



الجامعة الإسلامية
الاندونيسية

Pengembangan Model Named Entity Recognition Untuk Pengenalan Entitas Pada Data Obat Indonesia

Dede Brahma Arianto

18917109

Tesis diajukan sebagai syarat untuk meraih gelar Magister Komputer

Konsentrasi Sains Data

Program Studi Informatika Program Magister

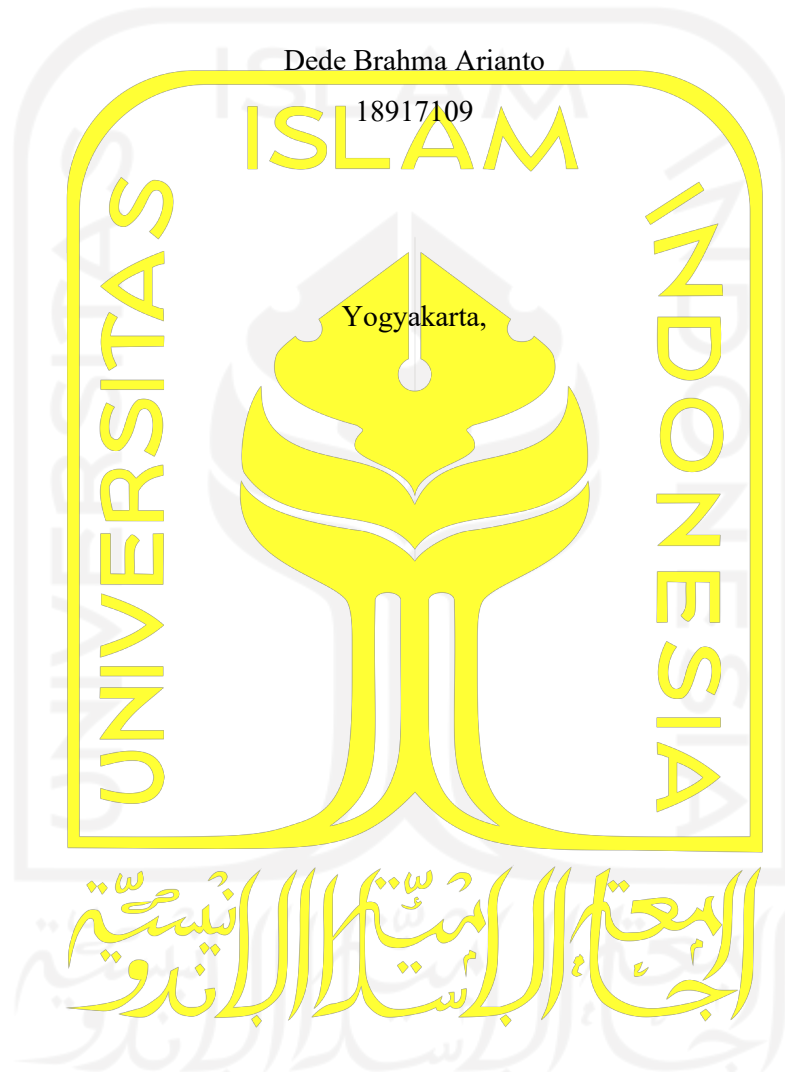
Fakultas Teknologi Industri

Universitas Islam Indonesia

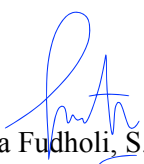
2023

Lembar Pengesahan Pembimbing

**Pengembangan Model Named Entity Recognition Untuk Pengenalan Entitas Pada
Data Obat Indonesia**



Pembimbing


Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D.

Lembar Pengesahan Penguji

Pengembangan Model Named Entity Recognition Untuk Pengenalan Entitas Pada Data Obat Indonesia

Dede Brahma Arianto

18917109

Yogyakarta, Januari 2023

Tim Penguji,

Dhomas Hatta Fudholi, ST., M.Eng, Ph.D

Ketua

Chandra Kusuma Dewa, S.Kom., M.Cs., Ph.D

Anggota I

Ahmad M Rafie Pratama, ST., MIT., Ph.D

Anggota II

Mengetahui,

Ketua Program Studi Informatika Program Magister

Universitas Islam Indonesia



Irving Vitra Paputungan, S.T., M.Sc., Ph.D

Abstrak

Pengembangan Model Named Entity Recognition Untuk Pengenalan Entitas Pada Data Obat Indonesia

Dalam beberapa tahun terakhir *Named Entity Recognition* (NER) menjadi teknik penting dalam pengolahan ekstraksi informasi pada berbagai macam domain. Penelitian NER dengan pendekatan *deep learning* menjadi *state-of-the-art* saat ini dan semakin terus dikembangkan. Di bawah domain medis, penelitian ini bertujuan untuk mengekstrak entitas produk obat, kandungan obat, dan komposisi obat yang digunakan sebagai kategori untuk mengklasifikasikan entitas yang diekstraksi. Kontribusi utama penelitian ini terletak pada pengembangan arsitektur model NER dan evaluasi model. Penelitian ini menggunakan data obat indonesia sebagai objek penelitian dan metode yang digunakan adalah *Convolutional Neural Network–Bidirectional Gated Recurrent Unit* (CNN-BiGRU). Pada arsitektur model, CNN digunakan untuk mengekstraksi fitur karakter pada tiap kata dan BiGRU digunakan untuk menangkap informasi secara dua arah dengan tujuan mendapatkan konteks kata yang lebih baik sehingga klasifikasi entitas lebih optimal. Pada proses pelatihan model dilakukan dengan delapan *hyperparameter* berbeda untuk mendapatkan model terbaik. Berdasarkan hasil pengujian, hasil akurasi terbaik didapatkan pada model dengan pengaturan *hyperparameter* yaitu CNN *kernel size* adalah 7, CNN *filter* adalah 50, CNN *layer* adalah 1, GRU *unit* adalah 200 dan GRU *layer* adalah 1. Dengan hasil akurasi yang didapatkan pada F1-Score sebesar 87%. Pada hasil perbandingan menunjukkan bahwa model CNN-BiGRU memiliki nilai *training loss* dan *validation loss* yang lebih rendah dibandingkan model CNN-BiLSTM. Pada *training time* menunjukkan proses pelatihan pada model CNN-BiGRU memiliki *training time* lebih cepat dibandingkan model CNN-BiLSTM.

Kata kunci

named entity recognition, bidirectional, cnn-bigru, deep learning, obat.

Abstract

Model Development of Named Entity Recognition for Entity Recognition in Indonesian Drug Data

Recently, Named Entity Recognition (NER) has become an important technique in processing information extraction in various domains. NER research with a deep learning approach is currently state-of-the-art and continues to be developed. In medical domain, this study aims to extract the drug product entities, drug ingredient, and drug dose which are used as categories to classify the extracted entities. The main contribution of this research lies in the development of the NER model architecture and model evaluation. In this study, data on Indonesian drugs was used as the research object and the method used was the Convolutional Neural Network–Bidirectional Gated Recurrent Unit (CNN-BiGRU). In the architecture model, CNN is used to extract character features in each word and BiGRU is used to be able to retrieve information in two directions with the aim of getting better word context so that entity classification is more optimal. The training process is carried out with eight different hyperparameters to get the best model. Based on the test results, the best accuracy results were obtained for the model with hyperparameter settings such as the CNN kernel size is 7, the CNN filter is 50, the CNN layer is 1, the GRU unit is 200 and the GRU layer is 1. The accuracy results obtained on the F1-Score are 87%. The comparison results show that the CNN-BiGRU model has smaller training loss and validation loss values than the CNN-BiLSTM model. The training time shows that the training process on the CNN-BiGRU model has a faster training time than the CNN-BiLSTM model.

Keywords

named entity recognition, bidirectional, cnn-bigru, deep learning, drug

Pernyataan Keaslian Tulisan

Dengan ini saya menyatakan bahwa tesis ini merupakan tulisan asli dari penulis, dan tidak berisi material yang telah diterbitkan sebelumnya atau tulisan dari penulis lain terkecuali referensi atas material tersebut telah disebutkan dalam tesis. Apabila ada kontribusi dari penulis lain dalam tesis ini, maka penulis lain tersebut secara eksplisit telah disebutkan dalam tesis ini.

Dengan ini saya juga menyatakan bahwa segala kontribusi dari pihak lain terhadap tesis ini, termasuk bantuan analisis statistik, desain survei, analisis data, prosedur teknis yang bersifat signifikan, dan segala bentuk aktivitas penelitian yang dipergunakan atau dilaporkan dalam tesis ini telah secara eksplisit disebutkan dalam tesis ini.

Segala bentuk hak cipta yang terdapat dalam material dokumen tesis ini berada dalam kepemilikan pemilik hak cipta masing-masing. Apabila dibutuhkan, penulis juga telah mendapatkan izin dari pemilik hak cipta untuk menggunakan ulang materialnya dalam tesis ini.

Yogyakarta, 23 Januari 2023



Dede Brahma Arianto

Daftar Publikasi

Publikasi yang menjadi bagian dari tesis

Fudholi, D. H., Nayoan, R. A. N., Hidayatullah, A, F., & Arianto, D. B. (2022). A HYBRYD CNN-BILSTM MODEL FOR DRUG NAMED ENTITY RECOGNITION. *Journal of Engineering Science and Technology (JESTEC)* Vol.17, No.1 (2022)

Sitasi publikasi 1

Kontributor	Jenis Kontribusi
Dhomas Hatta Fudholi	Mendesain eksperimen Menulis dan mengedit <i>paper</i>
Royan Abida Nur Nayoan	Membuat model dan menulis <i>paper</i>
Ahmad Fathan Hidayatullah	Membuat model Menulis dan mengedit <i>paper</i>
Dede Brahma Arianto	Membuat model dan menulis <i>paper</i>

Halaman Kontribusi

Dalam penulisan tesis ini Dosen Pembimbing, dan Dosen-dosen Penguji memberikan masukan-masukan sebagai perbaikan dari cara penulisan tesis serta memberikan saran tentang data yang akan diolah dan dianalisis.



Halaman Persembahan

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

Kupersembahkan Tesis ku ini dengan setulus hati & sepenuh jiwa untuk :

Istriku Marina Octavia tercinta, kedua Orang tuaku yang selalu mengiringi langkah kakiku dengan do'a, yang tak pernah putus asa untuk mengurus, mendidik, menasehati, menyemangati, membimbing serta memberikan motivasi kepadaku.



Kata Pengantar

Alhamdulillahirabbil'amin, segala puji syukur kehadirat Allah SWT senantiasa penulis panjatkan karena atas limpahan rahmat dan hidayah-Nya, tesis yang berjudul “Pengembangan Model *Named Entity Recognition* Untuk Pengenalan Entitas Pada Data Obat Indonesia” dapat berjalan dengan lancar dalam penyelesaiannya. Tesis ini diajukan sebagai bagian dalam menyelesaikan studi dan sebagai salah satu syarat untuk memperoleh gelar Magister Komputer pada Program Studi Magister Teknik Informatika Fakultas Teknologi Industri Universitas Islam Indonesia.

Dalam penyelesaian Tesis ini, penulis banyak mendapatkan bantuan dari berbagai pihak, untuk itu penulis menyampaikan ucapan terima kasih setulusnya kepada:

1. Bapak Prof. Fathul Wahid, S.T., M.Sc., Ph.D., selaku Rektor Universitas Islam Indonesia.
2. Bapak Irving Vitra Papatungan, S.T., M.Sc., Ph.D., selaku Ketua Program Studi Teknik Informatika Program Studi Magister Universitas Islam Indonesia.
3. Bapak Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D., selaku Dosen Pembimbing yang telah memberikan banyak pengarahan dan masukan dalam penyusunan dan penyempurnaan Tesis ini.
4. Dewan Penguji Tesis yang telah memberikan banyak pengarahan dan masukan dalam penyusunan dan penyempurnaan Tesis ini.
5. Bapak/Ibu Dosen Program Studi Magister Teknik Informatika yang telah memberikan bekal ilmu pengetahuan kepada penulis, semoga ilmunya menjadi amal jariyah di dunia maupun akhirat.
6. Bapak/Ibu Staff Akademik Program Pascasarjana Fakultas Teknologi Universitas Islam Indonesia, yang telah membantu dalam segala urusan administrasi di kampus.
7. Sahabat-sahabat seperjuangan Sains Data yang telah turut serta membantu, mendukung, menyemangati serta memberikan masukan kepada penulis selama menjalani studi di Program Pascasarjana Magister Teknik Informatika Fakultas Teknologi Industri Universitas Islam Indonesia Yogyakarta.
8. Teman-teman Magister Teknik Informatika Universitas Islam Indonesia Yogyakarta se-angkatan.

Penulis menyadari bahwa dalam penulisan tesis ini masih banyak kelemahan dan kekurangan, untuk itu kritik dan saran yang sifatnya membangun sangat penulis harapkan agar tesis ini dapat menjadi lebih baik.

Daftar Isi

Lembar Pengesahan Pembimbing	i
Lembar Pengesahan Penguji.....	ii
Abstrak	iii
Abstract.....	iv
Daftar Publikasi	vi
Halaman Kontribusi.....	vii
Halaman Persembahan	viii
Kata Pengantar.....	ix
Daftar Isi	x
Daftar Tabel.....	xiii
Daftar Gambar	xiv
Glosarium	xvi
BAB 1 Pendahuluan.....	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	3
1.3 Batasan Masalah.....	4
1.4 Tujuan Penelitian.....	4
1.5 Sistematika Penulisan.....	4
BAB 2 Tinjauan Pustaka.....	5
2.1 Penelitian Terdahulu	5
2.2 Konsep Pengetahuan	6
2.2.1 Obat.....	6
2.2.2 Natural Language Processing	8
2.2.3 Named Entity Recognition.....	8
2.2.4 Part of Speech Tagging.....	10

2.2.5	BIO Tagging	10
2.2.6	Deep Learning.....	11
2.2.7	Convolution Neural Network	14
2.2.8	Recurrent Neural Network.....	18
2.2.9	Long Short Term Memory	20
2.2.10	Gated Recurrent Unit	22
2.2.11	Bidirectional.....	26
BAB 3 Metodologi.....		29
3.1	Sumber Data.....	30
3.2	Data Preprocessing.....	30
3.3	POS Tagging	31
3.4	Entity Tagging.....	34
3.5	Distribusi Data.....	36
3.6	Glove Pre-trained Word Embedding.....	36
3.7	Membangun Model CNN-BiGRU	39
3.8	Pelatihan Model.....	42
3.9	Evaluasi Model.....	43
3.10	Logarithmic Loss Model	44
BAB 4 Hasil dan Pembahasan		45
4.1	Data yang digunakan.....	45
4.2	Hasil Preprocessing	45
4.3	Hasil POS Tagging.....	50
4.4	Hasil Entity Tagging	52
4.5	Hasil Training dan Akurasi	54
4.6	Perbandingan Hasil Training.....	59
BAB 5 Kesimpulan dan Saran		64
5.1	Kesimpulan.....	64

5.2	Saran.....	64
	Daftar Pustaka.....	65



Daftar Tabel

Tabel 2.1 Penelitian Terkait.....	6
Tabel 3.1 Daftar Alamat Website Sumber Nama Obat	30
Tabel 3.2 Kumpulan POS Tag pada Pustaka Kumparan NLP	32
Tabel 3.3 Proses POS Tagging.....	34
Tabel 3.4 Pengaturan Parameter Utama	42
Tabel 3.5 Pengaturan Hyperparameter	43
Tabel 4.1 Hasil Hapus Tanggal	46
Tabel 4.2 Hasil Hapus Tanda Baca	47
Tabel 4.3 Hasil Hapus Karakter ASCII.....	48
Tabel 4.4 Hasil Normalisasi Teks.....	49
Tabel 4.5 Hasil Tokenizing	49
Tabel 4.6 Hasil Hapus Stopword.....	50
Tabel 4.7 Jumlah Kalimat Dan Token Data Set.....	54
Tabel 4.8 Jumlah Entitas Pada Data Set.....	54
Tabel 4.9 Hasil Akurasi Training Model.....	55
Tabel 4.10 Hasil NER Obat.....	58
Tabel 4.11 Hasil Perbandingan Nilai Loss	63
Tabel 4.12 Hasil Perbandingan Training Time	63

Daftar Gambar

Gambar 1.1 Jumlah Data Pada Internet	1
Gambar 2.1 Jumlah Data Obat Tahun 2022	7
Gambar 2.2 Named Entity Recognition	9
Gambar 2.3 NER BIO Tagging	11
Gambar 2.4 Deep Learning	12
Gambar 2.5 Perbedaan Machine Learning dan Deep Learning.....	13
Gambar 2.6 Arsitektur CNN Pada Klasifikasi Teks.....	15
Gambar 2.7 Contoh Convolutional Layer Filter 3x3.....	16
Gambar 2.8 Contoh Max Pooling.....	17
Gambar 2.9 Fully Connected Layer	18
Gambar 2.10 Struktur RNN.....	19
Gambar 2.11 Proses Alur Kerja RNN	19
Gambar 2.12 Ilustrasi Vanishing Gradient Problem	20
Gambar 2.13 Struktur LSTM	21
Gambar 2.14 Struktur GRU.....	23
Gambar 2.15 <i>Reset gate</i> Pada GRU	24
Gambar 2.16 Update Gate Pada GRU	24
Gambar 2.17 Current Memory Content Pada GRU	25
Gambar 2.18 Final Memory Content Pada GRU	26
Gambar 2.19 Arsitektur Bidirectional RNN.....	27
Gambar 2.20 Penerapan Bidirectional GRU	27
Gambar 3.1 Metodologi Penelitian.....	29
Gambar 3.2 Pola Chunking dan Chinking dalam Grammar.....	35
Gambar 3.3 Format Dataset Entity Tagging.....	36
Gambar 3.4 Co-occurrence Matriks pada Kalimat	37
Gambar 3.5 Model Konseptual GloVe	38
Gambar 3.6 Dataset GloVe Bahasa Indonesia.....	39
Gambar 3.7 CNN Character Features.....	40
Gambar 3.8 Model CNN-BiGRU	41
Gambar 4.1 Data Obat	45
Gambar 4.2 Kode Program Menghapus Tanggal	46

Gambar 4.3 Kode Program Menghapus Tanda Baca	46
Gambar 4.4 Kode Program Menghapus karakter ASCII.....	47
Gambar 4.5 Kode Program Normalisasi Teks.....	48
Gambar 4.6 Kode Program Tokenizing.....	49
Gambar 4.7 Kode Program Menghapus Stopword.....	50
Gambar 4.8 Kode Program POS Tagging	51
Gambar 4.9 Hasil POS Tagging	51
Gambar 4.10 Kode Program Entity Tagging.....	52
Gambar 4.11 Hasil Entity Tagging.....	53
Gambar 4.12 Data Log Hasil Akurasi Pelatihan Model 2.....	56
Gambar 4.13 Hasil Implementasi Pada Aplikasi DRUG NER	57
Gambar 4.14 Grafik Perbandingan Nilai Training Loss.....	59
Gambar 4.15 Grafik Perbandingan Nilai Validation Loss	60
Gambar 4.16 Data Log Pelatihan Model CNN-BiGRU	61
Gambar 4.17 Data Log Pelatihan Model CNN-BiLSTM.....	61
Gambar 4.18 Grafik Perbandingan Training Time.....	62
Gambar 4.19 Data Log Training Time Model CNN-BiGRU.....	62
Gambar 4.20 Data Log Training Time Model CNN-BiLSTM	62

Glosarium

NER	Named Entity Recognition
CNN	Convolution Neural Network
LSTM	Long Short Term Memory
GRU	Gated Recurrent Unit
BiLSTM	Bidirectional Long Short Term Memory
BiGRU	Bidirectional Gated Recurrent Unit
POS	Part of Speech
Hyperparameter	Variabel yang mempengaruhi keluaran model

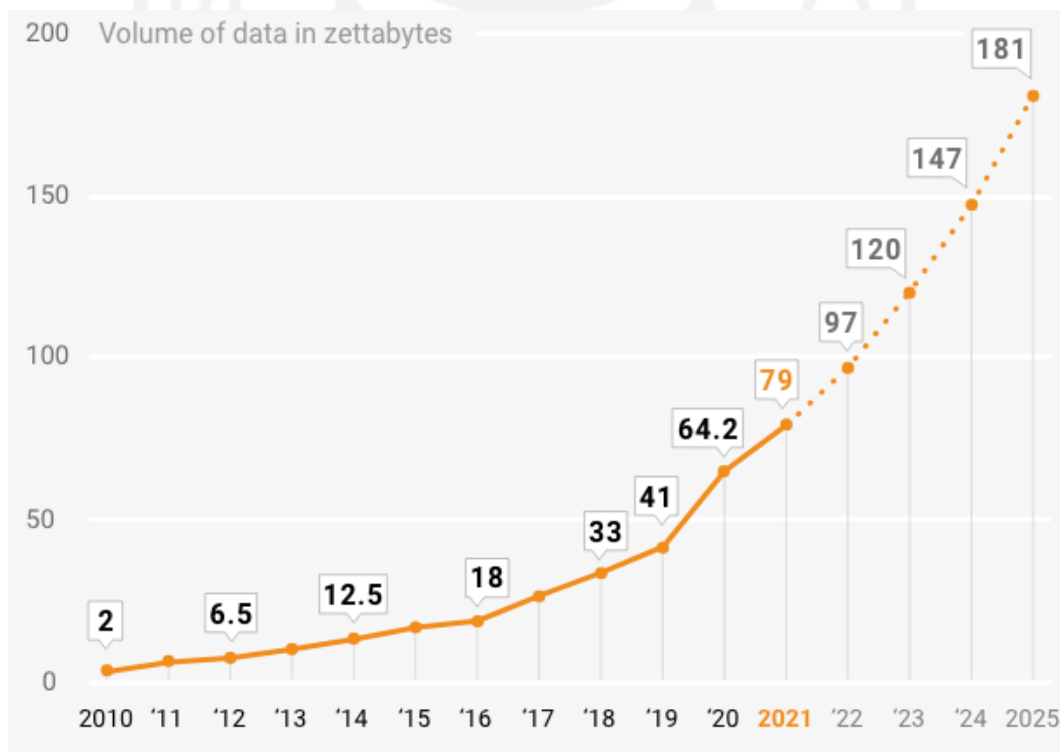


BAB 1

Pendahuluan

1.1 Latar Belakang

Obat telah menjadi bagian penting dalam kehidupan. Obat tidak hanya berfungsi untuk menyembuhkan penyakit. Namun penggunaan obat yang tidak sesuai dengan aturan yang diberikan dapat menyebabkan efek buruk pada pengguna. Oleh karena itu mengetahui informasi sifat, kandungan dan takaran penggunaan obat sangat diperlukan. Dengan perkembangan teknologi digital, informasi tentang obat dapat dengan mudah ditemukan pada internet. Pada tahun 2021 jumlah data di internet mencapai 79 *zettabyte* dan diperkirakan akan terus meningkat pada tahun-tahun berikutnya. Tren peningkatan jumlah data ditunjukkan pada Gambar 1.1.



Gambar 1.1 Jumlah Data Pada Internet¹

Pada Gambar 1.1 di atas menjelaskan bahwa pada setiap tahunnya jumlah data terus meningkat. Hal ini menyebabkan informasi yang tersebar semakin luas, namun pada

¹ <https://firstsiteguide.com/big-data-stats/>

kenyataannya sebagian besar data yang didapatkan melalui internet memiliki informasi yang kurang dapat diandalkan. Hal ini disebabkan oleh kesulitan dalam mengekstraksi informasi yang berharga. Ekstraksi informasi bertujuan untuk mencari fakta-fakta yang berkaitan dengan kejadian, entitas, atau keterhubungan dalam bentuk informasi yang terstruktur pada sebuah data teks (Piskorski & Yangarber, 2013).

Salah satu teknik yang dapat digunakan untuk mengekstraksi informasi adalah NER. NER digunakan untuk mengidentifikasi nama entitas dari sebuah teks dan mengklasifikasikannya ke dalam kategori yang telah ditentukan. NER telah banyak diimplementasikan pada berbagai macam bidang. Pada bidang biomedis, NER digunakan untuk mengidentifikasi entitas biologis seperti bahan kimia, obat, penyakit, gen, dan lain-lain. Berikut adalah contoh kategori entitas pada NER seperti nama orang (PER), organisasi (ORG), lokasi (LOC), ekspresi waktu (DATE), kuantitas (MG) dan lain-lain (Kale & Govilkar, 2017).

Dalam beberapa tahun terakhir, *deep learning* menjadi model pendekatan yang sangat dominan dalam hal penelitian NER. Hal ini dikarenakan model yang dibangun dengan menggunakan *deep learning* memiliki hasil akurasi yang lebih baik (Li et al., 2020). Salah satu algoritma *deep learning* yang populer saat ini adalah CNN. CNN dapat menangkap informasi fitur dari data teks dan juga dapat menangkap informasi berupa imbuhan dari sebuah kata. Penggunaan CNN dalam mengekstrak fitur karakter pada bidang NER memiliki hasil yang baik (Bolucu et al., 2019). Algoritma lain dalam *deep learning* adalah GRU. GRU mampu mempelajari representasi frasa bahasa secara semantik dan mampu mendapatkan informasi tentang konteks kalimat dari karakter yang berdekatan. Secara arsitektur GRU memiliki keunggulan dalam proses komputasi yang lebih cepat dibandingkan model *deep learning* lainnya seperti LSTM. Hal ini dikarenakan GRU menggunakan dua sistem gerbang pada pemrosesannya, sehingga proses komputasi menjadi lebih sederhana. Namun, pada pemrosesannya GRU hanya menerapkan pemrosesan satu arah atau disebut *unidirectional* yang berarti pemrosesan hanya memperoleh informasi dari masa lalu dan tidak dapat memperoleh informasi masa depan (Dai et al., 2019).

