TESIS

OPTIMALISASI Bi-LSTM DENGAN *GENETIC ALGORITHM* UNTUK MENDETEKSI ENTITAS LOKASI DAN WAKTU PADA DATA TEKS BAHASA INDONESIA TENTANG KEBAKARAN HUTAN

Disusun oleh:

Nama : **Dwi Ahmad Dzulhijjah**
NIM : **22.55.2321**
Konsentrasi : **Business Intelligence**

PROGRAM STUDI S2 INFORMATIKA
PROGRAM PASCASARJANA UNIVERSITAS AMIKOM YOGYAKARTA
YOGYAKARTA
2024

TESIS

OPTIMALISASI Bi-LSTM DENGAN *GENETIC ALGORITHM* UNTUK MENDETEKSI ENTITAS LOKASI DAN WAKTU PADA DATA TEKS BAHASA INDONESIA TENTANG KEBAKARAN HUTAN

Bi-LSTM WITH GENETIC ALGORITHM OPTIMIZATION FOR LOCATION AND TIME ENTITY RECOGNITION IN INDONESIAN FOREST FIRES TEXT DATA

Diajukan untuk memenuhi salah satu syarat memperoleh derajat Magister

Disusun oleh:

Nama : **Dwi Ahmad Dzulhijjah**
NIM : **22.55.2321**
Konsentrasi : **Business Intelligence**

PROGRAM STUDI S2 INFORMATIKA
PROGRAM PASCASARJANA UNIVERSITAS AMIKOM YOGYAKARTA
YOGYAKARTA
2024

HALAMAN PENGESAHAN

OPTIMALISASI Bi-LSTM DENGAN *GENETIC ALGORITHM* UNTUK MENDETEKSI ENTITAS LOKASI DAN WAKTU PADA DATA TEKS BAHASA INDONESIA TENTANG KEBAKARAN HUTAN

Bi-LSTM WITH GENETIC ALGORITHM OPTIMIZATION FOR LOCATION AND TIME ENTITY RECOGNITION IN INDONESIAN FOREST FIRES TEXT DATA

Dipersiapkan dan Disusun oleh

Dwi Ahmad Dzulhijjah

22.55.2321

Telah Diujikan dan Dipertahankan dalam Sidang Ujian Tesis
Program Studi S2 Informatika
Program Pascasarjana Universitas AMIKOM Yogyakarta
pada hari Senin, 10 Juni 2024

Tesis ini telah diterima sebagai salah satu persyaratan
untuk memperoleh gelar Magister Komputer

Yogyakarta, 10 Juni 2024
Rektor

Prof. Dr. M. Suyanto, M.M.
NIK. 190302001

HALAMAN PERSETUJUAN

OPTIMALISASI Bi-LSTM DENGAN *GENETIC ALGORITHM* UNTUK MENDETEKSI ENTITAS LOKASI DAN WAKTU PADA DATA TEKS BAHASA INDONESIA TENTANG KEBAKARAN HUTAN

Bi-LSTM WITH GENETIC ALGORITHM OPTIMIZATION FOR LOCATION AND TIME ENTITY RECOGNITION IN INDONESIAN FOREST FIRES TEXT DATA

Dipersiapkan dan Disusun oleh

Dwi Ahmad Dzulhijjah

22.55.2321

Telah Diujikan dan Dipertahankan dalam Sidang Ujian Tesis
Program Studi S2 Informatika
Program Pascasarjana Universitas AMIKOM Yogyakarta
pada hari Senin, 10 Juni 2024

Pembimbing Utama

Prof. Dr. Kusrini, M.Kom.
NIK. 190302106

Anggota Tim Penguji

Dhani Ariatmanto, S.Kom., M.Kom., Ph.D.
NIK. 190302197

M. Haaziq, S.Kom., M.Eng., Ph.D.
NIK. 190302024

Pembimbing Pendamping

Dr. Kumara Ari Yuana, S.T., M.T.
NIK. 190302575

Prof. Dr. Kusrini, M.Kom.
NIK. 190302106

Tesis ini telah diterima sebagai salah satu persyaratan
untuk memperoleh gelar Magister Komputer

Yogyakarta, 10 Juni 2024
Direktur Program Pascasarjana

Prof. Dr. Kusrini, M.Kom.
NIK. 190302106

HALAMAN PERNYATAAN KEASLIAN TESIS

Yang bertandatangan di bawah ini,

Nama mahasiswa : Dwi Ahmad Dzulhijjah
NIM : 22.55.2321
Konsentrasi : Business Intelligence

Menyatakan bahwa Tesis dengan judul berikut:
Optimalisasi Bi-LSTM dengan *Genetic Algorithm* untuk Mendeteksi Entitas Lokasi dan Waktu pada Data Teks Bahasa Indonesia tentang Kebakaran Hutan

Dosen Pembimbing Utama : Prof. Dr. Kusrini, M.Kom.
Dosen Pembimbing Pendamping : Dr. Kumara Ari Yuana, S.T., M.T.

1. Karya tulis ini adalah benar-benar ASLI dan BELUM PERNAH diajukan untuk mendapatkan gelar akademik, baik di Universitas AMIKOM Yogyakarta maupun di Perguruan Tinggi lainnya
2. Karya tulis ini merupakan gagasan, rumusan dan penelitian SAYA sendiri, tanpa bantuan pihak lain kecuali arahan dari Tim Dosen Pembimbing
3. Dalam karya tulis ini tidak terdapat karya atau pendapat orang lain, kecuali secara tertulis dengan jelas dicantumkan sebagai acuan dalam naskah dengan disebutkan nama pengarang dan disebutkan dalam Daftar Pustaka pada karya tulis ini
4. Perangkat lunak yang digunakan dalam penelitian ini sepenuhnya menjadi tanggung jawab SAYA, bukan tanggung jawab Universitas AMIKOM Yogyakarta
5. Pernyataan ini SAYA buat dengan sesungguhnya, apabila di kemudian hari terdapat penyimpangan dan ketidakbenaran dalam pernyataan ini, maka SAYA bersedia menerima SANKSI AKADEMIK dengan pencabutan gelar yang sudah diperoleh, serta sanksi lainnya sesuai dengan norma yang berlaku di Perguruan Tinggi

Yogyakarta, 10 Juni 2024
Yang Menyatakan,

Dwi Ahmad Dzulhijjah

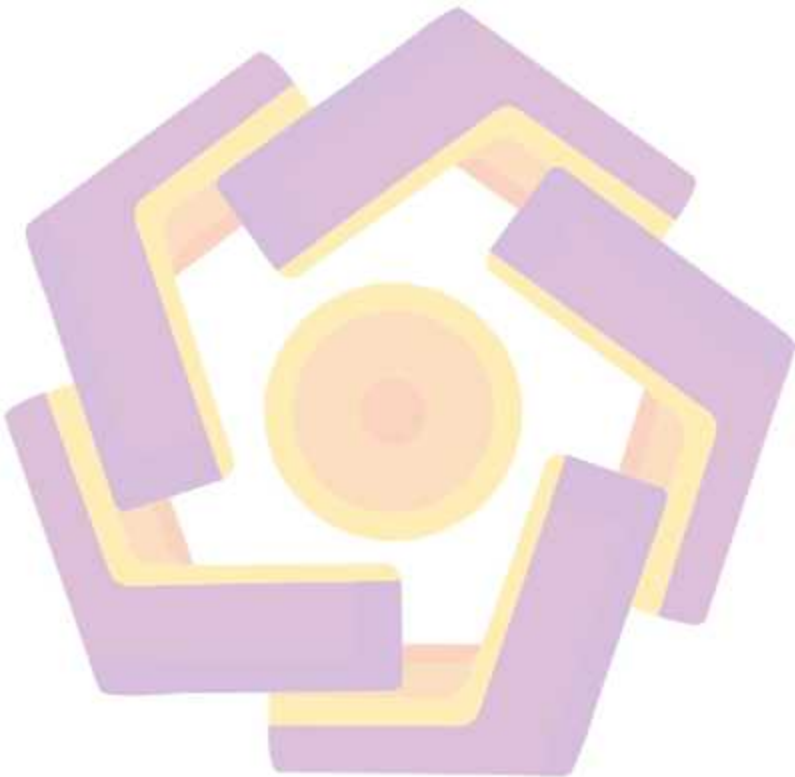
HALAMAN PERSEMBAHAN

Tesis ini saya persembahkan untuk segala dzat apapun yang sudah membantu menyelesaikan.

HALAMAN MOTTO

"Seize the day or die regretting the time you lost"

"Carpe diem!"



KATA PENGANTAR

Segala puji dan ungkapan syukur penulis ucapkan kepada siapapun dan apapun yang dapat menjadi pendorong, motivasi, dan *support* dalam menyelesaikan proyek tesis dan laporan tesis ini.

Pada kesempatan ini, penulis menyampaikan terima kasih kepada yang terhormat:

1. Prof. Dr. Kusrini, M.Kom. sebagai pembimbing utama dan inspirator penulis.
2. Dr. Kumara Ari Yuana, S.T., M.T. sebagai pembimbing pendamping.
3. Kedua orangtua saya, Bapak Suriansyah Abdurrazak dan Ibu Aspah sebagai *support system* dan alasan untuk hidup.
4. Saudari saya Eka Suci Fajariah dan saudara saya Trias Dzul Safawi yang menjadi *support system* utama.
5. Anggota studio Muhammad Ilham Alfianu, Fahri Albar, Syaifurrido, Davin, Rendy yang memotivasi dan pelipur lara.
6. Kepada staff Universitas Amikom Yogyakarta.
7. Kepada dosen-dosen pascasarjana Universitas Amikom.
8. Kepada teman-teman PJJ MTI selama berkuliah.
9. Konsosrium SILVANUS yang sudah menyediakan data untuk penelitian.
10. Kepada teman-teman rekan kerja yang sudah memaklumi kegiatan dan aktivitas saya, berikut kepada atasan-atasan dalam magang, kerja, dan project.

11. Kepada seluruh orang baik terlibat secara langsung maupun tidak langsung, baik fisik maupun mental.

Demikianlah kata pengantar saya untuk ucapa terimakasih dan rasa syukur atas segala bantuan, *support system*, dan kemudahan serta kemakluman dari semua pihak.

Yogyakarta, 13 Mei 2024

Penulis

DAFTAR ISI

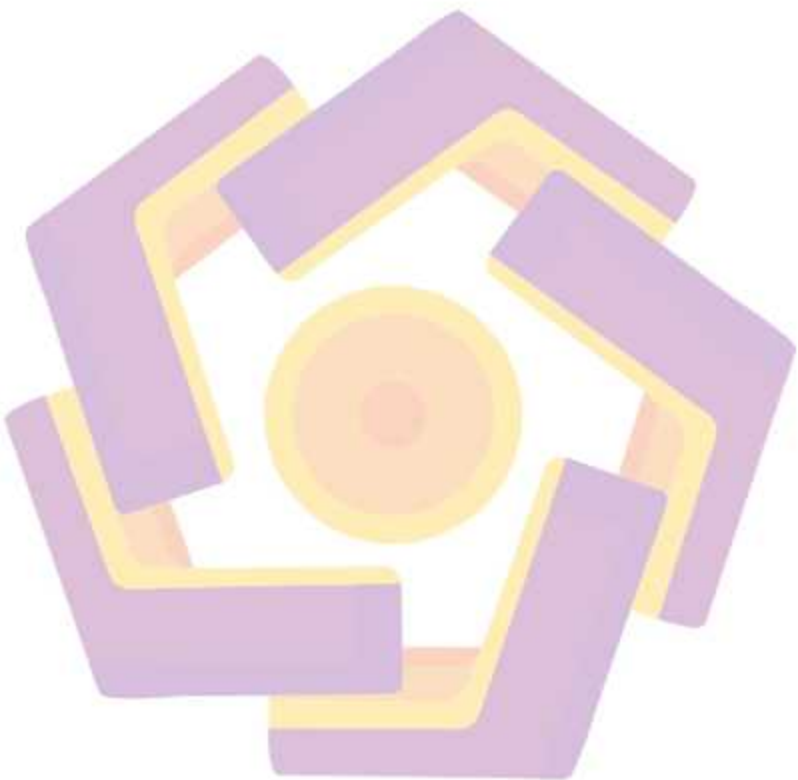
DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR ISTILAH

1. Algoritma Genetika: (Dari terjemahan : *Genetic Algorithm*) Merupakan metode komputasional yang terinspirasi oleh teori evolusi dan seleksi alam untuk menemukan solusi optimal untuk masalah yang kompleks.
2. LSTM (Long Short-Term Memory): Jenis arsitektur jaringan saraf rekurensial (RNN) yang dirancang untuk mengatasi masalah vanishing gradient dan mempertahankan informasi jangka panjang.
3. Bi-LSTM (Bidirectional Long Short-Term Memory): Varian dari LSTM yang memungkinkan model untuk memproses urutan data dalam dua arah, yaitu maju dan mundur.
4. *Hyperparameter*: Parameter yang nilainya tidak ditentukan selama proses pembelajaran model dan harus diatur sebelum pelatihan dimulai, seperti learning rate, jumlah unit, dll.
5. *NER (Named Entity Recognition)*: Tugas dalam pemrosesan bahasa alami (NLP) yang bertujuan untuk menemukan dan mengkategorikan entitas penting dalam teks, seperti nama orang, tempat, tanggal, dll.
6. Kebakaran Hutan: Kejadian di mana api menyebar secara luas di hutan, sering kali menyebabkan kerusakan lingkungan yang serius.
7. Lokasi dan Waktu: Entitas dalam teks yang merujuk pada informasi tentang tempat dan waktu tertentu.
8. *Units*: Jumlah neuron dalam suatu lapisan jaringan saraf.
9. *Dropout*: Teknik pencegahan overfitting dalam jaringan saraf dengan secara acak mengabaikan sebagian unit selama pelatihan.
10. *Embedding Dimension*: Jumlah dimensi yang digunakan untuk merepresentasikan kata atau fitur dalam ruang vektor.
11. *Learning Rate*: Parameter yang mengontrol seberapa besar langkah yang diambil saat memperbarui bobot model selama pelatihan.
12. *Batch Size*: Jumlah contoh data yang diproses dalam satu iterasi pelatihan.
13. *Epoch*: Satu iterasi penuh dari seluruh dataset yang digunakan dalam pelatihan model.
14. *Layer*: Struktur yang terdiri dari satu atau beberapa unit pemrosesan dalam jaringan saraf.

15. Regularisasi Kernel: Teknik untuk mengendalikan kompleksitas model dengan menambahkan *penalty* ke fungsi objektif berdasarkan norma L2 dari bobot.



INTISARI

Kebakaran hutan merupakan bencana yang sering terjadi di Indonesia dan menimbulkan dampak yang signifikan. Deteksi dini lokasi dan waktu kebakaran hutan sangat penting untuk upaya penanggulangan yang efektif. Penelitian ini bertujuan untuk meningkatkan akurasi deteksi entitas lokasi dan waktu pada data teks bahasa Indonesia tentang kebakaran hutan dengan mengoptimalkan model Bidirectional Long Short-Term Memory (Bi-LSTM) menggunakan algoritma genetika (GA).

Model Bi-LSTM dilatih pada data teks yang telah di anotasi dengan entitas lokasi dan waktu. Algoritma GA digunakan untuk mengoptimalkan *hyperparameter* model Bi-LSTM, seperti *units, dropout, embedding dimension, learning rate, batch size, epoch, layer,* dan *regularisasi kernel*. Eksperimen dilakukan dengan empat skema, skema 1 *hyperparameter default*, skema 2 dengan populasi 10 dan generasi 25 dengan total 250 evaluasi, skema 3 dengan 25 populasi dan 50 generasi serta 1250 evaluasi, dan skema 4 dengan 50 populasi dan 20 generasi dengan total 1000 evaluasi.

Hasil penelitian secara eksperimental menunjukkan bahwa optimasi model Bi-LSTM dengan GA dapat meningkatkan akurasi deteksi entitas lokasi dan waktu pada data teks bahasa Indonesia tentang kebakaran hutan. Skema 4 menghasilkan performa terbaik dengan F1 Score 0.965 dan Loss 0.165, yang lebih tinggi dibandingkan Skema 3 dengan F1 Score 0.963 dan Loss 0.151, Skema 2 dengan F1 Score 0.946 dan Loss 0.242, dan Skema 1 (Default) dengan F1 Score 0.937 dan Loss 0.158. Walaupun pada uji coba hasil hyperparameter selanjutnya memiliki hasil berbeda, namun secara konsisten *hyperparameter* yang dioptimasi memiliki keunggulan lebih baik daripada secara *default*, hal ini dibuktikan dengan uji K-Fold Cross Validation dengan K=5 dan K=10 menunjukkan peningkatan rata-rata F1 Score dan Loss pada skema optimasi dibandingkan skema non-optimasi.

Kata kunci: Kebakaran hutan, deteksi entitas, Bi-LSTM, Algoritma genetika, Bahasa Indonesia

ABSTRACT

Forest fires are a frequently occurring disaster in Indonesia, causing significant impacts. Early detection of the location and timing of forest fires is crucial for effective mitigation efforts. This research aims to improve the accuracy of detecting location and time entities in Indonesian text data about forest fires by optimizing the Bidirectional Long Short-Term Memory (Bi-LSTM) model using a Genetic Algorithm (GA). The Bi-LSTM model is trained on text data that has been annotated with location and time entities. The GA algorithm is used to optimize the hyperparameters of the Bi-LSTM model, such as units, dropout, embedding dimension, learning rate, batch size, epoch, layer, and kernel regularization. Experiments are conducted with four schemes: scheme 1 with default hyperparameters, scheme 2 with a population of 10 and 25 generations totaling 250 evaluations, scheme 3 with a population of 25 and 50 generations totaling 1250 evaluations, and scheme 4 with a population of 50 and 20 generations totaling 1000 evaluations. Experimental results show that optimizing the Bi-LSTM model with GA can improve the accuracy of detecting location and time entities in Indonesian text data about forest fires. Scheme 4 achieves the best performance with an F1 Score of 0.973 and Loss of 0.003, higher than Scheme 2 with an F1 Score of 0.967 and Loss of 0.025, Scheme 3 with an F1 Score of 0.937 and Loss of 0.169, and Scheme 1 (Default) with an F1 Score of 0.90 and Loss of 0.150. Although subsequent hyperparameter test results vary, consistently, optimized hyperparameters have better advantages than default ones, as evidenced by K-Fold Cross-Validation tests with K=5 and K=10, showing an increase in average F1 Score and Loss in the optimization schemes compared to non-optimized schemes.

Keyword: Forest fires, Entity detection, Bi-LSTM, Genetic algorithm, Indonesian language

BAB I

PENDAHULUAN

1.1. Latar Belakang Masalah

Indonesia adalah salah satu negara dengan keragaman lingkungan yang luar biasa, termasuk hutan tropis yang luas. Namun, hutan-hutan ini sering menghadapi ancaman serius, seperti kebakaran hutan, yang dapat mengancam ekosistemnya, serta kesehatan dan kehidupan manusia. Kebakaran hutan telah menjadi permasalahan yang sering terjadi di berbagai wilayah Indonesia, terutama pada musim kemarau. Selain itu, kebakaran hutan juga dapat menyebabkan masalah serius terkait dengan kualitas udara, hilangnya habitat satwa liar, dan kerugian ekonomi yang signifikan (Pasai, Miswar., 2020).

Dalam penanganan kebakaran hutan, waktu dan lokasi merupakan informasi kunci yang sangat penting (Salsabila dkk., 2020). Penentuan lokasi dan waktu kebakaran hutan dengan cepat dan tepat sangat diperlukan untuk mengkoordinasikan upaya pemadaman, menginformasikan masyarakat, serta merencanakan tindakan selanjutnya. Sayangnya, pengumpulan informasi ini dari berbagai sumber seperti laporan masyarakat, satelit, atau sensor cuaca seringkali memerlukan waktu yang cukup lama, dan dapat terhambat oleh keterbatasan sumber daya manusia (Fatayat dan Risanto,. 2020).

Penggunaan teknologi pemrosesan bahasa alami *(Natural Language Processing - NLP)* dalam mendeteksi entitas lokasi dan waktu pada data teks dapat membantu dalam mempercepat pengumpulan informasi yang diperlukan

untuk penanganan kebakaran hutan. (Kurniawan dan Putra., 2020). Salah satu metode yang umum digunakan dalam *NLP* adalah penggunaan model berbasis *LSTM (Long Short-Term Memory)*, yang mampu memahami konteks dan urutan dalam teks (Pallavi., 2022).

Meskipun LSTM telah terbukti efektif dalam berbagai tugas *NLP*, perlu ada upaya untuk mengoptimalkan kinerja model ini, terutama ketika digunakan dalam bahasa Indonesia yang memiliki ciri khas tersendiri (Budi dan Suryono., 2023). Dalam konteks ini, penggunaan algoritma genetika dapat menjadi pendekatan yang potensial untuk mengoptimalkan parameter LSTM dan meningkatkan kemampuannya dalam mendeteksi entitas lokasi dan waktu dalam bahasa Indonesia.

Genetic algorithm atau dalam tesis ini juga disebut algoritma genetika (disingkat GA) merupakan algoritma optimasi yang dapat digunakan untuk merancang parameter-parameter arsitektur model *neural network* (Kumar dkk., 2021). Pada penelitian tinjauan studi yang telah dilakukan sebelumnya bahwasanya algoritma genetika dapat digunakan untuk mengoptimasi dan memilih parameter-parameter arsitektur model *Convolutional Neural network (CNN)* secara otomatis walau penggunanya bukan ahli dalam merancang CNN (Sun ,dkk., 2020). Terdapat studi terkini terkait pemanfaatan lain algoritma genetika dalam masalah *time series* dengan mengoptimasi model CNN, LSTM, dan RNN untuk memprediksi PM2.5 dan polutan udara dengan hasil optimal, metode yang diusulkan mengungguli konfigurasi lainnya, dengan kesalahan *MSE*

yang dikurangi sebesar 13,38% dan 55,30% untuk kinerja pengujian (Erden, C.,2023).

Terdapat banyak penelitian menggunakan teks atau korpus tweet (terjemahan : cuitan) yang diambil dari Twitter.com atau X.com menggunakan algoritma deep leaming untuk pengenalan entitas bernama pada beragam bahasa. Terdapat penelitian yang menggunakan deep learning untuk pengenalan entitas bernama untuk teks bahasa Urdu yang menunjukkan peningkatan performa F1 score sekitar 6.26% (Haq dkk., 2023). Pada penelitian (Majumder dkk., 2021) bahkan menggunakan multi-bahasa korpus dari Twitter untuk mendapatkan karakter dan entitas bernama dari bahasa *hindi-english*. Selain itu juga ada bahasa lain seperti bahasa arab (Alzaidi dkk., 2022), bahasa spanyol (Garcia Le Pera, J. L., 2020), dan bahasa indonesia (Budi dan Suryono.,2023).

