlah atribut dengan entitas lain dalam suatu sistem yang terintegrasi. ERD digunakan oleh perancang sistem untuk memodelkan data yang nantinya akan dikembangkan menjadi basis data. ERD ini juga merupakan model konseptual yang dapat mendeskripsikan hubungan antara file yang digunakan untuk memodelkan struktur data serta hubungan antar data [15].

1.10 Converter

Convert adalah perintah untuk mengubah suatu file menjadi format yang berbeda agar dapat dibaca oleh aplikasi yang bersangkutan [2].

1.11 *Tesseract*

Tesseract merupakan *free engine Optical Character Recognition* (OCR) yang dirilis dibawah lisensi *Apache* dan pengembangannya disponsori oleh *Google*. *Tesseract* saat ini merupakan salah satu *engine* OCR *open source* yang paling akurat dibanding dengan *engine* yang lain. *Tesseract* dapat membaca berbagai format gambar dan mengkonversinya ke teks. Selain gambar, *Tesseract* juga dapat membaca file PDF [16].

1.12 UML

UML adalah salah satu bahasa pemodelan untuk memodelkan suatu perangkat lunak agar perangkat lunak tersebut dapat tergambar secara detail. Keuntungan menggunakan UML yaitu perangkat lunak yang dimodelkan akan terdokumentasi dengan baik maka ketika perangkat lunak sudah berjalan lama dan

berganti developer maka developer yang baru akan mudah mengerti dengan membaca UML tersebut [17].

Pada UML terdapat beberapa diagram yang digunakan pada penelitian ini, yaitu [17]:

1. *Use case diagram*

   Use case diagram adalah diagram untuk mengambarkan fungsi yang ada pada sebuah perangkat lunak. Tujuan dari *use case diagram* untuk menggambarkan relasi antara fungsi pada perangkat lunak dengan aktor atau bisa disebut pengguna.

2. *Use case scenario*

   Use case scenario adalah deskripsi dari sebuah *use case*. Penjelasan ini diperlukan untuk mendetailkan fungsi agar lebih dimengerti oleh pembaca terutama *developer*.

3. *Activity diagram*

   Activity diagram digunakan untuk menggambarkan alur dari proses, bisnis proses, alur pada *use case diagram*.

4. *Class diagram*

   Class diagram digunakan untuk menggambarkan *class* apa saja yang nantinya akan ada pada perangkat lunak tersebut. *Class* yang digambarkan berasal dari objek-objek pada perangkat lunak tersebut.

5. *Sequence diagram*

   Sequence diagram digunakan untuk menggambarkan komunikasi antar *class* yang dibuat pada *class* diagram dan juga komunikasi antar *class* tersebut dengan aktor.

1.13 Bahasa Pemrograman

Bahasa pemrograman merupakan kumpulan aturan yang disusun sedemikian rupa sehingga memungkinkan pengguna komputer membuat program yang dapat dijalankan dengan aturan tersebut [8]. Beberapa contoh bahasa pemrograman yang banyak dipakai, yaitu C, C++, C#, Pascal, Java, JavaScript, PHP, SQL, Python dan Bahasa Pemrograman yang digunakan pada pembangunan sumber daya jenis kata ini adalah menggunakan PHP, JavaScript dan SQL.

1.14 Python

Pada saat ini bahasa pemrograman python sering digunakan untuk melakukan penelitian atau pembuatan aplikasi machine learning dikarenakan banyak library yang mendukung dan bahasa pemrograman python sama seperti matematika. Kelebihan dari bahasa pemrograman python yaitu banyaknya library yang sedang dikembangkan, mudah untuk dipelajari dan dapat diintegrasikan dengan bahasa pemrograman lain seperti C, C++ atau java. Walaupun begitu bahasa pemrograman python memiliki beberapa kelemahan yaitu lemah dalam bidang teknologi mobile, lemah dalam pengembangan database layer dan run-time *error* (*error* hanya terlihat pada saat program sudah dijalankan) [18].

Python adalah bahasa pemrograman yang bersifat *open source*. Bahasa pemrograman ini dioptimalisasikan untuk *software quality*, *developer productivity*, *program portability*, dan *component integration*. Python telah digunakan untuk mengembangkan berbagai macam perangkat lunak, seperti *internet scripting*, *systems programming*, *user interfaces*, *product customization*, *numberic programming*, dan lain-lain [19].

1.15 Notepad++

Notepad++ adalah program aplikasi yang berguna untuk mengedit tes dan skrip kode pemrograman seperti HTML, CSS, PHP, XML, Java, dan lain-lain yang bekerja pada sistem operasi windows. Kelebihan Notepad++ jika dibandingkan dengan notepad bawaan windows adalah memiliki kelengkapan fitur untuk mempermudah pengguna saat mengedit kode termasuk saat mengedit kode HTML dan kode CSS.

1.16 Flow Chart

Flow chart atau Bagan Alir Program merupakan bagan yang menjelaskan secara rinci langkah-langkah dari proses program. *Flow chart* dibuat menggunakan simbol-simbol yang terdapat pada daftar simbol sebelumnya. *Flow chart* ini akan menggambarkan logika setiap langkah proses dari sebuah program, dimana logika ini bisa dianalogikan dengan logika yang ada pada kehidupan sehari-hari [20].

1.17 One hot pada library keras

Metode word embedding yang digunakan pada penelitian ini adalah *one hot* menggunakan *library keras* [21]. Kutipan sequence pada library keras dapat dilihat pada gambar 2.13 dibawah ini :

Keras provides the one_hot() function that you can use to tokenize and integer encode a text document in one step. The name suggests that it will create a one-hot encoding of the document, which is not the case.
Instead, the function is a wrapper for the hashing_trick() function described in the next section. The function returns an integer encoded version of the document. The use of a hash function means that there may be collisions and not all words will be assigned unique integer values.

Gambar 2.13 sequence from keras

Pada gambar 2.14 menjelaskan bahwa merubah kata menjadi vector dalam bentuk *sequence* integer yang mana tidak bisa dimulai dari angka 0 sehingga dari kata pertama dimulai dari angka 1.

Returns

A list of integer word indices (unicity non-guaranteed).

`0` is a reserved index that won't be assigned to any word.

Gambar 2.14 sequence rules form keras