Pada dasarnya penerapan NER bergantung pada konteks kata di dalam sebuah kalimat. Dua kata sebelum dan sesudah sangat membantu untuk memprediksi entitas dengan memperoleh informasi dalam konteks masa lalu dan masa depan. Pemrosesan dua arah atau *bidirectional* adalah solusi untuk dapat menangkap informasi dari masa lalu dan masa depan. Penerapan *bidirectional* sebelumnya telah berhasil diimplementasikan oleh (Fudholi et al., 2022). Pada penelitiannya model yang digunakan adalah CNN-BiLSTM. Model yang

dibangun berhasil melakukan klasifikasi entitas dengan memanfaatkan fitur *chunking* dan format “BIO” pada ekstraksi entitas data obat. Hasil pengujian penelitian ini menunjukkan bahwa penerapan CNN-BiLSTM dapat mengidentifikasi kata berdasarkan jenis entitasnya dengan baik. Namun pada penelitian ini model pembentukan entitas atau *grammar* model masih belum optimal. Hasil pengujian menunjukkan bahwa ada beberapa kata yang tidak berhasil di klasifikasi ke dalam entitas. BiGRU adalah alternatif lain dalam menerapkan pemrosesan dua arah dengan tujuan menangkap informasi yang lebih banyak dan menghasilkan klasifikasi entitas yang lebih baik lagi. BiGRU telah banyak diterapkan pada bidang NLP, seperti *machine translation*, *question answering*, dan *sequence labeling* (Ayifu et al., 2019). Menurut (Goyal et al., 2022) pada penelitiannya menjelaskan bahwa model BiGRU memiliki hasil yang lebih baik dari model *Recurrent Neural Network* (RNN) lainnya seperti LSTM, GRU dan BiLSTM.

Berdasarkan latar belakang di atas, penulis melakukan penelitian tentang pengembangan model NER untuk mengidentifikasi dan mengklasifikasikan kata secara otomatis pada kalimat yang memuat informasi tentang obat di Indonesia. Penelitian ini menggunakan pendekatan *deep learning* berbasis algoritma *Convolutional Neural Network–Bidirectional GRU* (CNN-BiGRU). Algoritma CNN-BiGRU dipilih karena dapat menangkap informasi fitur pada data teks seperti kata imbuhan dan dapat menangkap informasi dalam konteks masa lalu dan masa depan namun dengan komputasi yang lebih cepat. Kontribusi utama dalam penelitian ini adalah pembuatan model CNN-BiGRU untuk mengidentifikasi entitas pada data obat Indonesia dengan menggunakan pembentukan entitas atau *grammar* model berbahasa Indonesia. Selain itu juga memberikan gambaran secara komprehensif terhadap evaluasi model yang harapannya dapat dijadikan landasan saintifik bagi kasus-kasus yang serupa.

1.2 Rumusan Masalah

Rumusan masalah dalam penelitian ini adalah:

1. Bagaimana melakukan indentifikasi entitas pada data obat Indonesia seperti nama obat, kandungan obat, dan komposisi obat ?
2. Bagaimana membangun model NER pada data obat di Indonesia dengan pendekatan *deep learning* menggunakan metode CNN-BiGRU ?

1.3 Batasan Masalah

Batasan masalah dalam penelitian adalah sebagai berikut :

1. Penelitian ini menggunakan data obat di Indonesia yang terdaftar pada website Klik Dokter, Sehatq dan Hello Sehat.
2. Penelitian ini mengidentifikasi beberapa entitas yaitu obat, kandungan dan komposisi.
3. Penelitian ini menggunakan format BIO pada setiap jenis entitas

1.4 Tujuan Penelitian

Tujuan penelitian ini adalah menghasilkan model NER yang dapat digunakan untuk menentukan jenis entitas kata pada kalimat berbahasa Indonesia yang mengandung informasi tentang obat di Indonesia.

1.5 Sistematika Penulisan

Sistematika penulisan ini disusun untuk memberikan gambaran umum tentang penelitian yang dijalankan. Sistematika penulisan ini adalah sebagai berikut:

1. BAB I PENDAHULUAN

Pada bagian ini berisi tentang pengantar terhadap permasalahan yang akan dibahas, didalamnya menguraikan tentang latar belakang permasalahan, rumusan masalah, menentukan batasan masalah yang akan dibahas, tujuan dari penelitian, dan sistematika penulisan.

2. BAB II TINJAUAN PUSTAKA

Pada bab ini menjelaskan teori-teori yang digunakan di dalam penelitian. Setiap teori yang digunakan didalam penelitian akan dijabarkan pada penelitian ini.

3. BAB III METODOLOGI PENELITIAN

Pada bab ini menampilkan metode yang akan digunakan dan langkah-langkah atau prosedur di dalam penelitian.

4. BAB IV HASIL DAN PEMBAHASAN

Pada bab ini menjelaskan hasil dan pembahasan dalam penelitian, mulai dari data yang digunakan, hasil training dan evaluasi, sampai hasil perbandingan model.

5. BAB V KESIMPULAN DAN SARAN

Pada bab ini menjelaskan kesimpulan yang didapatkan setelah melakukan penelitian dan memuat point-point saran untuk dapat ditingkatkan.

BAB 2

Tinjauan Pustaka

2.1 Penelitian Terdahulu

Beberapa penelitian tentang NER berbasis *deep learning* telah dilakukan sebelumnya, seperti ditunjukkan Tabel 2.1. Penelitian oleh (Wu et al., 2018) menggunakan dua model *deep learning* yaitu CNN dan RNN untuk mengekstraksi entitas data teks klinis. Format BIO digunakan untuk mengkonversi anotasi entitas menjadi *sequence labeling tags*. Penelitian ini menggunakan enam entitas untuk diidentifikasi. Kemudian model CNN juga diterapkan oleh (Fudholi et al., 2022) pada NER obat. Penelitian ini menggunakan model CNN-BiLSTM untuk mengekstraksi entitas obat. Model yang dibangun berhasil mendeteksi kata dan *character-level feature* untuk mengidentifikasi tiga jenis entitas dengan baik. Penelitian ini menggunakan fitur *tag chunk* dan tag BIO pada ekstraksi entitas.

Penelitian dengan BiGRU dilakukan oleh (Gridach & Haddad, 2018). Penelitian ini mengkombinasikan model BiGRU-CRF untuk mengekstraksi entitas pada Bahasa Arab. Penggunaan fitur *word embedding* dan *character-level embedding* digunakan untuk mengidentifikasi morfologi kata-kata pada bahasa Arab yang berbeda dan langka. Penelitian berhasil mengidentifikasi tiga entitas berbeda dengan fitur *chunking* dan format BIO. Penelitian oleh (Ni et al., 2021) menggabungkan CNN dengan dua lapisan BiGRU pada data teks biomedis. Model yang dirancang berhasil mengidentifikasi kata ke dalam enam entitas berbeda dan memiliki kemampuan dalam memuat *input* yang lebih beragam. Penerapan BiGRU lainnya dilakukan oleh (Ayifu et al., 2019). Penelitian ini menggabungkan model BiGRU-CNN-CRF untuk mengekstrak entitas pada multibahasa yaitu Uyghur, Kazak dan Kyrgyz. Model yang dibangun menggunakan format IOBES pada ekstraksi entitas. Penggunaan fitur *word embedding* dan *character-level embedding* digunakan untuk mengidentifikasi morfologi kata yang kompleks. Penelitian ini mengidentifikasi tiga entitas berbeda pada masing-masing bahasa.

Tabel 2.1 Penelitian Terkait

No	Sub Tema	Algoritma	Domain Penelitian	Pustaka
1	<i>Clinical Named Entity Recognition Using Deep Learning Models</i>	CNN, RNN	Data teks klinis i2b2 2010	(Wu et al., 2018)
2	<i>A Hybrid Cnn-Bilstm Model for Drug Named Entity Recognition</i>	CNN-BiLSTM	Data obat Indonesia	(Fudholi et al., 2022)
3	<i>Arabic Named Entity Recognition: A Bidirectional GRU-CRF Approach</i>	BiGRU-CRF	Dataset bahasa Arab ANERcorp	(Gridach & Haddad, 2018)
4	<i>StaResGRU-CNN with CMedLMs: A Stacked Residual GRU-CNN with Pre-trained Biomedical Language Models for Predictive Intelligence</i>	Stacked Residual GRU-CNN	Data klinis Cina CCKS 2019	(Ni et al., 2021)
5	<i>Multilingual Named Entity Recognition Based on the BiGRU-CNN-CRF hybrid model</i>	BiGRU-CNN-CRF	Dataset bahasa Uygur, Kazak, Kyrgyz UKKNERDATA	(Ayifu et al., 2019)

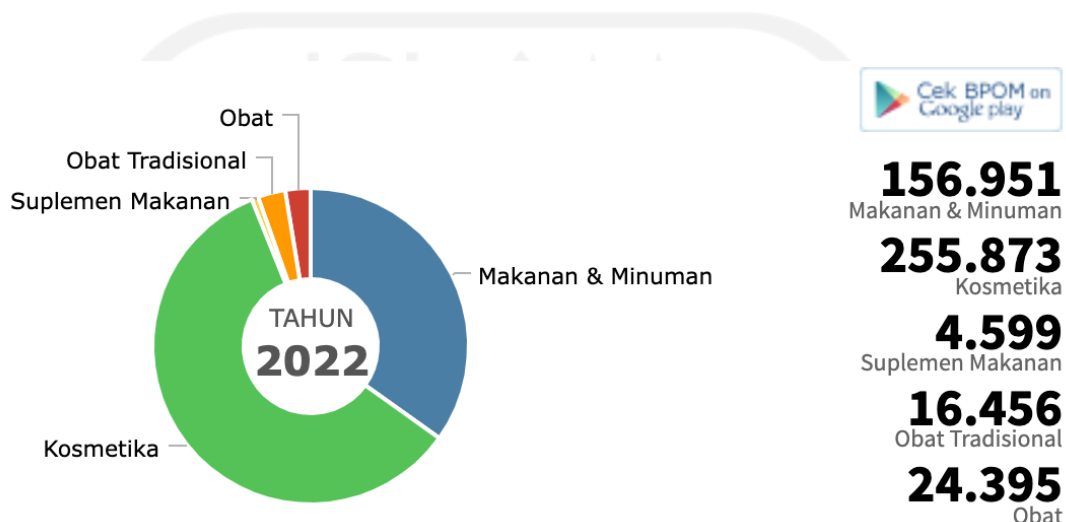
2.2 Konsep Pengetahuan

2.2.1 Obat

Berdasarkan Undang-Undang Nomor 36 Tahun 2009 tentang kesehatan, Obat adalah bahan atau paduan bahan, termasuk produk biologi yang digunakan untuk mempengaruhi atau menyelidiki sistem fisiologi atau keadaan patologi dalam rangka penetapan diagnosis,

pengecahan, penyembuhan, pemulihan, peningkatan kesehatan dan kontrasepsi, untuk manusia. Menurut (Zeenot, 2013) obat tidak hanya berfungsi untuk mendiagnosa, mencegah maupun menyembuhkan berbagai jenis penyakit. Namun penggunaan obat harus sesuai dengan aturan yang diberikan oleh para ahli, sesuai waktu dan dosis. Karena hal ini dapat menyebabkan efek buruk hingga keracunan.

Menurut data milik Badan Pengawas Obat dan Makanan (BPOM) tercatat 24.395 data obat yang telah teregistrasi hingga tahun 2022. Seperti ditunjukkan pada Gambar 2.1



Gambar 2.1 Jumlah Data Obat Tahun 2022²

Pada dasarnya obat merupakan bahan yang hanya dengan takaran tertentu dan dengan penggunaan yang tepat dapat dimanfaatkan untuk mendiagnosa, mencegah penyakit, menyembuhkan atau memelihara kesehatan. Oleh karena itu sebelum menggunakan obat, harus diketahui sifat dan cara penggunaannya agar tepat, aman dan rasional. Besarnya efektivitas obat tergantung pada dosis dan kepekaan organ tubuh. Setiap orang berbeda kepekaan dan kebutuhan dosis obatnya. Tetapi secara umum dapat dikelompokkan, yaitu dosis bayi, anak-anak, dewasa dan orang tua. Komposisi atau kandungan obat merupakan informasi tentang zat aktif yang terdapat di dalam sediaan obat, disebut juga dengan zat aktif atau zat berkhasiat. Komposisi dapat berupa zat tunggal atau kombinasi dari berbagai macam zat aktif dan bahan tambahan lain (Kemenkes RI, 2018).

² <https://cekbpom.pom.go.id/>

2.2.2 Natural Language Processing

Natural Language Processing (NLP) adalah cabang kecerdasan buatan yang digunakan untuk memecahkan permasalahan tugas-tugas rumit yang berkaitan dengan bahasa alami. NLP melibatkan desain dan implementasi model, sistem, dan algoritma dengan tujuan utama yaitu menerjemahkan bahasa alami manusia ke dalam perintah yang dapat dijalankan oleh komputer (Kang et al., 2020).

Pada perkembangannya NLP menjadi salah satu domain paling aktif dalam penelitian *deep learning*. Metode *deep learning* telah terbukti bekerja dengan baik pada berbagai tugas NLP seperti *language modeling*, *machine translation*, *POS tagging*, *NER*, *sentiment analysis*, dan *paraphrase detection*. Aspek yang paling menarik dari metode *deep learning* adalah kemampuannya untuk memproses data berupa teks dengan memanfaatkan konsep *embedding*. Konsep ini mengacu pada representasi informasi simbolik dalam teks pada tingkat kata, tingkat frasa, dan bahkan tingkat kalimat dengan mengubahnya kedalam bentuk vektor (Deng, 2014).

Menurut (Lauriola et al., 2022) saat ini tugas NLP yang banyak digunakan dapat dikategorikan dalam kelas-kelas berikut:

1. *Sequence classification*, digunakan untuk pemecahan permasalahan dengan beberapa urutan *input* dan tugasnya adalah memprediksi kategori dalam urutan tersebut. Contohnya seperti analisis sentimen, kategorisasi dokumen dan *question answering*.
2. *Pairwise sequence classification*, digunakan untuk membandingkan dan mengklasifikasikan dua urutan yang berbeda berdasarkan kemiripannya. Contohnya seperti Quora Question Pairs yang tujuannya adalah untuk menemukan pertanyaan duplikat.
3. *Word labeling*, digunakan untuk memberikan label pada kata yang bertujuan untuk memberikan konteks atau makna pada kata. Contohnya seperti *NER* dan *POS tagging*.
4. *Sequence2sequence*, digunakan untuk memberikan *output* secara berurutan dari hasil urutan *input* biasanya berupa kalimat. Contohnya seperti mesin penterjemah dan *story generator*.

2.2.3 Named Entity Recognition

Named Entity Recognition (NER) adalah salah satu tugas dalam NLP. Tujuan utama NER adalah untuk menemukan dan mengklasifikasi entitas pada teks dokumen kedalam kategori entitas yang telah ditentukan, misalnya entitas nama orang, organisasi, lokasi dan entitas

domain yang spesifik (Patil et al., 2020). Pada Gambar 2.2 menunjukkan contoh dari NER yang diklasifikasikan menjadi beberapa kategori, yaitu orang (PERSON), organisasi (ORG), tanggal (DATE), dan kebangsaan atau agama atau kelompok politik (GPE).

F.B.I. Agent Peter Strzok PERSON, Who Criticized Trump PERSON in Texts, Is Fired GPE - The New York Times ORG SectionsSEARCHSkip to contentSkip to site indexPoliticsSubscribeLog InSubscribeLog InToday's PaperAdvertisementSupported ORG byF.B.I. Agent Peter Strzok PERSON, Who Criticized Trump PERSON in Texts, Is FiredImagePeter Strzok, a top F.B.I. GPE counterintelligence agent who was taken off the special counsel investigation after his disparaging texts about President Trump PERSON were uncovered, was fired. CreditT.J. Kirkpatrick PERSON for The New York TimesBy Adam Goldman ORG and Michael S. SchmidtAug PERSON. 13 CARDINAL, 2018WASHINGTON CARDINAL - Peter Strzok PERSON, the F.B.I. GPE senior counterintelligence agent who disparaged President Trump PERSON in inflammatory text messages and helped oversee the Hillary Clinton PERSON email and Russia GPE investigations, has been fired for violating bureau policies, Mr. Strzok PERSON's lawyer said Monday DATE. Mr. Trump and his allies seized on the texts — exchanged during the 2016 DATE campaign with a former F.B.I. GPE lawyer, Lisa Page — in PERSON assailing the Russia GPE investigation as an illegitimate "witch hunt." Mr. Strzok PERSON, who rose over 20 years DATE at the F.B.I. GPE to become one of its most experienced counterintelligence agents, was a key figure in the early months DATE of the inquiry. Along with writing the texts, Mr. Strzok PERSON was accused of sending a highly sensitive search warrant to his personal email account. The F.B.I. GPE had been under immense political pressure by Mr. Trump PERSON to dismiss Mr. Strzok PERSON, who was removed last summer DATE from the staff of the special counsel, Robert S. Mueller III PERSON. The president has repeatedly denounced Mr. Strzok PERSON in posts on Twitter EVENT, and on Monday DATE expressed satisfaction that he had been sacked. Mr. Trump's ORG victory traces back to June DATE, when Mr. Strzok PERSON's conduct was laid out in a wide-ranging inspector general's report on how the F.B.I. GPE handled the investigation of Hillary Clinton's PERSON emails in the run-up to the 2016 DATE election. The report was critical of Mr. Strzok PERSON's conduct in sending the

Gambar 2.2 Named Entity Recognition³

NER tidak hanya digunakan untuk menyelesaikan masalah ekstraksi informasi, tetapi juga memainkan peran penting dalam berbagai aplikasi NLP lainnya seperti *information retrieval*, *automatic text summarization*, *question answering*, *machine translation*, dan lain-lain. Dalam perkembangannya berbagai macam teknik telah digunakan untuk membangun aplikasi NER. Berikut adalah beberapa teknik yang digunakan untuk membuat NER (Li et al., 2020) :

1. *Rule-based approaches*, teknik ini tidak memerlukan data beranotasi karena mengandalkan aturan pada pola-pola posisi kata yang telah didefinisikan.
2. *Unsupervised learning approaches*, teknik ini menggunakan data yang tidak berlabel untuk mengambil keputusan.
3. *Feature-based supervised learning approaches*, teknik ini menggunakan data yang telah diberi label yang digunakan untuk melatih model agar dapat mengenali entitas.
4. *Deep-learning based approaches*, teknik ini secara otomatis dapat menemukan klasifikasi entitas dari *input* baru secara *end-to-end*.

³ <https://towardsdatascience.com/named-entity-recognition-with-nltk-and-spacy-8c4a7d88e7da>

Untuk dapat membantu proses NER dalam menemukan dan mengklasifikasikan suatu entitas diperlukan proses pemberian label atau tanda pada suatu kata. Proses ini disebut POS *tagging*. POS *tagging* dianggap sebagai proses klasifikasi urutan label untuk setiap kata dalam kalimat.

2.2.4 Part of Speech Tagging

Part of Speech (POS) tagging adalah proses pemberian label atau disebut juga dengan *tag* dengan mengkategorikan setiap kata dalam sebuah kalimat kedalam kelas kata seperti kata kerja, kata sifat, kata keterangan, kata pengganti, kata penghubung dan lainnya (Rashel et al., 2014). *Tag* yang diberikan pada suatu kata dalam kalimat merupakan representasi dari kelas kata pada konteks kata tersebut. Kumpulan dari *tag* atau kelas kata yang ada disebut *tagset*.

Pemberian *tag* pada kata dapat dilakukan dengan dua cara. Pertama, secara manual yaitu dengan memberikan *tag* pada masing-masing kata pada kumpulan data yang kita miliki. Kedua, secara otomatis yaitu dengan menggunakan beberapa pendekatan seperti *rule-based* dan *probability-based*. *Rule-based tagging* dilakukan dengan bantuan ahli linguistik untuk membuat aturan-aturan yang biasa digunakan manusia. Sedangkan *probability-based tagging* dilakukan dengan melibatkan database dalam ukuran besar sebagai data pelatihan untuk kemudian memprediksi kelas kata secara probabilistik (Pisceldo, 2009).

Dalam penerapannya POS *tagging* diaplikasikan pada setiap kata termasuk pada tanda baca seperti tanda titik, tanda koma, dan tanda baca lainnya. Oleh karena itu untuk dapat memisahkan tanda baca dan kata-kata pada dokumen teks diperlukan proses tokenisasi. Tokenisasi adalah proses pemecahan teks menjadi kata-kata yang lebih kecil dan beberapa kata yang lebih kecil disebut token⁴.

2.2.5 BIO Tagging

Beginning, Inside, Outside (BIO) tagging adalah format pemberian *tag* untuk menandai token. Biasanya proses ini menggunakan *chunking* dan *chinking*. *Chunking* adalah proses untuk mengidentifikasi struktur kalimat kemudian mengelompokkannya berdasarkan frasa atau disebut *chunk*. Sedangkan *chinking* adalah proses menghilangkan token yang tidak dibutuhkan didalam *chunk*. Format BIO merupakan format yang umum digunakan sebagai penanda *tag*. Pada format BIO dimana B-*begin* digunakan untuk menunjukkan awalan dari

⁴ <https://nlp.stanford.edu/IR-book/html/htmledition/tokenization-1.html>

entitas, *I-inside* digunakan untuk menunjukkan kata setelahnya sebagai entitas lanjutan, dan *O-other* digunakan untuk menunjukkan kata yang berada diluar satu entitas (Ramshaw & Marcus, 1995).

Format BIO mewakili informasi dari sebuah kalimat sederhana dan juga notasi yang sama seperti tanda kurung dan bergantung pada proses tokenisasi yang digunakan. Penggunaan format BIO telah banyak digunakan pada aplikasi NER di berbagai domain. Pada Gambar 2.3 menunjukkan penerapan format BIO pada domain Bahasa Indonesia (Wintaka et al., 2019). Dapat dilihat bahwa kata “Kementerian” ditandai sebagai B-ORG karena dikenali sebagai awalan kata organisasi. Kemudian diikuti kata “keuangan” ditandai sebagai I-ORG karena masih dalam frasa yang sama dan dikenali sebagai organisasi. Kata “Indonesia” dikenali ditandai B-LOC karena dikenali sebagai awalan untuk lokasi dan kata yang tidak termasuk kedalam jenis entitas ditandai sebagai O.

Tweet	Tags
<i>Kementerian</i>	B-ORG
<i>keuangan</i>	I-ORG
<i>menyatakan</i>	O
<i>inflasi</i>	O
<i>di</i>	O
<i>Indonesia</i>	B-LOC
<i>rendah</i>	O

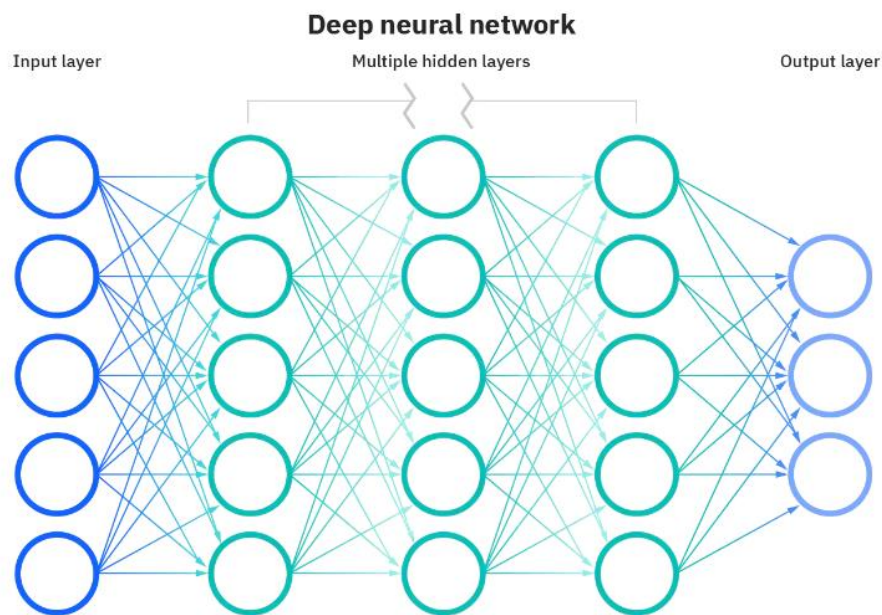
Gambar 2.3 NER BIO Tagging

2.2.6 Deep Learning

Deep learning merupakan sub bagian dari *machine learning* yang disempurnakan. Dimana pada arsitekturnya menggunakan *Artificial Neural network* (ANN) yang terdiri dari banyak lapisan. ANN memiliki sistem kerja seperti otak manusia, tiap *neuron* saling terhubung dan memproses data masukkan melalui banyak lapisan (Thomas & Sangeetha, 2020). *Deep learning* dirancang untuk memecahkan permasalahan pada kumpulan data yang besar dan kompleks seperti model matematika, pengenalan pola dan klasifikasi.

Pada dasarnya, *deep learning* merupakan jaringan saraf tiruan yang memiliki tiga atau lebih lapisan ANN. Sebagian besar metode *deep learning* menggunakan arsitektur jaringan saraf, karena itu model *deep learning* sering disebut sebagai *deep neural network*. Pada Gambar 2.4 menunjukkan arsitektur dari *deep learning*. Dimana sebuah *neural network*

memiliki elemen dasar berupa *input layer*, *hidden layer*, dan *output layer*. Dengan banyaknya jumlah lapisan atau koneksi berulang umumnya meningkatkan kedalaman jaringan untuk menyediakan berbagai tingkat representasi data dan ekstraksi fitur, yang disebut sebagai “*deep learning*” (Salehinejad et al., 2018). Dikatakan *deep learning* apabila suatu *neural network* ini memiliki lebih dari satu *hidden layer*.