Oleh karena itu penggunaan algoritma genetika digunakan untuk mengoptimalkan *hyperparameter* model LSTM dalam mendeteksi entitas lokasi dan waktu pada teks kebakaran hutan berbahasa indonesia. Dengan mengintegrasikan kedua teknologi yakni optimasi algoritma genetika dan LSTM, diharapkan dapat menciptakan solusi yang lebih efisien dan akurat dalam mendukung penanganan kebakaran hutan di Indonesia. Selain itu, kontribusi ilmiah dalam pengembangan metode NLP dan optimasi *hyperparameter* yang lebih efektif dalam bahasa Indonesia merupakan upaya dari penelitian ini.

1.2. Rumusan Masalah

Penelitian ini bertujuan untuk menyelesaikan beberapa masalah yang terkait dengan deteksi entitas lokasi dan waktu dalam teks berbahasa Indonesia yang berkaitan dengan kebakaran hutan menggunakan model Bidirectional Long Short-Term Memory (Bi-LSTM) yang dioptimalkan dengan algoritma genetika. Masalah yang ingin diselesaikan meliputi pengoptimalan model Bi-LSTM untuk meningkatkan kemampuannya dalam mengenali entitas lokasi dan waktu, mengevaluasi tingkat akurasi dari model yang telah dioptimalkan, serta membandingkan performa antara model Bi-LSTM dengan pengaturan default dan model Bi-LSTM yang dioptimalkan menggunakan algoritma genetika. Dari rumusan masalah tersebut, ditegaskan kembali secara eksplisit dalam pertanyaan riset berikut :

a. Bagaimana mengoptimalkan model Bidirectional Long Short-Term Memory (Bi-LSTM) menggunakan genetic algorithm (algoritma genetika) untuk mendeteksi entitas lokasi dan waktu pada data teks berbahasa Indonesia yang berkaitan dengan kebakaran hutan?
b. Berapa tingkat akurasi yang dapat dicapai dengan model Bi-LSTM yang telah dioptimalkan menggunakan genetic algorithm (algoritma genetika) dalam mendeteksi entitas lokasi dan waktu pada data teks bahasa Indonesia tentang kebakaran hutan?
c. Bagaimana perbandingan model Bi-LSTM secara default dan Bi-LSTM yang parameternya sudah dioptimasi dengan algoritma genetika?

1.3. Batasan Masalah

Batasan-batasan penelitian termuat dalam poin-poin sebagai berikut :

a. Dataset yang digunakan merupakan korpus *tweet* dari X.com atau yang sebelumnya disebut sebagai Twitter.com.
b. Dataset latih diakuisisi tidak secara realtime.
c. Skema *tagging* menggunakan *encoding* Begin Inside Outside (BIO) (Lin,dkk., 2020).
d. Dataset ini fokus pada penelitian untuk deteksi entitas waktu dan lokasi.
e. Dataset yang digunakan sepanjang 19846 ROW.
f. Dataset menggunakan kolom *sentence* untuk urutan sekuensial, *word* untuk kata dan *tag* untuk label.
g. Optimasi Bi-LSTM oleh GA (*searching space*) adalah dari faktor *hyperparameter* : *units, dropout, embedding dimension, learning rate, batch size, epoch, layer*, dan regularisasi kernel.
h. Jumlah atau angka dalam *hyperparameter* menggunakan asumsi umum dalam Bi-LSTM.
i. Epoch yang digunakan sebagai *hyperparameter* dibahas dalam hasil *preleminary experiment.* Epoch berdasarkan *preleminary experiment* tidak dapat menjadi acuan dalam penentuan jumlah epoch pada pencarian *hyperparameter*.
j. Pertimbangan pemilihan epoch berikut batasannya sebagai *hyperparameter* dalam ruang pencarian algoritma genetika tidak menggunakan jumlah yang banyak karena keterbatasan komputasi.

1.4. Tujuan Penelitian

Penelitian ini dilaksanakan dengan tujuan sebagai berikut :

a. Mengoptimasi *hyperparameter* Bi-LSTM dengan algoritma genetika.
b. Untuk mengetahui bagaimana cara algoritma genetika mengoptimasi Bi-LSTM.
c. Untuk mengetahui tingkat akurasi setelah optimasi Bi-LSTM menggunakan algoritma genetika.
d. Untuk mengetahui solusi optimal pada Bi-LSTM menggunakan algoritma genetika melalui *hyperparameternya*

1.5. Manfaat Penelitian

Penelitian ini memiliki ekspektasi dalam pemanfaatan hasil penelitian sebagai berikut :

a. Manfaat utama penelitian ini untuk menemukan parameter optimal saat mengatur arsitektur Bi-LSTM secara umum untuk korpus bahasa Indonesia dan terkhusus untuk mendeteksi entitas waktu dan lokasi kebakaran hutan.
b. Kontribusi pada riset dan penelitian adalah membuka peluang riset optimasi algoritma genetika pada model Bi-LSTM terutama untuk korpus bahasa Indonesia pada penelitian Named Entity Recognition dan terkhusus untuk parameter-parameter yang mempengaruhi akurasi pada NER.
c. Manfaat yang bisa didapatkan masyarakat adalah potensi implementasi untuk proyek dalam mendeteksi kebencanaan secara dini untuk lokasi dan waktu.

1.6. Catatan Kepenulisan

a. Selanjutnya untuk istilah "pengenalan entias bernama" sebagai terjemahan dari "*named entity recognition*" akan disebut sebagai "NER".

b. Selanjutnya untuk istilah "algoritma genetika" sebagai terjemahan dari "*genetic algorithm*" akan disebut sebagai "AG".

c. Selanjutnya untuk istilah "Long Short Term Memory" ataupun "Bi-stacked Long Short Term Memory" sebagai "LSTM" atau "Bi-LSTM".

d. Terdapat singkatan-singkatan nama metode atau istilah yang tidak bisa disebutkan dalam catatan kepenulisan ini, terutama dalam bahasan Tinjauan Pustaka, Landasan Teori, maupun Hasil dan pembahasan.

BAB II

TINJAUAN PUSTAKA

2.1. Tinjauan Pustaka

Tinjauan pustaka pada penelitian ini memperkenalkan serangkaian penelitian terkait yang telah dilakukan dalam bidang *named entity recognition (NER)*, penggunaan algoritma optimasi dalam masalah *hyperparameter*, penggunaan algoritma genetika dalam *hyperparameter*, dan penggunaan algoritma Bidirectional Long Short-Term Memory (Bi-LSTM) dalam memproses teks serta algoritma genetika dalam mengoptimasi permasalahan matematis dan algoritmis. Tinjaun pustaka untuk tesis ini tidak eksplisit menyebutkan pengembangan atau peningkatan dari penelitian tertentu karena penelitian dalam tesis ini juga mencakup *preleminary experiment*. Pengembangan penelitian dalam tesis ini mencakup pengembangan secara khusus *preleminary experiment* dan secara umum terkait pengembangan *hyperparameter optimization* dalam konteks *named entity recognition* (terjemahan : pengenalan etntitas bernama). Selanjutnya dibahas penelitian-penelitian yang menjadi landasan dan inspirasi pada tesis ini.

Terdapat penelitian yang menggarisbawahi pentingnya NER dalam pengambilan informasi dari dokumen teks dan bagaimana NER membantu dalam mengkategorikan entitas seperti orang, organisasi, dan lokasi dalam teks melalui algoritma Bi-LSTM (Pallavi dkk., 2022). Metode yang digunakan dalam penelitian Pallavi dkk., (2022) menggunakan LSTM untuk *chunking* dan

mengekstraksi kata dari serangkaian teks. Penelitian tersebut dapat memperkuat landasan penggunaan LSTM dalam ekstraksi entitas bernama dan *chunking*.

Metode atau model dalam NER tidak terbatas hanya pada LSTM saja. Terdapat meode-metode dalam NER yang dapat digunakan termasuk CNN, GRU, LSTM, Bi-LSTM, BRNN, MLP, CRF, Transformer, dan kombinasi di antaranya seperti GRAM-CNN dan MTM-CW (Song dkk., 2020 ; Li dkk., 2020). Ji dkk. (2022) melakukan studi literatur yang membandingkan performa berbagai algoritma pada berbagai dataset dengan ukuran yang berbeda. Hasil dari penelitian ini menunjukkan variasi performa antara algoritma-algoritma tersebut. Misalnya, penggunaan CNN pada dataset CoNLL03 mencapai akurasi sebesar 89.86%, sementara LSTM mencapai 90.10%, MLP 91.17%, Transformer 92.8%, GRU 91.93%, RNN 93.47%, dan ID-CNN 90.65%. Sementara itu, BRNN menggunakan dataset ONTONOTES5.0 mencapai akurasi sebesar 87.21%. Penelitian ini juga mengungkapkan bahwa algoritma LSTM cenderung stabil dalam performa F1-Score, dengan nilai tertinggi sebesar 93.5% ketika digunakan bersama Tag Encoder CRF untuk dataset CoNLL03. Tersebut merupakan alasan secara literasi dalam pemilihan algoritma LSTM pada penelitian ini dengan menekankan perbandingan LSTM dengan metode lainnya. Tabel 2.1 Perbandingan Algoritma untuk NER.

Tabel 2.1. Perbandingan Performa Algoritma NER

No	Algoritma	Dataset	Performa (F1-Score)
1	CNN	CoNLL03	89.86%
2	LSTM	CoNLL03	90.1%
3	MLP	CoNLL03	91.17%
4	Transformer	CoNLL03	92.8%
5	GRU	CoNLL03	91.93%

Tabel 2.1. Lanjutan

6	RNN	CoNLL03	93.47%
7	ID-CNN	CoNLL03	90.65%
8	BRNN	ONTONOTESS5.0	87.21%
9	LSTM;tag	CoNLL03	93.5%

LSTM (Long Short-Term Memory) memiliki beberapa kelebihan yang membuatnya unggul dalam tugas-tugas pemrosesan bahasa alami, terutama dalam mengatasi masalah vanishing gradient yang sering ditemui pada RNN standar, memungkinkan LSTM untuk mempertahankan informasi jangka panjang yang esensial dalam analisis data sekuensial. Kemampuan LSTM untuk mengingat informasi penting dari waktu yang lama dan melupakan informasi yang tidak relevan membuatnya sangat efektif dalam menangkap konteks dari data yang panjang, yang sangat penting dalam tugas-tugas seperti Named Entity Recognition (NER). Selain itu, arsitektur LSTM yang fleksibel memungkinkan integrasi dengan teknik lain, seperti embedding kata dan Conditional Random Fields (CRF), untuk meningkatkan akurasi model. LSTM juga cenderung lebih efisien secara komputasi dibandingkan model Transformer, menjadikannya pilihan yang baik dalam situasi dengan sumber daya terbatas. Fleksibilitas dan kemudahan penyesuaian LSTM dengan berbagai jenis data dan tugas NER semakin menegaskan relevansi dan stabilitasnya dalam berbagai aplikasi pemrosesan bahasa alami. Keterbatasan sumber daya komputasi menjadi pertimbangan penting ketika memilih model untuk pemrosesan bahasa alami, terutama di lingkungan dengan kapasitas komputasi yang terbatas. Meskipun model yang lebih canggih seperti Transformer, contohnya BERT, menawarkan kinerja yang

luar biasa dalam berbagai tugas NLP, mereka sering kali membutuhkan perangkat keras yang lebih kuat dan waktu pelatihan yang lebih lama. Dalam konteks ini, LSTM menawarkan solusi yang lebih praktis dan efisien karena dapat mencapai hasil yang baik dengan kebutuhan komputasi yang lebih rendah. Oleh karena itu, dalam banyak kasus di mana ketersediaan sumber daya komputasi menjadi kendala, LSTM tetap menjadi pilihan yang sangat relevan dan efektif.

Masih dengan alasan literasi, berikut singkat alasan secara praktis penggunaan Bi-LSTM untuk data *sequential* berupa teks dari serangkaian penelitian-penelitian yang berfokus pada efektifitas Bi-LSTM dalam kasus NER. Bi-LSTM telah terbukti efektif dalam menganalisis data teks yang bersifat *sequential*, seperti dalam tugas Named Entity Recognition (NER) (Jehangir dkk., 2023). Dengan kemampuannya untuk mengingat informasi jangka panjang dan mengatasi masalah *vanishing gradient*, Bi-LSTM mampu memodelkan ketergantungan jarak jauh antara kata dalam teks (Eligüzel dkk., 2022). Hal ini memungkinkan Bi-LSTM untuk secara akurat mengidentifikasi entitas yang tersembunyi di dalam teks berkelanjutan, seperti nama orang, tempat, dan organisasi. Dengan demikian, penerapan Bi-LSTM dalam NER dapat meningkatkan kinerja sistem pengenalan entitas dalam data teks.

Penelitian-penelitian yang disebutkan dalam tinjauan pustaka ini tidak hanya menjadi landasan dasar dalam pemilihan Bi-LSTM untuk kasus NER, berikut penelitian-penelitian terkini juga menerapkan upaya peningkatan Bi-LSTM dalam NER namun dengan cara mengkombinasikan NER. Contoh dari kombinasi-kombinasi metode Bi-LSTM seperti Bi-LSTM dengan CRF, Bi-LSTM

dengan CNN, pengembangan ASTRAL : LSTM modifikasi dengan CNN dan lain-lain. Penekanan upaya peningkatan LSTM yang disebut pada paragraf selanjutnya menekankan bahwa upaya-upaya peningkatan LSTM untuk NER tidak melalui pendekatan *hyperparameter*. Selanjutnya juga pembahasan terkait literatur yang digunakan tidak menggunakan korpus bahasa, sebagai pertimbangan penelitian bahwa bahasa dapat mempengaruhi kinerja NER (Li dkk., 2020) secara spesifik dalam bahasa indonesia yang memiliki karakter khas (Budi dan Suryono, 2023).

Meskipun penelitian yang dilakukan oleh (Xu dan Wang., 2022) mengenai Named Entity Recognition (NER) dengan model Bi-LSTM+CRF telah memberikan kontribusi yang signifikan dalam meningkatkan kinerja model dalam tugas NER, terdapat beberapa kekurangan yang dapat dijadikan dasar untuk pengembangan penelitian lebih lanjut. Salah satu kekurangan utama adalah fokus penelitian yang tidak secara khusus menangani bahasa Indonesia dan pengoptimalan *hyperparameter* model Bi-LSTM, sehingga belum memberikan solusi yang optimal untuk permasalahan mendeteksi entitas lokasi dan waktu pada data teks berbahasa Indonesia, terutama terkait kebakaran hutan. Oleh karena itu, penelitian yang akan diajukan ini akan memberikan kontribusi lebih lanjut dengan merinci implementasi AG dalam mengoptimalkan model Bi-LSTM untuk kasus spesifik bahasa Indonesia, mencakup pemrosesan data teks yang berkaitan dengan kebakaran hutan, dan mengatasi keterbatasan yang mungkin belum terpenuhi dalam penelitian sebelumnya.

Penelitian yang dilakukan oleh (Ronran dan Lee., 2020) telah memberikan wawasan yang berharga terkait tugas yang menantang dalam pemrosesan bahasa alami, khususnya dalam pengenalan entitas bernama (NER). Penelitian ini menyoroti peran teknik deep neural network (DNN), khususnya dengan menerapkan model Bidirectional LSTM-CRF, dalam mengekstrak entitas bernama dalam teks. Fokus utama penelitian adalah untuk menginvestigasi efek dari word embedding menggunakan Glove dan Fasttext dalam meningkatkan kinerja NER. Selain itu, penelitian ini juga mengeksplorasi dampak dari kombinasi fitur input tambahan berupa kata dan karakter pada model Convolutional Neural Network (CNN) dan BiLSTM dengan atau tanpa Conditional Random Field (CRF). Pada penelitian ini, tidak dilakukan pra-pemrosesan data atau penggunaan leksikon tambahan untuk perbaikan lebih lanjut, sehingga menitikberatkan pada efektivitas fitur kata dan karakter dalam NER. Hasil eksperimen menunjukkan bahwa penggunaan Glove 840B sebagai word embedding bersama dengan pola kata dan pola karakter untuk CNN, serta dua lapisan Bidirectional LSTM dengan CRF, menghasilkan hasil terbaik dengan nilai F1 mencapai 91.10% pada dataset CoNLL-2003. Pencapaian tersebut mengungguli hasil state-of-the-art (SOTA) dari Chiu, menyoroti potensi signifikan dari kombinasi fitur input yang diusulkan oleh penelitian ini. Meskipun demikian, perlu diperhatikan bahwa penelitian ini tidak melakukan pra-pemrosesan data dan tidak menggunakan leksikon tambahan, sehingga ada potensi untuk pengembangan lebih lanjut yang dapat dijelajahi untuk meningkatkan hasil

lebih lanjut dalam mendukung tugas NER misal dengan mengoptimasi *hyperparameternya* dan optimasi pra-pemrosesannya.

Penelitian oleh (Wang dkk., 2020) menyajikan pendekatan baru untuk meningkatkan NER melalui pengenalan sistem Adversarial Trained LSTM-CNN (ASTRAL). Pengenalan entitas bernama dalam data teks yang tidak terstruktur adalah tugas yang kompleks, dan para penulis menangani tantangan ini dengan mengusulkan model yang meningkatkan desain struktural dan proses pelatihan metode NER yang sudah ada. Gated-CNN diperkenalkan untuk menangkap informasi spasial antar kata yang berdekatan, dan metode "adversarial training" khusus digunakan untuk mengatasi masalah overfitting dalam NER. Ini melibatkan penambahan variasi pada variabel jaringan selama pelatihan, meningkatkan keragaman model, generalisasi, dan ketahanan. Evaluasi pada tiga benchmark, CoNLL-03, OntoNotes 5.0, dan WNUT-17, menunjukkan bahwa ASTRAL mencapai hasil terbaik. Studi kasus lebih lanjut menyoroti kemampuan sistem ini untuk konvergen lebih cepat dan kurang rentan terhadap overfitting.

Sama halnya dari penelitian (Cho dkk., 2020) membahas pentingnya *Named Entity Recognition* (NER) dalam bidang biomedis. Para penulis mengusulkan model NER deep learning yang efektif merepresentasikan token kata biomedis melalui desain combinatorial *feature embedding*. Model ini, berbasis Bidirectional Long Short-Term Memory (bi-LSTM) dengan Conditional Random Field (CRF), mengintegrasikan dua representasi tingkat karakter yang berbeda yang diekstrak dari Convolutional Neural Network (CNN) dan bi-LSTM. Mekanisme perhatian diterapkan untuk fokus pada token-token yang relevan

dalam kalimat, mengatasi masalah ketergantungan jangka panjang dari model LSTM. Evaluasi pada dataset benchmark, JNLPBA dan NCBI-Disease, menunjukkan kinerja model yang relatif lebih tinggi dengan F1-score sebesar 86,93% untuk NCBI-Disease dan kinerja yang kompetitif untuk JNLPBA dengan F1-score sebesar 75,31%. Studi ini berkontribusi pada kemajuan model NER biomedis, terutama dalam penanganan yang efektif terhadap jenis entitas baru dan tidak terlihat dalam data biomedis yang terus berkembang.

Selanjutnya pembahasan terkait dipilihnya algoritma optimasi AG sebagai upaya pengoptimalan *hyperparameter* LSTM. Selain membahas alasan praktis juga membahas hasil eksperimen dari studi-studi lainnya. Juga disebutkan upaya optimasi *hyperparameter LSTM* dengan AG tapi tidak menggunakan studi kasus NER, hal ini mempertegas bahwa penelitian dalam tesis ini masih belum dilakukan.

Secara umum terdapat banyak algoritma optimasi untuk masalah pencarian. Algoritma ini disebut sebagai *metaheuristic optimization*, algoritma yang paling banyak menjadi kajian penelitian yakni *Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Cuckoo Search (CS), and Harmony Search (HS) algorithms* (Halim dkk., 2021). Pemanfaatan secara khusus optimasi untuk pencarian adalah mencari *hyperparameter* dalam sebuah arsitektur pembelajaran mendalam (*deep learning)* dibahas dalam penelitian Bischl dkk. (2023), terdapat berbagai algoritma yang dapat digunakan untuk mengoptimasi *hyperparameter* algoritma *deep learning* seperti *Grid search and random search algorithm, Evolution strategies, Bayesian optimization, Multifidelity and hyperband, dan*

Gradient-based optimization (Bischl dkk., 2023). Upaya eksperimental untuk mencari nilai optimal dari *hyperparameter* LSTM sebagai arsitektur *deep learning* dilakukan melalui berbagai algoritma. Beberapa contoh dari penerapan algoritma optimasi untuk LSTM misalnya dengan Particle Swarm Optimization (PSO) (Wang dkk., 2020), Grey Wolf Optimizer (Aufa dkk., 2020), Bayesian Optimization (Thoppil, 2022), Tuna Swarm Optimization (Tuerxun dkk., 2022) dan Algoritma Genetika (Shahid, 2021).

Algoritma Genetika menjadi algoritma yang dapat diandalkan dalam pencarian hyperparameter dari sebuah deep learning, hal ini karena selain melakukan pencarian secara acak juga membatasi pencarian dengan kriteria yang sudah ditentukan (Lee dkk., 2021). Algoritma genetika merupakan algoritma yang populer (Wang dan Sobey, 2020) dan secara efektif dapat digunakan dalam berbagai bidang permasalahan optimasi (Gen dan Lin, 2023), algoritma genetika juga mampu diterapkan dalam bentuk yang sudah digabung dengan algoritma lainnya (Sohail, 2023), algoritma genetika sangat menjanjikan dalam mengoptimasi masalah pencarian (Katoch, 2021).

Pada penelitian yang menerapkan optimasi terhadap Bi-LSTM oleh Kara (2021) yang memperkenalkan metode untuk meramalkan wabah influenza dengan menggunakan jaringan Bi-LSTM dan algoritma genetika. Mereka menunjukkan bahwa kombinasi model ini berhasil mengungguli metode pembelajaran mesin tradisional dalam meramalkan wabah influenza. Penelitian ini mencakup penggunaan Bi-LSTM dan algoritma genetika, yang relevan dengan aspek optimasi dalam penelitian ini. Namun yang dibawa oleh topik ini optimasi GA

terhadap peramalan berdasarkan waktu serial dan bukan sekuensial teks seperti NER.

Selain kasus serial waktu juga ada penerapan algoritma genetika untuk mengoptimasi LSTM dalam konteks data koordinat 2D oleh (Palconit dkk., 2020), dalam penelitiannya melakukan prediksi pergerakan ikan di perairan yang keruh menggunakan algoritma genetika dan LSTM. Hasil penelitian ini menunjukkan bahwa model ini mampu memprediksi pergerakan ikan dengan tingkat akurasi yang dapat diterima. Ini relevan dengan penelitian ini karena melibatkan penggunaan LSTM dalam konteks pemodelan pergerakan. Hasil penelitian menunjukkan ketidakakuratan prediksi trayek ikan yang dapat diterima, dengan Mean Absolute Percentage Error (MAPE) berkisar antara 2.8% hingga 30.5% untuk algoritma genetika dan 3.33% hingga 12.74% untuk LSTM. Meskipun demikian, tingkat akurasi yang diperoleh masih dianggap dapat diterima, membuka peluang untuk lebih jauh mengembangkan studi ini dalam konteks pelacakan ikan.