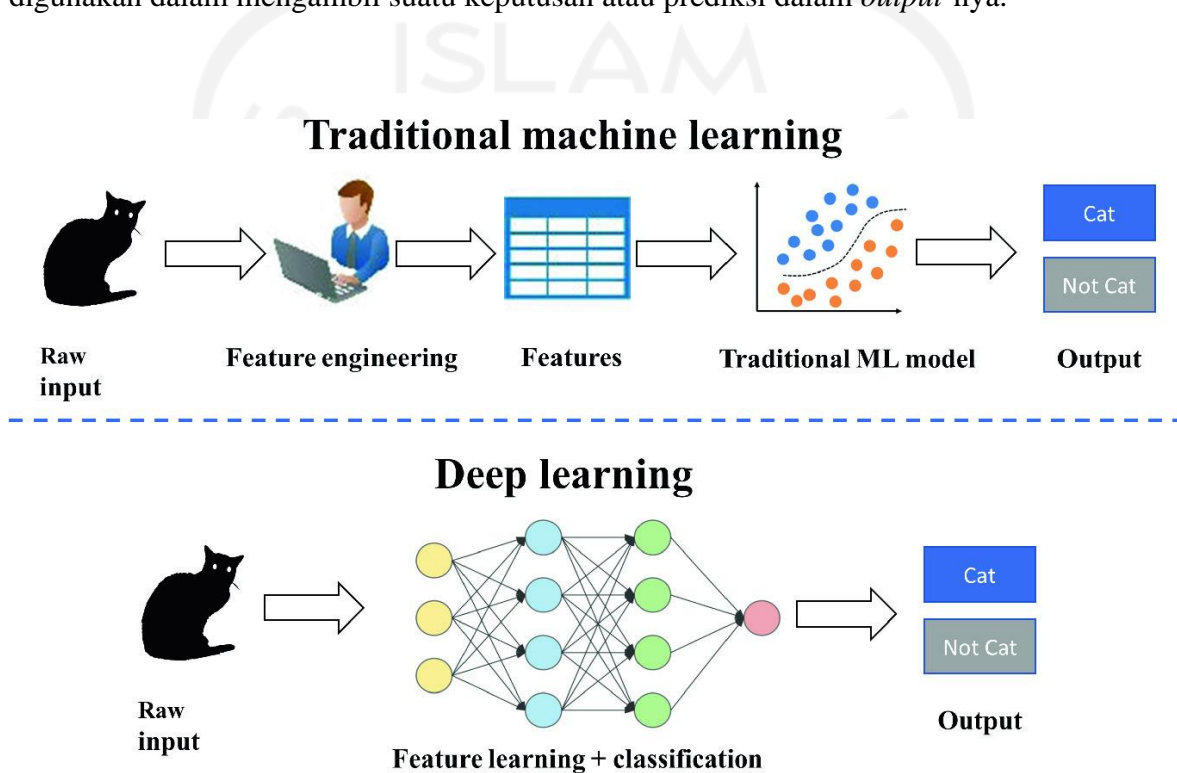
Gambar 2.4 Deep Learning⁵

Deep learning adalah pendekatan yang telah diterapkan secara luas dalam berbagai domain. Adapun pengaplikasian *deep learning* seperti *semantic parsing*, *transfer learning*, *natural language processing*, *computer vision* dan masih banyak lagi. Secara historis, konsep *deep learning* berasal dari penelitian berbasis ANN yang dikembangkan dengan menambahkan banyak *hidden layer* didalamnya. Pada awalnya konsep ini dipopulerkan pada tahun 1980-an, dan saat ini telah menjadi algoritma yang banyak digunakan untuk *learning parameter* (Deng, 2014). Ada tiga alasan penting kenapa *deep learning* banyak digunakan saat ini (Guo et al., 2016) :

1. kemampuan pemrosesan chip yang meningkat secara dramatis (misalnya GPU units)
2. biaya perangkat keras untuk komputasi yang semakin turun secara signifikan
3. kemajuan besar dalam berbagai penelitian dengan algoritma *machine learning*.

⁵ <https://www.ibm.com/cloud/learn/neural-networks>

Deep learning berbeda dengan *machine learning traditional*. Pada Gambar 2.5 menunjukkan perbedaan mendasar dari cara kerja *deep learning* dan *machine learning traditional* (Wang & Cao, 2021). Dalam *machine learning traditional*, pemrogram harus sangat spesifik memberi tahu komputer jenis hal apa yang harus dicari untuk mengambil suatu keputusan dalam *output*-nya, sedangkan *deep learning* membangun set fitur (*extraction feature*) secara otomatis tanpa perlu dilakukan secara manual oleh manusia untuk digunakan dalam mengambil suatu keputusan atau prediksi dalam *output*-nya.



Gambar 2.5 Perbedaan Machine Learning dan Deep Learning

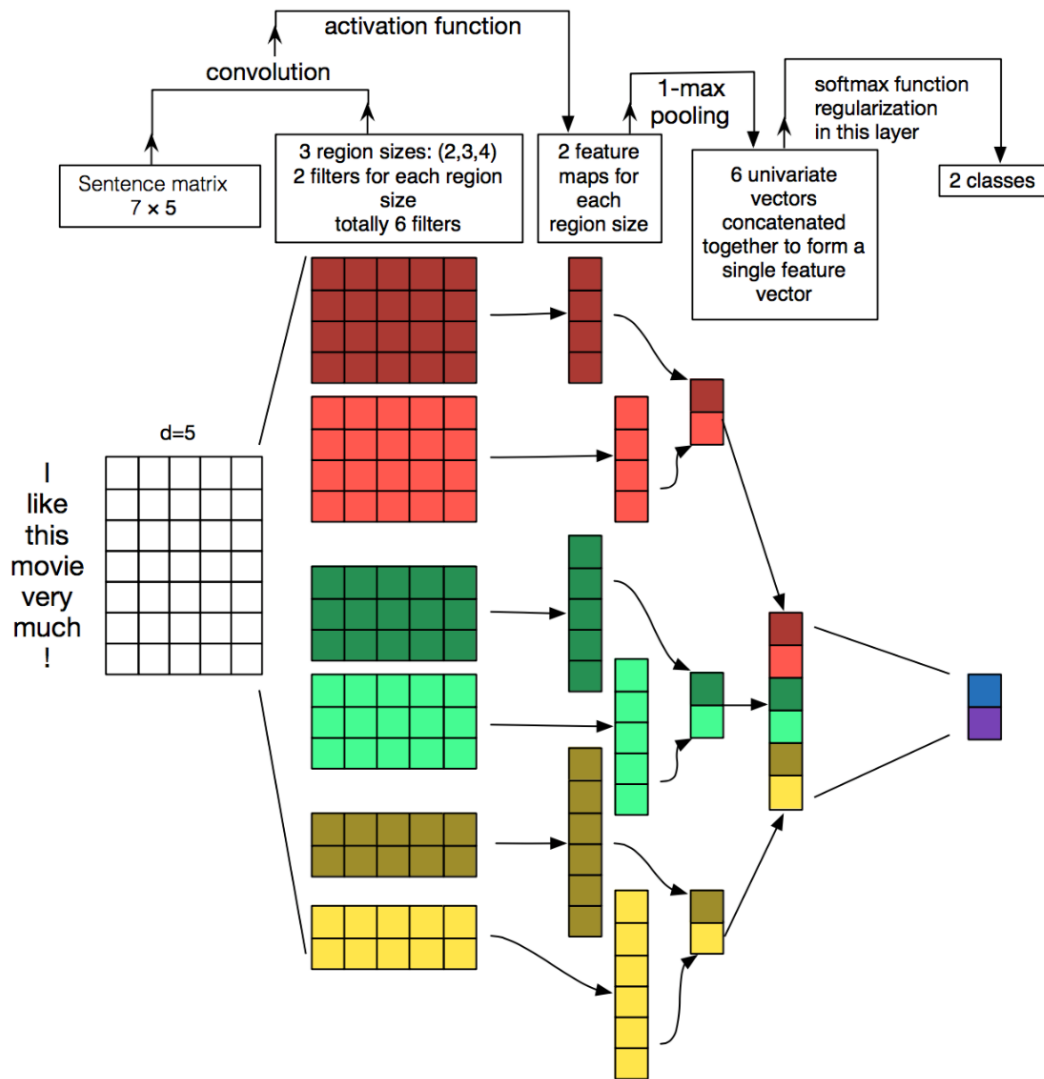
Deep learning secara umum dibagi kedalam tiga jenis arsitektur. Pertama, *Multi Layer Perceptron* (MLP) yang memiliki konsep dasar meniru sistem kerja otak manusia. MLP digunakan untuk memprediksi data yang memiliki label. Kedua, *Convolutional Neural network* (CNN) yang banyak digunakan untuk memprediksi data berupa gambar namun saat ini banyak penelitian pada CNN untuk melakukan klasifikasi dokumen teks. Ketiga, *Recurrent Neural network* (RNN) digunakan untuk memecahkan permasalahan data sekuensial seperti data teks.

2.2.7 Convolution Neural Network

Convolutional Neural Network (CNN) adalah salah satu arsitektur *deep learning* yang dapat diterapkan untuk melakukan klasifikasi dokumen teks. CNN awalnya digunakan sebagai metode untuk melakukan klasifikasi pada sebuah gambar. Namun saat ini CNN telah banyak diterapkan dalam berbagai tugas NLP (Kalchbrenner et al., 2014).

Secara historis CNN pertama kali diperkenalkan oleh Lecun, Bottou, Bengio dan Haffner pada tahun 1998. Penelitian ini ditujukan untuk *computer vision* dengan memanfaatkan lapisan dengan *filter* konvolusi pada *local features* (Lecun et al., 1998). Kemudian pada tahun 2012 CNN kembali dikembangkan dan melahirkan sebuah dataset yang disebut ImageNet. Dataset ini berisi banyak gambar yang dapat digunakan sebagai model pelatihan (Krizhevsky et al., 2012). Dan saat ini CNN telah digunakan juga dalam tugas NLP salah satunya adalah klasifikasi kalimat dengan menggunakan lapisan konvolusi pada CNN. Model CNN terbukti efektif untuk NLP termasuk untuk melakukan tugas klasifikasi teks (Kim, 2014).

Pada Gambar 2.6 menunjukkan arsitektur CNN pada klasifikasi teks. Hal pertama yang dilakukan dalam tahap klasifikasi teks pada CNN adalah mengambil *input* dalam kalimat atau dokumen yang direpresentasikan sebagai matriks. Setiap baris pada matriks merupakan kata tunggal (token) akan ditransformasi menjadi vektor. *Input* kalimat dijadikan matriks dengan ukuran 7x5 kemudian masuk ke dalam 3 *region size* yang masing-masing memiliki 2 *filter*. Setiap *filter* melakukan konvolusi pada matriks kalimat dan menghasilkan *feature maps*. Selanjutnya akan dilakukan *max-pooling* dengan mengambil nilai terbesar pada setiap *map*. Maka akan menghasilkan 6 *univariate feature vector* dari seluruh *map* sebanyak 6 dan akan digabungkan untuk membentuk *single feature vector*. Kemudian *feature vektor* akan digunakan sebagai *input* untuk dilakukan klasifikasi pada *layer softmax* (Zhang & Wallace, 2016).



Gambar 2.6 Arsitektur CNN Pada Klasifikasi Teks

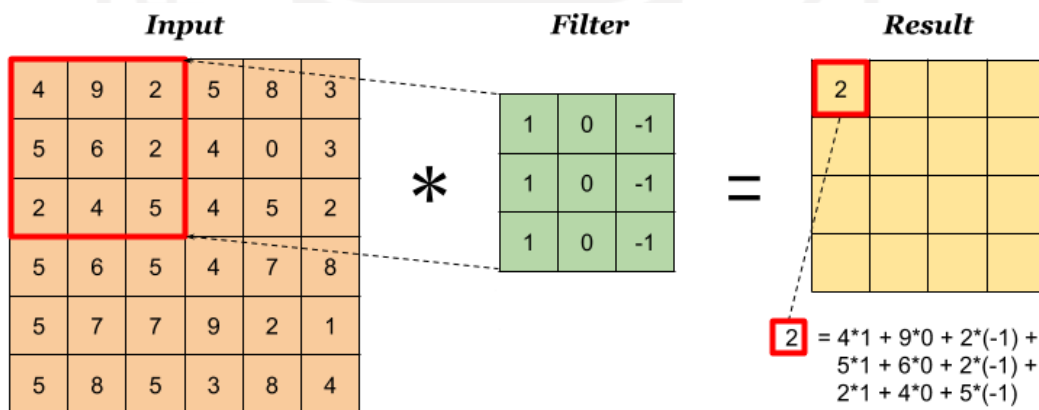
Arsitektur CNN mempunyai kelebihan yaitu, dapat mempertahankan informasi susunan kata dalam sebuah kalimat, serta aktivasi nonlinier dalam CNN dapat mempelajari karakteristik yang lebih abstrak (X. Qiu & Huang, 2015). Struktur pada CNN terdiri dari banyak lapisan atau disebut dengan *layer* sehingga struktur ini dapat secara otomatis mendapatkan karakteristik representasi dari sebuah data. Struktur CNN terdiri atas ekstraksi fitur yang terdiri atas *convolutional layer*, *pooling layer* dan *fully-connected layer*. Pada *convolutional layer* digunakan untuk mengekstraksi fitur dari data teks. sedangkan pada *pooling layer* mengurangi dimensi dan mengurangi waktu komputasi. Hasil fitur yang telah diekstraksi kemudian dimasukkan ke dalam *fully-connected layer* untuk proses klasifikasi (You et al., 2017). Untuk memahami bagaimana CNN bekerja secara lebih detail berikut penjelasan setiap *layer* yang ada pada CNN.

1. Convolutional Layer

Convolutional layer bertujuan untuk menangkap fitur dari sebuah *input*. *Convolutional layer* menerapkan matriks filter satu dimensi yang melewati tiap baris fitur dalam matriks. *Convolution* dijalankan pada data *input* dengan menggeser kernel konvolusi (*filter*) dan dilakukan perkalian matriks pada setiap *input* yang dilewati oleh *filter*. Untuk mendapatkan matriks pada *convolutional layer output*-nya menggunakan persamaan (2.1).

$$(N - m + 1) * (N - m + 1) \quad (2.1)$$

Pada persamaan (2.1) di atas menunjukkan jika matriks dengan ukuran $N * N$ sebagai *input*, dan $m * m$ sebagai *filter* ω . Maka pada Gambar 2.7 menjelaskan matriks pada *layer output* adalah $(6-3+1) * (6-3+1) = 4*4$. Pada hasil *output*-nya dilakukan perhitungan perkalian antara nilai matriks dengan nilai pada *filter* kemudian dijumlahkan. Pada *convolutional layer* terdapat *padding* sebagai parameter untuk menambahkan nilai pada satu kernel. Kemudian parameter *stride* yang digunakan sebagai nilai jumlah pergeseran kernel pada proses konvolusi.



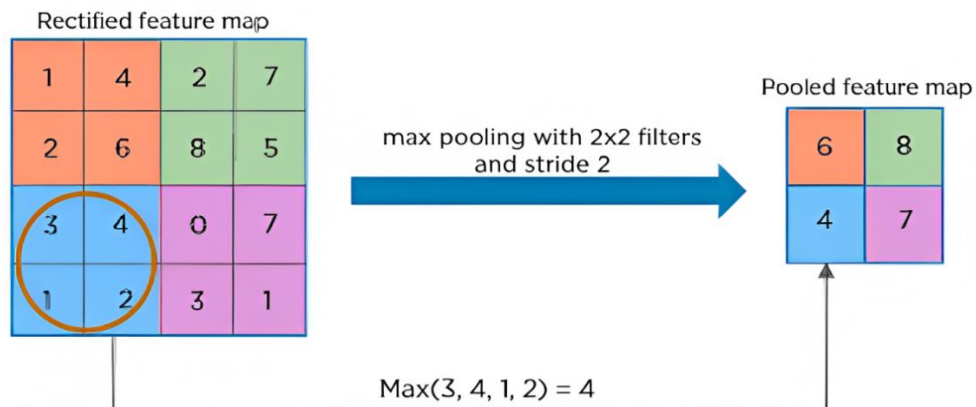
Gambar 2.7 Contoh Convolutional Layer Filter 3x3⁶

2. Pooling Layer

Pooling layer bertujuan untuk mengurangi dimensi pada matriks atau disebut juga *down sampling*. *Pooling layer* terdiri dari dua tipe yaitu *max pooling* dan *average pooling*. *Max pooling* adalah fungsi untuk mencari nilai terbesar pada ukuran *pooling*, sedangkan *average pooling* mencari nilai rata-rata dari keseluruhan nilai pada *pooling* (Albawi et al., 2017). Pada prinsipnya *pooling layer* terdiri dari *filter* dan *stride* yang bergeser pada

⁶ <https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/>

area *feature map*. Pada Gambar 2.8 menunjukkan bagaimana *max pooling* melakukan proses *down sampling* menggunakan filter berukuran 2×2 yang diaplikasikan dengan *stride* sebanyak 2 dan ukuran matriks *input* (4×4) Sehingga matriks yang dihasilkan pada *output* berukuran (2×2). Pada *max pooling* setiap pergerakan kernel akan diambil nilai tertinggi. Pada matriks ke-3 ditunjukkan dengan kotak berwarna biru berisikan anggota 3, 4, 1, 2 menghasilkan *output* dengan nilai tertinggi 4.

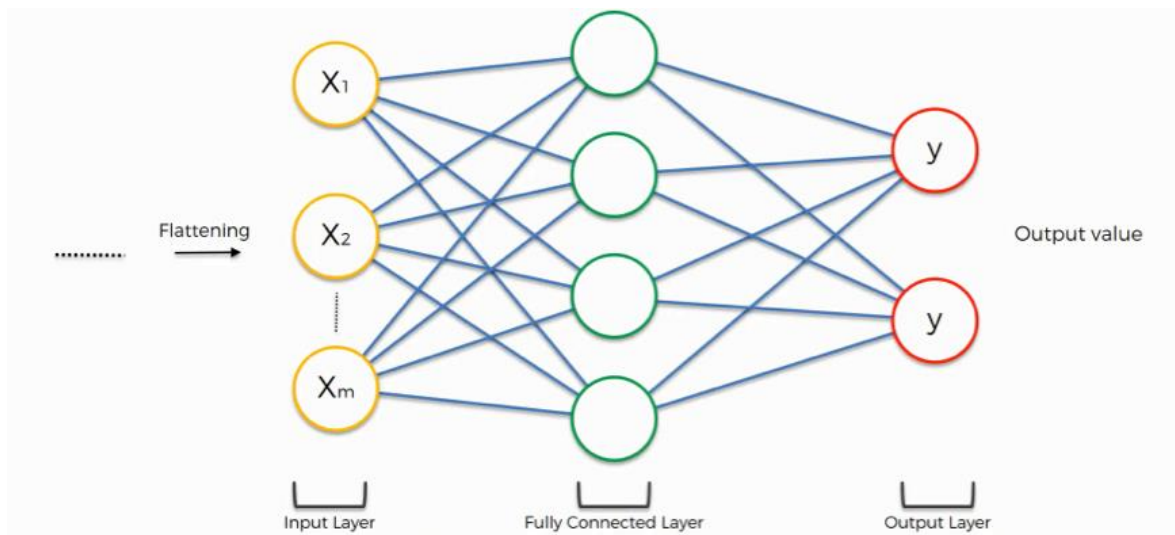


Gambar 2.8 Contoh Max Pooling⁷

3. Fully Connected Layer

Fully Connected Layer bertujuan untuk melakukan transformasi pada dimensi data agar dapat diklasifikasikan secara linier. Pada Gambar 2.9 menunjukkan bagaimana *fully connected layer* menghubungkan semua *neuron* aktivasi sebelumnya (*previous layer*) pada *neuron* pada lapisan *fully connected layer*. Setiap aktivasi dari lapisan sebelumnya masih berbentuk multidimensional array sehingga perlu diubah menjadi data satu dimensi dengan proses *flatten* atau *reshape* sebelum dapat dihubungkan ke semua *neuron* di lapisan *fully connected layer*. *Fully connected layer* pada intinya adalah sebuah *neural network multilayer perceptron* (MLP), yang memiliki beberapa *hidden layer*, *activation function*, *output layer*.

⁷ <https://www.simplilearn.com/tutorials/deep-learning-tutorial/convolutional-neural-network>

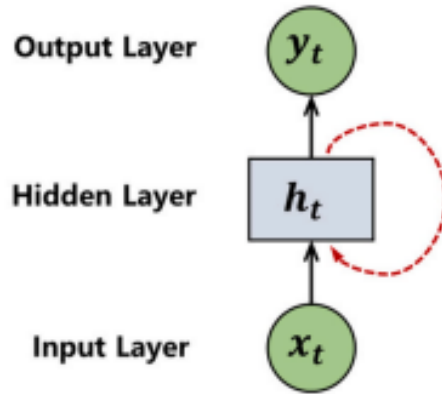


Gambar 2.9 Fully Connected Layer⁸

2.2.8 Recurrent Neural Network

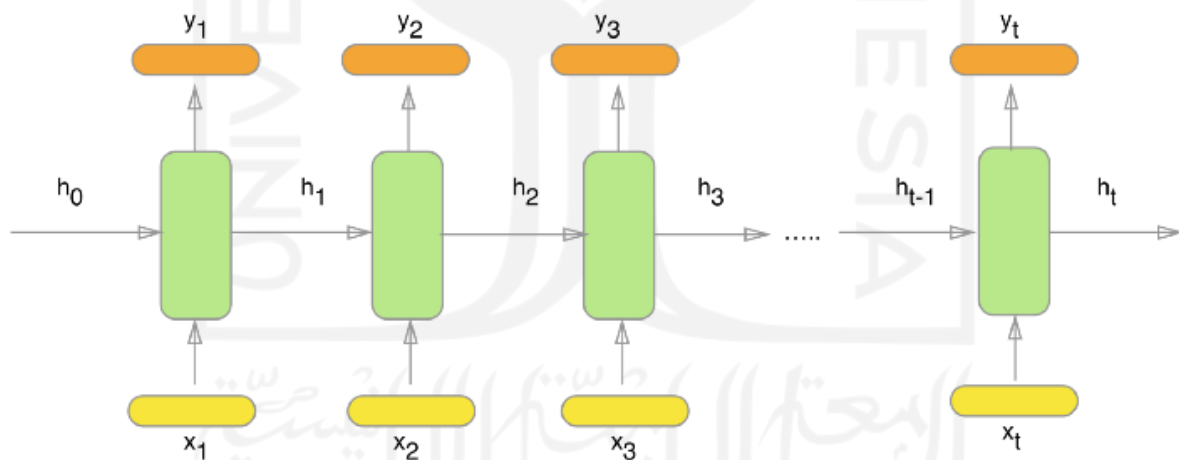
Recurrent Neural Network (RNN) adalah bagian dari *neural network* yang dirancang untuk memproses data berurutan atau disebut juga *sequential data*. RNN memproses urutan *input* dalam bentuk *sequence* dan tidak mengharuskan memiliki ukuran yang sama antara *input* dan *output* yang ditangani. Setiap *output* RNN telah dihitung dengan berulang kali dengan menjalankan fungsi yang sama (Onan, 2022). RNN merupakan model dinamis yang telah digunakan dalam domain yang beragam seperti musik, teks, dan data penangkapan gerak. RNN dapat dilatih untuk pembuatan *sequences* dengan memproses urutan data satu langkah pada satu waktu dan memprediksi apa yang terjadi selanjutnya. Dengan asumsi prediksi probabilistik, urutan baru dapat dihasilkan dari jaringan terlatih dengan pengambilan sampel berulang dari distribusi *output* jaringan, kemudian memasukkan sampel sebagai *input* pada langkah berikutnya (Graves, 2014). Gambar 2.10 menunjukkan struktur umum pada RNN.

⁸ <https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-4-full-connection>



Gambar 2.10 Struktur RNN

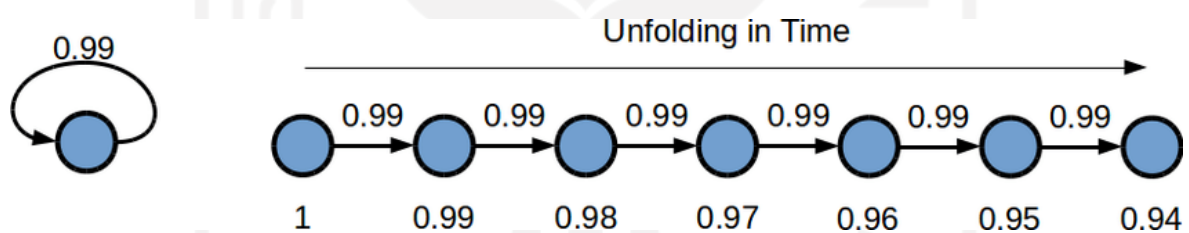
Pada Gambar 2.10 di atas menunjukkan bahwa didalam arsitekturnya RNN menyimpan informasi dari masa lalu dengan melakukan proses *looping*. Hal ini membuat RNN dapat menyimpan informasi dari masa lalu yang kemudian digunakan untuk menghasilkan *output* baru. *Output* yang dihasilkan tidak hanya merupakan fungsi dari sampel itu saja, tetapi juga berdasarkan *state internal* yang merupakan hasil dari pemrosesan sampel-sampel sebelumnya atau setelahnya seperti yang ditunjukkan pada Gambar 2.11.



Gambar 2.11 Proses Alur Kerja RNN

Pada Gambar 2.11 di atas variabel x_t adalah vektor *input* pada setiap langkahnya atau *time step*. Variable h_t adalah *hidden state* pada setiap *time step* yang akan dihitung dengan *hidden state* sebelumnya. Variabel y_t adalah *output* pada setiap *time step*. *Hidden state* atau disebut juga memori pada jaringan berfungsi untuk menyimpan hasil perhitungan yang telah dilakukan. Secara garis besar RNN menjalankan tugas yang sama pada setiap langkahnya dan memproses *output* dari informasi sebelumnya.

RNN mampu menangani deretan data yang saling terkait sebarang panjangnya. Namun prakteknya, RNN mengalami kesulitan jika deretan data sudah mencapai ukuran yang sangat besar. Penelitian yang dilakukan oleh Bengio, Simard dan Frasconi pada tahun 1994 menemukan beberapa alasan mendasar mengapa RNN mengalami kesulitan pada deretan data yang sangat panjang. Sehingga *hidden layer* yang terbentuk dari *input sequence* menjadi panjang dan menyulitkan proses pelatihan. Permasalahan ini kemudian disebut sebagai *vanishing gradient problem*. *Vanishing gradient problem* adalah keadaan yang disebabkan oleh pembelajaran data yang semakin lambat dan sulit karena nilai gradien yang terlalu kecil (Bengio et al., 1994). *Vanishing gradient problem* disebabkan karena nilai yang digunakan untuk memperbarui bobot atau disebut gradien semakin mengecil hingga lapisan terakhir membuat nilai bobot tidak berubah sehingga menyebabkan tidak pernah memperoleh hasil yang lebih baik atau konvergen. Sebaliknya gradien yang semakin membesar menyebabkan nilai bobot pada beberapa lapisan juga ikut membesar sehingga algoritma optimasi menjadi divergen atau disebut *exploding gradient*⁹. Pada Gambar 2.12 menjelaskan bahwa selama proses pelatihan nilai gradien mengalami penyusutan dan menyebabkan hilangnya nilai yang disimpan dalam setiap *time step* (Bayer, 2017).



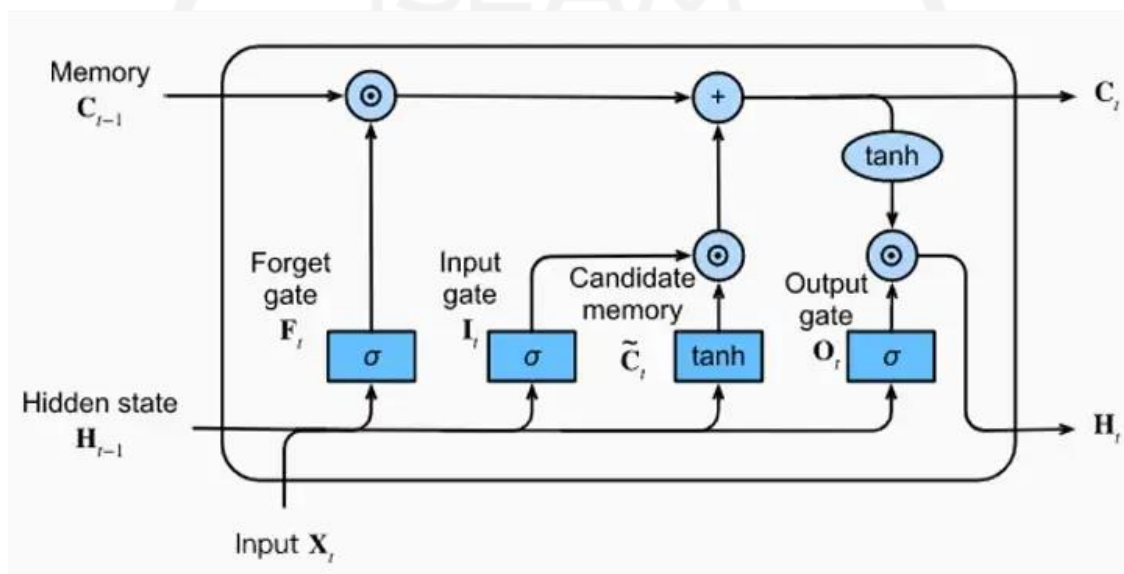
Gambar 2.12 Ilustrasi Vanishing Gradient Problem

2.2.9 Long Short Term Memory

Permasalahan RNN untuk deretan data yang sangat panjang, kemudian diselesaikan dengan algoritma *Long Short Term Memory* (LSTM) yang dikenalkan oleh (Hochreiter & Schmidhuber, 1997). LSTM merupakan pengembangan dari RNN yang dirancang khusus untuk mengatasi masalah *vanishing gradient problem*. Pada dasarnya arsitektur pada LSTM mirip dengan RNN, namun di dalam LSTM unit penjumlahan di *hidden layer* digantikan oleh blok memori. Blok LSTM juga dapat digabung dengan unit penjumlahan biasa, meskipun hal ini biasanya tidak diperlukan. *Output layer* yang sama dapat digunakan untuk jaringan LSTM seperti pada RNN (Graves, 2012).

⁹ <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

Arsitektur LSTM terdiri dari satu set subnet yang terhubung secara berulang, yang dikenal sebagai blok memori atau disebut *cell*. *Cell* dapat dianggap sebagai versi chip memori yang dapat dibedakan dalam komputer digital. Model LSTM menyaring informasi melalui struktur gerbang untuk mempertahankan dan memperbarui keadaan sel memori (J. Qiu et al., 2020). Pada Gambar 2.13 menunjukkan bahwa di dalam arsitektur LSTM terdapat tiga gerbang yang mengendalikan penggunaan dan memperbarui informasi terdahulu yaitu *input gate*, *forget gate*, dan *ouput gate*. *Cell* memori dan tiga gerbang dirancang untuk dapat membaca, menyimpan, dan memperbarui informasi terdahulu.



Gambar 2.13 Struktur LSTM¹⁰

Ide utama pada LSTM adalah jalur yang menghubungkan konteks lama C_{t-1} ke konteks baru C_t dengan tujuan meneruskan nilai pada C_{t-1} ke C_t dengan modifikasi minimal atau jika diperlukan saja. Langkah awal pada LSTM adalah memutuskan informasi yang akan disimpan atau dibuang pada C_{t-1} dengan menggunakan *forget gate*. Perhitungan nilai pada *forget gate* ditunjukkan pada persamaan (2.2)

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f) \quad (2.2)$$

Persamaan (2.2) di atas menunjukkan dimana f_t adalah nilai dari *forget gate*, W_f adalah bobot untuk nilai *input* pada waktu ke t , x_t adalah nilai *input* pada waktu ke t , h_{t-1} adalah nilai *output* dari waktu ke $t-1$, b_f adalah bias pada *forget gate* dan σ adalah fungsi sigmoid. Selanjutnya adalah proses *input gate*. Pada *input gate* akan menentukan nilai yang

¹⁰ https://d2l.ai/chapter_recurrent-modern/lstm.html

akan di ubah ke *cell state*. Perhitungan dari nilai pada *input gate* dan kandidat dari *cell state* dilakukan dengan menggunakan persamaan (2.3) dan (2.4).

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i) \quad (2.3)$$

$$\tilde{C}_t = \tanh(W_c.[h_{t-1}, x_t] + b_c) \quad (2.4)$$

Pada persamaan (2.3) menunjukkan dimana i_t adalah nilai dari *input gate*, W_i adalah bobot untuk nilai *input* pada waktu ke t , x_t adalah nilai *input* pada waktu ke t , h_{t-1} adalah nilai *output* dari waktu ke $t-1$ dan b_i adalah bias pada *input gate* dan σ adalah fungsi sigmoid. Persamaan (2.4) menunjukkan dimana \tilde{C}_t adalah nilai kandidat cell state, W_c adalah bobot untuk nilai *input* pada *cell* ke c , x_t adalah nilai *input* pada waktu ke t , h_{t-1} adalah nilai *output* dari cell c dan b_i adalah bias pada cell ke c dan \tanh adalah fungsi *hyperbolic tangent*. Langkah selanjutnya adalah memperbarui nilai *cell state* yang terbaru dengan menggunakan nilai-nilai yang sudah dihitung sebelumnya. Perhitungan pada *cell state* menggunakan persamaan (2.5).

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (2.5)$$

Pada persamaan (2.5) menunjukkan dimana C_t adalah nilai *memory cell state*, f_t adalah nilai *forget gate*, C_{t-1} adalah nilai *memory cell state* pada *cell* sebelumnya, i_t adalah nilai dari *input gate* dan \tilde{C}_t adalah nilai kandidat *memory cell state*. Setelah dihasilkan *memory cell state* yang baru, nilai dari *output gate* dapat dihitung dengan menggunakan persamaan (2.6) dan (2.7).

$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o) \quad (2.6)$$

$$h_t = o_t * \tanh(C_t) \quad (2.7)$$

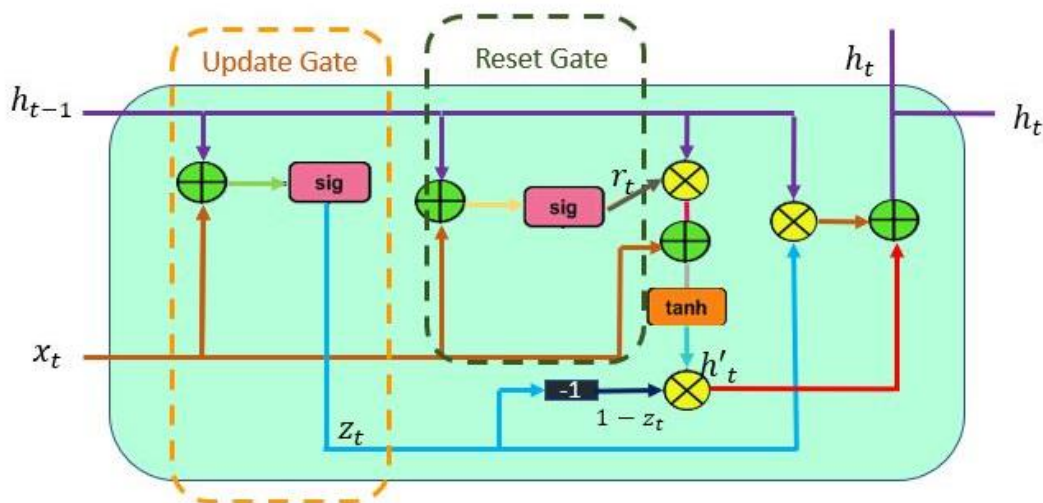
Pada persamaan (2.6) menunjukkan dimana o_t adalah nilai dari *output gate*, W_o adalah bobot untuk nilai *input* pada waktu ke t , h_{t-1} adalah nilai *output* dari waktu ke $t-1$, x_t adalah nilai *input* pada waktu ke t dan b_o adalah bias pada *output gate* dan σ adalah fungsi sigmoid. Persamaan (2.7) menunjukkan dimana h_t adalah *output* final, o_t adalah nilai *output gate*, C_t adalah nilai *memory cell state* yang baru dan \tanh adalah fungsi *hyperbolic tangent*.

2.2.10 Gated Recurrent Unit

Gated Recurrent Unit (GRU) merupakan salah satu tipe dari RNN atau varian lain dari LSTM yang digunakan untuk klasifikasi *sequential* data atau *time series*. Seperti pada LSTM, GRU juga dirancang untuk dapat memecahkan permasalahan *vanishing gradient problem* (Luo, 2019). Secara arsitektur GRU dan LSTM memiliki kesamaan dengan mengadopsi sistem gerbang (*gate*) dalam memproses data. Mekanisme gerbang terbukti

mampu menangani permasalahan pada RNN yaitu *vanishing gradient problem* (Dey & Salem, 2017).

GRU diperkenalkan pertama kali oleh Cho et al pada tahun 2014. Pada penelitiannya dijelaskan bahwa GRU mampu mempelajari representasi frasa bahasa secara semantik dan sintaksis yang diperkenalkan pada tugas terjemahan mesin (Cho et al., 2014). Pada Gambar 2.14 menunjukkan arsitektur pada GRU. Di dalam arsitekturnya GRU mengalami penyederhanaan jika dibandingkan pada LSTM karena GRU hanya menggunakan dua gerbang yaitu *update gate* dan *reset gate*. *Update gate* bertindak mirip dengan *forget gate* dan *input gate* pada LSTM.



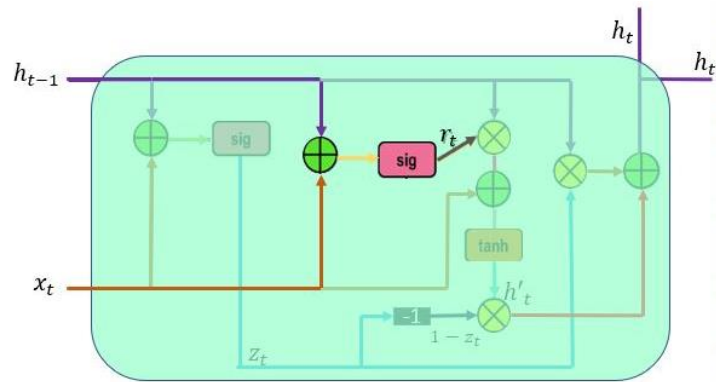
Gambar 2.14 Struktur GRU¹¹

Pada arsitektur GRU alur informasi melewati beberapa tahapan. Pertama, *reset gate* berfungsi untuk memutuskan berapa banyak informasi masa lalu yang akan dilupakan. Pada Gambar 2.15 menunjukkan alur *reset gate* pada GRU. *Reset gate* terdiri dari satu lapisan *neural network* dengan fungsi aktivasi *sigmoid* yang digunakan untuk menyaring nilai antara 0 dan 1. Perhitungan pada *reset gate* menggunakan persamaan (2.8).

$$r_t = \sigma(W_r.[h_{t-1}, x_t] + b_r) \quad (2.8)$$

Pada persamaan (2.8) menunjukkan r_t adalah nilai dari *reset gate*, t adalah *time step*, σ adalah fungsi sigmoid, W_r adalah nilai bobot matriks pada *reset gate*, x_t adalah nilai *input* pada waktu ke t , h_{t-1} adalah nilai *output* dari waktu ke $t-1$ atau nilai pada *hidden state* sebelumnya dan b_r adalah bias pada *reset gate*.

¹¹ <https://www.pluralsight.com/guides/lstm-versus-gru-units-in-rnn>

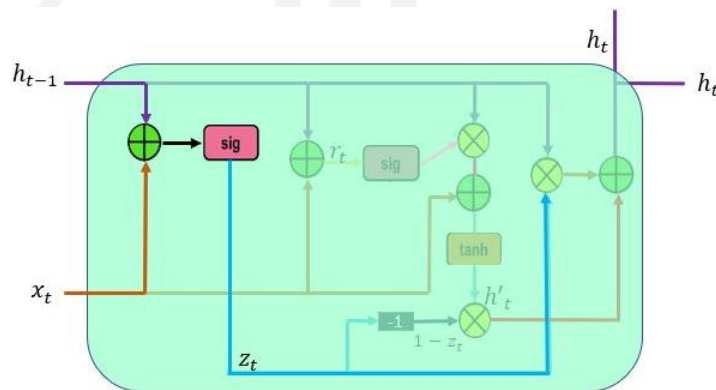


Gambar 2.15 Reset gate Pada GRU¹²

Kedua adalah *update gate* berfungsi untuk memutuskan informasi apa yang harus dibuang dan informasi baru apa yang akan ditambahkan. Pada *update gate* nilai didapatkan menggunakan persamaan (2.9).

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z) \quad (2.9)$$

Pada persamaan (2.9) menunjukkan z_t adalah nilai dari *update gate*, t adalah *time step*, σ adalah fungsi sigmoid, W_z adalah nilai bobot matriks pada *update gate*, x_t adalah nilai *input* pada waktu ke t , h_{t-1} adalah nilai *output* dari waktu ke $t-1$ atau nilai pada *hidden state* sebelumnya dan b_z adalah bias pada *update gate*. Pada Gambar 2.16 menunjukkan alur *update gate* pada GRU.



Gambar 2.16 Update Gate Pada GRU¹³

Selanjutnya ketiga adalah *current memory content*. Pada tahap ini akan menghasilkan nilai baru pada *memory content* dengan menggunakan beberapa langkah perhitungan sebagai berikut¹⁴ :

¹² <https://www.pluralsight.com/guides/lstm-versus-gru-units-in-rnn>

¹³ Ibid

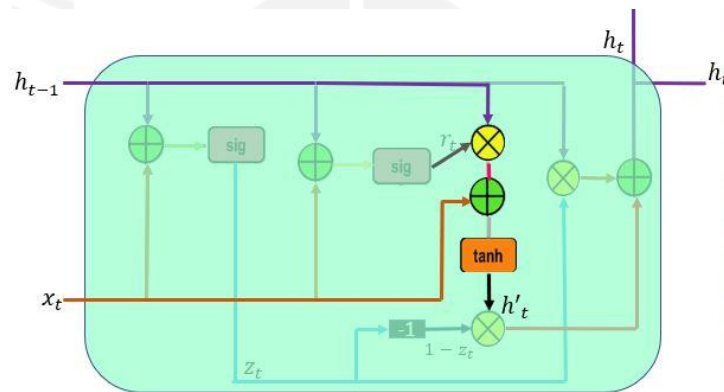
¹⁴ <https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be>

1. Mengalikan vektor *input* dan *hidden state* dengan bobot
2. Menggunakan *element-wise multiplication* (Hadamard) untuk menghitung nilai antara *reset gate* dan *hidden state* sebelumnya.
3. Menjumlahkan hasil pada langkah 1 dan 2
4. Menggunakan fungsi aktivasi tanh untuk mendapatkan nilai akhir

Perhitungan pada *current memory content* menggunakan persamaan (2.10).

$$h'_t = \tanh(W_h[r_t \odot h_{t-1}, x_t] + b_h) \quad (2.10)$$

Pada persamaan (2.10) menunjukkan h'_t adalah nilai dari *current memory*, t adalah *time step*, W_h adalah nilai bobot matriks pada *reset gate*, x_t adalah nilai *reset gate* pada waktu ke t , x_t adalah nilai *input* vector, h_{t-1} adalah nilai pada *hidden state* sebelumnya, \odot adalah fungsi hadamard dan b_h adalah bias pada waktu ke t . Pada Gambar 2.17 menunjukkan alur *current memory content* pada GRU.



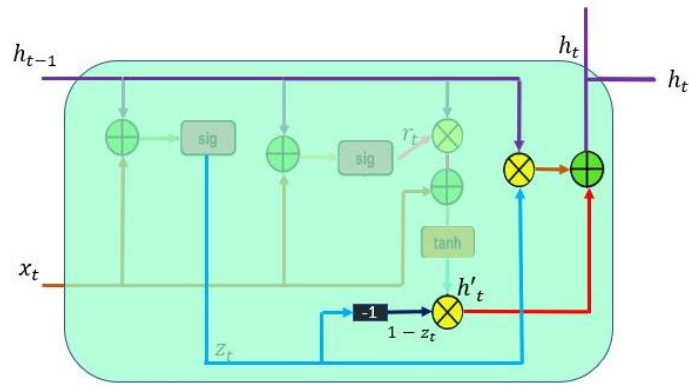
Gambar 2.17 Current Memory Content Pada GRU¹⁵

Selanjutnya tahap terakhir adalah *final memory content*. Seperti ditunjukkan Pada Gambar 2.18. Pada tahap ini jaringan akan menghitung antara nilai *update gate* menggunakan fungsi hadamard dengan nilai *hidden state* sebelumnya, kemudian dijumlahkan dengan hasil perhitungan antara nilai *update gate* menggunakan fungsi hadamard dengan nilai *current memory*. Perhitungan pada *final memory content* menggunakan persamaan (2.11).

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot h'_t \quad (2.11)$$

Pada persamaan (2.11) menunjukkan h_t adalah nilai dari final memory, t adalah *time step*, \odot adalah fungsi Hadamard, z_t adalah nilai *update gate* pada waktu ke t , h_{t-1} adalah nilai pada *hidden state* sebelumnya dan h'_t adalah nilai pada *current memory*.

¹⁵ <https://www.pluralsight.com/guides/lstm-versus-gru-units-in-rnn>



Gambar 2.18 Final Memory Content Pada GRU¹⁶

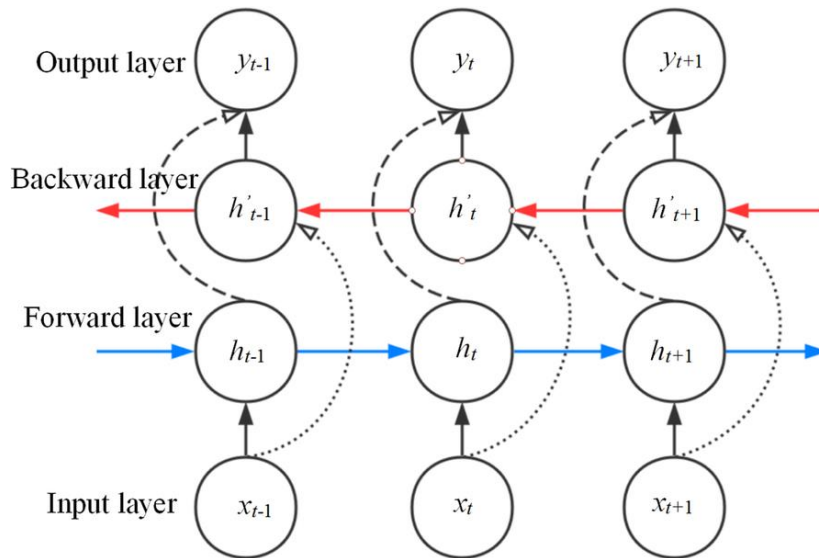
Menurut (Chung et al., 2014) arsitektur GRU menghasilkan model yang setara dengan LSTM, akan tetapi GRU secara komputasi lebih efisien. Hal ini disebabkan karena GRU menggunakan dua gerbang sehingga pelatihan dan konvergensinya lebih cepat dari LSTM. Dan saat ini GRU telah banyak digunakan dalam aplikasi NLP karena memiliki komputasi yang lebih sederhana.

2.2.11 Bidirectional

Pada dasarnya arsitektur RNN, LSTM dan GRU menerapkan pemrosesan satu arah atau disebut *unidirectional*. Pemrosesan ini memiliki keterbatasan hanya dapat merepresentasikan data pada masa lalu untuk menghasilkan *output*. *Output* yang dihasilkan dari pemrosesan ini terkadang memiliki ambiguitas yang cukup tinggi. Pada tugas NER, mempelajari data masa lalu dan juga mempelajari data masa depan sangat berguna untuk dapat memiliki pemahaman konteks yang lebih baik (Gridach & Haddad, 2018).

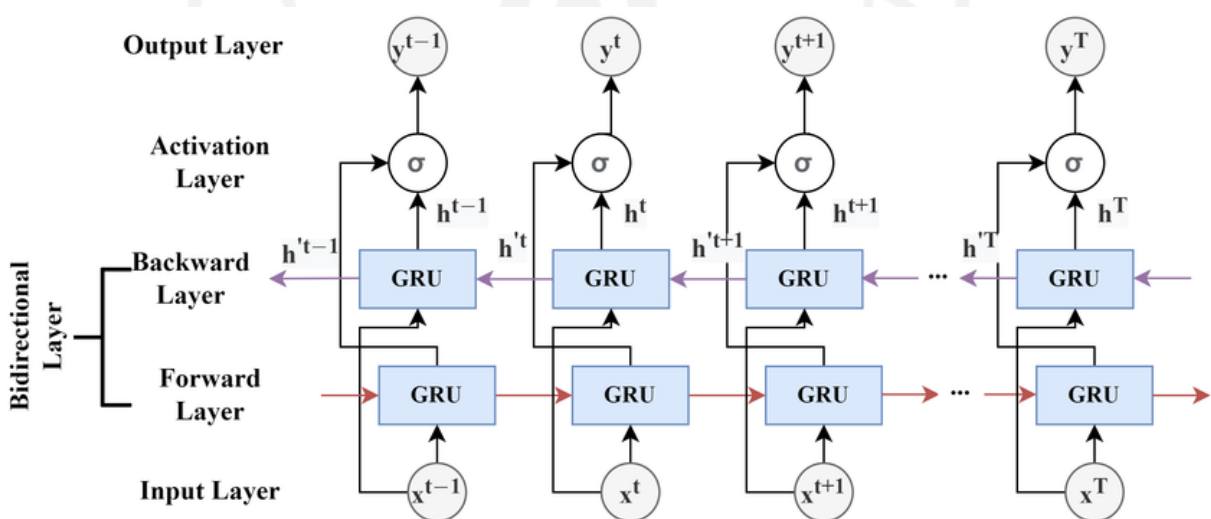
Pemrosesan dua arah atau disebut *bidirectional* adalah pengembangan dari *unidirectional* yang dirancang untuk dapat menangkap informasi pada masa lalu dan informasi masa depan. Teknik arsitektur *bidirectional* pertama kali diperkenalkan oleh Schuster dan Paliwal pada tahun 1997 dengan menggunakan algoritma RNN. Pada Gambar 2.19 menunjukkan arsitektur *bidirectional* RNN. Dapat dilihat bahwa ide dasar *bidirectional* adalah pada setiap urutan *input* dilewatkan melalui lapisan maju (*forward layer*) dan lapisan mundur (*backward layer*), dan kemudian *output* keduanya terhubung dalam lapisan *output* yang sama (Schuster & Paliwal, 1997).

¹⁶ <https://www.pluralsight.com/guides/lstm-versus-gru-units-in-rnn>



Gambar 2.19 Arsitektur Bidirectional RNN

Pada perkembangannya *bidirectional* telah banyak digunakan dalam tugas NLP dengan menggunakan beberapa algoritma *deep learning*. (Gridach & Haddad, 2018) menjelaskan bahwa *bidirectional* GRU (BiGRU) dapat menangkap konteks informasi pada teks antar kalimat lebih baik, sehingga informasi fitur pada seluruh teks dapat lebih dimanfaatkan dan entitas dapat dikenali secara efektif. Arsitektur yang digunakan terdiri dari dua *layer* GRU dan *layer* CRF yang digunakan untuk menggabungkan vektor *output* dari kedua *layer* GRU. Pada Gambar 2.20 menunjukkan contoh penerapan *bidirectional* menggunakan GRU.