Dalam penelitian (Zhang dkk., 2020) memfokuskan penelitian mereka pada prediksi produksi susu harian sapi perah menggunakan algoritma GA-LSTM. Hasil eksperimen menunjukkan bahwa model GA-LSTM lebih akurat dan stabil dalam memprediksi produksi susu harian. Ini relevan dengan penelitian ini karena melibatkan penggunaan LSTM dan algoritma genetika dalam konteks prediksi.

Dalam penelitian terkait prediksi beban listrik, peneliti menghadapi permasalahan prediksi beban listrik dengan menggabungkan keunggulan Long-Short Term Memory (LSTM) *neural network*, yang mampu merekam informasi

jangka panjang dan jangka pendek, dengan algoritma genetika (GA) untuk mengatasi kesulitan menentukan parameter LSTM. Pendekatan ini menggunakan tingkat pembelajaran dan jumlah iterasi sebagai kromosom, yang kemudian dioptimalkan melalui proses seleksi, persilangan, dan mutasi GA. Hasil simulasi menunjukkan peningkatan akurasi prediksi sebesar 63% dibandingkan dengan LSTM standar, menyoroti potensi penggunaan GA untuk meningkatkan kinerja model prediksi beban listrik. Penelitian ini memberikan wawasan komprehensif tentang bagaimana integrasi LSTM dan GA dapat menjadi solusi efektif untuk meningkatkan ketepatan prediksi beban listrik dalam konteks jaringan tenaga.

Terakhir, penelitian oleh (Ashok dkk., 2020) memperkenalkan model *deep neural network* untuk mendeteksi sindiran dalam data teks. Mereka mengoptimalkan model dengan menggunakan LSTM dan algoritma genetika. Penelitian ini relevan karena menggabungkan penggunaan LSTM dan algoritma genetika dalam konteks pemrosesan teks dan klasifikasi teks.

Keaslian dari penelitian ini ditampilkan pada sub bab 2.3 Keaslian Penelitian. Keaslian penelitian tidak berfokus pada landasan teoritis pemilihan algoritma, melainkan upaya ringkas terhadap penelitian paling mirip atau serupa dalam penelitian ini.

2.2. Ringkasan Tinjauan Pustaka

Dari tinjauan pustaka ini selain memperkuat landasan dalam pemilihan algoritma genetika dan alasan pemilihan LSTM secara literasi juga disebutkan secara logis dan praktis algoritma.

Algoritma LSTM dapat mengatasi *vanishing gradient* dan arsitekturnya mampu memprediksi atau mengklasifikasi data sekuensial, secara praktis dalam data teks untuk NER. Dari komparasi studi literasi LSTM dengan metode lainnya (baik LSTM yang berdiri sendiri (terpisah) maupun kombinasi) menunjukkan efektifitas LSTM dan keunggulan dibandingkan CNN, GRU, BRNN, MLP, CRF dan Transformer.

Algoritma genetika (AG) bekerja dengan cara mencari nilai terbaik dari setiap kemungkinan-kemungkinan yang terjadi sehingga juga disebut sebagai *meta heuristik*. AG mencegah kemungkinan acak yang tak terarah dengan membatasi ruang pencarian dengan nilai tertentu dan batas *score* yang disebut *fitness score*. AG menjadi dasar algoritma optimasi berbasis prinsip evolusi, sehingga pertimbangan utama pengguanaan algoritma optimasi dapat dilakukan dengan memilih AG terlebih dahulu. AG dapat diimplementasikan mencari nilai terbaik yakni F1 Score untuk *hyperparameter* Bi-LSTM dengan cara menerapkan angka acak terlebih dahulu pada *hyperparameter* kemudian diuji coba sebanyak n kali dalam lingkungan eksperimen evolusi, hingga ditemukan *hyperparameter* terbaik.

Berdasarkan tinjauan pustaka yang ada, terdapat upaya peningkatan performa Bi-LSTM baik dengan *hyperparameter*, kombinasi Bi-LSTM, maupun tahap *preprocessing*. Penelitian dalam tesis ini berfokus pada upaya peningkatan performa melalui pencarian *hyperparameter* terbaik dalam arsitektur BiBi-LSTM melalui algoritma genetika terkhusus NER dalam bahasa Indonesia. Walaupun

sudah ada penelitian optimasi *hyperparameter* untuk Bi-LSTM, tidak memberikan wawasan terkait data teks terkhusus untuk tugas NER.

Keaslian dari penelitian ini menekankan pada upaya optimasi *hyperparameter* Bi-LSTM untuk kasus NER pada teks berbahasa Indonesia dengan konteks kebakaran hutan. *Preleminary experiment* dilakukan sebagai pertimbangan dan bukti pengembangan dari penelitian sebelumnya. Keaslian penelitian dimuat secara spesifik di tabel 2.2.

2.3. Keaslian Penelitian

Tabel 2.2. Matriks literatur review dan posisi penelitian
Optimalisasi Bi-LSTM dengan Algoritma Genetika untuk Mendeteksi Entitas Lokasi dan Waktu pada Data Teks Bahasa Indonesia Tentang Kebakaran Hutan

No	Judul	Peneliti, Media Publikasi, dan Tahun	Tujuan Penelitian	Kesimpulan	Saran atau Kelemahan	Perbandingan
1	*LSTM Based Named Entity Chunking and Entity Extraction*	B. G. Pallavi, E. R. Kumar, R. Karnati, R. A. Kumar, 2022 First International Conference on Artificial Intelligence Trends and Pattern Recognition (ICAITPR), 2022	Penelitian ini bertujuan untuk mengembangkan metode ekstraksi entitas berbasis LSTM untuk Named Entity Recognition (NER).	Penelitian ini berhasil menunjukkan bahwa model LSTM dapat efektif digunakan dalam tugas NER dan pengembangan berikutnya dapat mempertimbangkan peningkatan performa model.	Saran penelitian selanjutnya adalah mempertimbangkan optimalisasi model LSTM untuk tugas NER. Kelemahan penelitian ini adalah kurangnya analisis perbandingan dengan metode lain dalam konteks yang sama.	Penelitian yang dilakukan Pallavi dkk., mengekstraksi entitas secara umum terhadap teks dokumen menggunakan metode LSTM dan tanpa mengoptimasi model LSTM. Adapun penelitian yang diajukan dalam tesis ini fokus pada teks twitter (*tweet*) yang memiliki jumlah karakter terbatas dalam 150 karakter dan terdapat optimasi metode LSTM dengan menggunakan algoritma genetika.
2	*The Study of NER Methods Based on Bi-LSTM+CRF Model*	A. Xu, C. Wang, CIBDA 2022; 3rd International Conference on	Penelitian ini bertujuan untuk meningkatkan metode NER dengan menggunakan model	Hasil penelitian menunjukkan bahwa model Bi-LSTM+CRF dengan fitur karakter dapat secara signifikan	Saran penelitian selanjutnya adalah menerapkan metode active learning untuk tugas NER dengan data	Penelitian oleh A.Xu dan Wang membandingkan dua model NER, yaitu Bi-LSTM+CRF dengan model yang tidak memakai CRF.

Tabel 2.2. Lanjutan

		Computer Information and Big Data Applications, 2022	Bi-LSTM+CRF dan fitur karakter.	meningkatkan performa dalam tugas NER.	terbatas. Kelemahan penelitian ini adalah tidak mempertimbangkan perbandingan dengan metode NER konvensional dengan optimasi algoritma.	Penelitian tersebut membuka peluang untuk mengkombinasi ataupun mengoptimasi model NER baik dengan variasi model lainnya ataupun menggunakan optimasi algoritma tertentu. Pada penelitian tesis yang diajukan berupaya untuk mengoptimasi fokus pada metode LSTM yang dioptimasi *hyperparameternya* dengan algoritma genetika.
3	*Effect of Character and Word Features in Bidirectional LSTM-CRF for NER*	C. Ronran, S. Lee, 2020 IEEE International Conference on Big Data and Smart Computing (BigComp), 2020	Penelitian ini berfokus pada penggunaan fitur karakter dan kata dalam model Bidirectional LSTM-CRF untuk NER.	Hasil eksperimen menunjukkan bahwa penggunaan Glove 840B sebagai word embedding bersama dengan pola kata dan pola karakter untuk CNN, serta dua lapisan Bidirectional LSTM dengan CRF, menghasilkan hasil terbaik dengan nilai F1	Kelemahan penelitian ini adalah kurangnya analisis eksperimental yang mendalam pada pengaruh fitur karakter dan kata terhadap performa model. Penelitian ini tidak melakukan pra-pemrosesan data atau penggunaan leksikon tambahan,	Penelitian oleh Ronran dan Lee memberikan peluang terhadap *preprocessing* data yakni fokus pada komponen data teks yang akan diwakili dengan vektor angka menggunakan metode Glove dan Fasttext. Namun dalam LSTM yang digunakan tidak ada optimasi yang dilakukan untuk memperoleh upaya akurasi menggunakan

Tabel 2.2. Lanjutan

				mencapai 91.10% pada dataset CoNLL-2003.	menitikberatkan pada efektivitas fitur kata dan karakter dalam NER.	*hyperparameter*.
4	*ASTRAL: Adversarial Trained LSTM-CNN for Named Entity Recognition*	Wang, J., Xu, W., Fu, X., Xu, G., & Wu, Y, Knowledge-Based Systems, 197, 105842, 2020	Meningkatkan NER melalui pengenalan sistem ASTRAL.	ASTRAL mencapai hasil terbaik dalam evaluasi pada tiga benchmark. Kemampuan konvergensi cepat dan kurang rentan terhadap overfitting.	Implementasi Gated-CNN dan metode Adversarial training dapat meningkatkan desain struktural dan proses pelatihan NER.	Pada penelitian Wang dkk., kombinasi Adversarial Trained LSTM-CNN dapat meningkatkan performa dalam kasus NER sehingga baik dalam menghindari overfitting. Penelitian ini menggabungkan model LSTM dengan model lainnya tanpa mempertimbangkan *hyperparameter* yang terdapat pada LSTM.
5	*Combinatorial feature embedding based on CNN and LSTM for biomedical named entity recognition*	Cho, M., Ha, J., Park, C., & Park, S, Journal of biomedical informatics. 103, 10338, 2020.	Mengusulkan model NER deep learning berbasis bi-LSTM dengan CRF untuk data biomedis.	Model mencapai kinerja yang relatif lebih tinggi pada dataset benchmark, menunjukkan efektivitas dalam penanganan entitas baru dalam data biomedis.	Penggunaan dua representasi tingkat karakter yang berbeda dan mekanisme perhatian untuk mengatasi ketergantungan jangka panjang dari model LSTM.	Penelitian oleh Cho dkk. berfokus pada *preprocessing dataset* untuk *feature embedding*. CNN dan LSTM terlibat untuk menentukan entitas bernama pada dataset biomedis.
6	*Multi-step influenza outbreak forecasting using deep LSTM*	Ahmet Kara, Expert Systems with	Penelitian ini bertujuan untuk meramalkan wabah	Penelitian ini berhasil mengembangkan model hibrid yang	Saran penelitian selanjutnya adalah mempertimbangkan	Penelitian yang dilakukan oleh Ahmet Kara, yang dipublikasikan dalam Expert

Tabel 2.2. Lanjutan

	network and genetic algorithm.	Applications, 2021	influenza dengan memanfaatkan jaringan LSTM dan algoritma genetika.	mengungguli metode machine learning lain dalam meramalkan wabah influenza.	penggunaan model ini dalam konteks kesehatan masyarakat. Kelemahan penelitian ini adalah kurangnya perbandingan dengan model statistik.	Systems with Applications pada tahun 2021, bertujuan untuk meramalkan wabah influenza dengan memanfaatkan jaringan LSTM dan algoritma genetika. Penelitian tersebut memberikan peluang untuk mengkombinasikan LSTM dan algoritma genetika pada kasus pengenalan entitas bernama yang memiliki dataset berupa sekuansial *non-timeseries*.
7	*Towards Tracking: Investigation of Genetic Algorithm and LSTM as Fish Trajectory Predictors in Turbid Water*	M. G. B. Palconit dkk., 2020 IEEE REGION 10 CONFERENCE (TENCON), 2020	Penelitian ini bertujuan untuk melacak perilaku ikan dengan meramalkan lintasan koordinat 2D ikan.	Hasil penelitian menunjukkan bahwa algoritma genetika dan LSTM dapat digunakan untuk meramalkan lintasan ikan dengan tingkat ketidakakuratan yang dapat diterima.	Saran penelitian selanjutnya adalah mempertimbangkan peningkatan dalam ramalan lintasan ikan dalam kondisi lingkungan bawah air yang lebih kompleks. Kelemahan penelitian ini adalah tingkat ketidakakuratan dalam	Penelitian yang dilakukan oleh M. G. B. Palconit dkk., dan dipresentasikan dalam IEEE REGION 10 CONFERENCE (TENCON) tahun 2020, bertujuan untuk melacak perilaku ikan dengan meramalkan lintasan koordinat 2D ikan. Dataset yang digunakan merupakan sekuensial

Tabel 2.2. Lanjutan

					ramalan.	koordinat 2D. Terdapat peluang baru penggunaan LSTM dan algoritma genetika dalam konteks dataset sekuensial *vector text*.
8	*Daily milk yield prediction of dairy cows based on the GA-LSTM algorithm*	W. Zhang, K. Yang, N. Yu, T. Cheng, J. Liu, 2020 15th IEEE International Conference on Signal Processing (ICSP), 2020	Penelitian ini bertujuan untuk meramalkan hasil produksi susu harian sapi berdasarkan algoritma GA-LSTM.	Hasil penelitian menunjukkan bahwa GA-LSTM dapat meningkatkan akurasi dalam meramalkan hasil produksi susu dibandingkan dengan LSTM konvensional.	Penggunaan algoritma genetika dalam konteks prediksi terhadap LSTM konvensional dan modifikasi.	Penelitian yang dilakukan oleh W. Zhang dkk., dan dipresentasikan dalam 15th IEEE International Conference on Signal Processing (ICSP) tahun 2020, bertujuan untuk meramalkan hasil produksi susu harian sapi berdasarkan algoritma GA-LSTM. Penelitian ini dapat memberikan wawasan tentang aplikasi algoritma genetika dalam mengoptimalkan model LSTM untuk prediksi. Hasil akurasi yang baik bisa diuji lebih lanjut untuk konteks NER.
9	*Short-term Load Forecasting of Long-short Term Memory Neural Network Based*	W. Li, C. Zang, D. Liu, P. Zeng, 2020 IEEE 4th Conference on	Penelitian ini bertujuan untuk meramalkan beban listrik jangka pendek	Hasil penelitian menunjukkan bahwa penggunaan GA dalam mengoptimasi parameter	Saran penelitian selanjutnya adalah mempertimbangkan aplikasi model ini dalam	Penelitian yang dilakukan oleh W. Li dkk., dan dipresentasikan dalam 2020 IEEE 4th Conference on

Tabel 2.2. Lanjutan

	on Genetic Algorithm	Energy Internet and Energy System Integration (EI2), 2020	menggunakan jaringan LSTM dengan optimasi algoritma genetika.	LSTM dapat meningkatkan akurasi dalam meramalkan beban listrik.	perencanaan dan manajemen jaringan listrik. Kelemahan penelitian ini adalah kurangnya perbandingan dengan metode tradisional.	Energy Internet and Energy System Integration (EI2), bertujuan untuk meramalkan beban listrik jangka pendek menggunakan jaringan LSTM dengan optimasi algoritma genetika. Penelitian ini berfokus pada peramalan namun memberikan peluang LSTM dioptimasi dengan algoritma genetika sehingga lebih baik termasuk dalam konteks *dataset* berupa kata-kata atau *vector text sequential*.
10	*Sarcasm Detection using Genetic Optimization on LSTM with CNN*	D. M. Ashok, A. Nidhi Ghanshyam, S. S. Salim, D. Burhanuddin Mazahir, B. S. Thakare, 2020 International Conference for Emerging Technology (INCET), 2020	Penelitian ini bertujuan untuk mendeteksi sindiran dalam data besar.	Hasil penelitian menunjukkan bahwa model deep learning dengan LSTM yang di-optimasi menggunakan algoritma genetika dapat digunakan untuk mendeteksi sindiran.	Saran penelitian selanjutnya adalah mempertimbangkan aplikasi lebih lanjut dalam analisis sentimen dan pengolahan bahasa alami. Kelemahan penelitian ini adalah kurangnya analisis perbandingan dengan metode lain.	Penelitian yang dilakukan oleh D. M. Ashok dkk., dan dipresentasikan dalam International Conference for Emerging Technology (INCET) tahun 2020, bertujuan untuk mendeteksi sindiran dalam data besar. Penelitian ini memberikan wawasan LSTM yang dioptimasi dengan algoritma genetika untuk kasus mendeteksi bahasa sindiran.

2.4. Landasan Teori

2.4.1. Kebakaran Hutan dan Permasalahannya

Kebakaran Hutan adalah fenomena yang telah menjadi masalah serius di seluruh dunia. Kebakaran hutan terjadi ketika api membakar di daerah hutan atau lahan gambut, dan hal ini dapat memiliki dampak ekologis yang serius, termasuk kerusakan habitat alam dan emisi gas rumah kaca. Kebakaran hutan juga memiliki dampak sosial dan ekonomi yang signifikan. Kebakaran hutan telah menjadi permasalahan yang sering terjadi di berbagai wilayah Indonesia, terutama pada musim kemarau. Selain itu, kebakaran hutan juga dapat menyebabkan masalah serius terkait dengan kualitas udara, hilangnya habitat satwa liar, dan kerugian ekonomi yang signifikan (Pasai, 2020).

2.4.2. Teks *Tweet* dalam X.com atau Twitter.com

Teks *tweet* adalah pesan singkat yang seringkali digunakan dalam platform media sosial seperti Twitter (Burgess dan Baym, 2022), walaupun media sosial Twitter.com berubah nama brand menjadi X.com ini tidak menghilangkan arti dari *tweet post* (Stokel-Walker, 2023). Platform ini memungkinkan pengguna untuk berbagi informasi secara cepat dan luas, termasuk informasi tentang peristiwa seperti kebakaran hutan. Teks *tweet* umumnya terbatas dalam panjangnya, dan analisis teks *tweet* memerlukan pemahaman tentang bahasa yang digunakan dan konteks sosial di balik pesan-pesan tersebut.

2.4.3. Penggunaan Bahasa Indonesia dalam Twitter

Bahasa Indonesia, sebagai bahasa nasional dan penyatuan bangsa Indonesia (Saputra dan Fitri, 2022), menjadi komunikasi utama dalam

menyampaikan teks *tweet* di Indonesia. Meskipun variasi dalam bentuk bahasa resmi, slank, dan alay banyak ditemui (Rosalina dkk., 2020), pemahaman mendalam terhadap Bahasa Indonesia, termasuk struktur kalimat, tata bahasa, dan penggunaan kata-kata, menjadi krusial dalam menganalisis teks *tweet* yang berkaitan dengan isu kebakaran hutan. Perlu ditekankan bahwa ragam kata Bahasa Indonesia sering mengalami perubahan makna melalui ameliorasi, serapan bahasa asing, plesetan, peyorasi, dan berbagai fenomena lainnya, khususnya dalam konteks platform media sosial seperti Twitter (Prayudi dan Nasution, 2020). Dengan demikian, pemahaman mendalam terhadap dinamika bahasa Indonesia di Twitter menjadi aspek kritis dalam menafsirkan pesan-pesan terkait kebakaran hutan.

2.4.4. Kecerdasan Buatan dan Pembelajaran Mesin

Kecerdasan buatan (*Artificial Intelligence/AI*) adalah disiplin ilmu komputer yang bertujuan mengembangkan sistem cerdas mirip manusia (Zhang dan Lu, 2021). Pembelajaran Mesin (*Machine Learning/ML*), sebagai cabang utama AI, memungkinkan sistem untuk belajar dari data tanpa pemrograman eksplisit. Ada tiga jenis utama pembelajaran mesin: *supervised learning, unsupervised learning,* dan *reinforcement learning* (Michalski dkk., 2013).

Keterkaitan antara *AI, ML, dan Natural Language Processing (NLP)* muncul dalam pengembangan sistem yang dapat memproses dan memahami bahasa manusia. NLP fokus pada interaksi komputer dengan bahasa alami, dan pembelajaran mesin meningkatkan kemampuan sistem dalam memproses teks (Khan dkk., 2016).

Named Entity Recognition (NER) adalah tugas dalam NLP yang melibatkan identifikasi entitas bernama dalam teks. Pembelajaran mesin, terutama model NLP, menjadi kunci dalam pengembangan sistem NER yang efektif. Dengan mengintegrasikan kecerdasan buatan, pembelajaran mesin, NLP, dan NER, kita dapat menciptakan sistem yang mampu mengekstraksi informasi entitas bernama dari teks dengan tingkat akurasi yang tinggi. Sistem ini memiliki aplikasi luas, seperti analisis sentimen, pengolahan otomatis dokumen, dan penyaringan informasi dalam jumlah besar data teks (Sun dkk., 2020).

2.4.5. *Natural Language Processing* (NLP)

Natural Language Processing (NLP) adalah cabang dari kecerdasan buatan yang memungkinkan komputer untuk memahami dan memproses bahasa manusia (Khan dkk., 2016). NLP memiliki peran krusial dalam penelitian ini karena digunakan untuk menganalisis teks tweet dalam bahasa Indonesia terkait kebakaran hutan. Dalam konteks ini, NLP mencakup berbagai tugas, seperti pemrosesan teks, pemahaman konteks, dan ekstraksi informasi (Jurafsky dan James, 2000).

Salah satu tren utama dalam NLP adalah Named Entity Recognition (NER). NER adalah subtugasb penting yang memainkan peran kunci dalam menemukan dan mengklasifikasikan entitas seperti nama organisasi, nama orang, atau lokasi dalam teks (Jurafsky & James, 2000). Dalam konteks ekstraksi informasi, NER menjadi kegiatan vital di berbagai domain.

Terkait tren NER, pertumbuhan besar dan ketersediaan data menimbulkan tantangan dalam mengekstrak informasi yang berguna dari dokumen berbahasa

alami. NER dapat secara otomatis mengklasifikasikan artikel dan mengungkap individu, organisasi, dan tempat yang disebutkan dalam teks (Sharma dkk., 2022) . Seiring perkembangan dalam penelitian NER, teknik-teknik yang digunakan dalam dekade terakhir menjadi fokus utama. Sebuah kajian menyeluruh tentang NER, seperti yang diusulkan oleh (Jurafsky dan James, 2000), menggambarkan perkembangan terbaru dalam teknik-teknik NER, menyajikan alat, dataset, teknik, tantangan, dan arah masa depan dalam bidang ini (Sharma dkk., 2022).

Pemrosesan bahasa alami (NLP) merupakan landasan teori yang esensial dalam konteks ini, memungkinkan pemahaman dan analisis teks dalam bahasa manusia. Seiring dengan itu, NER mengambil peran lebih spesifik, fokus pada pengenalan entitas tertentu dalam teks, menjadi bagian integral dari ekstraksi informasi yang lebih luas. Kombinasi NLP dan NER membuka pintu bagi pemahaman yang lebih mendalam tentang teks bahasa manusia dan ekstraksi informasi yang berharga.