Gambar 2.20 Penerapan Bidirectional GRU

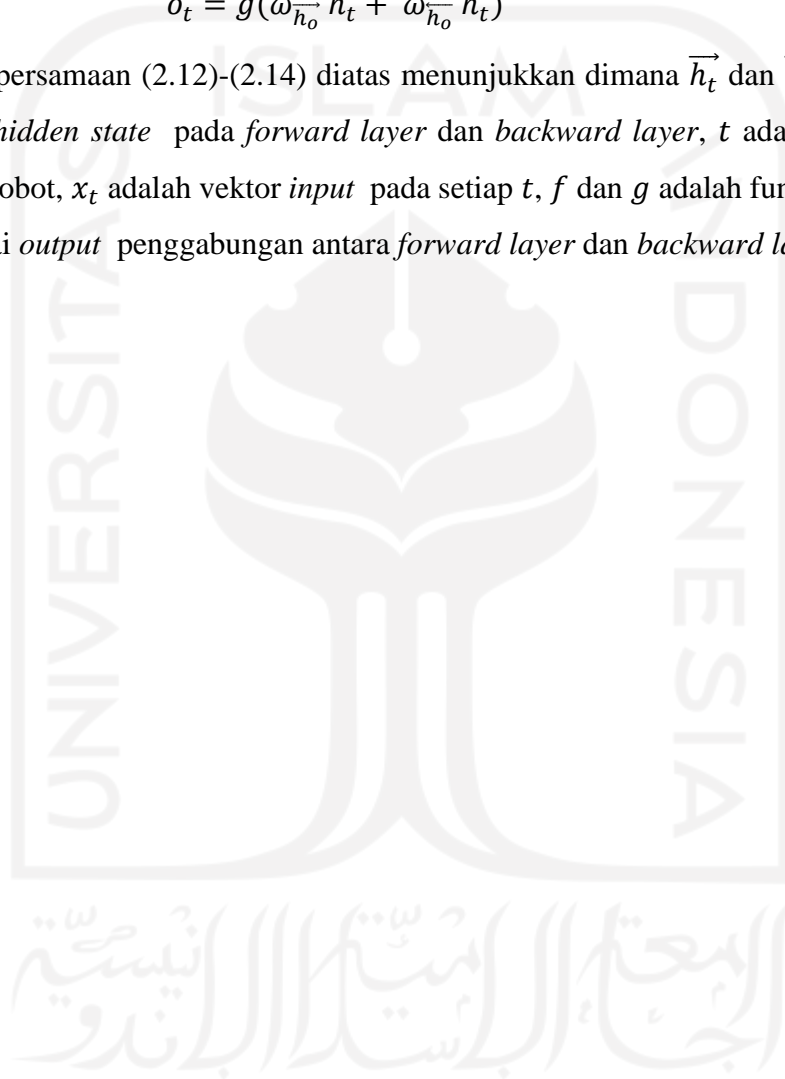
Pada Gambar 2.20 di atas menunjukkan bahwa pada arsitektur bidirectional GRU *forward layer* akan menghitung nilai output pada *hidden state* dari arah depan ke belakang. Sedangkan *backward layer* akan menghitung nilai *output* pada *hidden state* dari arah belakang ke depan. Perhitungan pada BiGRU menggunakan persamaan (2.12) – (2.14).

$$\vec{h}_t = f(\omega_{\vec{h}_t} x_t + \omega_{\vec{h}_t} \vec{h}_{t-1}) \quad (2.12)$$

$$\overleftarrow{h}_t = f(\omega_{\overleftarrow{h}_t} x_t + \omega_{\overleftarrow{h}_t} \overleftarrow{h}'_{t+1}) \quad (2.13)$$

$$o_t = g(\omega_{\vec{h}_o} \vec{h}_t + \omega_{\overleftarrow{h}_o} \overleftarrow{h}_t) \quad (2.14)$$

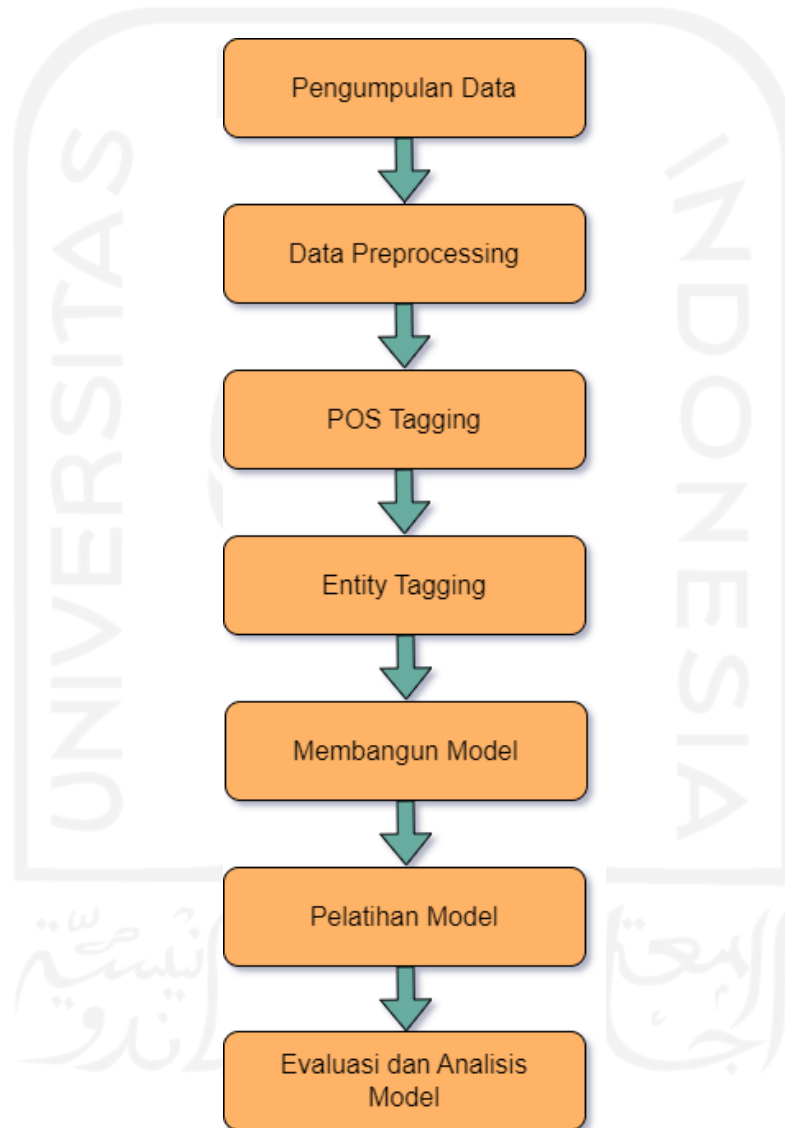
Pada persamaan (2.12)-(2.14) diatas menunjukkan dimana \vec{h}_t dan \overleftarrow{h}_t adalah vektor *output* dari *hidden state* pada *forward layer* dan *backward layer*, t adalah *time step*, ω adalah nilai bobot, x_t adalah vektor *input* pada setiap t , f dan g adalah fungsi aktivasi, dan o_t adalah nilai *output* penggabungan antara *forward layer* dan *backward layer*.



BAB 3

Metodologi

Proses penelitian ini menggunakan metodologi, sehingga dapat diketahui urutan langkah-langkah secara sistematis yang dapat dijadikan sebagai pedoman dalam menyelesaikan permasalahan. Adapun langkah-langkah pada penelitian ini dapat dilihat pada Gambar 3.1.



Gambar 3.1 Metodologi Penelitian

Pada Gambar 3.1 di atas menjelaskan alur metodologi yang akan dilakukan dari awal hingga akhir pada penelitian ini. Penjelasan pada setiap tahapan diatas akan dijelaskan pada sub bab dibawah ini.

3.1 Sumber Data

Penelitian ini menggunakan data pada penelitian (Fudholi et al., 2022). Data berhasil dikumpulkan melalui tahapan awal dengan mengidentifikasi beberapa situs web kesehatan Indonesia yang menyediakan daftar nama obat. Alamat situs web sumber nama obat dapat dilihat pada Tabel 3.1. Kemudian daftar nama obat dari situs web digunakan sebagai kata kunci untuk mengambil informasi yang lebih detail dengan menggunakan cuplikan Google. Untuk menampilkan cuplikan singkat pada Google ditambahkan kata kunci sederhana “kandungan” yang berarti “isi” kemudian diikuti nama obatnya. Sebagai contoh, kata kunci “kandungan paramex” memberikan cuplikan singkat “Tiap tablet Paramex mengandung 250 mg paracetamol, 150 mg propifenazon, 50 mg kafein, dan 1 mg dexchlorpheniramine. Paramex jenis ini digunakan untuk meredakan sakit kepala dan sakit gigi.”. Jumlah data yang dikumpulkan sebanyak 18.075 kalimat yang berisi tentang kata yang mengandung obat.

Tabel 3.1 Daftar Alamat Website Sumber Nama Obat

Nama Website	Alamat Website
Klik Dokter	https://www.klikdokter.com/obat
Sehatq	https://www.sehatq.com/obat
Hello Sehat	https://www.hellosehat.com/obatan-suplemen/obat

3.2 Data Preprocessing

Data *preprocessing* merupakan proses untuk membersihkan data dengan mengurangi *noise*, menyeragamkan bentuk kata dan meminimalkan variansi pada data teks. Tujuan dari data *preprocessing* adalah untuk menyiapkan data teks menjadi lebih terstruktur dan sesuai untuk model yang digunakan (Hidayatullah & Ma’arif, 2017). Pada tahap *preprocessing* ini akan dilakukan beberapa proses diantaranya :

1. Menghapus tanggal

Proses ini dilakukan untuk menghapus informasi tanggal karena pada pada penelitian ini informasi tanggal tidak diperlukan.

2. Menghapus *punctuation* atau tanda baca

Proses ini dilakukan untuk menghapus tanda baca atau karakter khusus pada data teks seperti koma, titik, hashtags, tanda kutip dan lain-lain.

3. Menghapus karakter yang tidak sesuai standar *American Standard Code for Information Interchange* (ASCII)

ASCII merupakan kode standar yang digunakan dalam pertukaran informasi dalam komputer. Proses ini dilakukan untuk menghapus karakter seperti “Â” dan “â€”.

4. Normalisasi teks

Proses ini dilakukan untuk mengubah beberapa singkatan pengukuran kedalam pengukuran bentuk aslinya seperti “mg” menjadi “miligram”.

5. *Tokenizing*

Proses *tokenizing* adalah proses pemisahan kalimat teks menjadi potongan kata atau disebut token.

6. Menghapus *Stopword*

Proses ini dilakukan untuk menghilangkan kata yang tidak penting seperti ini, itu, jadi, dan lain-lain.

3.3 POS Tagging

Pada proses ini akan menetapkan setiap kata yang terdapat dalam kalimat kedalam kelas kata (*tag*) secara gramatikal berdasarkan definisi dan konteks. POS *tagging* bertujuan untuk mengkategorikan kelas kata menjadi kata benda, kata sifat, kata kerja dan lain-lain (Kumawat & Jain, 2015).

Pada penelitian ini proses POS *tagging* menggunakan pustaka Kumparan NLP. Pustaka Kumparan NLP menyediakan fungsi POS *tagging* untuk Bahasa Indonesia yang tersedia untuk umum (Juwantara et al., 2021). Pada penelitian sebelumnya yang dilakukan oleh (Fatimah et al., 2021). Pustaka Kumparan NLP digunakan untuk mengekstraksi entitas lokasi karena terdapat nama lokasi di Indonesia yang memiliki lebih dari satu kata seperti DKI Jakarta dan Jawa Tengah. Pada Tabel 3.2 menunjukkan kumpulan POS *tagging* Bahasa Indonesia pada pustaka Kumparan NLP.

Tabel 3.2 Kumpulan POS Tag pada Pustaka Kumparan NLP

Tag	Deskripsi	Contoh
ADV	Adverbs. Includes adverb, modal, and auxiliary verb	sangat, hanya, justru, boleh, harus, mesti
CC	Coordinating conjunction. Coordinating conjunction links two or more syntactically equivalent parts of a sentence. Coordinating conjunction can link independent clauses, phrases, or words.	dan, tetapi, atau
DT	Determiner/article. A grammatical unit which limits the potential referent of a noun phrase, whose basic role is to mark noun phrases as either definite or indefinite.	para, sang, si, ini, itu, nya
FW	Foreign word. Foreign word is a word which comes from foreign language and is not yet included in Indonesian dictionary	workshop, business, e-commerce
IN	Preposition. A preposition links word or phrase and constituent in front of that preposition and results prepositional phrase.	dalam, dengan, di, ke
JJ	Adjective. Adjectives are words which describe, modify, or specify some properties of the head noun of the phrase	bersih, panjang, jauh, marah
NEG	Negation	tidak, belum, jangan
NN	Noun. Nouns are words which refer to human, animal, thing, concept, or understanding	meja, kursi, monyet, perkumpulan
NNP	Proper Noun. Proper noun is a specific name of a person, thing, place, event, etc	Indonesia, Jakarta, Piala Dunia, Idul Fitri, Jokowi

NUM	Number. Includes cardinal and ordinal number	9876, 2019, 0,5, empat
PR	Pronoun. Includes personal pronoun and demonstrative pronoun	saya, kami, kita, kalian, ini, itu, nya, yang
RP	Particle. Particle which confirms interrogative, imperative, or declarative sentences	pun, lah, kah
SC	Subordinating Conjunction. Subordinating conjunction links two or more clauses and one of the clauses is a subordinate clause.	sejak, jika, seandainya, dengan, bahwa
SYM	Symbols and Punctuations	+,%,@
UH	Interjection. Interjection expresses feeling or state of mind and has no relation with other words syntactically.	ayo, nah, ah
VB	Verb. Includes transitive verbs, intransitive verbs, active verbs, passive verbs, and copulas.	tertidur, bekerja, membaca
ADJP	Adjective Phrase. A group of words headed by an adjective that describes a noun or a pronoun	sangat tinggi
DP	Date Phrase. Date written with whitespaces	1 Januari 2020
NP	Noun Phrase. A phrase that has a noun (or indefinite pronoun) as its head	Jakarta Pusat, Lionel Messi
NUMP	Number Phrase	10 juta
VP	Verb Phrase. A syntactic unit composed of at least one verb and its dependents	tidak makan

Berikut adalah contoh kalimat yang berisi tentang detail obat.

“Obat paramex mengandung 250 mg paracetamol dan 1 mg dexchlorpheniramine,”.

Hasil POS *tagging* dari kalimat diatas ditunjukkan pada Tabel 3.3. Seperti kata “paramex” ditandai dengan NNP karena dikenali sebagai kata benda spesifik. Kata “250” ditandai dengan NUM karena dikenali sebagai angka dan kata “mg” ditandai sebagai NN karena dikenali sebagai kata benda umum.

Tabel 3.3 Proses POS Tagging

No	Token	POS
1	Obat	NN
2	paramex	NNP
3	mengandung	VB
4	250	NUM
5	mg	NN
6	paracetamol	NN
7	dan	CC
8	1	NUM
9	mg	NN
10	dexchlorpheniramine	NN
11	,	SYM

3.4 Entity Tagging

Pada tahap ini akan membuat label entitas pada setiap kata dan disusun berdasarkan hasil klasifikasi kata pada POS *tagging*. Entitas yang digunakan pada penelitian ini adalah (OBAT) untuk menunjukkan nama obat, (KANDUNGAN) untuk menunjukkan bahan obat, (KOMPOSISI) untuk menunjukkan jumlah atau dosis obat, dan (O) untuk menunjukkan kata yang tidak memiliki entitas.

Untuk dapat mengekstrak beberapa informasi penting dari teks dilakukan proses *chunking* dan *chinking*. *Chunking* dilakukan setelah proses POS *tagging* karena membutuhkan POS *tagging* sebagai *input* untuk mengekstrak frasa. Pada Gambar 3.2 menunjukkan proses pembuatan aturan pada kelas kata menggunakan *regular expression* atau dikenal dengan “*grammar*”. Proses ini bertujuan untuk mendefinisikan pola *chunking*. Pola *chunking* yang digunakan pada penelitian ini menggunakan pola yang biasa digunakan dalam penulisan teks tentang obat dalam Bahasa Indonesia. Contoh *chunking* sebagai berikut

: {<NUM><NN|NUM><VB>*}. Pola *chunking* ini diawali dengan NUM kemudian diikuti dengan NN atau NUM kemudian diikuti VB atau *tag* lainnya. Digunakan untuk menandai sebuah kata ke dalam entitas KOMPOSISI. Pola *chinking* juga diterapkan dalam pembentukan entitas KOMPOSISI. Pola *chinking* diawali dengan karakter “{” dan diakhiri dengan “}”. Contoh *chinking* sebagai berikut : }<NUM><NN.*|NUM><VB>{. Pola ini diawali dengan NUM kemudian diikuti NN dan *tag* lainnya atau NUM kemudian diikuti VB.

```

grammar = r"""
KOMPOSISI: {<NUM><NN|NUM><VB>*}
           }<NUM><NN.*|NUM><VB>{
KANDUNGAN: {<VB><NN><NNP>*}
           }<VB><NN>{
KANDUNGAN: {<NNP>+?<VB>*<NNP>*}
           }<VB>*<NNP>*{
KANDUNGAN: {<VB>*<NN>*<VB><NNP>*}
           }<VB>*<NN>+?<VB>{
KANDUNGAN: {<VB>*<NN>*<VB><NN>+?<NNP>?}
           }<VB>*<NN>*<VB>{
OBAT: {<VB>*<NN>?<NNP>*}
      }<VB>*<NN>+?{
OBAT: {<NN.*>?<VB><VB>}
      }<VB><VB>{
OBAT: {<NNP>*<ADV>*<VB>}
      }<ADV>*<VB>{
OBAT: {<NNP>*<NN>*<VB><NN>*<VB>}
      }<VB><NN>*<VB>{
OBAT: {<NNP>+?<NUM><NNP|NUM|NN>}
OBAT: {<NNP|NN|FW>*<VB><KOMPOSISI>}
      }<VB><KOMPOSISI>{
OBAT: {<NN.*>*<VB><NUM>}
      }<VB><NUM>{
OBAT: {<NN.*|FW>*<VB><KANDUNGAN>}
      }<VB><KANDUNGAN>{
KANDUNGAN: {( <NNP><FW|NNP>*<KOMPOSISI> ) | ( <FW><NNP|FW>*<NN><KOMPOSISI> ) }
           }<KOMPOSISI>{
KANDUNGAN: {<OBAT><VB><KOMPOSISI><NN.*|FW>*}
           }<OBAT><VB><KOMPOSISI>{
.....

```

Gambar 3.2 Pola Chunking dan Chinking dalam Grammar

Selanjutnya didalam proses *chunking* penelitian ini menggunakan format BIO. Format BIO merupakan format yang umum digunakan sebagai penanda entitas. Tujuannya adalah untuk mendefinisikan suatu entitas yang menjadi bagian pada entitas lainnya. Pada Gambar 3.3 menunjukkan format dataset yang akan digunakan untuk proses pelatihan

model. Format dataset terdiri dari empat fitur di dalamnya seperti token, POS *tag*, *chunk tag* dan *entity tag*. Pada format ini kata “Paramex” ditandai sebagai B-OBAT karena Paramex merupakan kata awal yang dikenali sebagai produk obat. Kata “mengandung” ditandai sebagai O karena tidak dikenali dalam aturan *grammar*. Kata “Paracetamol” ditandai sebagai B-KANDUNGAN karena merupakan kata awal yang dikenali sebagai bahan obat. “250 miligram” ditandai sebagai B-KOMPOSISI dan I-KOMPOSISI karena kedua kata tersebut dikenali sebagai dosis obat berdasarkan aturan *grammar*. Kata “250” ditandai sebagai B-KOMPOSISI karena dikenal sebagai awal dosis obat dan diikuti dengan “miligram” yang ditandai sebagai I-KOMPOSISI karena terletak di dalam frasa yang sama.

Paramex	NNP	B-NP	B-OBAT
mengandung	VB	B-VB	O
Paracetamol	NNP	B-NP	B-KANDUNGAN
250	NUM	O	B-KOMPOSISI
miligram	NN	B-NP	I-KOMPOSISI

Gambar 3.3 Format Dataset Entity Tagging

3.5 Distribusi Data

Setelah proses *entity tagging* terbentuk selanjutnya melakukan distribusi data sesuai format dataset. Dataset ini yang akan menjadi data masukan dalam proses pelatihan model. Pembagian dataset dilakukan dengan membagi dataset menjadi dua bagian yaitu *training set* dan *test set*. (Gholamy et al., 2018) menjelaskan bahwa jika dataset dibagi sebagai berikut, 70%-80% data sebagai *training set* dan 20-30% data digunakan sebagai *test set*, maka model dapat memperoleh hasil terbaik. Pada penelitian ini menggunakan *training set* 70% dan *test set* 30% dari keseluruhan data.

3.6 Glove Pre-trained Word Embedding

Sebelum data teks diolah dengan model *deep learning*, data teks harus terlebih dahulu diubah ke dalam bentuk vektor angka. Proses pengubahan data teks ke dalam bentuk vektor angka disebut dengan *word embedding*. *Word embedding* adalah proses konversi kata ke dalam bentuk vektor yang merepresentasikan makna kata semantik atau kata yang memiliki konteks yang sama dengan kata yang lain (Bengio et al., 2000). Sebagai contoh kata “marah”

dan “mengamuk” lebih memiliki kedekatan jika dibandingkan dengan kata “marah” dan “bahagia”. *Word embedding* dapat dibuat langsung dari dataset yang dimiliki atau menggunakan *pre-trained word embedding* yang telah tersedia. *Pre-trained word embedding* adalah *word embedding* yang telah dilatih menggunakan dataset yang besar pada domain permasalahan tertentu yang dapat digunakan untuk menyelesaikan permasalahan lain yang serupa (Mikolov et al., 2013).

Salah satu metode dalam *word embedding* adalah *Global Vectors for Word Representation* (GloVe). GloVe pertama kali diperkenalkan oleh Pennington, Socher dan Manning pada tahun 2014. GloVe menggunakan metode faktorisasi matriks, matriks yang mewakili kemunculan atau ketiadaan kata-kata dalam suatu dokumen. GloVe menghasilkan *word embeddings* yang sangat baik. Hal ini terbukti dengan tingkat keberhasilan hingga 75% pada tes analogi kata (Pennington et al., 2014). GloVe memperoleh hubungan semantik antar kata berdasarkan *co-occurrence* matriks. Sebagai contoh diberikan korpus kata-kata dalam V , *co-occurrence* matriks sebagai X yang akan membentuk matriks $V \times V$. Di mana i adalah baris dan j adalah kolom dari X . X_{ij} adalah jumlah kemunculan kata j pada konteks kata. Gambar 3.4 menunjukkan contoh *co-occurrence* matriks pada kalimat.

	the	cat	sat	on	mat
the	0	1	0	1	1
cat	1	0	1	0	0
sat	0	1	0	1	0
on	1	0	1	0	0
mat	1	0	0	0	0

Gambar 3.4 Co-occurrence Matriks pada Kalimat¹⁷

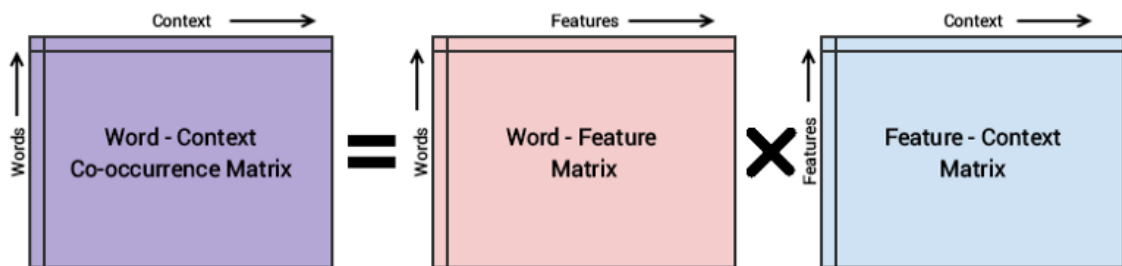
Pada gambar 3.4 di atas terlihat bahwa i (the) dengan j (cat) memiliki nilai koefisien 1 yang berarti kedua kata muncul secara bersamaan dalam ukuran *window*. Jika panjang *window* adalah s , maka $\mu = \{1, 2, \dots, s\}$ adalah jarak kata konteks dengan kata utama

¹⁷ <https://towardsdatascience.com/light-on-math-ml-intuitive-guide-to-understanding-glove-embeddings-b13b4f19c010>

sepanjang *window*. Untuk menghitung jumlah kemungkinan kata j yang muncul pada konteks kata i , menggunakan persamaan (3.1).

$$P_{ik} = X_{ik}/X_i \quad (3.1)$$

Pada persamaan (3.1) di atas P_{ik} menunjukkan probabilitas terjadinya kata i dan k secara bersamaan yang dihitung dengan membagi berapa kali i dan k muncul bersama X_{ik} dengan jumlah total kata i muncul dalam korpus X_i . Pada Gambar 3.5 menunjukkan model konseptual pada GloVe. Dimana nilai perhitungan pada *word-context matrix*s didapatkan dari hasil perhitungan nilai *word-feature* dan *feature-context*.



Gambar 3.5 Model Konseptual GloVe¹⁸

Pada penelitian ini menggunakan *pre-trained GloVe word embedding* dalam domain Bahasa Indonesia. *Pre-trained* ini dikembangkan oleh (Hanif, 2018) dalam Bahasa Indonesia. Pada Gambar 3.6 menunjukkan dataset *pre-trained GloVe Bahasa Indonesia*. *Pre-trained* yang dikembangkan telah berhasil diimplementasikan kedalam model untuk mengenali makna kata dan mengelompokkan kalimat pertanyaan dengan sangat baik. Dataset *pre-trained* ini memiliki dimensi vektor sebanyak 50.

¹⁸ <https://www.kdnuggets.com/2018/04/implementing-deep-learning-methods-feature-engineering-text-data-glove.html>

```

vectors.txt
x
1 dan -0.028669 -0.079521 -0.745437 0.947957 0.145600 0.208793 0.563256 0.144865 -1.168601 -0.770691 -0.868666 -0.099608 -0.677246
0.276379 -0.202523 -0.938004 0.869196 1.154606 -0.711352 1.530791 0.633017 1.103734 3.651248 -0.063392 -1.858380 1.355643
-0.342743 -0.638605 0.393076 -1.268856 1.055029 -0.375233 -1.428337 0.428032 1.361086 0.711733 1.206595 1.210484 -0.190727
0.194155 0.269840 -0.191284 -0.169478 -0.336127 -0.449392 0.552006 -0.037513 -0.216293 0.687252 -0.775978
2 yang 0.295120 0.139766 -0.561535 0.379670 -0.457635 0.926367 -0.305081 0.320839 -1.758719 -0.747167 -0.548925 0.207057 -1.654143
0.082712 -1.006004 -0.697750 0.261671 1.100704 -0.250909 1.697600 -0.109233 0.560439 3.805391 0.737748 -1.084196 1.394527
-1.000456 -0.015177 0.894618 -1.269922 0.632379 -0.410289 -1.089126 0.536642 1.314406 0.282170 1.189637 1.221083 -0.995803
0.512078 0.335196 -0.236668 -0.297434 -0.774030 -0.178452 0.063443 -0.109701 0.235066 0.868171 -0.899171
3 di 1.106462 0.004512 -0.690674 1.124802 -0.149371 1.124944 -0.567691 -0.298786 -2.716129 -0.373370 -0.898177 0.105004 -0.946431
0.248117 -0.540397 -0.767326 1.377455 0.732194 -0.466124 1.721660 -0.463452 0.204959 2.758096 0.380950 -2.298299 1.192526
-0.957032 -0.449725 -0.636391 -1.280624 0.826016 -1.147952 -1.024122 0.194116 0.540715 -0.073779 0.859183 1.470176 0.038010
0.933654 0.370973 -0.149733 -0.912499 -0.803927 -1.154939 1.048008 -1.006073 -0.377452 0.428321 0.672893
4 dari -0.186636 0.111410 -0.732223 0.065295 -0.844779 0.889617 -0.955966 0.288073 -2.016413 -0.345438 -0.626897 -0.290234 -1.369437
1.068053 -0.282120 -1.298875 0.324307 0.304477 -0.526821 1.966279 0.229065 0.595011 3.579488 0.706916 -1.535269 0.951961 -0.447950
0.218530 0.269990 -1.103329 0.791549 -0.949529 -0.887715 1.229542 1.253005 0.397025 1.250033 0.794312 -0.253100 0.211000 0.471364
-0.111975 -0.015543 -1.234242 -0.209678 0.719845 0.170422 0.297882 0.728826 -0.549682
5 pada 0.176362 -0.444578 -0.738165 0.387016 -0.490474 0.986875 -1.041134 0.842025 -1.523508 -1.352321 -0.612168 0.093491 -1.604917
0.785478 -0.430605 -1.379618 1.123060 1.036913 -0.337897 1.777915 -0.141289 0.758515 2.910556 -0.225810 -1.431380 1.123152
-0.150565 0.173488 0.308411 -1.046966 1.456711 -0.100640 -1.099367 0.356595 0.367295 0.264882 0.767646 1.617524 0.592090 0.649888
0.734666 0.219987 0.444486 -1.595183 0.033392 0.277143 -0.068231 -0.484764 1.069384 -0.261006
6 ini 0.831705 0.046489 -0.937325 0.450916 -0.526941 0.665796 -1.087795 0.157487 -1.852144 -0.945056 -0.596593 -0.039013 -1.681005
0.199372 -0.825560 -0.695409 1.104178 1.064445 -0.114196 1.649205 0.212960 0.162971 3.703525 0.608975 -1.487388 0.975682 -0.778560
0.239221 0.809415 -0.613960 0.399136 -0.437693 -1.185412 0.391578 0.533899 0.276906 1.294808 1.501956 -0.726748 0.607045 0.522597
0.327948 -0.588878 -1.083995 -0.265219 -0.014640 -0.094993 0.132099 0.932381 -1.251944
7 dengan -0.227211 0.383416 -0.301135 0.273031 0.322421 0.389767 -0.865624 0.588544 -1.344036 -1.427436 -0.743515 -0.286100
-0.307043 -0.085402 -0.369622 -1.266325 0.587680 0.908675 -0.136412 1.859955 0.249211 0.791411 3.342266 0.375537 -1.607914
1.364994 -0.442970 -0.293419 0.818938 -1.084524 1.409469 -0.019253 -1.575661 0.156550 1.427614 0.431279 1.335829 1.393825
-0.250521 0.219346 0.479620 -0.512203 0.086095 -0.352333 0.136520 -0.236814 0.180587 -0.202489 0.876362 -0.440473
8 untuk -0.836124 0.606331 -0.390495 0.066492 0.429015 0.629061 -0.271357 0.640551 -1.766504 -1.524977 -0.566473 -0.249516 -0.461937
1.027234 -0.758217 -0.995814 1.011672 1.327235 -0.776567 1.938334 -0.361684 0.739195 3.651254 -0.008335 -1.409534 0.275726
-0.306186 -0.577312 0.743647 -1.304509 1.210871 -0.096647 -1.850315 0.187839 0.290210 0.405537 0.325495 1.425676 0.227180
-0.308959 0.125034 -0.854986 0.013999 0.029465 -0.629842 0.528955 -0.033799 -0.506460 0.501647 -1.082697

```

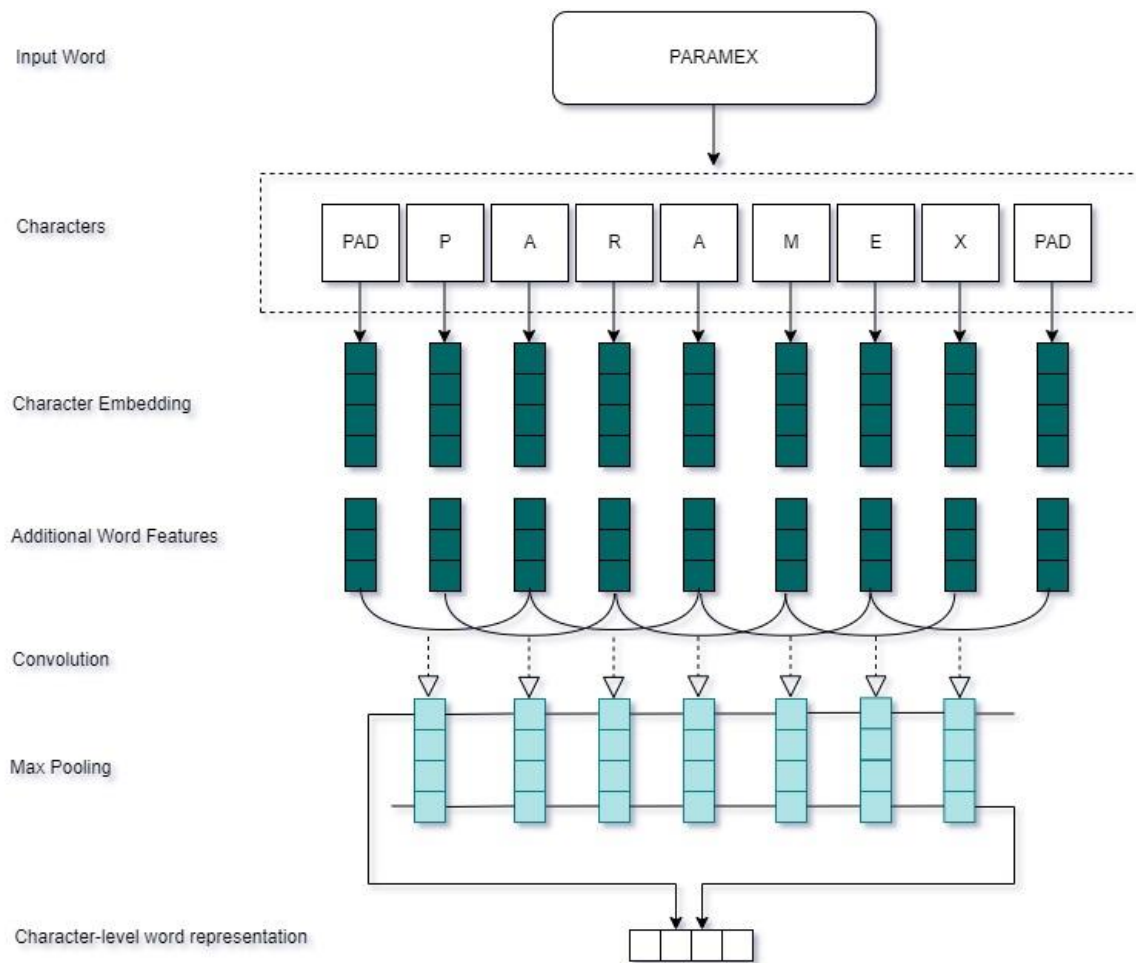
Gambar 3.6 Dataset GloVe Bahasa Indonesia

3.7 Membangun Model CNN-BiGRU

Sebagian besar data yang digunakan pada tugas NLP adalah kalimat atau dokumen yang direpresentasikan sebagai matriks. Setiap baris matriks berhubungan dengan satu token, biasanya sebuah kata, tetapi bisa juga berupa karakter. Artinya, setiap baris adalah vektor yang mewakili sebuah kata. CNN melakukan tahapan pemodelan dengan mentransformasi semua kata token dalam kalimat menjadi vektor menggunakan *word embedding* dalam kalimat secara berurutan (dos Santos & Guimarães, 2015). Kemudian CNN merangkum makna sebuah kalimat melalui *convolutional layer* dan *pooling layer*, hingga mencapai sebuah representasi *fixed-length vector* pada lapisan akhir.

Pada penelitian ini CNN digunakan untuk mengekstrak *character features*, yang sebelumnya telah berhasil diterapkan oleh (Chiu & Nichols, 2016) dengan menggunakan CNN dan BiLSTM pada tugas NER. Pada Gambar 3.7 menunjukkan proses CNN dalam mengekstraksi fitur *character-level*. Dimulai dengan proses menambahkan PADDING dan UNKNOWN. PADDING digunakan untuk memberikan vektor karakter pada token agar memiliki panjang yang sama untuk setiap token. UNKNOWN digunakan untuk menggantikan karakter yang tidak sesuai pada korpus. Selanjutnya pada *character embedding layer*, lapisan ini akan memasukkan nilai berdasarkan *lookup table* yang bertujuan untuk mengubah fitur diskrit seperti kata dan karakter menjadi representasi vektor kontinu. *Lookup table* diinisialisasi secara acak dengan nilai yang diambil dari distribusi seragam antara -0,5 dan 0,5 yang akan menghasilkan *character embedding* 25 dimensi.

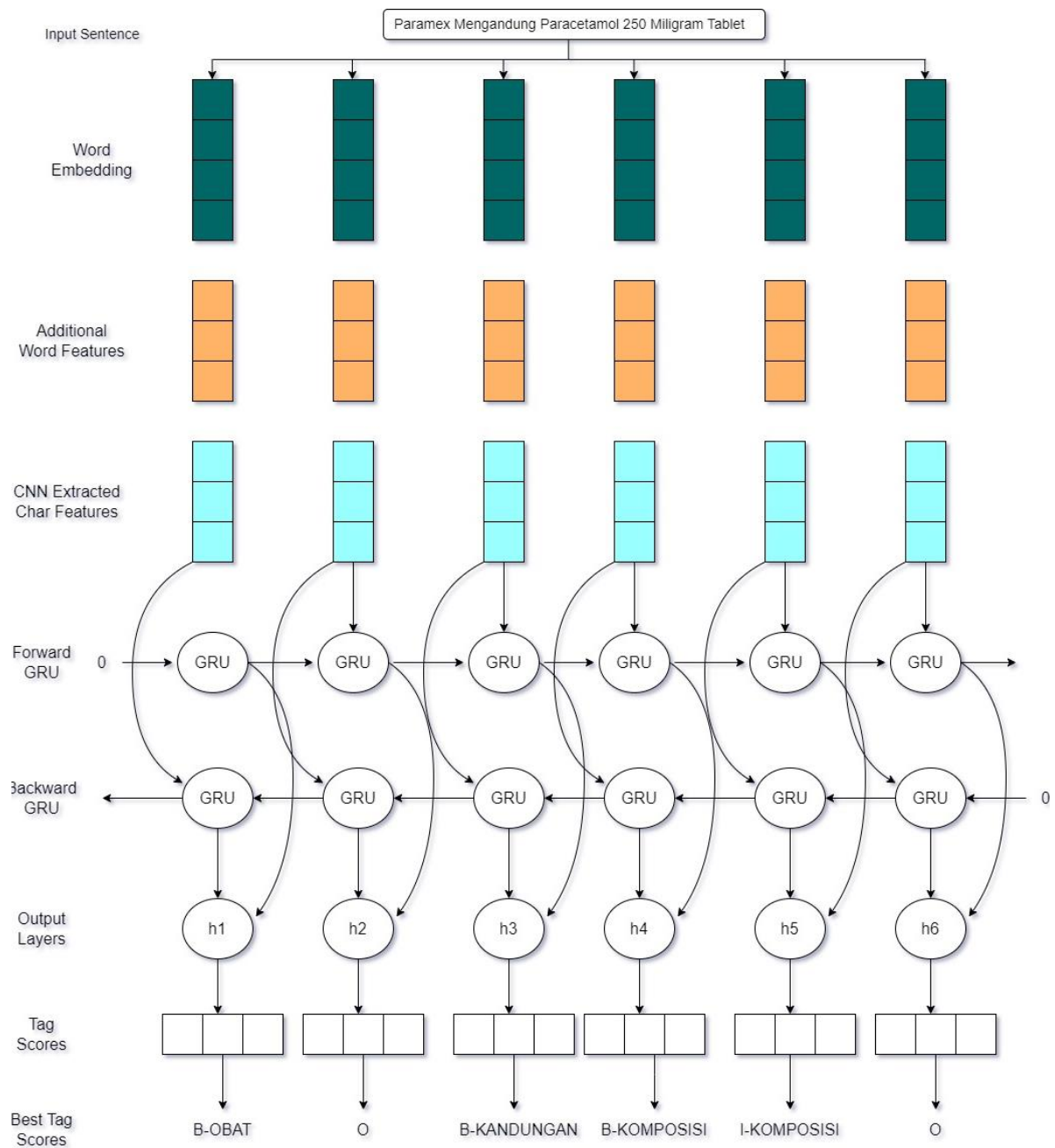
Selanjutnya *additional char features* digunakan untuk mengidentifikasi tipe karakter pada setiap token. Proses identifikasi menggunakan *lookup table* yang menampilkan vektor 4 dimensi yang mewakili tipe karakter seperti huruf besar, huruf kecil, tanda baca, dan lainnya. Kemudian *convolution layer* dan *max layer* digunakan untuk mengekstraksi vektor fitur baru dari vektor fitur per karakter hasil proses *embedding character* dan penambahan *additional character features*.



Gambar 3.7 CNN Character Features

Selanjutnya hasil ekstraksi *character feature* dihubungkan dengan menggunakan BiGRU. Model dengan struktur *bidirectional* memiliki kemampuan untuk mempelajari informasi dari data sebelumnya dan data selanjutnya sehingga memiliki pemahaman konteks yang lebih baik. BiGRU adalah salah satu jenis model dua arah yang ditentukan berdasarkan keadaan dua GRU yang searah dan arah yang berlawanan. Penggabungan model CNN-BiGRU telah berhasil diterapkan oleh (Xu et al., 2019) dalam melakukan klasifikasi data teks. BiGRU digunakan sebagai pemetaan konteks terstruktur pada kalimat sehingga entitas

dapat dikenali secara efektif. Kemudian model dilatih dengan GRU dua arah dan menggabungkan *output* keduanya untuk melakukan klasifikasi. Sehingga didapatkan hasil klasifikasi entitas berdasarkan nilai terbaik. Pada Gambar 3.8 menunjukkan rancangan model CNN-BiGRU pada penelitian ini.



Gambar 3.8 Model CNN-BiGRU

Pada Gambar 3.8 di atas menunjukkan bagaimana arsitektur pada model CNN-BiGRU melakukan proses klasifikasi entitas pada sebuah kalimat. Tahapan awal dimulai dengan kalimat "Paramex Mengandung Paracetamol 250 Miligram Tablet" menjadi masing-

masing token. Selanjutnya kata diubah menjadi representasi vektor. Kemudian pada *additional word features* kata akan dipisahkan berdasarkan *table lookup* berdasarkan empat vektor dimensi seperti huruf besar, huruf kecil, tanda baca dan lainnya. Selanjutnya pada *extracted character features* akan mengekstraksi vektor baru, yang merupakan vektor filter per karakter dengan menggunakan *convolution layer* dan *max pooling layer* pada CNN. Hasilnya kemudian digabungkan dan dimasukkan kedalam lapisan BiGRU dimana prosesnya menggunakan proses dua arah *forward layer* dan *backward layer*. Pada hasil akhir akan digabungkan untuk kemudian mengubah fitur kata menjadi *tag scores*. Didalam *tag scores* akan dilakukan perhitungan menggunakan *log-probabilities* untuk menghitung nilai probabilitas pada masing-masing entitas. Dapat dilihat bahwa pada *tag score* terdapat tiga nilai probabilitas. Hal ini disebabkan karena entitas yang terdefinisi adalah sebanyak tiga yaitu OBAT, KANDUNGAN dan KOMPOSISI. Adapun untuk *tag O* digunakan untuk mendefinisikan kata yang tidak termasuk pada jenis entitasnya. *Tag O* secara otomatis terdefinisi menggunakan format BIO.

3.8 Pelatihan Model

Pelatihan model dilakukan dengan beberapa kali percobaan dengan menggunakan delapan model yang berbeda. Proses ini bertujuan untuk mencari nilai akurasi yang paling baik. Pelatihan dilakukan dengan menggunakan pengaturan parameter utama dan *hyperparameter*. Pada Tabel 3.4 menunjukkan pengaturan parameter utama. Pada pengaturan parameter utama menggunakan *batch size*, *learning rate*, *dropout* dan *epoch*. *Batch size* digunakan untuk mengatur jumlah panjang kalimat sebagai *input*. Kemudian *learning rate* digunakan untuk mengukur nilai koreksi bobot dengan range 0 sampai 1. *Dropout* digunakan untuk mencegah terjadinya *overfitting* pada model dan bernilai 0 sampai 1. *Epoch* digunakan untuk mengatur jumlah banyaknya pelatihan yang dilakukan dalam sekali putaran yaitu dari awal sampai akhir dan dikembalikan lagi ke awal.

Tabel 3.4 Pengaturan Parameter Utama

No	Parameter	Nilai
1	Batch size	50
2	Learning rate	0.001
3	Dropout	0.5
4	Epoch	50

Selanjutnya pada Tabel 3.5 menunjukkan pengaturan *hyperparameter* untuk masing-masing model. Pada pengaturan *hyperparameter* tiap model akan dilatih dengan CNN *kernel size*, CNN *filter*, CNN *layer*, GRU *unit*, dan GRU *layer*. CNN *kernel size* digunakan untuk menentukan jumlah dimensi kernel pada saat konvolusi, CNN *filter* digunakan untuk menentukan jumlah *filter* yang digunakan untuk menampung nilai hasil pergeseran konvolusi, CNN *layer* digunakan untuk menentukan jumlah lapisan CNN yang digunakan dalam model. GRU *unit* digunakan untuk menentukan banyaknya jumlah *cell* pada lapisan GRU. Dan terakhir GRU *layer* digunakan untuk menentukan jumlah lapisan GRU yang digunakan untuk proses pelatihan.

Tabel 3.5 Pengaturan Hyperparameter

Model	CNN Kernel Size	CNN Filter	CNN Layer	GRU Unit	GRU Layer
1	3	30	1	200	2
2	7	50	1	200	1
3	7	50	1	200	2
4	7	50	2	200	1
5	3	30	1	100	1
6	3	30	2	100	1
7	3	50	2	200	1
8	7	30	2	200	1

3.9 Evaluasi Model

Evaluasi digunakan untuk mengukur apakah model mampu melakukan klasifikasi dengan baik atau tidak. Hasil dari proses evaluasi ini digunakan untuk mengetahui kinerja dari model. Model pengukuran menggunakan F-Score dengan *precision* dan *recall* untuk mencari nilai tingkat akurasi sebagai parameter keberhasilan model yang dikembangkan dan penerapan model. *Precision* adalah ukuran entitas yang ditemukan dalam korpus yang telah diklasifikasikan dengan benar. *Precision* didefinisikan pada persamaan (3.2) dimana T_p adalah jumlah dari *True Positives* dan F_p adalah jumlah *False Positive*.

$$P = \frac{T_p}{T_p + F_p} \quad (3.2)$$

Recall menunjukkan berapa banyak dari semua entitas yang seharusnya ditemukan oleh model NER. *Recall* didefinisikan pada persamaan (3.3) dimana T_p adalah jumlah dari *True Positives* dan F_n adalah jumlah *False Negative*.

$$R = \frac{T_p}{T_p + F_n} \quad (3.3)$$

F-Score adalah perbandingan rata-rata antara *precision* dan *recall*. F-Score didefinisikan pada persamaan (3.4) dimana β adalah bobot koefisien antara *precision* dan *recall* dengan nilai parameter adalah 1.

$$F_\beta = (1 + \beta^2) \frac{PR}{\beta^2 P + R} \quad (3.4)$$

3.10 Logarithmic Loss Model

Pada dasarnya didalam proses pembelajaran model *deep learning* adalah menerima *input* data dan memprediksi hasil dengan benar. Namun hasil prediksi yang dihasilkan perlu diukur untuk mengetahui seberapa baik algoritma yang digunakan dalam memodelkan data. *Loss function* adalah metode untuk menghitung nilai *loss* atau *error* yang digunakan untuk mengevaluasi seberapa baik algoritma dalam memprediksi keluaran yang tepat. Jika hasil prediksi menyimpang terlalu banyak dari hasil aktual *loss function* akan memiliki nilai yang tinggi. Sebaliknya jika hasil prediksi mendekati hasil *actual loss function* akan memiliki nilai yang rendah. Dengan kata lain *loss function* yang baik adalah model yang menghasilkan *error* paling rendah (Goodfellow et al., 2016).

Cross-entropy atau *logarithmic loss* adalah salah satu *loss function* yang digunakan untuk menghitung nilai *loss* pada tugas *multiclass classification*. Algoritma ini digunakan untuk mengukur kinerja model klasifikasi yang keluarannya berupa nilai probabilitas antara 0 dan 1. Nilai *Cross-entropy loss* akan meningkat apabila probabilitas yang diprediksi menyimpang dari label sebenarnya. Model yang sempurna akan memiliki nilai *log loss* atau nilai *cross-entropy loss* 0. *Cross-entropy* secara matematis dirumuskan dengan persamaan (3.5) dimana H adalah fungsi *cross-entropy*, P adalah nilai target dan Q adalah nilai prediksi.

$$H(P, Q) = - \sum_i^x P(i) \log(Q(i)) \quad (3.5)$$

BAB 4

Hasil dan Pembahasan

4.1 Data yang digunakan

Penelitian ini menggunakan dataset yang berasal dari cuplikan Google dengan menggunakan kata kunci daftar obat. Daftar obat diambil melalui beberapa situs web kesehatan dapat dilihat pada bab 3. Data yang berhasil dikumpulkan sebanyak 18,075 kalimat dan disimpan dalam bentuk file csv. Gambar 4.1 menunjukkan data yang telah terkumpul.

```
SCRAPED SNIPPETS x
1 DETAILS_MEDS
2 tiap kapsul abajos mengandung zat aktif (nama generik) sebagai berikut :paracetamol 500 mg.thiamine HCl 50 mg.pyridoxine HCl 100
mg.cyanocobalamin 100 mcg.
3 "20 Apr 2019 · Kandungan Abate. Kandungan yang ada di dalam bubuk abate adalah Temephos dengan kadar 1%. Jadi, setiap satu abate
yang berisi 1 ..."
4 "Abbotic mengandung zat aktif Clarithromycin, suatu antibiotik golongan macrolide yang mempunyai spektrum luas, aktif terhadap
bakteri gram negatif maupun gram positif. Dalam penggunaan obat ini HARUS SESUAI DENGAN PETUNJUK DOKTER."
5 "Informasi terlengkap tentang ABC MENTHOL CONE 20 GR. Komposisi, Fungsi, Kegunaan,Ulasan, Efek Samping, Indikasi, Aturan Pakai, dan
Dosis."
6 "Komposisi Abdelyn Drop mengandung : Vitamin A 2,000 UI Vitamin B1 2 mg Vitamin B2 3 mg Vitamin B6 2 mg Vitamin B12 2 mcg Vitamin
D3 400 IU Nikotinamid 20 mg L-lysine HCl 25 mg D-panthenol 5 mg Kontraindikasi Abdelyn Drop tidak boleh diberikan pada : Pasien
yang hipersensitif terhadap komponen ini."
7 Tiap kemasan Abilify Tablet mengandung zat aktif (nama generik) sebagai berikut :Aripiprazole 5 mg / tablet.Aripiprazole 10 mg /
tablet.Aripiprazole 15 mg / tablet.
8 "ABIXA TABLET 10 MGPengobatan pasien dengan penyakit Alzheimer.Deskripsi. -Memantine Hcl 10 mg.5 mg setiap hari, Max: 20 mg /
hari.Bersamaan dengan makanan atau tidak.Dus, 2 Strip @ 14 Tablet.Wanita menyusui.Gangguan fungsi ginjal dan epilepsi.Item
lainnya..."
9 "Abixim adalah obat yang digunakan untuk mengobati infeksi saluran pernafasan, infeksi kulit dan jaringan lunak, serta infeksi
saluran kemih (isk) dan kelamin.
10 ...
11 Abixim mengandung zat aktif (nama generik) sebagai berikut :Cefixime 100 mg/caplet.Cefixime 200 mg/caplet.Cefixime 100 mg/5 mL
syrup kering."
12 A-B VASK merupakan obat dengan kandungan Amlodipine dalam bentuk tablet. Obat ini digunakan untuk pengobatan hipertensi dan angina.
Dalam penggunaan obat ini harus SESUAI DENGAN PETUNJUK DOKTER. PENGGUNAAN OBAT INI HARUS SESUAI DENGAN PETUNJUK DOKTER.
13 "2 Mei 2018 · Acarbose adalah obat antidiabetes yang digunakan untuk menangani diabetes tipe 2. Acarbose berfungsi untuk mengontrol
kadar gula darah ...
14
15 Golongan: Antidiabetes
16 Manfaat: Mengontrol kadar gula dalam darah pada penderita diabetes tipe 2
17 Dikonsumsi oleh: Dewasa
18 Kategori: Obat resep"
19 "Kemasan Acdat cream dipasarkan dengan kemasan sebagai berikut : Tube 5 gram cream 2 % Kandungan Setiap kemasan obat Acdat cream
mengandung zat aktif sebagai berikut : Fusidic acid 2 % atau 20 mg/gram cream Dosis Dosis pada orang dewasa, dioleskan sebayak 3-4
kali dalam sehari, sampai dengan sembuh."
```

Gambar 4.1 Data Obat

4.2 Hasil Preprocessing

Selanjutnya data yang telah dikumpulkan kemudian dilakukan tahapan data *preprocessing*. Data *preprocessing* merupakan salah satu tahapan penting untuk menyiapkan data yang sesuai pada model yang digunakan. Data *preprocessing* yang dilakukan antara lain :

1. Menghapus tanggal

Pada penelitian ini informasi tanggal tidak diperlukan karena tidak relevan dengan kata yang mengandung obat. Pada Gambar 4.2 menunjukkan potongan kode program untuk menghapus tanggal. Dapat dilihat bahwa proses menghapus tanggal menggunakan aturan *regular expression* yang didalamnya mengandung angka dan karakter.

```
[4] def removeDate(str):
    temp = re.sub('[0-9]{1,2} [w]{4} [0-9]{5}', ' ', str)
    return temp
```

Gambar 4.2 Kode Program Menghapus Tanggal

Pada Tabel 4.1 menunjukkan bahwa *preprocessing* berhasil menghapus kata “2 Mei 2018” yang merupakan kata yang menerangkan sebagai tanggal.

Tabel 4.1 Hasil Hapus Tanggal

Sebelum	Sesudah
2 Mei 2018 · Acarbose adalah obat antidiabetes yang digunakan untuk menangani diabetes tipe 2. Acarbose berfungsi untuk mengontrol kadar gula darah ...	· Acarbose adalah obat antidiabetes yang digunakan untuk menangani diabetes tipe 2. Acarbose berfungsi untuk mengontrol kadar gula darah ...

2. Menghapus *punctuation* atau tanda baca

Proses ini dilakukan untuk menghapus tanda baca atau karakter khusus pada data teks seperti koma, titik, hashtags, tanda kutip dan lain-lain. Pada Gambar 4.3 menunjukkan potongan kode program untuk menghapus tanda baca. Dapat dilihat bahwa untuk menghapus tanda baca perlu mendefinisikan jenis tanda baca apa saja yang akan dihapus ke dalam sebuah variabel.

```
[17] def removePunct(s):
    symb = [';', '?', '!', '&', '\n', '\xa0', '.', '|', '(', ')', '@', '/', ':', 'x', 'X', '+']
    for sy in symb:
        s = s.replace(sy, " ")
    return s
```

Gambar 4.3 Kode Program Menghapus Tanda Baca

Pada Tabel 4.2 menunjukkan *preprocessing* berhasil menghapus tanda baca pada karakter “(”, “)” dan “/”.