Dalam NLP, baik LSTM ataupun Bi-LSTM digunakan dalam berbagai tugas seperti terjemahan mesin, analisis sentimen, pengenalan entitas, dan masih banyak lagi. Ketika menangani teks, LSTM membantu dalam memodelkan dan memahami dependensi temporal antara kata-kata dalam sebuah kalimat, yang penting untuk memahami makna kontekstual. Dengan struktur internalnya yang memungkinkan pengingatan informasi dalam jangka panjang, LSTM dapat menangani masalah dalam bahasa yang melibatkan dependensi jarak jauh (Zhang, 2018). Selanjutnya akan dibahas terkait *Named Entity Recognition (NER)* sebagai sub *Natural Language Processing (NLP)*.

2.4.6. *Named Entity Recognition* (NER)

Pengenalan Entitas Bernama (*Named Entity Recognition*, NER) mewakili sebuah teknik mutakhir dalam ranah pemrosesan bahasa alami yang memfokuskan upayanya pada identifikasi dan kategorisasi entitas tertentu dalam suatu teks (Sun dkk., 2020). Entitas yang diacu dapat melibatkan berbagai jenis, seperti orang, lokasi, organisasi, waktu, prosedur klinis, hingga protein biologis (Yadav, V., & Bethard, S. , 2019). Dalam konteks penelitian ini, NER digunakan sebagai alat untuk mengenali serta mengekstraksi entitas waktu dan lokasi dalam teks tweet yang terkait dengan fenomena kebakaran hutan.

Teknologi NER memiliki implikasi luas, seringkali berperan sebagai tahap awal dalam berbagai aplikasi yang melibatkan pemahaman dan analisis teks. Aplikasi-aplikasi tersebut termasuk, namun tidak terbatas pada, sistem pertanyaan jawaban, pengambilan informasi, resolusi ko-referensi, dan pemodelan topik (Yadav, V., & Bethard, S. , 2019). Oleh karena itu, pemantauan terhadap kemajuan terkini dalam bidang NER menjadi sangat relevan, terutama dalam konteks penggunaan arsitektur NER berbasis jaringan saraf yang telah mencapai kinerja terbaik dengan minimnya rekayasa fitur.

Sejak NER pertama kali dikembangkan sebagai pemrosesan teks (Grishman dan Sundheim, 1996), telah banyak tugas bersama dan kumpulan data yang dibuat khusus untuk NER. CoNLL 2002 (Tjong Kim Sang, 2002) dan CoNLL 2003 (Tjong Kim Sang dan De Meulder, 2003) dibuat dari artikel berita baru dalam empat bahasa berbeda (Spanyol, Belanda, Inggris, dan Jerman) dan

difokuskan pada empat entitas utama: PER (orang), LOC (lokasi), ORG (organisasi), dan MISC (miscellaneous, mencakup semua jenis entitas lainnya).

Tugas bersama NER juga telah diorganisir untuk berbagai bahasa lain, termasuk bahasa India (Rajeev Sangal dan Singh, 2008), Arab (Shaalan, 2014), Jerman (Benikova dkk., 2014), dan bahasa Slavia (Piskorski dkk., 2017). Perlu dicatat bahwa ekspansi cakupan tugas bersama NER ke berbagai bahasa ini menunjukkan pentingnya menyesuaikan teknik NER dengan struktur dan konteks bahasa yang beragam, mencerminkan sifat dinamis penelitian dan aplikasi di bidang ini.

Selanjutnya perkembangan NER mencakup pergeseran dari pendekatan awal yang didasarkan pada aturan buatan, leksikon, fitur ortografis, dan ontologi, menuju sistem-sistem NER yang lebih canggih berbasis rekayasa fitur dan pembelajaran mesin (Nadeau dan Sekine, 2007). Dalam beberapa tahun terakhir, terutama sejak karya Collobert dkk. pada tahun 2011, pendekatan berbasis jaringan saraf tanpa rekayasa fitur yang signifikan telah menjadi pilihan populer. Keberhasilan model-model ini terletak pada kemampuannya untuk mencapai kinerja tinggi tanpa bergantung pada sumber daya domain khusus seperti leksikon atau ontologi, menjadikannya lebih adaptif dalam berbagai domain. Seiring dengan itu, beragam arsitektur saraf telah diusulkan, kebanyakan diantaranya didasarkan pada penggunaan jaringan saraf rekuren (RNN : *Recurren neural network*) yang memproses informasi dari karakter, sub-kata, dan/atau embedding kata (Sun dkk., 2020). Dalam penerapan RNN terdapat *hyperparameter* yang bisa dioptimasi menggunakan algoritma optimisasi.

2.4.7. Optimasi

Optimasi adalah disiplin lintas-bidang yang bertujuan untuk menemukan solusi terbaik dari berbagai alternatif yang mungkin dengan menggunakan sumber daya yang terbatas. Dengan memanfaatkan konsep matematika seperti teori graf, analisis kalkulus, dan program linier, bersama dengan algoritma seperti algoritma genetika dan pengoptimalan heuristik, optimasi memungkinkan untuk merumuskan masalah dalam kerangka kerja formal dan mengeksplorasi ruang solusi dengan cara yang efisien. Dari aplikasi dalam sistem manufaktur hingga pengelolaan rantai pasokan, dari perencanaan jadwal hingga desain jaringan, optimasi menjadi instrumen penting dalam mengambil keputusan yang cerdas dan efektif dalam berbagai konteks, mengoptimalkan kinerja sistem, mengurangi biaya, dan meningkatkan efisiensi secara keseluruhan (Fanani dan Swara, 2023).

2.4.8. Algoritma Optimasi

Algoritma optimisasi merujuk pada sekumpulan metode matematis dan komputasional yang bertujuan untuk mencari solusi optimal atau mendekati solusi terbaik dari suatu masalah (Kohenderfer dan Wheeler,2019). Tujuan utamanya adalah mengoptimalkan fungsi tujuan, yang bisa berupa maksimisasi atau minimisasi, dengan menggunakan berbagai teknik dan pendekatan. Proses algoritma optimisasi melibatkan pencarian ruang solusi, di mana eksplorasi dan eksploitasi digunakan untuk menemukan berbagai solusi potensial atau memperbaiki solusi yang sudah ditemukan. Fungsi tujuan, sebagai kriteria yang akan dioptimalkan, menjadi fokus dalam mencari nilai terendah atau tertinggi tergantung pada jenis masalah. Selain itu, algoritma optimisasi harus memiliki

kriteria konvergensi untuk menentukan kapan proses pencarian dapat dihentikan, yang melibatkan pemantauan perubahan nilai fungsi tujuan atau perubahan dalam ruang solusi. Jenis algoritma optimisasi bervariasi, termasuk algoritma genetika, algoritma evolusioner, optimisasi gradien, dan metaheuristik (Kohenderfer dan Wheeler, 2019). Aplikasi algoritma optimisasi sangat luas, mencakup pengoptimalan parameter model machine learning dan deep learning (Soydaner, 2020) , penjadwalan tugas, perencanaan rute, dan banyak lagi (Diwekar, U.M., 2020). Dengan kontribusinya yang signifikan, algoritma optimisasi membentuk dasar untuk mencari solusi terbaik dalam berbagai konteks masalah yang kompleks .

2.4.9. Algoritma Genetika (AG)

*Genetic algorithm (*atau juga disebut algoritma genetik (AG) dalam penelitian dan tulisan ini) merupakan teknik atau algoritma optimisasi yang mengambil inspirasi dari mekanisme seleksi alamiah dan proses genetika dalam evolusi biologis. Dikembangkan oleh John Holland pada tahun 1975, AG bertujuan untuk menemukan solusi optimal atau mendekati solusi terbaik dari suatu masalah dengan mensimulasikan konsep-konsep evolusi biologis (Lambora dkk., 2019).

Dalam AG, solusi potensial dari masalah, yang disebut individu, direpresentasikan sebagai kromosom. Populasi awal terdiri dari kumpulan individu, dan proses evolusi dimulai dengan reproduksi dan seleksi. Individu dipilih untuk reproduksi berdasarkan pada *fitness*, yang mencerminkan seberapa baik suatu individu dapat menyelesaikan masalah tertentu. Proses reproduksi

melibatkan pencampuran genetik antara individu untuk menghasilkan keturunan (Hluck, G., 2019).

Operator utama dalam AG melibatkan *crossover*, yaitu pemindahan genetik antara dua kromosom, dan mutasi, yaitu perubahan acak dalam gen. *Crossover* memungkinkan kombinasi sifat-sifat baik dari dua individu, sementara mutasi memberikan variasi acak/*random* untuk mencegah konvergensi terlalu dini. Fungsi *fitness* mengevaluasi kinerja individu dalam menyelesaikan masalah, dan nilai fitness digunakan sebagai dasar untuk seleksi (Hluck, 2019).

Algoritma genetik berjalan melalui berbagai generasi, dan konvergensi terjadi ketika populasi mencapai solusi yang optimal atau mendekati solusi terbaik. Kriteria konvergensi umumnya melibatkan pemantauan perubahan nilai fungsi tujuan atau stagnasi dalam populasi. AG telah sukses diterapkan dalam berbagai bidang, termasuk optimisasi kombinatorial, perencanaan jadwal, desain mesin, dan pengoptimalkan parameter model dalam machine learning. Dengan kemampuannya menangani masalah kompleks dan ruang pencarian yang besar, algoritma genetik menjadi salah satu pendekatan yang kuat dan populer dalam dunia optimisasi dan komputasi evolusioner (Hluck., 2019).

Prediksi kata berikutnya merupakan permasalahan penting dalam ranah Pemrosesan Bahasa Alami (NLP), khususnya dalam kecerdasan buatan modern. Hal ini menarik perhatian baik dari segi ilmiah maupun industri, karena menjadi inti dari berbagai proses, seperti otokoreksi, pembangkitan teks, dan prediksi ulasan, dan sebagainya. Saat ini, pendekatan yang paling efisien dan umum digunakan adalah klasifikasi, menggunakan Jaringan Saraf Tiruan (ANNs). Salah

satu kelemahan utama dari ANNs adalah penyetelan fine-tuning pada hyperparameter mereka, suatu prosedur yang sangat penting untuk kinerja model. Di sisi lain, pendekatan yang biasanya digunakan untuk fine-tuning entah terlalu mahal secara komputasional (misalnya, *grid search*) atau kurang efisien (misalnya, *trial & error*). Sebagai respons terhadap permasalahan tersebut, makalah ini menyajikan suatu pendekatan algoritma genetika sederhana, yang digunakan untuk penyetelan hyperparameter pada model bahasa umum, dan mencapai efisiensi penyetelan tanpa melakukan pencarian yang sangat mendalam (Gorgolis, 2019).

Ilustrasi pada Gambar 2.1. berikut merupakan alur dari algoritma genetika berdasarkan Putra (2018).

Gambar 2.1. Alur Algoritma Genetika

Berdasarkan diagram Gambar 2.1 yang Anda berikan, ini adalah alur algoritma genetika yang dijelaskan langkah demi langkah:

1. Mulai: Proses dimulai dengan titik awal di mana algoritma genetika dilaksanakan.

2. Tentukan Jumlah Populasi dan Generasi: Langkah pertama adalah menentukan jumlah populasi awal dan jumlah generasi yang akan dievaluasi. Ini mungkin melibatkan keputusan tentang berapa banyak individu yang akan ada dalam populasi awal dan berapa banyak generasi yang akan diperlakukan dalam proses evolusi.

3. Tentukan Ruang Pencarian: Setelah menentukan parameter populasi dan generasi, langkah selanjutnya adalah menentukan ruang pencarian atau domain solusi dari masalah yang dihadapi. Ruang pencarian ini mungkin berupa rentang nilai-nilai yang valid untuk setiap variabel dalam solusi.

4. Inisiasi Populasi pada Generasi Awal: Populasi awal dibentuk dengan membuat sejumlah individu yang secara acak dihasilkan. Setiap individu dalam populasi ini mewakili satu solusi potensial untuk masalah yang dihadapi.

5. Evaluasi Populasi: Setelah populasi awal terbentuk, langkah selanjutnya adalah mengevaluasi setiap individu dalam populasi berdasarkan pada seberapa baik mereka memecahkan masalah yang diberikan. Evaluasi ini mungkin menggunakan fungsi objektif yang telah ditentukan.

6. Lakukan Cross Over dan Mutasi dan Bentuk Populasi: Setelah evaluasi populasi awal selesai, langkah berikutnya adalah melakukan operasi crossover dan mutasi untuk menghasilkan generasi berikutnya dari individu. Crossover melibatkan pertukaran informasi genetik antara individu, sedangkan mutasi melibatkan perubahan acak dalam gen-gen individu.

7. Evaluasi Populasi dan Ambil Individu Terbaik: Setelah operasi crossover dan mutasi selesai, populasi generasi baru dievaluasi kembali untuk menentukan

seberapa baik mereka memecahkan masalah. Kemudian, individu terbaik dalam populasi ini diambil untuk dipertahankan ke dalam generasi berikutnya.

8. Selesai: Proses ini berlanjut secara iteratif dengan langkah-langkah 5-7 sampai kriteria berhenti terpenuhi. Kriteria berhenti ini mungkin berupa jumlah generasi yang telah dievaluasi atau penemuan solusi yang memuaskan. Setelah kriteria berhenti tercapai, algoritma dianggap selesai, dan solusi terbaik yang ditemukan dianggap sebagai solusi akhir dari masalah yang diberikan.

Dengan demikian, diagram Gambar 2.1 memberikan visualisasi dari alur algoritma genetika dari mulai hingga selesai dengan menyoroti setiap langkah penting dalam prosesnya. Dalam konteks optimasi *hyperparameter*, hyperparameter-hyperparameter seperti jumlah unit LSTM, panjang sekuens input, kecepatan pembelajaran, dan dropout rate dapat dianggap sebagai variabel-gen dalam algoritma genetika. Proses inisiasi populasi awal akan menghasilkan berbagai kombinasi hyperparameter secara acak. Kemudian, evaluasi dilakukan dengan melatih model LSTM menggunakan setiap kombinasi hyperparameter dan mengukur performanya pada data validasi. Individu-individu terbaik dalam populasi diidentifikasi berdasarkan performa model mereka. Operasi crossover dan mutasi digunakan untuk menghasilkan generasi berikutnya dari kombinasi hyperparameter, dengan harapan menghasilkan kombinasi yang lebih baik. Proses ini berlanjut secara iteratif hingga kriteria berhenti terpenuhi, seperti jumlah generasi yang ditentukan atau penemuan hyperparameter yang memuaskan. Dengan demikian, algoritma genetika memberikan pendekatan sistematis untuk

mengeksplorasi ruang hyperparameter dan mengoptimalkan kinerja model Bi-LSTM.

2.4.10. Jaringan Saraf Tiruan (*Artificial Neural Network/NN*)

Neuron atau node yang saling terhubung dalam lapisan input, lapisan tersembunyi, dan lapisan output, NN memproses informasi dengan mengalirkan sinyal melalui koneksi sinaptik yang memiliki bobot. Setiap neuron menerima input, mengalikannya dengan bobot yang sesuai, dan menghasilkan output melalui fungsi aktivasi non-linear. Jaringan ini belajar melalui pelatihan dengan menyesuaikan bobot sinaptik menggunakan algoritma seperti backpropagation, yang mengoptimalkan kinerja jaringan berdasarkan perbedaan antara output yang dihasilkan dan output yang diharapkan. Dalam konteks deep learning, NN dengan beberapa lapisan tersembunyi, dikenal sebagai *deep neural network* (DNN), mampu memodelkan representasi data yang semakin kompleks, menjadikannya efektif dalam tugas seperti pengenalan gambar, bahasa alami, dan pemrosesan data kompleks lainnya (Islam dan Jin, 2019).

Implementasi jaringan syaraf tiruan telah membawa perubahan signifikan dalam bidang Pemrosesan Bahasa Alami (*Natural Language Processing/NLP*) dan Pengenalan Entitas Bernama (*Named Entity Recognition/NER*). Dalam NLP, NN digunakan untuk menganalisis dan memahami struktur bahasa manusia, memungkinkan aplikasi seperti penerjemahan otomatis, analisis sentimen, dan pembangkitan teks yang lebih canggih. Dalam NER khususnya, NN telah memberikan kontribusi besar dalam mengidentifikasi dan mengklasifikasikan entitas bernama seperti nama orang, tempat, dan organisasi dalam teks. Model NN,

terutama yang berbasis arsitektur deep learning, dapat mengekstraksi pola kompleks dan kontekstual dari data teks, meningkatkan akurasi pengenalan entitas bernama dalam berbagai bahasa dan konteks. Implementasi ini tidak hanya meningkatkan kinerja sistem NLP dan NER, tetapi juga membuka peluang baru untuk aplikasi yang lebih canggih dalam pengelolaan informasi berbasis teks (Alshemali dan Kalita, 2020).

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

Terdapat tantangan dalam pembelajaran informasi jangka panjang melalui rekurensi dengan algoritma backpropagation (RNN). Dalam konteks ini, Hochreiter (1997) menganalisis masalah aliran balik kesalahan yang tidak memadai dan semakin mengecil seiring waktu. Sebagai solusi, mereka memperkenalkan Long Short-Term Memory (LSTM), metode berbasis gradien yang efisien.

LSTM memotong gradien di tempat-tempat yang tidak merugikan pembelajaran, mengatasi keterlambatan waktu minimal lebih dari 1000 langkah waktu diskrit dengan menjaga aliran kesalahan konstan melalui "karusel kesalahan konstan" dalam unit-unit khusus. Pintu multiplicative gate dalam LSTM memberikan fleksibilitas dalam pengelolaan informasi jangka panjang (Hochreiter dan Schmidhuber, 1997).

Keunggulan LSTM terletak pada sifatnya yang lokal dalam ruang dan waktu, mengurangi kompleksitas komputasinya per langkah waktu dan bobot menjadi O(1). Eksperimen dengan data buatan menunjukkan bahwa LSTM menghasilkan lebih banyak percobaan yang berhasil dan belajar lebih cepat

dibandingkan dengan metode-metode pembelajaran rekurensi sebelumnya. LSTM juga berhasil menyelesaikan tugas-tugas buatan yang melibatkan keterlambatan waktu yang panjang (Hochreiter dan Schmidhuber, 1997).

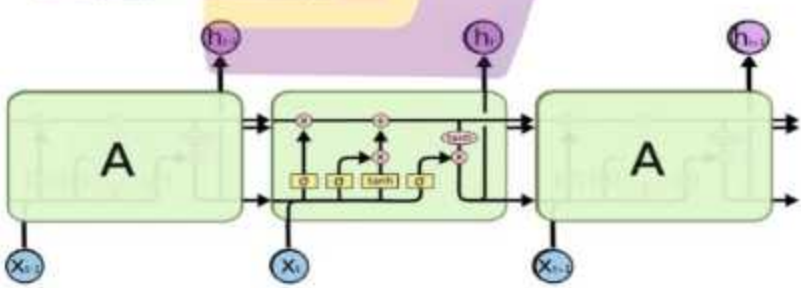
Sebagai ilustrasi pada Gambar 2.2, pemahaman ini melibatkan pemahaman terhadap peran masing-masing variabel dan elemen. Variabel input dan hidden states memberikan informasi pada setiap langkah waktu, sementara "A" sebagai matriks bobot mengontrol aliran informasi. Fungsi "tanh" memodifikasi nilai output, memberikan stabilitas pada proses pembelajaran (Nosouhian, S., 2021).

Dalam arsitektur LSTM terdapat tiga gate utama yang mengatur alur informasi, yaitu sebagai berikut (Arifoglu dan Bouchachia, 2018):

a. Forget gate, merupakan gate yang memutuskan informasi mana yang akan dihapus dari cell.

b. Input gate, merupakan gate yang memutuskan nilai dari input untuk di perbarui pada state memori.

c. Output gate, merupakan gate yang memutuskan apa yang akan dihasilkan sesuai dengan input dan memori pada cell.



Gambar 2.2. Arsitektur Dasar LSTM

Dalam struktur LSTM seperti yang tergambar pada Gambar 2.2, terdapat tiga gerbang utama: gerbang masukan (i_t), gerbang lupa (f_t), dan gerbang keluaran (o_t), yang bertugas mengatur aliran informasi masuk dan keluar dari sel memori. Implementasi LSTM telah berhasil dalam berbagai aplikasi, terutama dalam pengenalan ucapan, sintesis ucapan, pemodelan bahasa dan terjemahan, serta pengenalan tulisan tangan. Rumus-rumus yang digunakan ditandai dalam persamaan (2.1) hingga (2.6), dengan rumus khusus untuk menghitung gerbang lupa dinyatakan dalam persamaan (2.1). Pada Gambar 2.3 Merupakan ilustrasi *Forget Gate*.

Gambar 2.3. Forget Gate

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \quad (2.1)$$

Keterangan:

f_t = *forget gate*

σ = fungsi *sigmoid*

W_f = nilai bobot untuk *forget gate*

U_f = nilai bobot berulang untuk *forget gate*

h_{t-1} = nilai output sebelum orde ke t

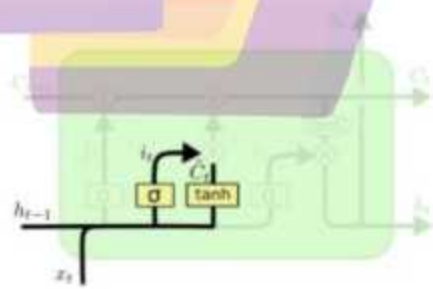
x_t = nilai input pada orde ke t

Keterangan (lanjutan persamaan 2.1) :

b_f = nilai bias pada *forget gate*

Pada Gambar 2.3, tahap pertama dalam LSTM adalah menentukan apakah informasi atau data akan dihapus atau disimpan oleh cell state (Ct-1) menggunakan fungsi sigmoid. Proses ini melibatkan dua data masukan: data masukan pada waktu tertentu (xt) dan data keluaran pada sel sebelumnya (*h*t-1), yang dikalikan dengan matrix bobot (Uf) dan diikuti dengan penambahan bias. Keluaran dari proses ini kemudian dimasukkan ke dalam fungsi aktivasi sigmoid, menghasilkan output biner (σ). Jika output sigmoid mendekati 0, informasi tersebut dihilangkan atau dilupakan oleh cell state; jika mendekati 1, informasi tersebut disimpan untuk digunakan pada tahap berikutnya.

Langkah selanjutnya dalam tahapan untuk menentukan informasi baru yang harus disimpan pada status sel ditunjukkan pada Gambar 2.4. Pertama, lapisan yang disebut sebagai lapisan gerbang masukan digunakan untuk menentukan nilai mana yang diperbarui, dengan menggunakan formula (2.2), di mana informasi diproses melalui gerbang masukan (it).



Gambar 2.4. Langkah pada input gate layer dan tanh layer (Olah, 2015)

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \quad (2.2)$$

Keterangan (persamaan 2.2):

i_t = *input gate*

σ = fungsi *sigmoid*

Keterangan (lanjutan persamaan 2.2) :

W_i = nilai bobot untuk *input gate*

U_i = nilai bobot berulang untuk *input gate*

h_{t-1} = nilai output sebelum orde ke *t*

x_t = nilai input pada orde ke *t*

b_i = nilai bias pada *input gate*

Pada tahap ini, dengan it sebagai input gate dan σ sebagai fungsi sigmoid, nilai bobot pada input gate W_i, data input pada waktu ke-t, serta nilai keluaran dari langkah waktu sebelumnya (h_{t-1}), dan b_i sebagai bias. Proses yang dilakukan adalah menentukan informasi yang akan diperbarui pada status sel menggunakan fungsi aktivasi sigmoid (σ), dan menyaring nilai yang akan disimpan dengan gerbang lupa (forget gate) menggunakan input (*ht*-1) dan (x_t).