Tabel 4.2 Hasil Hapus Tanda Baca

Sebelum	Sesudah
Tiap kemasan Abilify Tablet mengandung zat aktif (nama generik) sebagai berikut :Aripiprazole 5 mg / tablet Aripiprazole 10 mg / tablet Aripiprazole 15 mg / tablet	Tiap kemasan Abilify Tablet mengandung zat aktif nama generik sebagai berikut Aripiprazole 5 mg tablet Aripiprazole 10 mg tablet Aripiprazole 15 mg tablet

3. Menghapus karakter ASCII

Karakter ASCII merupakan kode standar yang digunakan dalam pertukaran informasi dalam komputer. Proses ini dilakukan untuk menghapus karakter seperti “Â” dan “â€”. Pada Gambar 4.4 menunjukkan potongan kode program untuk menghapus karakter ASCII. Dapat dilihat bahwa proses menghapus karakter ASCII menggunakan pustaka string untuk menyaring karakter ASCII yang ada pada kalimat.

```
[14] def nonASCII(s):
    printable = set(string.printable)
    return ''.join(filter(lambda x: x in printable, s))
```

Gambar 4.4 Kode Program Menghapus karakter ASCII

Pada Tabel 4.3 menunjukkan *preprocessing* berhasil menghapus karakter ASCII pada karakter “¥”, “α” dan “©”.

Tabel 4.3 Hasil Hapus Karakter ASCII

Sebelum	Sesudah
¥ Tiap kaplet salut selaput mengandung (α -Ketoisoleucine) garam kalsium 67 mg, (α -Ketoleucine) garam kalsium 101 mg, (α -Ketophenylalanine) garam kalsium 68 mg, (α -Hydroxymethionine) garam kalsium 59 mg, (α -Ketovaline) garam kalsium 86 mg, L-tryptophan 23 mg, L-threonine 53 mg, L-histidine 38 mg, L-tyrosine 30 mg, L- ©	Tiap kaplet salut selaput mengandung (-Ketoisoleucine) garam kalsium 67 mg, (-Ketoleucine) garam kalsium 101 mg, (-Ketophenylalanine) garam kalsium 68 mg, (-Hydroxymethionine) garam kalsium 59 mg, (-Ketovaline) garam kalsium 86 mg, L-tryptophan 23 mg, L-threonine 53 mg, L-histidine 38 mg, L-tyrosine 30 mg, L-

4. Normalisasi Teks

Proses ini dilakukan untuk menormalisasikan kata penting yang dibutuhkan yang masih belum sesuai dengan format penulisan. Seperti contoh “3mg” perlu dipisah karena mengandung keterangan komposisi obat. Pada Gambar 4.5 menunjukkan potongan kode program untuk normalisasi teks. Dapat dilihat bahwa proses normalisasi teks menggunakan aturan *regular expression* yang didalamnya mengandung angka dan karakter.

```
[9] def splitMeasure(s):
    temp = re.sub('([0-9]mg|[0-9]Mg|[0-9]MG|[0-9]ML|[0-9]mL|[0-9]ml)',r'\1',
                re.sub('(mg+|Mg+|Ml+|ml+|MG+|ML+)',r'\1', s)).split()
    res = ' '.join(temp).replace('mg.', 'mg. ')
    return res
```

Gambar 4.5 Kode Program Normalisasi Teks

Pada Tabel 4.4 menunjukkan *preprocessing* berhasil melakukan normalisasi teks pada kata “5MI” yang dinormalisasikan menjadi “5”, “MI”. Proses ini menjadi penting karena kata tersebut menunjukkan keterangan komposisi. Dimana keterangan komposisi sangat diperlukan dalam penelitian ini.

Tabel 4.4 Hasil Normalisasi Teks

Sebelum	Sesudah
Tiap sendok takar 5Ml Anakonidin Syrup mengandung zat aktif nama generik sebagai berikut Pseudoephedrine 7.5Mg	Tiap sendok takar 5 Ml Anakonidin Syrup mengandung zat aktif nama generik sebagai berikut Pseudoephedrine 7.5 Mg

5. Tokenizing

Proses pemisahan kalimat teks menjadi potongan kata atau disebut token. Pada Gambar 4.6 menunjukkan potongan kode program untuk *tokenizing*. Dapat dilihat bahwa proses *tokenizing* menggunakan pustaka *nltk* untuk dapat melakukan proses *tokenizing* kata.

```
[33] def splitSentence(s):
      sent_text = nltk.word_tokenize(s)
      return sent_text
```

Gambar 4.6 Kode Program Tokenizing

Pada Tabel 4.5 menunjukkan *preprocessing* berhasil melakukan *tokenizing* kata pada kalimat. Proses *tokenizing* memecah kata hingga level karakter seperti karakter “(”, “)” dan “,”.

Tabel 4.5 Hasil Tokenizing

Sebelum	Sesudah
Artro (plus) mengandung glucosamine HCl 500 mg	Artro
	(
	plus
)
	mengandung
	glucosamine
	HCL
	500
milligram	

6. Menghapus *Stopword*

Proses ini dilakukan untuk menghilangkan kata yang tidak penting seperti ini, itu, jadi, dan lain-lain. Pada Gambar 4.7 menunjukkan potongan kode program untuk menghapus *stopword*. Dapat dilihat bahwa proses menghapus *stopword* menggunakan file eksternal yang telah didefinisikan sebagai daftar kata yang tidak diperlukan.

```
[52] def Stopword(s):  
    read = pd.read_csv('/content/stopword_NER.csv')  
    more_stopword = read['Stopword'].tolist()  
    temp = []  
    for i in s.split():  
        if i.lower() not in more_stopword:  
            temp.append(i)  
    return ' '.join(temp)
```

Gambar 4.7 Kode Program Menghapus Stopword

Pada Tabel 4.6 menunjukkan *preprocessing* berhasil menghapus *stopword* dengan menghilangkan kata “sachet” dan “yang”.

Tabel 4.6 Hasil Hapus Stopword

Sebelum	Sesudah
Tiap Astamax sachet mengandung Natural Astaxanthin 2 mg, fish collagen 250 mg, yang menjadikan Astamax memiliki antioksidan yang paling kuat, 10 kali lipat dibandingkan antioksidan pada wortel	Astamax mengandung Natural Astaxanthin 2 mg, fish collagen 250 mg, menjadikan Astamax memiliki antioksidan paling kuat, 10 kali lipat dibandingkan antioksidan pada wortel

4.3 Hasil POS Tagging

Selanjutnya adalah proses POS *tagging* digunakan untuk mengkategorikan kelas kata pada data teks. Proses POS *tagging* menggunakan pustaka Kumparan NLP yang menyediakan fungsi POS *tagging* untuk Bahasa Indonesia. Gambar 4.8 menunjukkan potongan kode program untuk POS *tagging*.

```

import nltk
from nlp_id.postag import PosTag
postagger = PosTag()

def pos_tagging(sent):
    result = postagger.get_pos_tag(sent)
    return result

```

Gambar 4.8 Kode Program POS Tagging

Pada Gambar 4.8 di atas menunjukkan bahwa proses POS *tagging* menggunakan pustaka pada Kumparan dengan melakukan import terlebih dahulu dan mendefinisikan fungsi PosTag untuk menjalankan fungsi POS *tagging*. Pada Gambar 4.9 menunjukkan hasil kode program untuk POS *tagging*. Seperti kata “Aflucaps” ditandai sebagai NNP yang dikenal sebagai kata benda spesifik. Kata “380” ditandai sebagai NUM yang dikenal sebagai angka dan kata “milligram” ditandai sebagai NN yang dikenal sebagai kata benda.

Word	POS
Aflucaps	NNP
mengandung	VB
paracetamol	NNP
380	NUM
miligram	NN
chlorphenamine	NNP
maleate	FW
2	NUM
miligram	NN
ephedrine	NNP
HCl	NNP
7	NUM
miligram	NN

Gambar 4.9 Hasil POS Tagging

4.4 Hasil Entity Tagging

Selanjutnya adalah proses pelabelan entitas untuk memetakan setiap kata berdasarkan hasil POS *tagging*. Proses pembentukan *entity tagging* menggunakan *chunking* dan format BIO untuk mendefinisikan entitas yang saling terhubung. Pada Gambar 4.10 menunjukkan potongan kode program untuk *entity tagging*.

```
from nltk.chunk import conlltags2tree, tree2conlltags

clean_tok = []
BIOres = []

def nltk_parse_clause(sentence):
    grammar = r"""
    KOMPOSISI: {<NUM><NN|NUM><VB>*}
               }<NUM><NN.*|NUM><VB>{
    KANDUNGAN: {<VB><NN><NNP>*}
               }<VB><NN>{
    KANDUNGAN: {<NNP>+?<VB>*<NNP>*}
               }<VB>*<NNP>*{
    KANDUNGAN: {<VB>*<NN>*<VB><NNP>*}
               }<VB>*<NN>+?<VB>{
    KANDUNGAN: {<VB>*<NN>*<VB><NN>+?<NNP>?}
               }<VB>*<NN>*<VB>{
    OBAT: {<VB>*<NN>?<NNP>*}
          }<VB>*<NN>+?{
    OBAT: {<NN.*>?<VB><VB>}
          }<VB><VB>{
    OBAT: {<NNP>*<ADV>*<VB>}
          }<ADV>*<VB>{
    OBAT: {<NNP>*<NN>*<VB><NN>*<VB>}
          }<VB><NN>*<VB>{
    OBAT: {<NNP>+?<NUM><NNP|NUM|NN>}
    OBAT: {<NNP|NN|FW>*<VB><KOMPOSISI>}
          }<VB><KOMPOSISI>{
    OBAT: {<NN.*>*<VB><NUM>}
          }<VB><NUM>{
    OBAT: {<NN.*|FW>*<VB><KANDUNGAN>}
          }<VB><KANDUNGAN>{
    KANDUNGAN: {(<NNP><FW|NNP>*<KOMPOSISI>)|(<FW><NNP|FW>*<NN><KOMPOSISI>)}
               }<KOMPOSISI>{
    KANDUNGAN: {<OBAT><VB><KOMPOSISI><NN.*|FW>*}
               }<OBAT><VB><KOMPOSISI>{
    """
    cp = nltk.RegexpParser(grammar)
    parsed_sentence = cp.parse(sentence)
    return parsed_sentence

temp = nltk_parse_clause(sent_token)
clean_tok.append(temp)

for i in clean_tok:
    iob_tagged = tree2conlltags(i)
    BIOres.append(iob_tagged)
```

Gambar 4.10 Kode Program Entity Tagging

Pada Gambar 4.10 di atas menunjukkan bahwa proses pembentukan *entity tagging* menggunakan aturan *regular expression* dengan mendefinisikan pola *chunking* pada masing-masing jenis entitasnya. Dapat dilihat bahwa untuk membuat *grammar* menggunakan pustaka *nlk* dan memanggil fungsi *grammar* yang telah didefinisikan ke dalam fungsi *nlk.RegexParser*. Selanjutnya pada Gambar 4.11 menunjukkan hasil proses *entity tagging*.

Word	POS	Tag
Aflucaps	NNP	B-OBAT
mengandung	VB	O
paracetamol	NNP	B-KANDUNGAN
380	NUM	B-KOMPOSISI
miligram	NN	I-KOMPOSISI
chlorphenamine	NNP	B-KANDUNGAN
maleate	FW	I-KANDUNGAN
2	NUM	B-KOMPOSISI
miligram	NN	I-KOMPOSISI
ephedrine	NNP	B-KANDUNGAN
HCl	NNP	I-KANDUNGAN
7	NUM	B-KOMPOSISI
miligram	NN	I-KOMPOSISI

Gambar 4.11 Hasil Entity Tagging

Pada Gambar 4.11 di atas dapat dilihat bahwa pada kata “Aflucaps” berhasil ditandai sebagai B-OBAT. Karena aflucaps dikenali sebagai awalan kata obat. Kata “paracetamol” berhasil ditandai sebagai B-KANDUNGAN karena dikenali sebagai kandungan obat. Kata “380” dan “milligram” berhasil ditandai sebagai B-KOMPOSISI dan I-KOMPOSISI. Hal ini dikarenakan kedua kata tersebut merupakan frasa kata yang sama yang dikenali sebagai komposisi obat.

4.5 Hasil Training dan Akurasi

Proses pelatihan pada model menggunakan dataset yang telah didistribusikan dengan format 70% *training set* dan 30% *test set*. Pada Tabel 4.7 menunjukkan jumlah kalimat dan token pada dataset. Dapat dilihat bahwa jumlah data pada *training set* memiliki jumlah kalimat dan token lebih banyak.

Tabel 4.7 Jumlah Kalimat Dan Token Data Set

Data set	Jumlah Kalimat	Jumlah Token
Training set	10754	330160
Tes set	7320	141435

Selanjutnya pada Tabel 4.8 menjelaskan bahwa entitas *tag* yang digunakan pada penelitian ini adalah OBAT digunakan untuk menunjukkan nama obat, KANDUNGAN digunakan untuk menunjukkan kandungan apa saja yang terdapat pada obat, KOMPOSISI digunakan untuk menunjukkan besaran jumlah dosis pada obat, dan O digunakan untuk menunjukkan kata yang tidak memiliki makna. Pada entitas *tag* O memiliki jumlah proporsi yang lebih tinggi dari entitas *tag* lainnya. Hal ini disebabkan karena dalam kalimat-kalimat yang terkumpul terdapat kata-kata yang digunakan untuk menjelaskan tentang kegunaan obat atau menjelaskan dosis penggunaan obat. Dan pada penelitian ini kata-kata tersebut tidak berkaitan dengan entitas *tag* yang diperlukan. Oleh karena itu, kata-kata yang tidak terkait ini ditandai sebagai *tag* O dan memiliki jumlah proporsi yang tinggi.

Tabel 4.8 Jumlah Entitas Pada Data Set

Data set	OBAT	KANDUNGAN	KOMPOSISI	O
Training set	15229	20013	8817	281700
Test set	7898	13891	10818	100817

Kemudian dataset digunakan untuk melakukan pelatihan pada model dengan menggunakan delapan model berbeda. Setiap model menjalankan proses pelatihan dengan parameter utama untuk semua model, dan *hyperparameter* yang berbeda pada masing-

masing model. Hasil *output* dari proses pelatihan model adalah akurasi dengan F1-Score. Dan menampilkan hasil *output* akurasi untuk kedua data set. Pada Tabel 4.9 menunjukkan hasil akurasi pada masing-masing model.

Tabel 4.9 Hasil Akurasi Training Model

Model	Dataset	Precision	Recall	F1 score
1	Train set	0.805	0.836	0.858
	Test set	0.811	0.897	0.864
2	Train set	0.845	0.890	0.866
	Test set	0.863	0.894	0.873
3	Train set	0.857	0.885	0.862
	Test set	0.864	0.870	0.869
4	Train set	0.821	0.851	0.853
	Test set	0.839	0.866	0.861
5	Train set	0.875	0.888	0.849
	Test set	0.880	0.892	0.857
6	Train set	0.854	0.867	0.852
	Test set	0.868	0.871	0.858
7	Train set	0.842	0.879	0.861
	Test set	0.861	0.886	0.868
8	Train set	0.838	0.881	0.857
	Test set	0.842	0.853	0.860

Berdasarkan hasil akurasi pelatihan pada Tabel 4.9 di atas dapat disimpulkan bahwa model 2 memiliki akurasi terbaik sebesar 0.873. Pada model 2 dapat menghasilkan akurasi

terbaik dengan menggunakan *hyperparameter* pada CNN *kernel size* adalah 7, CNN *filter* adalah 50, CNN *layer* adalah 1, GRU *unit* adalah 200 dan GRU *layer* adalah 1. Pengaturan *hyperparameter* pada masing-masing model memiliki pengaruh terhadap besaran akurasi yang dicapai pada suatu model. Pada Gambar 4.12 menunjukkan data log hasil akurasi pada pelatihan model 2.

```
Epoch 49/50
100/100 [=====] - 34s 372ms/step
Epoch 50/50
100/100 [=====] - 38s 380ms/step

9677/9677 [=====] - 257s 27ms/step
Train-data: Prec: 0.845, Rec: 0.890, F1: 0.866
9155/9155 [=====] - 257s 27ms/step
Test-data: Prec: 0.863, Rec: 0.894, F1: 0.873
```

Gambar 4.12 Data Log Hasil Akurasi Pelatihan Model 2

Berdasarkan Gambar 4.12 di atas menunjukkan bahwa hasil akurasi untuk F1-Score yang didapatkan pada model 2 sebesar 0.873. Seperti yang ditunjukkan pada Tabel 4.9 sebelumnya model 2 merupakan model dengan hasil akurasi F1-Score tertinggi. Oleh karena itu, model 2 digunakan untuk diimplementasikan sebagai bukti penelitian dalam menyimpulkan model NER untuk obat pada dunia nyata. Implementasi NER dilakukan menggunakan aplikasi DRUG NER dengan memasukkan kalimat sebagai *input*. Gambar 4.13 menunjukkan hasil implementasi dengan menggunakan aplikasi DRUG NER.

Your text here

Paramex mengandung 250 mg paracetamol dan kafein 50 mg

Extract

	Tokens	Entity Tags
0	Paramex	B-OBAT
1	mengandung	O
2	250	B-KOMPOSISI
3	mg	I-KOMPOSISI
4	paracetamol	B-KANDUNGAN
5	dan	O
6	kafein	B-KANDUNGAN
7	50	B-KOMPOSISI
8	mg	I-KOMPOSISI

Gambar 4.13 Hasil Implementasi Pada Aplikasi DRUG NER

Pada Gambar 4.13 di atas dapat dilihat bahwa implementasi dengan menggunakan kalimat “Paramex mengandung 250 mg paracetamol dan kafein 50 mg” yang digunakan sebagai *input* berhasil diidentifikasi berdasarkan jenis entitasnya. Kata “Paramex” berhasil ditandai sebagai B-OBAT karena Paramex merupakan kata awal yang dikenali sebagai produk obat. Kata “mengandung” ditandai sebagai O karena tidak dikenali dalam aturan *grammar*. Kata “250” ditandai sebagai B-KOMPOSISI karena dikenal sebagai awal dosis obat dan diikuti dengan kata “mg” yang kemudian ditandai sebagai I-KOMPOSISI karena terletak di dalam frasa yang sama. Kata “Paracetamol” ditandai sebagai B-KANDUNGAN karena merupakan kata awal yang dikenali sebagai bahan obat. Kata “dan” ditandai sebagai O karena tidak dikenali dalam aturan *grammar*. Kata “Kafein” ditandai sebagai B-KANDUNGAN karena merupakan kata awal yang dikenali sebagai bahan obat. Kata “50” ditandai sebagai B-KOMPOSISI karena dikenal sebagai awal dosis obat dan kata “mg” ditandai sebagai I-KOMPOSISI karena terletak di dalam frasa yang sama. Selanjutnya dilakukan tahap implementasi lanjutan dengan memasukkan dua kalimat berbeda dimana pada masing-masing kalimat adalah kalimat yang mengandung kata obat didalamnya. Seperti ditunjukkan pada Tabel 4.10.

Tabel 4.10 Hasil NER Obat

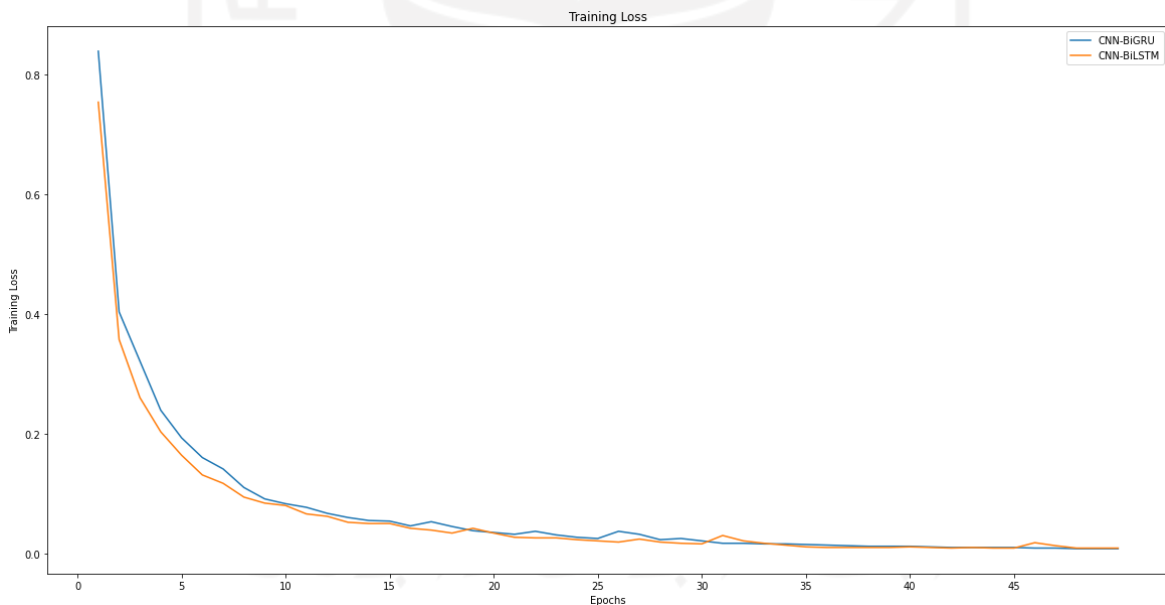
No	Kalimat	NER
1	Efisol-C 10 tablet mengandung Dequalinium chloride 0.25 mg dan Asorbic (Vitamin C) 50 mg	[('Efisol-C', 'B-OBAT'), ('10', 'I-OBAT'), ('Tablet', 'I-OBAT'), ('mengandung', 'O'), ('Dequalinium', 'B-KANDUNGAN'), ('chloride', 'I-KANDUNGAN'), ('0.25', 'B-KOMPOSISI'), ('mg', 'I-KOMPOSISI'), ('dan', 'O'), ('Asorbic', 'B-OBAT'), ('(', 'O'), ('Vitamin', 'B-KANDUNGAN'), ('C', 'I-KANDUNGAN'), (',', 'I-KANDUNGAN'), ('50', 'B-KOMPOSISI'), ('mg', 'I-KOMPOSISI)']
2	Red Yeast Rice 300 mg yang terkandung dalam Angkak kapsul berguna untuk mengatasi demam berdarah	[('Red', 'B-OBAT'), ('Yeast', 'I-OBAT'), ('Rice', 'I-OBAT'), ('300', 'B-KOMPOSISI'), ('mg', 'I-KOMPOSISI'), ('yang', 'O'), ('dikandung', 'O'), ('dalam', 'O'), ('Angkak', 'B-OBAT'), ('kapsul', 'I-OBAT'), ('berguna', 'O'), ('untuk', 'O'), ('mengatasi', 'O'), ('demam', 'O'), ('berdarah', 'O)']

Pada Tabel 4.10 di atas, model NER diuji dengan memasukkan dua kalimat berbeda. Pada kalimat pertama yang digunakan adalah “Efisol-C 10 tablet mengandung Dequalinium chloride 0.25 mg dan Asorbic (Vitamin C) 50 mg”. Hasil NER pada kalimat pertama menunjukkan hasil klasifikasi entitas yang sangat baik. Seperti pada kata “Vitamin C” berhasil ditandai sebagai B-KANDUNGAN dan I-KANDUNGAN karena dikenali sebagai bahan obat. Kemudian pada kalimat kedua yang digunakan adalah “Red Yeast Rice 300 mg yang terkandung dalam Angkak kapsul berguna untuk mengatasi demam berdarah”. Hasil NER pada kalimat kedua juga menunjukkan hasil klasifikasi entitas yang sangat baik. Seperti pada kata “ANGKAK” berhasil ditandai sebagai B-OBAT karena dikenali sebagai produk obat.

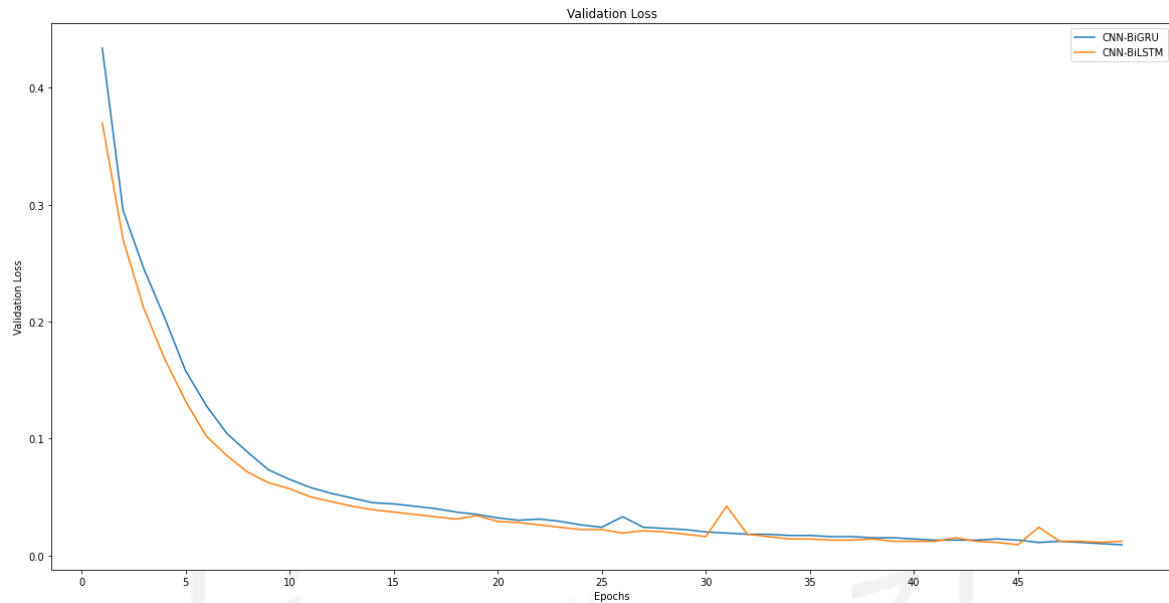
4.6 Perbandingan Hasil Training

Untuk mendukung analisa evaluasi, maka dilakukan perbandingan hasil pelatihan model. Tujuan dari perbandingan model adalah untuk mengukur performa dari model yang telah dibangun. Model CNN-BiLSTM pada penelitian (Fudholi et al., 2022) digunakan sebagai model pembanding dalam penelitian ini. Model CNN-BiLSTM telah terbukti berhasil mengekstrak entitas pada data obat Indonesia. Oleh karena itu, mengacu pada domain yang sama penelitian ini melakukan perbandingan pada model CNN-BiGRU dan model CNN-BiLSTM.

Perbandingan yang dilakukan pada penelitian ini adalah. Pertama, perbandingan pada nilai loss. Pengukuran nilai loss digunakan untuk mengetahui seberapa optimal model dalam memprediksi entitas dengan mengukur nilai kesalahan. Nilai loss yang paling kecil menunjukkan hasil model yang optimal. Dari hasil analisa evaluasi model didapatkan grafik perbandingan nilai *training loss* dan *validation loss* pada masing-masing model untuk setiap *epoch*-nya. Pada Gambar 4.14 menunjukkan grafik nilai *training loss* dan Gambar 4.15 menunjukkan grafik *validation loss* pada masing-masing model.