Selanjutnya, lapisan tanh digunakan untuk menghasilkan nilai kandidat baru ($\widetilde{C}_t$), dengan menggunakan formula (2.3).

$$\widetilde{C}_t = tanh(W_c x_t + U_c h_{t-1} + b_c) \quad (2.3)$$

Keterangan:

$\widetilde{C}_t$ – nilai baru yang dapat ditambahkan ke cell state

$tanh$ – fungsi tanh

Wc – nilai bobot untuk cell state

Uc – nilai bobot berulang untuk cell state

$ht-1$ – nilai output sebelum order ke t

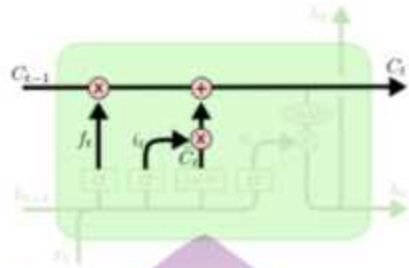
xt – nilai input pada orde ke t

bc – nilai bias untuk cell state

Dengan nilai baru ($\tilde{C}_t$) yang dihasilkan, menggunakan fungsi tanh, dengan & sebagai nilai bobot pada cell state, (x_t) sebagai nilai masukan pada waktu ke-t, U_c sebagai nilai matriks pada gerbang masukan, ht-1 sebagai nilai keluaran dari langkah waktu sebelumnya, dan b_c sebagai bias. Nilai yang dihasilkan oleh lapisan tanh berada di antara -1 dan 1, yang mencakup semua kemungkinan nilai dari ht-1 dan x_t . Kemudian, nilai-nilai tersebut dikalikan bersama untuk menghasilkan informasi yang penting.

Setelah menentukan informasi baru yang harus disimpan pada status sel, langkah selanjutnya adalah memperbarui status sel sebelumnya (Ct-1) menjadi status sel baru (Ct).

Dengan Ct sebagai cell state, ft sebagai forget gate, Ct-1 sebagai status sel sebelumnya, it sebagai input gate, $\tilde{C}_t$ sebagai nilai baru, dan * sebagai operasi perkalian. Pada langkah sebelumnya, Ct-1 dikalikan dengan ft, hasil perkalian tersebut ditambahkan dengan it dikali $\tilde{C}_t$. Ini dilakukan untuk memperbarui nilai di setiap status sel, yang merupakan proses pada Gambar 2.5.



Gambar 2.5. Memperbaharui Nilai Cell State (Olah, 2015)

$$c_t = f_t * c_{t-1} + i_t * \widetilde{C}_t) \quad (2.4)$$

Keterangan (Persamaan 2.4):

Ct – cell state

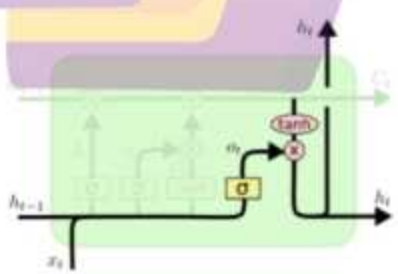
ft – forget state

$Ct-1$ – cell state sebelum orde ke t

it – input gate

$\widetilde{C}\,t$ – nilai baru yang dapat ditambahkan ke cell state

Pada langkah terakhir, gerbang keluaran (ot) bertugas untuk mengekstrak informasi yang berguna dari cell state, sesuai dengan yang diilustrasikan pada Gambar 2.6.



Gambar 2.6. Output Gate (Olah, 2015)

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \quad (2.5)$$

Keterangan:

ot – output gate

σ – fungsi sigmoid

Wo – nilai bobot untuk output gate

Uo – nilai bobot berulang untuk output gate

$ht-1$ – nilai output sebelum orde ke t

xt – nilai input pada orde ke t

bo – nilai bias pada output gate

Dengan ot sebagai output gate, σ sebagai fungsi sigmoid, 'Wo sebagai nilai bobot pada output gate, Xt sebagai data masukan pada waktu ke-t, Uo sebagai nilai matriks pada gerbang keluaran, *h*t-1 sebagai nilai keluaran pada langkah waktu sebelumnya, dan ,bo sebagai bias.

Langkah pertama, fungsi aktivasi sigmoid digunakan untuk memutuskan informasi yang akan dihasilkan. Selanjutnya, nilai keluaran dari cell state dimasukkan ke dalam lapisan tanh (yang berfungsi sebagai nilai pengganti antara 1 dan -1), kemudian dikalikan dengan output dari sigmoid untuk memutuskan informasi yang akan disimpan menggunakan data masukan (*h*t-1) dan (!), sehingga diperoleh informasi untuk dikirimkan sebagai hasil dan data masukan pada sel selanjutnya. Untuk menentukan gerbang keluaran, dapat digunakan formula (2.5). Langkah terakhir adalah menghitung hasil keluaran yang dimasukkan ke dalam fungsi tanh, kemudian hasil tersebut dikalikan dengan nilai keluaran dari gerbang sigmoid menggunakan formula (2.6).

$$h_t = o_o * tanh\,(c_t) \quad (2.6)$$

Keterangan:

ht – nilai output orde t

ot – output gate

$tanh$ – fungsi tanh

Ct – cell state

Dengan ht adalah hidden state, ot adalah output gate, tanh merupakan fungsi tanh, ∘ adalah operasi perkalian, dan Ct adalah status sel.

Dalam keseluruhan, LSTM (Long Short-Term Memory) membawa kontribusi yang signifikan dalam mengatasi tantangan memodelkan dan memprediksi urutan data kompleks. Dengan struktur yang terdiri dari input gate, forget gate, dan output gate, LSTM mampu mengatasi masalah "vanishing gradient" dan mempertahankan informasi yang relevan dalam urutan data panjang. Implementasi LSTM telah menunjukkan hasil yang baik dalam berbagai aplikasi seperti pengenalan ucapan, sintesis ucapan, pemodelan bahasa dan terjemahan, serta pengenalan tulisan tangan. Dengan demikian, LSTM tetap menjadi salah satu alat yang sangat berguna dalam dunia pemrosesan bahasa alami, pengenalan pola, dan tugas-tugas lain yang melibatkan urutan data.

BAB III

METODE PENELITIAN

3.1. Jenis, Sifat, dan Pendekatan Penelitian

Penelitian yang dilakukan dalam tesis ini merupakan penelitian eksperimental, di mana peneliti menggunakan pendekatan eksperimental untuk menyelidiki dan menguji hipotesis terkait optimalisasi model Bi-LSTM dengan algoritma genetika dalam mendeteksi entitas lokasi dan waktu pada data teks berbahasa Indonesia mengenai kebakaran hutan. Metode eksperimental memungkinkan untuk fokus pada pengendalian variabel-variabel tertentu guna mengidentifikasi hubungan sebab-akibat dalam fenomena tersebut (Hasibuan dan Zainal, 2007). Eksperimen ini dilakukan dengan tujuan menguji efektivitas algoritma genetika dalam meningkatkan kinerja model Bi-LSTM. Pendekatan ini memberikan keleluasaan untuk secara aktif memanipulasi variabel-variabel tersebut, sehingga hasilnya dapat digunakan untuk memvalidasi atau menyanggah hipotesis penelitian kami (Sudipa dkk., 2023).

Pendekatan kausal bertujuan untuk mengeksplorasi dampak dari variabel bebas, yaitu penerapan algoritma genetika pada model Bi-LSTM, terhadap variabel terikat, yaitu efektivitas deteksi entitas lokasi dan waktu pada data teks Bahasa Indonesia tentang kebakaran hutan. Dengan demikian, melalui penelitian diharapkan dapat menyumbangkan wawasan yang lebih mendalam mengenai pengaruh algoritma genetika terhadap peningkatan kinerja model Bi-LSTM dalam konteks aplikasi deteksi entitas pada data teks yang bersifat kausal tersebut.

Pendekatan kuantitatif yang dipilih dalam penelitian ini berasal dari ilmu alam. Kami menggunakan metode ini untuk memahami bagaimana sesuatu dibangun dan bekerja, dengan membangun penjelasan dari data numerik. Pendekatan ini bersifat obyektif, berorientasi pada verifikasi, dan melibatkan observasi yang dikontrol. Umumnya, metode ini juga melibatkan generalisasi dari hasil penelitian. Dengan pendekatan ini, kami berharap dapat memberikan kontribusi signifikan dalam pemahaman mengenai deteksi entitas lokasi dan waktu pada data teks Bahasa Indonesia terkait kebakaran hutan.

3.2. Metode Pengumpulan Data

Data bersumber dari tim peneliti konsorsium SILVANUS (https://silvanus-project.eu/about/). Data yang diperoleh sudah memiliki *tag* dalam skema *encoding* Begin-Inside-Outside (BIO) dengan pemberian *tag* secara manual oleh tim peneliti SILVANUS. Dalam bentuk dataset, datum sudah dalam bentuk token-token dengan penanda berupa *sentence* untuk menandai bahwa bagian datum atau kata yang *ditokenisasi* masih dalam satu kalimat atau korpus.

Tim SILVANUS melakukan metode pengumpulan data melalui API Twitter dengan menerapkan filter "Kebakaran Hutan" saat melakukan pencaian. Setiap *tweet* yang memenuhi kriteria filter diunduh dan disimpan dalam format CSV untuk memudahkan analisis. Data yang dikumpulkan akan diberikan tag NER (Named Entity Recognition) dengan skema *encoding* BIO untuk menandai waktu dan lokasi pada setiap kata secara manual..

Jenis data yang dikumpulkan melibatkan teks tweet, tanggal pembuatan tweet, dan informasi lokasi geografis (jika tersedia). Data ini menjadi dasar untuk analisis lebih lanjut terkait deteksi entitas lokasi dan waktu pada data teks berbahasa Indonesia mengenai kebakaran hutan.

Metode persiapan dan pengolahan data dalam penelitian ini dimulai dengan tahap tokenisasi kalimat atau korpus. Dalam langkah ini, data teks awal dipisahkan menjadi kalimat-kalimat terpisah, yang selanjutnya dipecah menjadi token-token, mewakili unit dasar seperti kata-kata atau subkata. Proses selanjutnya melibatkan pelabelan manual untuk tag Named Entity Recognition (NER) BIO, di mana setiap token diberi label yang menunjukkan apakah itu merupakan awal entitas (B), bagian dalam entitas (I), atau token di luar entitas (O). Para labeler terlibat secara cermat dalam proses pelabelan, memastikan konsistensi dan memperhatikan konteks saat menetapkan label pada entitas yang berbeda. Proses pelabelan manual ini merupakan langkah kritis untuk memastikan kualitas data yang digunakan dalam penelitian ini.

3.3. Metode Analisis Data

Metode analisis data dalam penelitian ini dirancang untuk merespon setiap aspek dari rumusan masalah. Pertama-tama, tahap preprocessing data dilaksanakan untuk memastikan kebersihan dataset sebagai data latih. Langkah-langkah ini termasuk konversi huruf besar ke huruf kecil dan penghapusan karakter khusus, menghasilkan dataset yang terstruktur dengan baik dan siap untuk digunakan dalam pengoptimalan.

Setelah tahap preprocessing selesai dilakukan pengoptimalan model Bi-LSTM dengan algoritma genetika. Algoritma genetika digunakan untuk mengeksplorasi dan menggabungkan solusi potensial secara iteratif. Proses ini menciptakan populasi model Bi-LSTM yang beragam, di mana setiap individu mewakili konfigurasi unik dari faktor *hyperparameter* seperti *units, dropout, embedding dimension, learning rate, batch size, epoch, layer*, dan regularisasi kernel.

Melalui serangkaian generasi, algoritma genetika mengevaluasi kinerja setiap individu berdasarkan tingkat akurasi mereka dalam tugas deteksi entitas lokasi dan waktu pada data teks terkait kebakaran hutan. Model Bi-LSTM yang memberikan hasil terbaik memiliki peluang lebih besar untuk bertahan dan berkembang dalam populasi. Solusi sukses ini kemudian menjadi dasar untuk menciptakan generasi baru model, dan siklus ini berulang hingga konfigurasi optimal tercapai.

Pendekatan pengoptimalan menggunakan algoritma genetika tidak hanya melibatkan penyesuaian acak dari parameter. Sebaliknya, proses ini melibatkan penemuan kombinasi parameter yang secara efektif mengoptimalkan kinerja model Bi-LSTM dalam konteks deteksi entitas pada data teks berbahasa Indonesia mengenai kebakaran hutan. Pendekatan ini memastikan bahwa model Bi-LSTM dapat disesuaikan secara spesifik untuk menangani tugas yang kompleks ini.

Langkah berikutnya setelah pengoptimalan model Bidirectional Long Short-Term Memory (Bi-LSTM) dengan algoritma genetika adalah evaluasi kinerja model. Analisis data dilakukan pada dataset yang telah diberi label, dan

performa model diukur terhadap kemampuannya dalam mengenali dan mengklasifikasikan entitas waktu dan lokasi pada data teks berbahasa Indonesia yang berkaitan dengan kebakaran hutan. Evaluasi ini melibatkan penggunaan matriks evaluasi, termasuk *Confusion Matrix, Precision, Recall, dan F1-score*, yang memberikan gambaran rinci tentang hasil klasifikasi model. Selain itu, fungsi kerugian (*loss function*) seperti *Cross-Entropy Loss* digunakan untuk mengevaluasi sejauh mana distribusi probabilitas prediksi model mendekati distribusi probabilitas sebenarnya. Akurasi keseluruhan dan Area di Bawah Kurva ROC (AUC-ROC) juga dianalisis untuk memberikan pemahaman menyeluruh tentang kinerja model. Dengan menganalisis metrik-metrik ini, dapat diperoleh wawasan yang mendalam tentang sejauh mana tingkat akurasi yang dapat dicapai oleh model Bi-LSTM dalam tugas deteksi entitas pada data teks berbahasa Indonesia tentang kebakaran hutan.

Untuk mengidentifikasi faktor-faktor yang mempengaruhi kinerja model Bi-LSTM, langkah-langkah eksperimental yang sistematis diperlukan. Pertama, dilakukan eksperimen variasi parameter hyperparameter seperti *units, dropout, embedding dimension, learning rate, batch size, epoch, layer,* dan regularisasi kernel. Setiap parameter divariasikan dalam rentang nilai yang mungkin, dan eksperimen ini diulang berulang kali untuk melihat bagaimana perubahan nilai pada masing-masing parameter mempengaruhi kinerja model. Rancangan eksperimen ini perlu sistematis agar dapat mengevaluasi setiap parameter secara terpisah dan menghindari kompleksitas interaksi antar parameter.

Setelah melakukan eksperimen, metrik kinerja seperti akurasi, *precision, recall, F1-score,* dan fungsi kerugian dievaluasi pada setiap variasi parameter. Hasil eksperimen dicatat dan dianalisis untuk mengidentifikasi tren atau pola yang muncul. Melalui analisis statistik dan visualisasi, ditelusuri apakah terdapat "puncak" atau "lembah" dalam kinerja model terkait dengan kombinasi tertentu dari hyperparameter. Selanjutnya, dilakukan analisis sensitivitas untuk mengidentifikasi parameter yang paling mempengaruhi kinerja model, dengan mengubah satu parameter sambil menjaga parameter lain konstan.

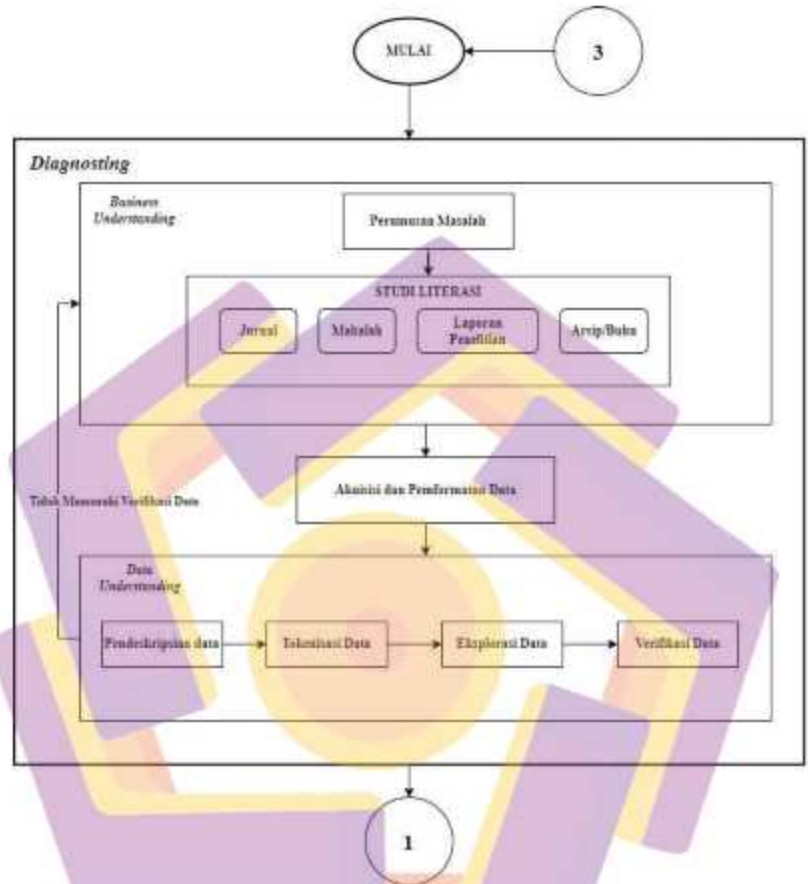
Proses identifikasi faktor-faktor tersebut menjadi inti analisis untuk memahami lebih dalam tentang variabel mana yang memiliki dampak signifikan terhadap hasil deteksi entitas lokasi dan waktu dalam bahasa Indonesia. Dengan melibatkan langkah-langkah eksperimental dan analisis yang cermat, diharapkan penelitian ini dapat memberikan pemahaman yang lebih komprehensif tentang cara mengoptimalkan model Bi-LSTM untuk tugas deteksi entitas pada data teks berbahasa Indonesia yang berkaitan dengan kebakaran hutan.

3.4. Alur Penelitian

Alur penelitian dibagi menjadi lima bagian utama yakni *diagnosing, action planning, action taking, evaluation dan specifying learning* dengan memasukkan tahapan-tahapan CRISP-DM (Kusrini, 2009) alur penelitian ini diilustrasikan pada Gambar 3.1-12. CRISP-DM (Cross-Industry Standard Process for Data Mining) adalah metodologi yang umum digunakan dalam proses penggalian data atau data mining. Metodologi ini terdiri dari enam tahap utama, yaitu pemahaman bisnis,

pemahaman data, persiapan data, pemodelan, evaluasi, dan penyiapan. Pada tahap pemahaman bisnis, tujuan dan kebutuhan bisnis yang ingin dicapai dengan analisis data ditetapkan. Tahap pemahaman data melibatkan eksplorasi awal terhadap dataset yang ada. Kemudian, pada tahap persiapan data, data disiapkan dan dibersihkan untuk analisis lebih lanjut. Tahap pemodelan melibatkan pengembangan model analisis data, diikuti oleh tahap evaluasi untuk mengukur performa model. Terakhir, pada tahap penyiapan, hasil analisis diimplementasikan ke dalam lingkungan bisnis. CRISP-DM memberikan kerangka kerja yang sistematis dan terstruktur untuk mendukung proses penggalian data dalam konteks berbagai industri.

Pada Gambar 3.1, penelitian dimulai dengan tahap diagnostik, dengan tahap pertama adalah business understanding yang mencakup perumusan masalah. Proses ini kemudian ditinjau dan dipelajari melalui studi literatur yang mencakup pemahaman dan referensi dari berbagai sumber ilmiah seperti paper, makalah, buku, dan laporan ilmiah. Setelah tahap business understanding, peneliti mengambil data dari tim peneliti SILVANUS, yang sudah diformat dan diberikan tag secara manual untuk entitas lokasi dan waktu dengan skema Begin Inside Outside (BIO). Selanjutnya, data masuk ke tahap data understanding.



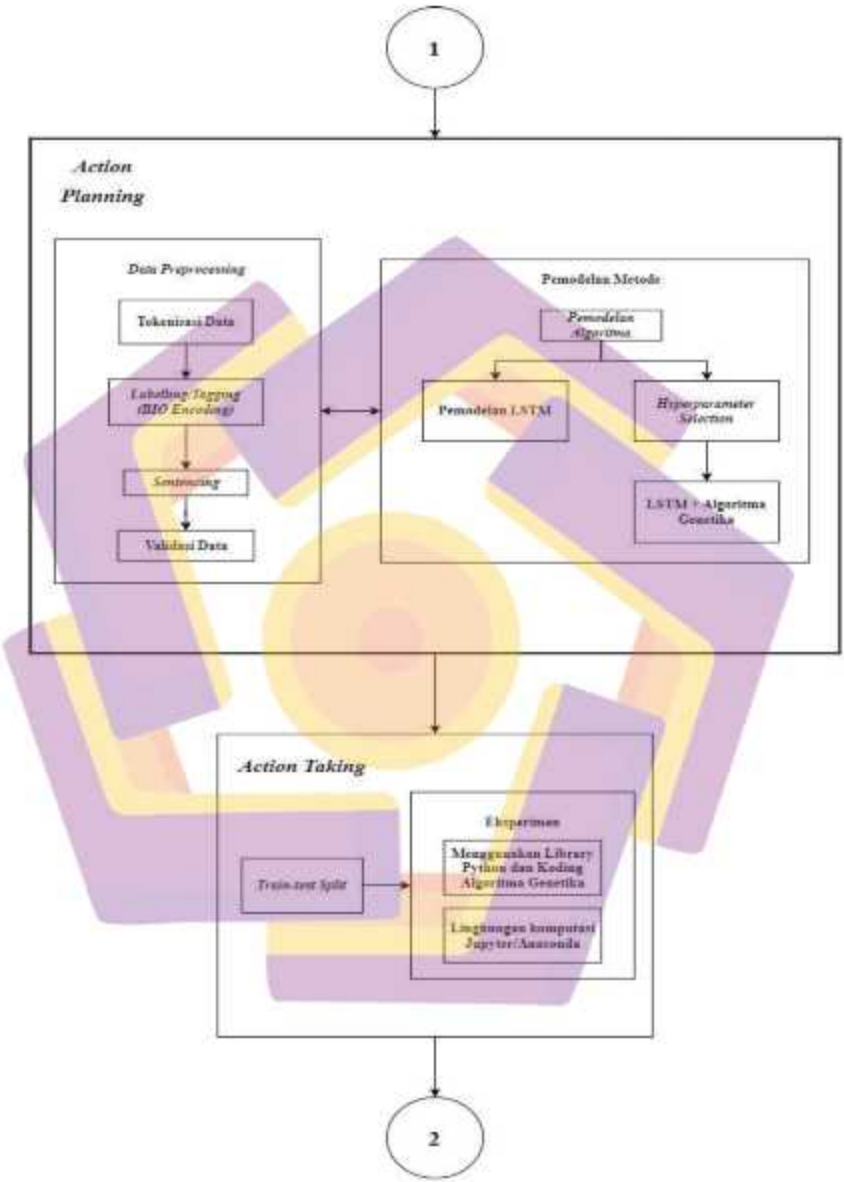
Gambar 3.1. Alur Penelitian Bagian 1

Berdasarkan Gambar 3.1, dalam tahap data understanding, langkah pertama adalah pendeskripsian data dengan tujuan memahami karakteristik dan struktur data. Proses ini diikuti oleh tahap tokenisasi data, di mana data dibagi menjadi unit-token untuk analisis lebih lanjut. Eksplorasi data dilakukan untuk mengeksplorasi pola atau tren yang mungkin terkandung dalam dataset.

Selanjutnya, verifikasi data dilakukan untuk memastikan bahwa data memenuhi kriteria yang diperlukan, terutama terkait dengan penggunaan Named Entity Recognition (NER). Jika data tidak memenuhi verifikasi, penelitian akan kembali ke tahap business understanding untuk penyesuaian. Dalam diagram tersebut, terdapat suatu lingkaran yang mengarah ke alur "mulai" dengan angka 3, menunjukkan adanya proses lain yang dapat menyebabkan kembali ke langkah 3 dalam metodologi penelitian. Ini menggambarkan siklus iteratif yang mungkin terjadi dalam proses penelitian untuk memastikan bahwa data dan pemahaman bisnis saling mendukung.

Langkah selanjutnya adalah menuju langkah 1, yang terdiri dari action planning dan action taking. Gambar 3.2 Alur Penelitian Bagian 2 memberikan ilustrasi visual tentang kedua tahap ini, menyoroti proses berkelanjutan dalam perjalanan penelitian.

Pada Gambar 3.2 nampak ilustrasi dengan dua bagian yakni action planning dan action taking. Pada action planning, peneliti merencanakan tindakan yang akan diambil berdasarkan pemahaman bisnis dan hasil dari tahap sebelumnya. Sedangkan pada action taking, rencana tersebut diimplementasikan, meliputi penerapan model analisis data dalam proses eksperimen.



Gambar 3.2. Alur Penelitian Bagian 2

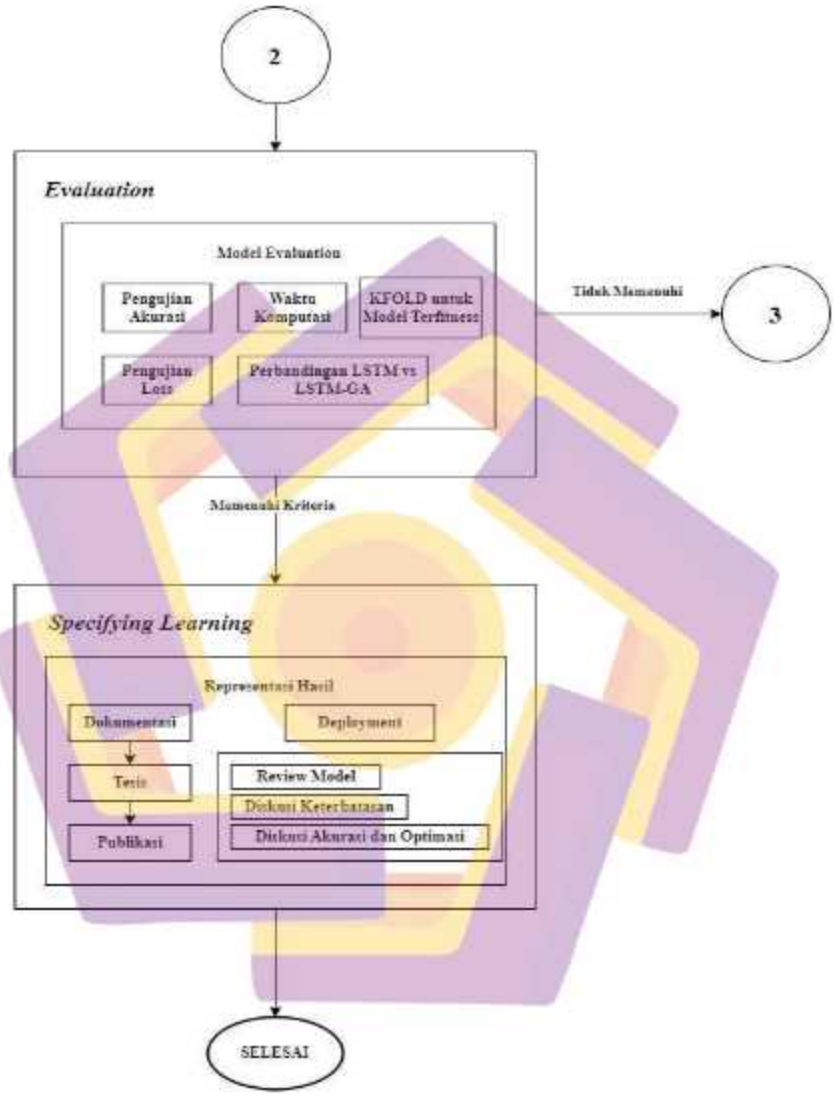
Pada Gambar 3.2 bagian data preprocessing pada action planning merupakan fase krusial dalam mempersiapkan dataset sebelum memasuki tahap pemodelan. Tokenisasi data menjadi langkah awal di mana teks atau data non-struktural dibagi menjadi unit-token, mempermudah analisis dan pemrosesan selanjutnya. Labelling dan tagging dilakukan untuk memberikan label atau tanda pada data, khususnya terkait entitas seperti lokasi atau waktu, sehingga model dapat mengenali dan memahami konteks informasi dengan lebih baik. Langkah selanjutnya adalah sentencing, di mana data dikelompokkan berdasarkan struktur kalimat atau frase untuk memudahkan pemahaman dan analisis. Proses ini membantu menciptakan suatu representasi yang lebih terstruktur dari data mentah. Selain itu, validasi data juga menjadi aspek penting dalam action planning, di mana data diverifikasi untuk memastikan kesesuaian dengan kriteria yang ditetapkan sebelumnya, terutama terkait dengan tujuan penggunaan Named Entity Recognition (NER). Dengan demikian, bagian data preprocessing pada action planning mencakup serangkaian langkah yang bersifat esensial untuk memastikan keakuratan, struktur, dan kualitas data sebelum masuk ke fase pemodelan.

Bagian selanjutnya, Pemodelan Metode, terdapat bagan "Pemodelan Algoritma" dengan dua aspek fokus, yaitu metode Bi-LSTM (Bidirectional Long Short-Term Memory) dan Hyperparameter Selection. Bi-LSTM, sebagai model rekurensi yang efektif, memungkinkan penanganan informasi temporal dan pola-pola kompleks dalam dataset. Hyperparameter Selection, diikuti oleh penggunaan Algoritma Genetika, menjadi kunci dalam mengoptimalkan parameter-model untuk meningkatkan performa dan ketepatan prediksi. Alur dengan panah dua

arah dalam ilustrasi mencerminkan pentingnya validasi data, memberikan fleksibilitas untuk kembali ke tahap preprocessing jika terdapat kebutuhan perbaikan atau penyesuaian. Dengan demikian, Pemodelan Metode menjadi fase kritis dalam menggambarkan bagaimana pendekatan ini tidak hanya mengadopsi model pemodelan yang canggih, seperti Bi-LSTM, tetapi juga menitikberatkan pada pengoptimalan parameter untuk meraih hasil yang optimal.

Selanjutnya, pada bagian action taking, proses train-test split dilakukan untuk mempersiapkan data. Eksperimen dilakukan dengan menggunakan library Python untuk Bi-LSTM dan Algoritma Genetika, yang diimplementasikan secara manual dalam lingkungan komputasi Jupyter Notebook atau Anaconda. Tahap ini merupakan langkah kunci dalam mengaplikasikan rencana yang telah disusun pada action planning.

Alur penelitian di Gambar 1.2 mengarah ke alur ke-2, menandakan bahwa hasil eksperimen dan evaluasi model akan mempengaruhi langkah-langkah berikutnya. Pada alur ke-2, terdapat evaluation untuk menilai kinerja model, dan jika tidak memenuhi kriteria, alur menuju ke langkah 3, yang melibatkan penajaman spesifikasi atau redefinisi pembelajaran.



Gambar 3.3 Alur Penelitian Bagian 3

Pada Gambar 3.2. Alur Penelitian Bagian 3, fokus terletak pada evaluasi model dengan sub model evaluation yang mencakup sejumlah metrik penting.

Pengujian akurasi dan pengujian loss digunakan untuk mengukur sejauh mana model mampu memberikan prediksi yang tepat dan efisien. Waktu komputasi dievaluasi untuk memahami performa waktu dari model tersebut, sedangkan KFOLD digunakan untuk menilai seberapa baik model dapat menyesuaikan diri dengan dataset yang berbeda. Perbandingan antara model Bi-LSTM dan Bi-LSTM-GA juga diterapkan untuk mendapatkan wawasan tentang kemungkinan peningkatan performa melalui optimasi genetika.

Jika hasil evaluasi menunjukkan tidak memenuhi kriteria, alur secara iteratif kembali ke tahap 3 dengan arah menuju "mulai". Namun, jika model memenuhi kriteria, alur berlanjut ke bagian specifying learning. Di dalamnya, ada sub bagian "representasi hasil", yang melibatkan deployment model ke tim backend untuk diterapkan dalam pengenalan entitas waktu dan lokasi kebakaran hutan. Proses ini didokumentasikan dengan penyusunan tesis dan publikasi ilmiah. Bagian review dan diskusi menyoroti evaluasi model, membahas keterbatasan, akurasi, dan strategi optimasi yang diterapkan.

Alur penelitian kemudian mengarah ke tahap selesai, menandakan penutupan dari iterasi penelitian yang sistematis, ilmiah, dan komprehensif. Ini mencerminkan pendekatan metodologis yang mendalam dan terstruktur dalam mengatasi tantangan pemodelan serta implementasi solusi dalam pengenalan entitas waktu dan lokasi kebakaran hutan.

BAB IV

HASIL PENELITIAN DAN PEMBAHASAN

4.1. Perencanaan Aksi (*Action Planning)*

Pada sub bab ini akan membahas terkait perencanaan aksi untuk menyelesaikan masalah penelitian. Perencanaan aksi (terjemahan dari : *Action Planning*) akan terdiri dari dua bagian utama yakni *data preprocessing* dan *data modelling/experiment setting*. Pada *data preprocessing* akan dilakukan tokenisasi data, *labelling, sentencing* dan validasi data.

Pada bagian ini tidak memuat pembahasan sumber data. Namun sebagai penjelas, bahwasanya data diperoleh dari tim peneliti SILVANUS, yang sudah diformat dan diberikan tag secara manual untuk entitas lokasi dan waktu dengan skema Begin Inside Outside (BIO).

Adapun lingkungan perangkat lunak yang digunakan adalah dengan rincian sebagai berikut :

1. Bahasa pemrograman Python
2. JupyterNotebook sebagai *IDE*.
3. Berikut *library* penunjang yang digunakan dalam eksperimental seperti Pandas, Matplotlib, Numpy, Tensorflow dan Keras.

4.1.1. *Data Preprocessing*

Data yang diperoleh dan diakuisisi disimpan dalam bentuk format Comma Separated Values (CSV) atau .CSV. Selanjutnya data diakuisisi menggunakan bahasa pemrograman Python dan menggunakan library data akan berada dalam

format *encoding* pandas dengan nilai "latin1". Tabel 4.1. berikut merupakan cuplikan dari data dari dataset yang digunakan, di mana kaidah Tag menggunakan BIO dengan total 1103 *sentence*, data selengkapnya ada di Lampiran.

Tabel 4.1. Cuplikan *Dataset* dengan *Tagnya*

Sentence	Word	Tag
1	Azkiya	I-per
1	Dihni	I-per
1	Vika	B-per
1	Aria	B-per
2	10.2	I-tim
2	Yudhistira	I-per
2	11/1/2022,	B-tim
2	2021	B-tim
2	2020	B-tim
2	Indonesia	B-geo
2	di	O
2	WIB	O
3	hektare	I-qty
...	...	...
1102	juta	I-qty
1102	hektar	I-qty
1102	2,6	B-qty
1102	Indonesia	B-geo
1103	miliar	I-qty
1103	triliun.	I-qty
1103	US$16	B-qty
1103	Rp224	B-qty

Berikut merupakan baris kode pada tahap akuisisi dan pemanggilan *library*.

```
#Baris Kode Berikut ini untuk mengambil library yang dibutuhkan
%matplotlib inline  # Perintah khusus untuk Jupyter Notebook
agar plot matplotlib ditampilkan secara langsung di dalam
notebook
import matplotlib.pyplot as plt  # Modul untuk membuat plot
import pandas as pd  # Modul untuk manipulasi dan analisis data
import numpy as np  # Modul untuk operasi numerik
np.random.seed(0)  # Menetapkan seed agar hasil random dapat
direproduksi
plt.style.use("ggplot")  # Mengatur gaya plot menjadi "ggplot"
import tensorflow as tf  # Modul untuk pembelajaran mesin
menggunakan TensorFlow

Baris kode berikut untuk akuisisi dan membaca file data
data = pd.read_csv("Bersih_ner_karhutla.csv", encoding="latin1")
# Membaca data dari file CSV
data.head(20)  # Menampilkan 20 baris pertama dari data
```

Koding ini menggunakan beberapa modul Python, termasuk matplotlib, pandas, numpy, dan tensorflow. Koding ini dimulai dengan mendefinisikan %matplotlib inline, yang merupakan perintah khusus untuk Jupyter Notebook agar plot matplotlib ditampilkan secara langsung di dalam notebook. Selanjutnya, modul-modul yang diperlukan diimpor, dan gaya plot ggplot diterapkan.

Kemudian, data dibaca dari file CSV dengan nama "Bersih_ner_karhutla.csv" menggunakan pd.read_csv(). Data yang dibaca kemudian ditampilkan menggunakan data.head(20), yang menampilkan 20 baris pertama dari data tersebut.

Selanjutnya data akan disesuaikan labelnya dengan mengkonfirmasi bentuk tag secara manual, adapun data yang kosong akan diberikan nilai "O" yang

berarti statusnya Outside dalam skema BIO. Berikut adalah baris kode untuk mengisi data kosong dengan label "O".

```
data["Tag"].fillna("O", inplace=True)  # Mengisi nilai-nilai
yang hilang (NaN) dalam kolom "Tag" dengan "O"
data.head(20)  # Menampilkan 20 baris pertama dari data setelah
pengisian nilai yang hilang
```

Komentar ini menjelaskan bahwa perintah pertama mengisi nilai-nilai yang hilang (NaN) dalam kolom "Tag" dengan nilai "O". Kemudian, perintah kedua menampilkan 20 baris pertama dari data setelah pengisian nilai yang hilang.

Selanjutnya adalah pembentukan senarai yang akan memisahkan dan menandai akhir dari sebuah kalimat. Berikut adalah baris kodenya.

```
# Mengambil nilai unik dari kolom "Word" dalam data dan
mengonversinya menjadi list, lalu mengubahnya menjadi set untuk
menghilangkan duplikat
words = list(set(data["Word"].values))

# Menambahkan token khusus "ENDPAD" ke dalam list kata
words.append("ENDPAD")

# Menghitung jumlah kata unik dalam dataset
num_words = len(words)

# Mengambil nilai unik dari kolom "Tag" dalam data dan
mengonversinya menjadi list
tags = list(set(data["Tag"].values))

# Menghitung jumlah tag unik dalam dataset
num_tags = len(tags)
```

Koding ini bertujuan untuk mempersiapkan data yang akan digunakan dalam pemrosesan lebih lanjut, seperti dalam tugas pemrosesan bahasa alami atau tugas terkait NLP lainnya. Pertama, langkah-langkah untuk mempersiapkan token kata ('words') dilakukan dengan mengambil nilai unik dari kolom "Word" dalam dataset. Setelah itu, token khusus "ENDPAD" ditambahkan ke dalam list kata tersebut. Ini bertujuan untuk menandai akhir dari sebuah urutan kata dalam data.

Jumlah kata unik dalam dataset dihitung untuk keperluan penggunaan lebih lanjut dalam pemodelan. Selanjutnya, langkah serupa dilakukan untuk menyiapkan token tag ('tags'), dengan mengambil nilai unik dari kolom "Tag" dalam data. Jumlah tag unik dalam dataset juga dihitung. Dengan langkah-langkah ini, data menjadi siap untuk digunakan dalam proses pelatihan dan evaluasi model pemrosesan bahasa alami yang akan dilakukan selanjutnya.

Selanjutnya adalah proses *sentencing* yang dimuat dalam kelas dan fungsi-fungsi untuk mengelompokkan sebuah kalimat atau korpus ke dalam satu kesatuan berdasarkan dataset yang sudah diberikan penanda pada tiap korpusnya. Berikut adalah barisan kodenya.

```
class SentenceGetter(object):
    def __init__(self, data):
        self.n_sent = 1  # Inisialisasi indeks kalimat
        self.data = data  # Menyimpan dataset
        self.empty = False  # Indikator untuk dataset kosong

# Fungsi agregasi untuk mengambil pasangan kata dan tag dari
setiap baris dalam dataset
        agg_func = lambda s: [(w, t) for w, t in
zip(s["Word"].values.tolist(), s["Tag"].values.tolist())]

        # Mengelompokkan dataset berdasarkan kalimat dan
menerapkan fungsi agregasi
        self.grouped =
self.data.groupby("Sentence").apply(agg_func)

        # Mengonversi hasil pengelompokkan menjadi list kalimat
        self.sentences = [s for s in self.grouped]

    def get_next(self):
        try:
            # Mengambil kalimat berikutnya dari dataset
            s = self.grouped["Sentence: {}".format(self.n_sent)]
            self.n_sent += 1  # Menambahkan indeks kalimat
            return s
        except:
            return None
# Membuat objek SentenceGetter dengan dataset yang diberikan
getter = SentenceGetter(data)
# Mendapatkan list kalimat dari objek SentenceGetter
```

```
sentences = getter.sentences
word2idx = {w: i + 1 for i, w in enumerate(words)}  # Membuat
kamus untuk memetakan kata ke indeks
tag2idx = {t: i for i, t in enumerate(tags)}  # Membuat kamus
untuk memetakan tag ke indeks
```

Kelas SentenceGetter digunakan untuk memfasilitasi pengambilan kalimat-kalimat dari dataset. Dalam metode __init__, dataset disimpan dalam variabel self.data, kemudian dataset dikelompokkan berdasarkan kalimat dan fungsi agregasi diterapkan untuk mengambil pasangan kata dan tag dari setiap baris dalam dataset. Hasil pengelompokkan disimpan dalam variabel self.grouped, dan kemudian diubah menjadi list kalimat dalam variabel self.sentences.

Metode get_next digunakan untuk mengambil kalimat berikutnya dari dataset. Dengan mencoba mengakses kalimat berdasarkan indeks n_sent, kalimat berikutnya diambil dari dataset dan indeks n_sent ditingkatkan. Jika tidak ada lagi kalimat yang tersedia, metode ini mengembalikan None.

Setelah membuat objek SentenceGetter dengan dataset yang diberikan, list kalimat diambil dari objek tersebut dan disimpan dalam variabel sentences.

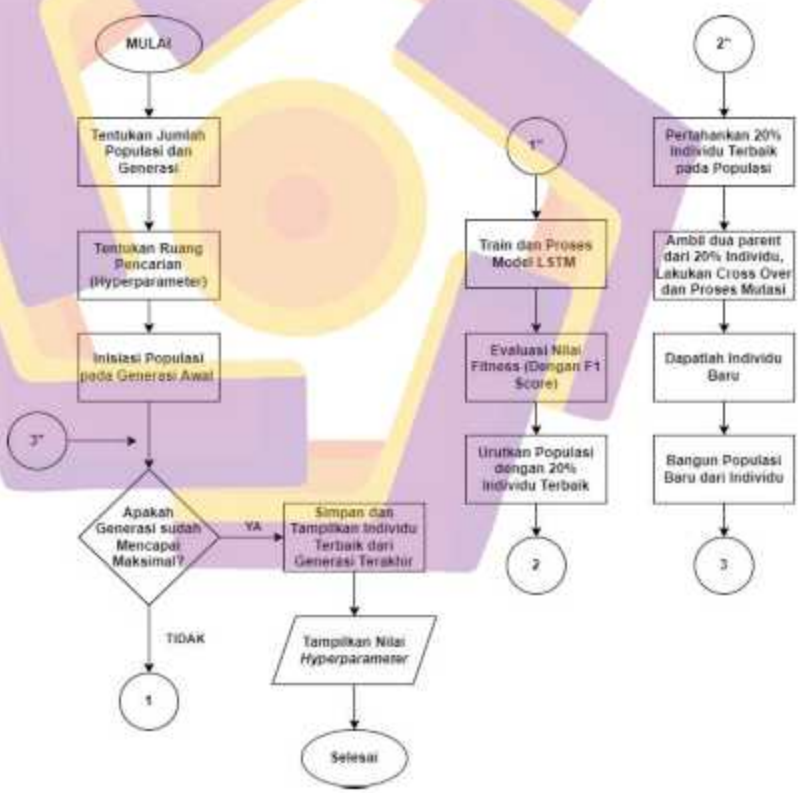
Kemudian, kamus word2idx dibuat untuk memetakan setiap kata ke indeksnya dalam list kata (words), sementara kamus tag2idx dibuat untuk memetakan setiap tag ke indeksnya dalam list tag (tags). Ini mempersiapkan data untuk digunakan dalam proses pembuatan model pemrosesan bahasa alami berikutnya.

Pada tahap validasi data, data sebelumnya sudah dikonfirmasi dan divalidasi bentuk, skema dan pelabelannya menggunakan software Google Spreadsheet. Dengan menggunakan perintah untuk melihat keunikan data maka

data yang nilainya unik atau berbeda dari ketentuan akan nampak. Koreksi bentuk kesalahan misalnya "B-pet" menjadi bentuk "B-per". Kasus ini terjadi karena ketidaktelitian atau kesalahan ketik (*typo).*

4.1.2. Algoritma Sistem

Algoritma dalam penelitian ini bertujuan untuk mencari nilai *hyperparameter* terbaik dari model Bi-LSTM melalui serangkaian langkah pencarian secara evolusi yakni dengan algoritma genetika, nampak pada Gambar 4.1.

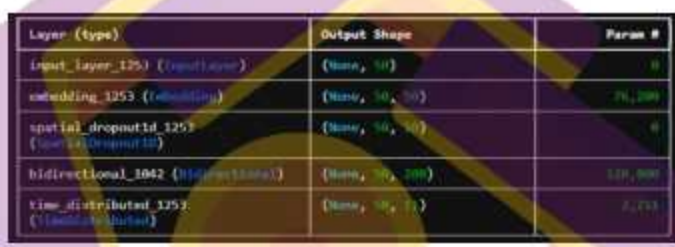


Gambar 4.1. Algoritma Sistem (Optimisasi *Hyperparameter Bi-LSTM* dengan AG*)*

4.1.3. Pemrograman Metode

Pada tahap ini perencanaan model algoritma akan menggunakan beberapa pengaturan, pertama pada implementasi Bi-LSTM yang hasil performanya nanti akan dikomparasi dengan Bi-LSTM yang sudah diseleksi *hyperparameternya* dengan Algoritma Genetika.