Gambar 4.14 Grafik Perbandingan Nilai Training Loss



Gambar 4.15 Grafik Perbandingan Nilai Validation Loss

Berdasarkan Gambar 4.14 dan Gambar 4.15 di atas menunjukkan bahwa nilai *training loss* dan *validation loss* baik pada model CNN-BiGRU maupun model CNN-BiLSTM nilai *validation loss* memiliki nilai lebih rendah dibandingkan nilai *training loss* pada awal-awal *epoch*. Jika dilihat pada penggunaan parameter utama pada bab 3, penelitian ini menggunakan *dropout regularization* untuk mencegah terjadinya *overfitting* pada saat dilakukan pelatihan model. Oleh karena itu nilai *validation loss* lebih rendah dibandingkan nilai *training loss* pada awal-awal *epoch* disebabkan oleh faktor lain yaitu distribusi data yang tidak seimbang. Dalam hal ini mengacu pada varian data yang ada pada *test set* karena memiliki varian data lebih sedikit dibandingkan varian data pada *training set*¹⁹.

Kemudian dapat dilihat bahwa nilai *training loss* dan *validation loss* pada model CNN-BiGRU terlihat semakin kecil pada *epoch* ke 28 hingga *epoch* ke 50. Pada model CNN-BiGRU nilai loss yang didapatkan cenderung stabil pada *epoch* ke 28 dan seterusnya. Nilai *training loss* dan *validation loss* terendah pada model CNN-BiGRU didapatkan pada *epoch* ke 50. Seperti ditunjukkan pada Gambar 4.16. Sedangkan pada model CNN-BiLSTM nilai *training loss* dan *validation loss* terlihat semakin kecil pada *epoch* ke 33. Dapat dilihat bahwa pada *epoch* ke 46 mengalami sedikit kenaikan kemudian turun kembali hingga *epoch* ke 50. Nilai *training loss* dan *validation loss* terendah pada model CNN-BiLSTM didapatkan pada *epoch* ke 45. Seperti ditunjukkan pada Gambar 4.17.

¹⁹ <https://pyimagesearch.com/2019/10/14/why-is-my-validation-loss-lower-than-my-training-loss/>

```

Epoch 46/50
102/102 [=====] - 246s 9ms/step - train_loss: 0.009025 - val_loss: 0.011333 - time: 77.549975
Epoch 47/50
102/102 [=====] - 247s 10ms/step - train_loss: 0.008515 - val_loss: 0.012204 - time: 77.023174
Epoch 48/50
102/102 [=====] - 246s 9ms/step - train_loss: 0.008492 - val_loss: 0.010502 - time: 77.254073
Epoch 49/50
102/102 [=====] - 246s 9ms/step - train_loss: 0.008275 - val_loss: 0.009578 - time: 77.068954
Epoch 50/50
102/102 [=====] - 245s 8ms/step - train_loss: 0.008234 - val_loss: 0.008679 - time: 77.036747

```

Gambar 4.16 Data Log Pelatihan Model CNN-BiGRU

```

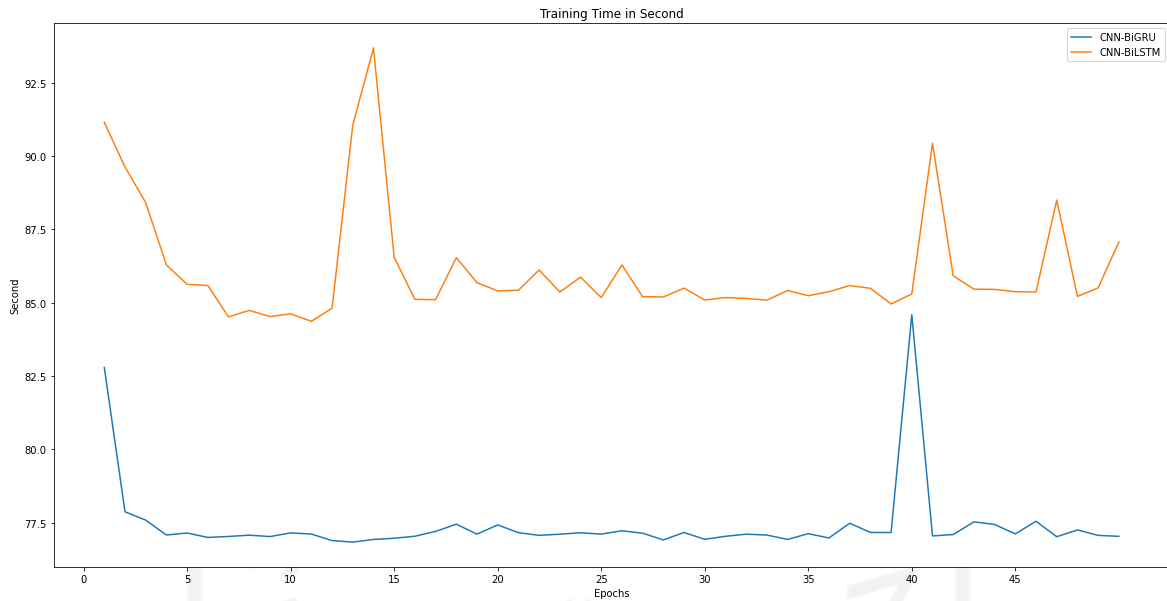
Epoch 44/50
102/102 [=====] - 246s 9ms/step - train_loss: 0.009023 - val_loss: 0.011081 - time: 85.457206
Epoch 45/50
102/102 [=====] - 267s 10ms/step - train_loss: 0.008721 - val_loss: 0.009194 - time: 85.376669
Epoch 46/50
102/102 [=====] - 256s 9ms/step - train_loss: 0.018222 - val_loss: 0.023541 - time: 85.370227
Epoch 47/50
102/102 [=====] - 250s 10ms/step - train_loss: 0.013284 - val_loss: 0.012146 - time: 88.500897
Epoch 48/50
102/102 [=====] - 246s 9ms/step - train_loss: 0.009312 - val_loss: 0.012026 - time: 85.219865
Epoch 49/50
102/102 [=====] - 246s 9ms/step - train_loss: 0.009125 - val_loss: 0.011460 - time: 85.507123
Epoch 50/50
102/102 [=====] - 260s 10ms/step - train_loss: 0.009060 - val_loss: 0.011662 - time: 87.074148

```

Gambar 4.17 Data Log Pelatihan Model CNN-BiLSTM

Pada Gambar 4.16 di atas menunjukkan bahwa nilai loss terendah pada model CNN-BiGRU didapatkan pada *epoch* ke 50 dengan nilai *training loss* sebesar **0.008234** dan nilai *validation loss* sebesar **0.008679**. Selanjutnya pada Gambar 4.17 di atas menunjukkan bahwa nilai loss terendah pada model CNN-BiLSTM didapatkan pada *epoch* ke 45 dengan nilai *training loss* sebesar **0.008721** dan nilai *validation loss* sebesar **0.009194**. Dapat dilihat bahwa hasil perbandingan pada nilai loss model CNN-BiGRU memiliki nilai loss yang lebih kecil dibandingkan nilai loss pada model CNN-BiLSTM.

Selanjutnya dilakukan perbandingan kedua yaitu perbandingan *training time* pada model CNN-BiGRU dan model CNN-BiLSTM. Pada penelitian ini lama waktu proses pelatihan dalam tiap *epoch* disimpan dan digunakan sebagai data untuk perbandingan. Data *training time* disimpan dalam bentuk csv, kemudian dilakukan visualisasi menggunakan data tersebut pada masing-masing model. Pada Gambar 4.18 menunjukkan grafik perbandingan *training time* pada model CNN-BiGRU dan CNN-BiLSTM.



Gambar 4.18 Grafik Perbandingan Training Time

Berdasarkan Gambar 4.18 di atas menunjukkan bahwa *training time* pada model CNN-BiGRU berada pada rentang waktu 76 sampai dengan 85 detik. Proses *training time* dengan waktu tercepat pada model CNN-BiGRU ditunjukkan pada *epoch* ke 13. Seperti ditunjukkan pada Gambar 4.19. Sedangkan proses *training time* pada model CNN-BiLSTM berada pada rentang waktu 84 sampai dengan 94 detik. Proses *training time* tercepat pada model CNN-BiLSTM ditunjukkan pada *epoch* ke 11. Seperti ditunjukkan pada Gambar 4.20.

```

Epoch 10/50
102/102 [=====] - 246s 8ms/step - train_loss: 0.083461 - val_loss: 0.064986 - time: 77.152558
Epoch 11/50
102/102 [=====] - 247s 8ms/step - train_loss: 0.077024 - val_loss: 0.057896 - time: 77.114078
Epoch 12/50
102/102 [=====] - 246s 9ms/step - train_loss: 0.067477 - val_loss: 0.053282 - time: 76.891689
Epoch 13/50
102/102 [=====] - 246s 9ms/step - train_loss: 0.059816 - val_loss: 0.049107 - time: 76.840465
Epoch 14/50
102/102 [=====] - 245s 8ms/step - train_loss: 0.055452 - val_loss: 0.045029 - time: 76.928996

```

Gambar 4.19 Data Log Training Time Model CNN-BiGRU

```

Epoch 10/50
102/102 [=====] - 246s 8ms/step - train_loss: 0.080427 - val_loss: 0.056988 - time: 84.622446
Epoch 11/50
102/102 [=====] - 247s 8ms/step - train_loss: 0.066348 - val_loss: 0.049980 - time: 84.369327
Epoch 12/50
102/102 [=====] - 246s 9ms/step - train_loss: 0.062245 - val_loss: 0.045974 - time: 84.812002
Epoch 13/50
102/102 [=====] - 246s 9ms/step - train_loss: 0.052207 - val_loss: 0.042490 - time: 91.058042
Epoch 14/50
102/102 [=====] - 245s 8ms/step - train_loss: 0.050350 - val_loss: 0.039076 - time: 93.691234

```

Gambar 4.20 Data Log Training Time Model CNN-BiLSTM

Pada Gambar 4.19 dan Gambar 4.20 di atas dapat dilihat bahwa *training time* pada model CNN-BiGRU memiliki waktu *training* tercepat pada *epoch* ke 13 dengan nilai sebesar **76.840464** detik. Sedangkan *training time* pada model CNN-BiLSTM memiliki waktu pelatihan tercepat pada *epoch* ke 11 dengan nilai sebesar **84.369327** detik. Dapat dilihat bahwa proses *training time* yang didapatkan oleh model CNN-BiGRU memiliki nilai lebih kecil dibandingkan *training time* pada model CNN-BiLSTM.

Dari dua pengujian yang telah dilakukan, didapatkan hasil perbandingan pada model CNN-BiGRU dan model CNN-BiLSTM. Pada Tabel 4.11 menunjukkan hasil perbandingan pada kedua model untuk nilai loss dan Tabel 4.12 menunjukkan hasil perbandingan *training time* pada masing-masing model. Berdasarkan hasil perbandingan nilai loss pada Tabel 4.11, dapat dilihat bahwa nilai *training loss* dan nilai *validation loss* pada model CNN-BiGRU memiliki nilai loss yang lebih kecil dari model CNN-BiLSTM. Selisih nilai *training loss* dari kedua model adalah **0.000487** dan selisih nilai *validation loss* dari kedua model adalah **0.000515**. Sedangkan berdasarkan pada Tabel 4.12 dapat dilihat bahwa model CNN-BiGRU memiliki waktu *training time* lebih cepat dibandingkan model CNN-BiLSTM. Selisih waktu *training time* dari kedua model adalah **7.528862** detik.

Tabel 4.11 Hasil Perbandingan Nilai Loss

No	Model	Jumlah Epoch	Training Loss	Validation Loss
1	CNN-BiGRU	50	0.008234	0.008679
2	CNN-BiLSTM	45	0.008721	0.009194

Tabel 4.12 Hasil Perbandingan Training Time

No	Model	Jumlah Epoch	Training Time
1	CNN-BiGRU	13	76.840465 detik
2	CNN-BiLSTM	11	84.369327 detik

BAB 5

Kesimpulan dan Saran

5.1 Kesimpulan

Pada penelitian ini menunjukkan bahwa model *named entity recognition* yang dibangun dengan pendekatan *deep learning* menggunakan algoritma CNN-BiGRU berhasil melakukan pengenalan entitas. Model yang dibangun dapat mengidentifikasi kalimat berbahasa Indonesia yang mengandung kata obat ke dalam jenis entitasnya seperti nama obat, kandungan obat, dan komposisi obat.

Berdasarkan eksperimen, penerapan POS *tagging* dan BIO *tagging* memudahkan proses pelabelan secara otomatis. Sehingga model yang dibangun berhasil mengklasifikasikan kata berdasarkan jenis entitasnya dengan baik. Pada proses pelatihan model, penggunaan parameter utama dan *hyperparameter* mempengaruhi hasil akurasi pada proses pembentukan model. Pada hasil perbandingan model menunjukkan bahwa model CNN-BiGRU memiliki nilai loss lebih rendah dan *training time* yang lebih cepat dibandingkan dengan model pada penelitian sebelumnya yaitu model CNN-BiLSTM.

5.2 Saran

Adapun saran untuk pengembangan penelitian selanjutnya adalah:

1. Penelitian ini difokuskan pada obat sintetik yang memiliki kandungan dan dosis. Untuk penelitian selanjutnya, dapat diperluas dengan pelabelan entitas mencakup ekstraksi entitas pada obat herbal.
2. Pada penelitian selanjutnya dimungkinkan juga untuk menambahkan beberapa kategori lagi untuk mengklasifikasikan entitas yang diekstraksi seperti reaksi obat yang merugikan dan efek samping obat.
3. Pada penelitian selanjutnya dapat ditambahkan algoritma lain seperti *conditional random field* untuk mendapatkan hasil akurasi yang lebih baik.

Daftar Pustaka

- Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017). Understanding of a convolutional neural network. *2017 International Conference on Engineering and Technology (ICET)*, 1–6. <https://doi.org/10.1109/ICEngTechnol.2017.8308186>
- Ayifu, M., Wushouer, S., & Palidan, M. (2019). Multilingual named entity recognition based on the BiGRU-CNN-CRF hybrid model. *International Journal of Information and Communication Technology*, *15*(3), 223. <https://doi.org/10.1504/IJICT.2019.102996>
- Bayer, J. (2017). *Development of a Modular Software Framework for Supporting Architects During Early Design Phases*. <https://doi.org/10.13140/RG.2.2.11442.09924>
- Bengio, Y., Ducharme, R., & Vincent, P. (2000). A Neural Probabilistic Language Model. *Advances in Neural Information Processing Systems*, *13*. <https://proceedings.neurips.cc/paper/2000/hash/728f206c2a01bf572b5940d7d9a8fa4c-Abstract.html>
- Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, *5*(2), 157–166. <https://doi.org/10.1109/72.279181>
- Bolucu, N., Akgol, D., & Tuc, S. (2019). Bidirectional LSTM-CNNs with Extended Features for Named Entity Recognition. *2019 Scientific Meeting on Electrical-Electronics & Biomedical Engineering and Computer Science (EBBT)*, 1–4. <https://doi.org/10.1109/EBBT.2019.8741631>
- Chiu, J. P. C., & Nichols, E. (2016). Named Entity Recognition with Bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, *4*, 357–370. https://doi.org/10.1162/tacl_a_00104

- Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1724–1734. <https://doi.org/10.3115/v1/D14-1179>
- Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling* (arXiv:1412.3555). arXiv. <https://doi.org/10.48550/arXiv.1412.3555>
- Dai, Z., Wang, X., Ni, P., Li, Y., Li, G., & Bai, X. (2019). Named Entity Recognition Using BERT BiLSTM CRF for Chinese Electronic Health Records. *2019 12th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, 1–5. <https://doi.org/10.1109/CISP-BMEI48845.2019.8965823>
- Deng, L. (2014). Deep Learning: Methods and Applications. *Foundations and Trends® in Signal Processing*, 7(3–4), 197–387. <https://doi.org/10.1561/20000000039>
- Dey, R., & Salem, F. M. (2017). Gate-variants of Gated Recurrent Unit (GRU) neural networks. *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, 1597–1600. <https://doi.org/10.1109/MWSCAS.2017.8053243>
- dos Santos, C., & Guimarães, V. (2015). Boosting Named Entity Recognition with Neural Character Embeddings. *Proceedings of the Fifth Named Entity Workshop*, 25–33. <https://doi.org/10.18653/v1/W15-3904>
- Fatimah, N., Budi, I., Santoso, A. B., & Putra, P. K. (2021). Analysis of Mental Health During the Covid-19 Pandemic in Indonesia using Twitter Data. *2021 8th International Conference on Advanced Informatics: Concepts, Theory and*

Applications (ICAICTA), 1–6.

<https://doi.org/10.1109/ICAICTA53211.2021.9640265>

Fudholi, D. H., Nayoan, R. A. N., Hidayatullah, A. F., & Arianto, D. B. (2022). *A HYBRID*

CNN-BILSTM MODEL FOR DRUG NAMED ENTITY RECOGNITION. 17, 15.

Gholamy, A., Kreinovich, V., & Kosheleva, O. (2018). Why 70/30 or 80/20 Relation

Between Training and Testing Sets: A Pedagogical Explanation. *Departmental*

Technical Reports (CS). https://scholarworks.utep.edu/cs_techrep/1209

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. The MIT Press.

Goyal, A., Gupta, V., & Kumar, M. (2022). Deep Learning-Based Named Entity

Recognition System Using Hybrid Embedding. *Cybernetics and Systems*, 0(0), 1–

23. <https://doi.org/10.1080/01969722.2022.2111506>

Graves, A. (2012). *Supervised Sequence Labelling with Recurrent Neural Networks* (Vol.

385). Springer Berlin Heidelberg. <https://doi.org/10.1007/978-3-642-24797-2>

Graves, A. (2014). *Generating Sequences With Recurrent Neural Networks*

(arXiv:1308.0850). arXiv. <http://arxiv.org/abs/1308.0850>

Gridach, M., & Haddad, H. (2018). Arabic Named Entity Recognition: A Bidirectional

GRU-CRF Approach. In A. Gelbukh (Ed.), *Computational Linguistics and*

Intelligent Text Processing (pp. 264–275). Springer International Publishing.

https://doi.org/10.1007/978-3-319-77113-7_21

Guo, Y., Liu, Y., Oerlemans, A., Lao, S., Wu, S., & Lew, M. S. (2016). Deep learning for

visual understanding: A review. *Neurocomputing*, 187, 27–48.

<https://doi.org/10.1016/j.neucom.2015.09.116>

Hanif, I. (2018, May 1). *Klasifikasi Perintah Bahasa Natural Menggunakan Global*

Vectors for Word Representations (GloVe), Convolutional Neural Networks, dan

Teknik Transfer Learning pada Aplikasi Chatbots.

- <https://www.semanticscholar.org/paper/Klasifikasi-Perintah-Bahasa-Natural-Menggunakan-for-Hanif/08a24731b3859a437e0ba97c64378ab69fb33bd4>
- Hidayatullah, A. F., & Ma'arif, M. R. (2017). Pre-processing Tasks in Indonesian Twitter Messages. *Journal of Physics: Conference Series*, 801, 012072.
<https://doi.org/10.1088/1742-6596/801/1/012072>
- Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8), 1735–1780. <https://doi.org/10.1162/neco.1997.9.8.1735>
- Juwantara, Z., Eddy, F., Sasmita, D. H., Khoolish, T. N., & Aldiyansyah, B. (2021). Kumparan NLP library. *Zenodo*.
- Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014). A Convolutional *Neural network* for Modelling Sentences. *ArXiv:1404.2188 [Cs]*.
<http://arxiv.org/abs/1404.2188>
- Kale, S., & Govilkar, S. (2017). Survey of Named Entity Recognition Techniques for Various Indian Regional Languages. *International Journal of Computer Applications*, 164(4), 37–43. <https://doi.org/10.5120/ijca2017913621>
- Kang, Y., Cai, Z., Tan, C.-W., Huang, Q., & Liu, H. (2020). Natural language processing (NLP) in management research: A literature review. *Journal of Management Analytics*, 7(2), 139–172. <https://doi.org/10.1080/23270012.2020.1756939>
- Kemenkes RI. (2018). *KEPUTUSAN MENTERI KESEHATAN REPUBLIK INDONESIA TAHUN 2018*.
- Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1746–1751. <https://doi.org/10.3115/v1/D14-1181>
- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing*

Systems, 25.

<https://papers.nips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html>

Kumawat, D., & Jain, V. (2015). POS Tagging Approaches: A Comparison. *International Journal of Computer Applications*, 118(6), 32–38. <https://doi.org/10.5120/20752-3148>

Lauriola, I., Lavelli, A., & Aiolli, F. (2022). An introduction to Deep Learning in Natural Language Processing: Models, techniques, and tools. *Neurocomputing*, 470, 443–456. <https://doi.org/10.1016/j.neucom.2021.05.103>

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278–2324. <https://doi.org/10.1109/5.726791>

Li, J., Sun, A., Han, J., & Li, C. (2020). *A Survey on Deep Learning for Named Entity Recognition* (arXiv:1812.09449). arXiv. <http://arxiv.org/abs/1812.09449>

Luo, L. (2019). Network text sentiment analysis method combining LDA text representation and GRU-CNN. *Personal and Ubiquitous Computing*, 23(3), 405–412. <https://doi.org/10.1007/s00779-018-1183-9>

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *Efficient Estimation of Word Representations in Vector Space* (arXiv:1301.3781). arXiv. <https://doi.org/10.48550/arXiv.1301.3781>

Ni, P., Li, G., Hung, P. C. K., & Chang, V. (2021). StaResGRU-CNN with CMedLMs: A stacked residual GRU-CNN with pre-trained biomedical language models for predictive intelligence. *Applied Soft Computing*, 113, 107975. <https://doi.org/10.1016/j.asoc.2021.107975>

- Onan, A. (2022). Bidirectional convolutional recurrent *neural network* architecture with group-wise enhancement mechanism for text sentiment classification. *Journal of King Saud University - Computer and Information Sciences*, 34(5), 2098–2117. <https://doi.org/10.1016/j.jksuci.2022.02.025>
- Patil, N., Patil, A., & Pawar, B. V. (2020). Named Entity Recognition using Conditional Random Fields. *Procedia Computer Science*, 167, 1181–1188. <https://doi.org/10.1016/j.procs.2020.03.431>
- Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1532–1543. <https://doi.org/10.3115/v1/D14-1162>
- Pisceldo, F. (2009). *Probabilistic Part Of Speech Tagging for Bahasa Indonesia*. <https://www.semanticscholar.org/paper/Probabilistic-Part-Of-Speech-Tagging-for-Bahasa-Pisceldo/260d6049f012c9405ba10fa32dc416382ee8ad6f>
- Piskorski, J., & Yangarber, R. (2013). Information Extraction: Past, Present and Future. In T. Poibeau, H. Saggion, J. Piskorski, & R. Yangarber (Eds.), *Multi-source, Multilingual Information Extraction and Summarization* (pp. 23–49). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-28569-1_2
- Qiu, J., Wang, B., & Zhou, C. (2020). Forecasting stock prices with long-short term memory *neural network* based on attention mechanism. *PLOS ONE*, 15(1), e0227222. <https://doi.org/10.1371/journal.pone.0227222>
- Qiu, X., & Huang, X. (2015, July 25). *Convolutional Neural Tensor Network Architecture for Community-Based Question Answering*. International Joint Conference on Artificial Intelligence. <https://www.semanticscholar.org/paper/Convolutional->

Neural-Tensor-Network-Architecture-Qiu-

Huang/7a9c4c98e361f3b4f4bfbeea7e6699917ce42091

- Ramshaw, L. A., & Marcus, M. P. (1995). *Text Chunking using Transformation-Based Learning* (arXiv:cmp-lg/9505040). arXiv. <http://arxiv.org/abs/cmp-lg/9505040>
- Rashel, F., Luthfi, A., Dinakaramani, A., & Manurung, R. (2014). Building an Indonesian rule-based part-of-speech tagger. *2014 International Conference on Asian Language Processing (IALP)*, 70–73. <https://doi.org/10.1109/IALP.2014.6973521>
- Salehinejad, H., Sankar, S., Barfett, J., Colak, E., & Valaee, S. (2018). *Recent Advances in Recurrent Neural Networks* (arXiv:1801.01078). arXiv. <http://arxiv.org/abs/1801.01078>
- Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11), 2673–2681. <https://doi.org/10.1109/78.650093>
- Thomas, A., & Sangeetha, S. (2020). Deep Learning Architectures for Named Entity Recognition: A Survey. In B. Pati, C. R. Panigrahi, R. Buyya, & K.-C. Li (Eds.), *Advanced Computing and Intelligent Engineering* (Vol. 1082, pp. 215–225). Springer Singapore. https://doi.org/10.1007/978-981-15-1081-6_18
- Wang, S., & Cao, J. (2021). AI and Deep Learning for Urban Computing. In W. Shi, M. F. Goodchild, M. Batty, M.-P. Kwan, & A. Zhang (Eds.), *Urban Informatics* (pp. 815–844). Springer. https://doi.org/10.1007/978-981-15-8983-6_43
- Wintaka, D. C., Bijaksana, M. A., & Asror, I. (2019). Named-Entity Recognition on Indonesian Tweets using Bidirectional LSTM-CRF. *Procedia Computer Science*, 157, 221–228. <https://doi.org/10.1016/j.procs.2019.08.161>

- Wu, Y., Jiang, M., Xu, J., Zhi, D., & Xu, H. (2018). Clinical Named Entity Recognition Using Deep Learning Models. *AMIA Annual Symposium Proceedings, 2017*, 1812–1819.
- Xu, J., Zheng, X., & Jiang, M. (2019). Gene Mutation Classification Using CNN and BiGRU Network. *2019 9th International Conference on Information Science and Technology (ICIST)*, 397–401. <https://doi.org/10.1109/ICIST.2019.8836846>
- You, Y., Gitman, I., & Ginsburg, B. (2017). Large Batch Training of Convolutional Networks. *ArXiv: Computer Vision and Pattern Recognition*.
<https://www.semanticscholar.org/paper/Large-Batch-Training-of-Convolutional-Networks-You-Gitman/1e3d18beaf3921f561e1b999780f29f2b23f3b7d>
- Zeenot, S. (2013). *Pengelolaan & penggunaan obat wajib apotek*. D-Medika.
<https://books.google.co.id/books?id=9MXRnQEACAAJ>
- Zhang, Y., & Wallace, B. (2016). *A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification* (arXiv:1510.03820). arXiv. <https://doi.org/10.48550/arXiv.1510.03820>