Implementasi Bi-LSTM yang digunakan adalah bidirectional Bi-LSTM (Bi-LSTM). Dengan konfigurasi berikut secara *default*.



Gambar 4.2. Konfigurasi *Hyperparameter* Secara Default

Adapun perintah pengkodean ini dengan baris kode sebagai berikut.

```
from tensorflow.keras import Model, Input  # Impor modul yang
diperlukan dari TensorFlow Keras
from tensorflow.keras.layers import LSTM, Embedding, Dense  #
Impor jenis layer yang akan digunakan
from tensorflow.keras.layers import TimeDistributed,
SpatialDropout1D, Bidirectional  # Impor jenis layer tambahan
yang akan digunakan

input_word = Input(shape=(max_len,))  # Mendefinisikan input
layer untuk model

# Membuat arsitektur model dengan menggunakan layer-layer yang
telah diimpor
model = Embedding(input_dim=num_words,
output_dim=50)(input_word)  # Layer embedding untuk representasi
kata
model = SpatialDropout1D(0.1)(model)  # Layer dropout spasial
untuk mengurangi overfitting
model = Bidirectional(LSTM(units=100, return_sequences=True,
recurrent_dropout=0.1))(model)  # Layer LSTM dengan dua arah
out = TimeDistributed(Dense(num_tags,
```

```
activation="softmax"))(model)  # Layer fully connected dengan
distribusi waktu
model = Model(input_word, out)  # Menginisialisasi model dengan
input dan output layer

from tensorflow.keras.callbacks import ModelCheckpoint,
EarlyStopping  # Impor callback yang akan digunakan
from livelossplot.tf_keras import PlotLossesCallback  # Impor
untuk plotting interaktif menggunakan livelossplot

model.compile(optimizer="adam",  # Mengkompilasi model dengan
pengoptimal Adam
              loss="sparse_categorical_crossentropy",  #
Menggunakan loss function sparse categorical crossentropy
              metrics=["accuracy"])  # Menggunakan metrik
akurasi untuk evaluasi model

# Mendefinisikan callbacks yang akan digunakan untuk memantau
proses pelatihan
chkpt = ModelCheckpoint("model_weights.weights.h5",
monitor='val_loss', verbose=1, save_best_only=True,
save_weights_only=True, mode='min')  # Menyimpan model dengan
bobot terbaik
early_stopping = EarlyStopping(monitor='val_accuracy',
min_delta=0, patience=1, verbose=0, mode='max', baseline=None,
restore_best_weights=False)  # Menghentikan pelatihan jika tidak
ada peningkatan

# Membuat list dari callbacks yang akan digunakan
callbacks = [PlotLossesCallback(), chkpt, early_stopping]
```

Kode ini bertujuan untuk membangun, melatih, dan mengevaluasi model jaringan saraf menggunakan TensorFlow Keras untuk tugas pemrosesan bahasa alami (NLP). Alur dimulai dengan definisi input layer menggunakan `Input` dari TensorFlow Keras, dengan `shape` yang sesuai dengan panjang maksimum dari kalimat yang akan diproses. Kemudian, arsitektur model dibangun dengan menggunakan layer-layer yang telah diimpor, termasuk `Embedding` untuk representasi kata, `SpatialDropout1D` untuk mengurangi overfitting, dan `Bidirectional LSTM` untuk memproses urutan secara dua arah. Output dari Bi-LSTM kemudian dihubungkan ke layer `TimeDistributed Dense` untuk memprediksi tag untuk setiap token dalam kalimat.

Selanjutnya, model dikompilasi dengan pengoptimal Adam dan fungsi loss 'sparse_categorical_crossentropy', serta metrik akurasi untuk evaluasi model. Callbacks juga ditentukan untuk memantau pelatihan, seperti 'ModelCheckpoint' untuk menyimpan model dengan bobot terbaik, dan 'EarlyStopping' untuk menghentikan pelatihan jika tidak ada peningkatan dalam metrik yang dimonitor.

Setelah pengaturan model, proses pelatihan dimulai dengan memanggil metode 'fit()' pada model. Data latih dan uji, batch size, jumlah epoch, dan callbacks yang telah ditentukan sebelumnya disertakan sebagai parameter. Selama pelatihan, grafik interaktif dari kerugian (loss) dan metrik akan ditampilkan secara langsung menggunakan 'PlotLossesCallback' dari livelossplot. Proses ini akan berlangsung selama jumlah epoch yang ditentukan, dan setelah selesai, model yang telah dilatih siap untuk digunakan untuk tugas pemrosesan bahasa alami, seperti pengenalan entitas dalam teks.

Pemodelan selanjutnya adalah dengan mengatur batasan-batasan *hyperparameter* yang digunakan yakni *units, dropout, embedding_dim, learning rate, batch size, layers, epoch* dan *kernel regularizer.*

Hyperparameter units merujuk pada jumlah *unit* atau neuron dalam lapisan Bi-LSTM atau *Dense* yang akan digunakan dalam model. Kemudian, *dropout* mengontrol tingkat *dropout* yang diterapkan pada model untuk mengurangi *overfitting*. embedding_dim adalah dimensi representasi kata dalam lapisan *embedding*. learning_rate adalah laju belajar yang mengontrol seberapa cepat model mengubah bobotnya selama pelatihan. batch_size menentukan jumlah sampel yang digunakan dalam satu iterasi pelatihan. epochs adalah jumlah iterasi

yang dilakukan selama pelatihan model. layers adalah jumlah lapisan dalam model, termasuk lapisan Bi-LSTM dan Dense. Terakhir, kernel_regularizer menentukan jenis regularisasi yang diterapkan pada bobot model, baik tidak ada regularisasi, regularisasi L1, atau regularisasi L2.

Proses mengatasi ketidakseimbangan data dalam konteks Named Entity Recognition (NER) lebih efektif dilakukan pada saat regularisasi model daripada melalui *preprocessing* karena regularisasi seperti dropout dan L2 regularization dapat membantu model untuk generalisasi lebih baik tanpa mengubah distribusi data alami. Manipulasi data melalui *oversampling* atau *undersampling* dapat memperkenalkan bias dan mengubah konteks data asli, yang bisa merugikan performa model dalam memahami pola dan konteks yang tepat. Sebaliknya, regularisasi menjaga integritas data dan membantu model fokus pada entitas yang kurang terwakili dengan menambahkan penalti pada kompleksitas model, sehingga mengurangi risiko *overfitting*. Menurut penelitian oleh Jie Yang dan Yue Zhang (2018), penggunaan regularisasi dalam model NER secara signifikan meningkatkan performa model tanpa perlu mengubah distribusi data melalui preprocessing.

Untuk penentuan epoch, dilakukan penyelidikan terhadap jumlah epoch dengan menerapkan *earlystop mechanism.* Saat model Bi-LSTM tidak memiliki performa secara signifikan maka *earlystop* akan menghentikan proses pembelajaran. Jumlah epoch dari pembelajaran akan diimplementasikan dalam koding Bi-LSTM berikutnya dan dievaluasi. Berikut cuplikan koding pasca epoch dari early stop diketahui.

```
# Melakukan pelatihan model dengan data yang telah disiapkan
sebelumnya
history = model.fit(
    x=x_train,
    y=y_train,
    validation_data=(x_test, y_test),
    batch_size=32,
    epochs=10,
    callbacks=callbacks,
    verbose=1
)
```

Hyperparameter yang disebutkan akan menjadi ruang pencarian (*Search Space*) dalam algoritma genetika. Pada tahapan perancangan model Bi-LSTM dan Algoritma Genetika, menggunakan beberapa *library* yang dideskripsikan dalam baris kode berikut.

```
# Import necessary modules
import numpy as np
import pandas as pd
import time
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
from tensorflow.keras.callbacks import ModelCheckpoint,
EarlyStopping
from livelossplot.tf_keras import PlotLossesCallback
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Embedding,
SpatialDropout1D, Bidirectional, LSTM, TimeDistributed, Dense
from tensorflow.keras.optimizers import Adam
```

Baris pertama dalam kode yang berfungsi untuk mengimpor semua modul yang diperlukan untuk membangun dan melatih model jaringan saraf menggunakan TensorFlow Keras. Modul numpy dan pandas digunakan untuk manipulasi data, time digunakan untuk mengukur waktu eksekusi, sedangkan train_test_split dan f1_score dari modul sklearn digunakan untuk pemrosesan data dan evaluasi kinerja model. Selanjutnya, modul ModelCheckpoint, EarlyStopping, dan PlotLossesCallback dari tensorflow.keras.callbacks diimpor untuk

mengaktifkan callback selama pelatihan model, seperti menyimpan model terbaik, menghentikan pelatihan jika tidak ada peningkatan, dan memantau kerugian dan metrik secara langsung selama pelatihan. Modul Model dari tensorflow.keras.models digunakan untuk mendefinisikan struktur model, sedangkan tensorflow.keras.layers digunakan untuk menambahkan layer-layer ke dalam model seperti layer embedding, dropout, Bi-LSTM, dan Dense. Terakhir, modul Adam dari tensorflow.keras.optimizers digunakan untuk mendefinisikan pengoptimal yang akan digunakan selama pelatihan model. Dengan mengimpor semua modul yang diperlukan, maka sistem siap untuk membangun dan melatih model jaringan saraf untuk tugas tertentu.

Selanjutnya mendefinisikan sebuah metode yang akan digunakan sebagai Bi-LSTM yang akan diimplementasikan dalam algoritma genetika sebagai berikut.

```
# Define your LSTM model creation and training within a function
def create_and_train_lstm_model(units, dropout, embedding_dim,
learning_rate, batch_size, epochs, layers, kernel_regularizer,
x_train, y_train, x_test, y_test):
    input_word = Input(shape=(max_len,))
    model = Embedding(input_dim=num_words,
output_dim=embedding_dim)(input_word)
    model = SpatialDropout1D(dropout)(model)

    for _ in range(layers - 1):
        model = Bidirectional(LSTM(units=units,
return_sequences=True, recurrent_dropout=dropout))(model)

    out = TimeDistributed(Dense(num_tags, activation="softmax",
kernel_regularizer=kernel_regularizer))(model)
    model = Model(input_word, out)

    optimizer = Adam(learning_rate=learning_rate)
    model.compile(optimizer=optimizer,
loss="sparse_categorical_crossentropy", metrics=["accuracy"])

    chkpt = ModelCheckpoint("model_weights.weights.h5",
monitor='val_loss', verbose=1, save_best_only=True,
save_weights_only=True, mode='min')
    early_stopping = EarlyStopping(monitor='val_accuracy',
min_delta=0, patience=1, verbose=0, mode='max', baseline=None,
```

```
restore_best_weights=False)

    start_time = time.time()
    history = model.fit(
        x=x_train,
        y=y_train,
        validation_data=(x_test, y_test),
        batch_size=batch_size,
        epochs=epochs,
        callbacks=[PlotLossesCallback(), chkpt, early_stopping],
        verbose=1
    )
    end_time = time.time()

    y_pred = model.predict(x_test)
    y_pred_argmax = np.argmax(y_pred, axis=-1)

    accuracy = model.evaluate(x_test, y_test)[1]
    f1 = f1_score(y_test.flatten(), y_pred_argmax.flatten(),
average='weighted')

    return accuracy, f1, history.history['loss'][-1],
history.history['accuracy'][-1], history.history['val_loss'][-
1], history.history['val_accuracy'][-1], end_time - start_time
```

Fungsi 'create_and_train_lstm_model' dibuat untuk membuat dan melatih model Bi-LSTM. Fungsi ini menerima sejumlah hyperparameter seperti jumlah unit Bi-LSTM, dropout, dimensi embedding, laju belajar, ukuran batch, jumlah epoch, jumlah lapisan, dan regularisasi kernel. Pertama, input layer untuk model Bi-LSTM didefinisikan sesuai dengan panjang maksimum dari kalimat yang akan diproses. Kemudian, layer embedding dan dropout diterapkan pada model. Selanjutnya, model Bi-LSTM dibuat dengan menggunakan jumlah unit dan dropout yang diberikan, dan jumlah lapisan Bi-LSTM ditentukan oleh parameter 'layers'. Output dari Bi-LSTM dihubungkan ke layer TimeDistributed Dense untuk melakukan klasifikasi untuk setiap token dalam kalimat. Model kemudian dikompilasi dengan menggunakan pengoptimal Adam dengan laju belajar yang telah ditentukan, serta fungsi loss sparse categorical crossentropy dan metrik

akurasi. Callbacks juga ditambahkan untuk memantau pelatihan model, termasuk `PlotLossesCallback` untuk memantau kerugian dan metrik secara langsung selama pelatihan. Setelah pelatihan selesai, waktu eksekusi, akurasi, dan skor F1 dari model dihitung dan dikembalikan bersama dengan beberapa statistik lainnya. Dengan fungsi ini, proses pembuatan dan pelatihan model Bi-LSTM dapat dilakukan secara fleksibel dengan mengatur hyperparameter yang sesuai.

Selanjutnya akan mendefinisikan fungsi *fitness* (terjemahan : kebugaran) yang akan digunakan sebagai evaluasi dalam setiap generasi dari eksekusi algoritma genetika. Berikut adalah baris kodenya.

```
# Define the custom fitness function
def custom_fitness(accuracy, f1):
    # You can modify this function based on your specific custom
fitness criteria
    return 2 * ((accuracy * f1)/(f1+accuracy))
```

Fungsi 'custom_fitness' ditentukan sebagai fungsi kebugaran khusus yang memanfaatkan akurasi dan skor F1 sebagai masukan. Tujuan dari fungsi ini adalah untuk menghasilkan skor kebugaran yang mengevaluasi performa model berdasarkan kriteria kustom tertentu. Dalam implementasi saat ini, fungsi menghitung skor kebugaran dengan menggunakan formula yang menggabungkan akurasi dan skor F1. Nilai kembalian dari fungsi ini adalah hasil dari perhitungan tersebut. Namun, fungsi ini dapat dimodifikasi sesuai dengan kriteria kebugaran khusus yang diperlukan untuk tujuan tertentu. Dengan cara ini, skor kebugaran dapat disesuaikan dengan kebutuhan spesifik dalam mengevaluasi dan memilih model terbaik.

Bagian algoritma genetika akan diimplementasikan dalam definisi sebuah metode. Berikut adalah baris kodenya.

```
# Define the genetic algorithm
def genetic_algorithm():
    population_size = 25
    generations = 50

    # Define the search space for hyperparameters
    search_space = {
        'units': [50, 100, 150],
        'dropout': [0.1, 0.2, 0.3],
        'embedding_dim': [50, 100, 150],
        'learning_rate': [0.001, 0.01, 0.1],
        'batch_size': [16, 32, 64],
        'epochs': [5, 10, 15],
        'layers': [1, 2, 3],
        'kernel_regularizer': [None, 'l1', 'l2'],
        # Add more hyperparameters and their ranges here
    }

    # Generate initial population with random
hyperparameters
    population = [{param: np.random.choice(values) for
param, values in search_space.items()} for _ in
range(population_size)]

    best_hyperparameters = None
    best_performance = 0.0
    history_per_generation = []

    for gen in range(generations):
        print(f"Generation {gen + 1} / {generations}")

        # Evaluate fitness for each individual in the
population
        fitness_scores =
[create_and_train_lstm_model(individual['units'],
individual['dropout'], individual['embedding_dim'],

individual['learning_rate'], individual['batch_size'],
individual['epochs'],

individual['layers'], individual['kernel_regularizer'],

x_train, y_train, x_test, y_test) for individual in
population]

        # Calculate custom fitness scores
```

```
        custom_fitness_scores = [custom_fitness(accuracy,
f1) for accuracy, f1, _, _, _, _, _ in fitness_scores]

        # Select top individuals based on their custom
fitness scores
        top_individuals =
sorted(range(len(custom_fitness_scores)), key=lambda k:
custom_fitness_scores[k],
reverse=True)[:int(population_size * 0.2)]

        # Update best hyperparameters and performance
        best_index = top_individuals[0]
        current_performance =
custom_fitness_scores[best_index]
        if current_performance > best_performance:
            best_performance = current_performance
            best_hyperparameters = population[best_index]

        # Store history of the best hyperparameters and
their performance
        history_per_generation.append({
            "generation": gen + 1,
            "best_hyperparameters": best_hyperparameters,
            "best_performance": best_performance,
            "hyperparameters_performance":
[{"hyperparameters": population[i], "performance":
custom_fitness_scores[i],
                                                "accuracy":
fitness_scores[i][0], "f1": fitness_scores[i][1],
                                                "loss":
fitness_scores[i][2], "accuracy_val":
fitness_scores[i][3],
                                                "loss_val":
fitness_scores[i][4], "accuracy": fitness_scores[i][5],
                                                "time":
fitness_scores[i][6]}
                                               for i in
range(population_size)]
        })

        # Crossover and mutation to create new individuals
        new_population = []
        for _ in range(population_size):
            parent1 =
population[np.random.choice(top_individuals)]
            parent2 =
population[np.random.choice(top_individuals)]

            # Apply crossover and mutation to create a new
individual
```

```
            child = {}
            for param in search_space:
                if np.random.rand() > 0.5:
                    child[param] = parent1[param]
                else:
                    child[param] = parent2[param]

                if np.random.rand() < 0.1:  # Mutation
                    child[param] =
np.random.choice(search_space[param])

            new_population.append(child)

        population = new_population

    return history_per_generation
```

Fungsi 'genetic_algorithm' didefinisikan untuk melaksanakan algoritma genetika dengan tujuan mencari kombinasi hyperparameter terbaik yang menghasilkan model dengan kinerja tertinggi berdasarkan skor kebugaran yang ditentukan. Algoritma dimulai dengan mendefinisikan ukuran populasi dan jumlah generasi yang akan dievaluasi. Kemudian, ruang pencarian hyperparameter ditentukan dalam variabel 'search_space', yang mencakup sejumlah hyperparameter yang akan dieksplorasi bersama dengan rentang nilai-nilainya.

Penerapan epoch dibatasi dengan angka 5, 10 dan 15. Alasan spesifiknya karena pada *preliminary experiment* ditemukan bahwa epoch terhenti pada epoch ke tiga ketika nilai *patience* = 1, terhenti pada epoch ke 7 ketika nilai patience = 5, dan terhenti pada angka 65 ketika patience = 10. Angka 5,10, dan 15 ditentukan secara acak dan tidak berdasarkan jumlah pada early stopping.

Populasi awal dibangkitkan dengan mengambil sampel acak dari ruang pencarian hyperparameter. Setiap individu dalam populasi dievaluasi dengan

menggunakan fungsi `create_and_train_lstm_model`, yang membuat dan melatih model Bi-LSTM dengan hyperparameter yang diberikan, serta menghitung kinerja model berdasarkan skor kebugaran yang ditentukan. Setelah evaluasi, individu-individu terbaik dipilih berdasarkan skor kebugaran mereka.

Selanjutnya, proses crossover dan mutasi diterapkan untuk menciptakan generasi baru dari individu-individu. Dua orang tua dipilih secara acak dari individu-individu terbaik, dan operasi crossover digunakan untuk menciptakan anak dengan menggabungkan atribut-atribut dari kedua orang tua. Selain itu, terdapat peluang mutasi untuk setiap atribut dari anak, yang memungkinkan perubahan acak pada nilai hyperparameter.

Proses ini diulangi selama sejumlah generasi yang telah ditentukan, dan setiap kali, individu-individu terbaik dan kinerja mereka direkam. Setelah selesai, hasil dari algoritma genetika berupa histori per generasi dari kombinasi hyperparameter terbaik dan kinerjanya dikembalikan. Dengan menggunakan algoritma genetika ini dapat menemukan kombinasi hyperparameter yang optimal untuk model Bi-LSTM berdasarkan kriteria evaluasi yang telah ditentukan.

Meskipun sudah ada algoritma dengan kemampuan adaptif untuk menentukan jumlah populasi, ini tidak menjadikan proses determinasi ukuran populasi secara acak diabaikan (Lobo dan Lima, 2007). Ukuran populasi tidak dapat ditentukan secara praktis kuantitasnya, sehingga perlu skema pembanding antar satu populasi dalam sebuah skema dengan yang lainnya (Eiben et al., 1999). Algoritma adaptasi adaptif lebih baik digunakan dalam sebuah skema

berkelanjutan sehingga dapat ditemukan konvergensi dari pencarian optimal algoritma genetika.

Eksperimen ini dilakukan dengan empat skema yang dirancang untuk mengevaluasi pengaruh ukuran populasi dan jumlah generasi dalam kinerja GA-Bi-LSTM dalam mendeteksi entitas teks kebakaran hutan. Ukuran populasi kecil (10 individu) memiliki waktu eksekusi cepat tetapi risiko konvergensi prematur tinggi (Reeves, 1993). Ukuran menengah (25 individu) memberikan keseimbangan antara diversitas dan kecepatan konvergensi (Eiben et al., 1999). Ukuran besar (50 individu) memungkinkan eksplorasi lebih luas tetapi dengan waktu komputasi lebih lama (Goldberg, 2006; Lobo dan Lima, 2005). Generasi yang lebih banyak memungkinkan pencarian lebih mendalam, meningkatkan peluang menemukan solusi optimal meskipun membutuhkan lebih banyak waktu komputasi (Lobo & Lima, 2005). Adapun eksperimen ini dilakukan dengan empat skema eksperimen yang ditunjukkan dalam Tabel 4.2. sebagai berikut.

Tabel 4.2. Skema Eksperimen Penelitian

No	Algoritma	Populasi	Generasi	Evaluasi
1	*Default* Bi-LSTM	-	-	1
2	GA - Bi-LSTM	10	25	250
3	GA - Bi-LSTM	25	50	1250
4	GA - Bi-LSTM	50	20	1000

Tabel 4.2. menunjukkan empat skema eksperimen yang dilakukan untuk membandingkan kinerja model Bi-LSTM default dengan model Bi-LSTM yang dioptimalkan menggunakan algoritma genetika (GA). Eksperimen ini dilakukan

hanya satu kali uji coba dengan mempertimbangkan keterbatasan komputasi dan waktu yang diperlukan sangat lama.

Eksperimen pertama, "Default Bi-LSTM," merupakan kontrol di mana model Bi-LSTM dibangun dengan menggunakan konfigurasi default tanpa penggunaan GA. Tidak ada populasi atau generasi yang dihasilkan dalam eksperimen ini, karena ini hanya menunjukkan kinerja model default.

Eksperimen berikutnya dilakukan dengan menggunakan GA untuk mengoptimalkan hyperparameter model Bi-LSTM. Dalam ketiga skema eksperimen selanjutnya, model Bi-LSTM dievaluasi menggunakan populasi dengan ukuran yang berbeda, yaitu 25, 10, dan 50 individu. Setiap populasi dievaluasi selama sejumlah generasi yang telah ditentukan, yaitu 50, 25, dan 20 generasi berturut-turut. Jumlah total evaluasi model dalam setiap skema eksperimen dihitung dengan mengalikan jumlah populasi dengan jumlah generasi.

Dengan menggunakan GA, skema eksperimen ini bertujuan untuk mencari kombinasi hyperparameter terbaik yang menghasilkan model dengan kinerja tertinggi berdasarkan skor kebugaran yang ditentukan. Dengan membandingkan hasil dari masing-masing skema eksperimen, penelitian ini dapat mengevaluasi apakah penggunaan GA berhasil meningkatkan kinerja model Bi-LSTM dalam tugas pemrosesan bahasa alami yang spesifik yang sedang diamati.

4.2. Pelaksanaan Aksi (*Action Taking)*

Sub bab pelaksanaan aksi berisikan eksekusi dan implementasi rancangan pada tahap perencanaan aksi (*action planning*). Pada sub-bab ini bagian-bagian eksekusi terdiri dari implementasi dari ketiga skema di tiap bahasannya. Pada

bagian ini terdapat tahap data splitting dengan komposisi 80% untuk data training dan 20% data testing untuk setiap skemannya.

Hasil dari pencarian *hyperparameter* terbaik dilampirkan pada lampiran. Adapun untuk memberikan gambaran, sekilas dari setiap pencarian ditampilkan dalam bentuk cuplikan tabel di tiap sub-bab.

4.2.1. *Preliminary Experiment :* Penentuan Jumlah Epoch

Pada tahap awal eksperimen dilakukan serangkaian percobaan untuk menentukan jumlah epoch yang optimal dalam pelatihan model Bi-LSTM. Sebuah model Bi-LSTM dianggap telah menguasai suatu dataset ketika performanya tidak menunjukkan peningkatan yang signifikan dari iterasi pelatihan sebelumny.

Untuk mencapai tujuan ini dilakukan pendekatan *early stopping*. Pendekatan early stopping memungkinkan untuk menghentikan proses pelatihan ketika performa model tidak lagi meningkat secara signifikan di setiap epoch baru. Dengan kata lain, ketika nilai kerugian (loss) di set validasi tidak lagi menurun atau bahkan mulai meningkat, itu menjadi tanda bahwa model telah mencapai titik di mana ia tidak lagi belajar secara signifikan dari data.deng. Early stopping juga dapat memberi gambaran pada grafik Bi-LSTM penguasaannya secara general. Upaya ini juga dapat meringankan biaya komputasi (Dodge dkk., 2020). Berikut adalah baris kode untuk eksperimen *preliminary* ini.

```
from tensorflow.keras.callbacks import ModelCheckpoint,
EarlyStopping
%%time

chkpt = ModelCheckpoint("model_weights.h5",
monitor='val_loss',verbose=1, save_best_only=True,
save_weights_only=True, mode='min')
early_stopping = EarlyStopping(monitor='val_accuracy',
```

```
min_delta=0, patience=10, verbose=0, mode='max',
baseline=None, restore_best_weights=False)

callbacks = [PlotLossesCallback(), chkpt, early_stopping]

history = model.fit(
    x=x_train,
    y=y_train,
    validation_data=(x_test,y_test),
    batch_size=32,
    epochs=200,
    callbacks=callbacks,
    verbose=1 )
```

Dalam melakukan preliminary experiment dataset dibagi menjadi data pelatihan dan data validasi. Model Bi-LSTM dilatih pada data pelatihan dan memonitor performanya pada data validasi setiap epoch. Penghentian dini (early stopping) diimplementasikan dengan memantau perubahan dalam performa model pada data validasi dari satu epoch ke epoch berikutnya.

Hasil eksperimen menunjukkan bahwa proses pelatihan terhenti pada epoch ke-3 ketika nilai patience adalah 1, pada epoch ke-7 ketika nilai patience adalah 5, dan pada epoch ke-65 ketika nilai patience adalah 10. Meskipun demikian, angka-angka ini tidak dijadikan batasan langsung dalam penerapan praktis. Hasil ini memberikan wawasan penting bahwa model mencapai puncak performa dalam jumlah epoch yang bervariasi tergantung pada kondisi pelatihan. Dengan demikian, pemilihan batas maksimum jumlah epoch dalam pelatihan model didasarkan pada hasil eksperimen ini untuk memastikan penggunaan yang efisien dari sumber daya komputasi tanpa mengorbankan kualitas hasil.

Dari eksperimen ini tidak dapat menjadi pertimbangan jumlah *epoch* dalam parameter *searching space* karena terdapat perbedaan signifikan jumlah *epoch* dalam percobaan jumlah patience yang berbeda. Sehingga dalam

implementasi selanjutnya jumlah *epoch* diterapkan pada *seacing space* secara acak antara 5,10, atau 15. Sehingga model diasumsikan dengan jumlah epoch tersbut sudah *signifikan* dan mempertimbangkan *hyperparameter lainnya.* Hal ini tidak menjadi permasalahan dalam efektifitas algoritma genetika untuk mencari hyperparameter terbaik, namun menjadi penyelidikan selanjutnya terkait dataset yang digunakan.

4.2.2. *Preliminary Experiment :* Implementasi Algoritma Bi-LSTM dengan Hyperparameter *Default* sebagai Kontrol

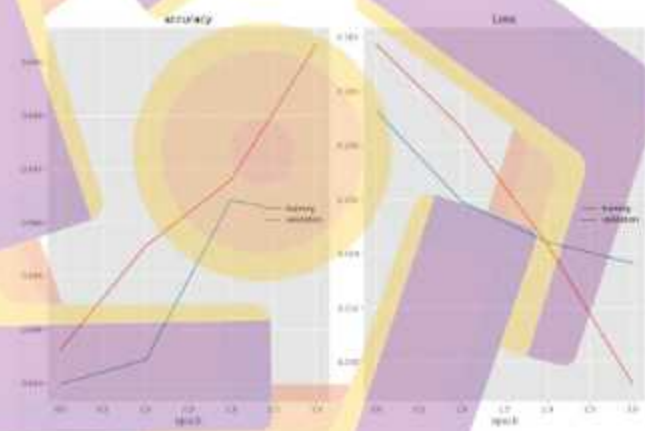
Skema eksperimen pertama mengimplementasikan model Bidirectional Long Short-Term Memory (Bi-LSTM) secara Bidireksional dengan parameter *default* dari *coding* dan rancangan *(action planning)* untuk melakukan deteksi terhadap entitas waktu dan lokasi yang berkaitan dengan kebakaran hutan. Implementasi model ini melibatkan penyesuaian parameter dan iterasi berulang guna mengoptimalkan kinerja model. Hasil dari proses pelatihan menunjukkan evolusi akurasi dan kerugian (loss) pada kumpulan data pelatihan dan validasi selama 10 iterasi (epochs). Berikut pada Gambar 4.3 merupakan cuplikan implementasi dalam IDE JupyterNotebook.

Gambar 4.3. Implementasi Koding Rancangan dalam IDE Jupyter Notebook

Pada tahap pelatihan, teramati bahwa akurasi pada kumpulan data pelatihan meningkat secara signifikan seiring dengan berjalannya iterasi. Dimulai dari 94.3% pada epoch pertama, akurasi terus meningkat hingga mencapai puncaknya pada 94.9% pada epoch terakhir. Sementara itu, akurasi pada kumpulan data validasi mencapai 94.3%, dan tetap stabil di sekitar 94.6% pada akhir proses pelatihan.

Kurva kerugian *(loss)* pada kumpulan data pelatihan menunjukkan penurunan yang konsisten selama seluruh iterasi, mencapai nilai minimum pada 0.150. Hal ini mencerminkan kemampuan model dalam mengurangi tingkat kesalahan pada data pelatihan. Fenomena serupa juga terjadi pada kurva kerugian pada kumpulan data validasi, yang menurun hingga mencapai nilai minimum di bawah 0.154. Ini menunjukkan bahwa model mampu melakukan generalisasi dengan baik terhadap data yang tidak pernah dilihat sebelumnya.

Ketika model dievaluasi pada data uji, ditemukan bahwa kerugian pada data uji adalah sebesar 0.154, sementara akurasi mencapai 94.6%. Ini memberikan gambaran tentang seberapa baik model dapat melakukan generalisasi informasi dari data pelatihan ke data yang baru. Visualisasi dari metrik-metrik ini dapat ditemukan pada Gambar 4.4, yang menggambarkan perubahan akurasi dan kerugian selama proses pelatihan dan evaluasi model. Hasil eksperimen ini memberikan wawasan mendalam tentang kemampuan model Jaringan Syaraf Tiruan Berbasis LSTM secara Bidireksional dalam mengenali entitas terkait kebakaran hutan.



Gambar 4.4. Grafik Performa Model

Grafik pada Gambar 4.4 merepresentasikan secara visual metrik-metrik tersebut selama proses pelatihan dan evaluasi model, yang membantu dalam pemahaman yang lebih baik terhadap kinerja model secara keseluruhan. Selanjutnya model dievaluasi dengan taraf F1 Score, hasil F1 Score menunjukkan sebesar 90.17%. Pada preleminary experiment untuk menentukan epoch, tidak ada

signifikansi yang berarti pada *epoch* ke 3, sehingga uji Bi-LSTM kontrol untuk pembanding algoritma genetika ini hanyalah gambaran dan hasil *default.*

4.2.3. Implementasi Algoritma Genetika pada Bi-LSTM (Skema 2-4)

Implementasi algoritma genetika pada model Bidirectional Long Short-Term Memory (Bi-LSTM) secara Bidireksional dalam konteks deteksi entitas terkait kebakaran hutan dilakukan dari skema eksperimen kedua hingga keempat. Algoritma genetika diterapkan pada skema ini untuk menemukan kombinasi *hyperparameter* optimal yang menghasilkan model dengan kinerja terbaik, berdasarkan skor kebugaran yang telah ditetapkan sebelumnya. Perbandingan generasi terbaik di tiap skema ditunjukkan pada Tabel 4.3.

Tabel 4.3. Hasil Konfigurasi *Hyperparameter* terbaik oleh Algoritma Genetika

No.	Konfigurasi *Hyperparameter* terbaik	*Fitness Rate*	*F1 Score*	*Loss*	Waktu Komputasi (detik)
1	{'units': 50, 'dropout': 0.3, 'embedding_dim': 50, 'learning_rate': 0.1, 'batch_size': 64, 'epochs': 15, 'layers': 2, 'kernel_regularizer': None}	97.45%	96.75%	0.025	9.77
2	{'units': 150, 'dropout': 0.2, 'embedding_dim': 150, 'learning_rate': 0.01, 'batch_size': 16, 'epochs': 10, 'layers': 2, 'kernel_regularizer': None}	97.72%	93.76%	0.169	29.49
3	{'units': 150, 'dropout': 0.1, 'embedding_dim': 150, 'learning_rate': 0.01, 'batch_size': 16, 'epochs': 15, 'layers': 2, 'kernel_regularizer':None}	97.74%	97.36	0.03	32.77

Pada skema kedua (No. 1 pada Tabel 4.3), digunakan populasi sebesar 10 individu dengan 25 generasi evaluasi, menghasilkan total 1250 kali evaluasi model. Hasil evaluasi pada generasi ke-10 menunjukkan bahwa kombinasi hyperparameter terbaik mencapai akurasi sebesar 97.15%, dengan skor F1 sebesar 96.75% dan loss sebesar 0.025. Selain itu, akurasi pada data validasi mencapai 99.13%, dengan loss sebesar 0.12. Proses pelatihan model dengan kombinasi hyperparameter terbaik memakan waktu sekitar 9.77 detik.

Skema ketiga dan keempat mengimplementasikan algoritma genetika dengan variasi dalam ukuran populasi dan jumlah generasi evaluasi. Pada skema ketiga, dilakukan evaluasi pada generasi ke-50 dengan populasi sebesar 25 individu, menghasilkan total 1250 kali evaluasi model. Hasil evaluasi menunjukkan bahwa model dengan kombinasi hyperparameter terbaik mencapai akurasi sebesar 94.36%, dengan skor F1 sebesar 93.76% dan loss sebesar 0.169. Pada skema keempat, dilakukan evaluasi pada generasi ke-20 dengan populasi sebesar 50 individu, menghasilkan total 1000 kali evaluasi model. Hasil evaluasi menunjukkan peningkatan kinerja, dengan model terbaik mencapai akurasi sebesar 97.74%, dengan skor F1 sebesar 97.36% dan loss sebesar 0.03.

Analisis ini membuktikan bahwa algoritma genetika mampu meningkatkan kinerja model Bi-LSTM dalam penyetelan *hyperparameter*, terutama dalam hal akurasi dan generalisasi terhadap data validasi. Variasi dalam ukuran populasi dan jumlah generasi evaluasi juga memengaruhi efisiensi dan kualitas hasil algoritma genetika. Oleh karena itu, penyetelan *hyperparameter*

dengan algoritma genetika merupakan pendekatan efektif untuk meningkatkan kinerja model dalam tugas pemrosesan bahasa alami.

4.2.4. Komparasi Hasil Performa Hyperparameter

Pada Tabel 4.4. ditampilkan hasil dari hyperparameter Bi-LSTM dari skema eksperimen pertama hingga keempat, di mana yang pertama adalah secara *default* dan kedua hingga keempat adalah *hyperparameter* yang sudah dioptimasi algoritma genetika.

Tabel 4.4. Hasil Komputasi Menggunakan empat skema *Hyperparameter*

Skema No	units	dropout	embedding_dim	learning_rate	batch_size	epochs	layers	kernel_regularizer	Loss	Akurasi	F1 Score	Waktu Komputasi (detik)
1	100	0.1	50	0.001	32	10	2	None	0.158	0.937	0.934	25.055
2	50	0.3	50	0.1	64	15	2	None	0.242	0.941	0.946	21.666
3	150	0.2	150	0.01	16	10	2	None	0.151	0.958	0.963	44.553
4	150	0.1	150	0.01	16	15	2	None	0.165	0.959	0.965	62.080

Meskipun performa model pada keempat skema hyperparameter tidak menunjukkan perubahan yang signifikan, tetapi tetap mengalami peningkatan dalam akurasi, *loss*, dan *F1 Score*. Meskipun tidak ada perubahan yang signifikan dalam hyperparameter, model masih mampu meningkatkan performanya. Namun, perlu diperhatikan bahwa waktu komputasi yang diperlukan cenderung tidak stabil, dengan kecenderungan peningkatan seiring dengan peningkatan akurasi dan F1 Score. Sebuah temuan menarik adalah bahwa skema kedua, meskipun memiliki *loss* terbesar, membutuhkan waktu komputasi yang paling pendek. Fenomena ini mungkin menunjukkan *trade-off* antara kecepatan komputasi dan kinerja model yang lebih baik.

Untuk memastikan keandalan dan konsistensi model, diperlukan validasi lanjutan seperti *k-fold cross validation*. *K-fold cross validation* membagi data menjadi beberapa subset dan melatih model pada subset-subset tersebut. Kinerja model kemudian diuji pada subset yang tidak digunakan dalam pelatihan. Proses ini diulang beberapa kali untuk memastikan bahwa kinerja model konsisten di berbagai sub-kelompok data. Dengan demikian, *k-fold cross validation* membantu memastikan bahwa model yang dihasilkan dapat digeneralisasikan dengan baik pada data baru (Anguita, D., Ghelardoni, L., Ghio, A., Oneto, L., & Ridella, S., 2012, April).

4.2.5. Uji Model dengan K-Fold Cross Validation

Untuk memahami keandalan, konsistensi, dan generalisasi model, dilakukan uji *K-Fold Cross Validation* dengan dua nilai K (5 dan 10). Pada K=5, data dibagi menjadi lima subset dan model dilatih dan diuji lima kali dengan

setiap subset sebagai set pengujian. Hal ini memberikan evaluasi rinci kinerja model di berbagai bagian data. Pada K=10, data dibagi menjadi sepuluh subset dan model dilatih dan diuji sepuluh kali dengan setiap subset sebagai set pengujian. Nilai K yang lebih besar menghasilkan estimasi performa model yang lebih akurat dan kemampuan generalisasinya ke data baru. Uji K-Fold Cross Validation ini memberikan wawasan komprehensif tentang keandalan, konsistensi, dan generalisasi model. Hasil uji K=5 disajikan pada Tabel 4.5.

Tabel 4.5. Uji Silang (Cross Validation) dengan K = 5

Skema No	units	dropout	embedding_dim	learning_rate	batch_size	epochs	layers	kernel_regularizer	Loss	Akurasi	F1 Score	Waktu Komputasi (detik)	fold
1	50	0.3	50	0.1	64	15	2	None	0.221	0.936	0.95	27.42	1
1	50	0.3	50	0.1	64	15	2	None	0.226	0.945	0.95	25.79	2
1	50	0.3	50	0.1	64	15	2	None	0.117	0.965	0.97	22.16	3
1	50	0.3	50	0.1	64	15	2	None	0.247	0.943	0.95	24.62	4
1	50	0.3	50	0.1	64	15	2	None	0.136	0.956	0.96	21.28	5
2	150	0.2	150	0.01	16	10	2	None	0.253	0.937	0.95	60.06	1
2	150	0.2	150	0.01	16	10	2	None	0.235	0.947	0.95	58.71	2
2	150	0.2	150	0.01	16	10	2	None	0.083	0.978	0.98	48.34	3
2	150	0.2	150	0.01	16	10	2	None	0.167	0.963	0.96	51.74	4
2	150	0.2	150	0.01	16	10	2	None	0.173	0.958	0.96	45.75	5
3	150	0.1	150	0.01	16	15	2	None	0.264	0.942	0.95	78.60	1
3	150	0.1	150	0.01	16	15	2	None	0.199	0.954	0.96	65.43	2
3	150	0.1	150	0.01	16	15	2	None	0.101	0.979	0.98	75.71	3
3	150	0.1	150	0.01	16	15	2	None	0.175	0.962	0.96	63.39	4
3	150	0.1	150	0.01	16	15	2	None	0.163	0.960	0.96	62.09	5

Tabel 4.5. Lanjutan													
4	100	0.1	50	0.01	32	10	2	None	0.247	0.937	0.95	29.90	1
4	100	0.1	50	0.01	32	10	2	None	0.227	0.937	0.94	28.73	2
4	100	0.1	50	0.01	32	10	2	None	0.119	0.969	0.97	31.25	3
4	100	0.1	50	0.01	32	10	2	None	0.173	0.955	0.96	26.10	4
4	100	0.1	50	0.01	32	10	2	None	0.140	0.955	0.96	27.56	5

Tabel 4.5 menampilkan hasil uji silang dengan K−5 untuk mengevaluasi performa empat skema hyperparameter model Bi-LSTM. Metrik performa yang dianalisis adalah F1 Score, Loss, dan Waktu Komputasi. Secara keseluruhan, Skema 2 (dengan hyperparameter dioptimasi algoritma genetika) menghasilkan F1 Score rata-rata tertinggi (0.961) dan loss rata-rata terendah (0.180). Skema 3 (dioptimasi genetika) mengikuti dengan F1 Score rata-rata 0.962 dan loss rata-rata 0.182. Skema 1 (default) dan Skema 4 (dioptimasi genetika) memiliki performa yang serupa dengan F1 Score rata-rata sekitar 0.954 dan loss rata-rata sekitar 0.189.

Terdapat variasi performa antar fold untuk setiap skema. Contohnya, Skema 2 memiliki F1 Score terbaik 0.980 (fold 3) dan terendah 0.943 (fold 4). Variasi ini menunjukkan pengaruh pembagian data pada performa model. Skema 3 memiliki waktu komputasi rata-rata tertinggi (69.04 detik), diikuti Skema 2 (52.92 detik) dan Skema 1 (24.25 detik). Skema 4 memiliki waktu komputasi rata-rata tercepat (27.56 detik). Kesimpulannya, Skema 2 dan 3 (dioptimasi genetika) menunjukkan performa terbaik dalam F1 Score dan Loss. Namun, Skema 3 memiliki waktu komputasi yang lebih tinggi. Skema 4 memiliki waktu komputasi tercepat, namun performanya sedikit lebih rendah.

Berdasarkan tabel 4.5, terlihat bahwa Skema 2 (dioptimalkan dengan algoritma genetika) menunjukkan performa yang lebih baik dibandingkan Skema 1 (tanpa algoritma genetika) dalam hal akurasi (F1 Score) dan efisiensi (loss). Skema 2 memiliki F1 Score rata-rata 0.961 dan loss rata-rata 0.182, sedangkan Skema 1 memiliki F1 Score rata-rata 0.954 dan loss rata-rata 0.189. Hal ini

menunjukkan bahwa Skema 2 lebih akurat dalam memprediksi kategori teks dan lebih baik dalam meminimalkan kesalahan prediksi. Namun, Skema 2 membutuhkan waktu komputasi rata-rata 52.92 detik, dua kali lipat dibandingkan Skema 1 yang hanya 24.25 detik. Peningkatan performa Skema 2 menunjukkan bahwa optimasi hyperparameter dengan algoritma genetika efektif, namun perlu dipertimbangkan trade-off antara akurasi, efisiensi, dan waktu komputasi. Skema 3 dan 4 juga dioptimalkan dengan algoritma genetika. Skema 3 memiliki F1 Score rata-rata 0.962 dan loss rata-rata 0.180, yang sedikit lebih baik dibandingkan Skema 2. Namun, Skema 3 membutuhkan waktu komputasi rata-rata 69.04 detik, jauh lebih lama dibandingkan Skema 2 dan Skema 4. Skema 4 memiliki F1 Score rata-rata 0.954 dan loss rata-rata 0.181, sedikit lebih rendah dibandingkan Skema 2, namun memiliki waktu komputasi tercepat, yaitu 28.71 detik. Skema 3 menunjukkan performa terbaik dalam F1 Score dan Loss, namun dengan trade-off waktu komputasi yang paling lama. Skema 4 memiliki waktu komputasi tercepat, namun performanya sedikit lebih rendah dibandingkan Skema 2.

Setelah menunjukkan peningkatan performa dengan Skema 2 (dioptimasi genetika) pada K=5, pengujian lanjutan dilakukan dengan K=10. Tujuannya untuk memperkuat validitas hasil, meningkatkan akurasi estimasi performa, dan memungkinkan perbandingan antar skema dengan lebih detail. Dengan K=10, model dievaluasi pada lebih banyak subset data, meningkatkan kepercayaan pada hasil dan mengurangi kemungkinan overfitting. K=10 juga menghasilkan estimasi F1 Score, Loss, dan Waktu Komputasi yang lebih stabil dan mendekati performa model pada data baru. Pengujian K=10 akan memberikan informasi yang lebih

komprehensif tentang skema *hyperparameter* terbaik untuk model bi-LSTM. Meskipun K−10 membutuhkan waktu komputasi yang lebih lama dibandingkan K−5, K−10 dipilih untuk menyeimbangkan kebutuhan akurasi dan efisiensi komputasi. Hasil pengujian K−10 akan disajikan pada tabel selanjutnya. Pengujian K−10 ini penting untuk memastikan bahwa Skema 2 (dioptimasi genetika) memang memberikan performa terbaik dan dapat digeneralisasikan ke data baru dengan baik. Tabel 4.6 merupakan uji K Fold Cross Validation dengan K − 10.

Tabel 4.6. Uji Silang (Cross Validation) dengan K = 10

Skema No	units	dropout	embedding_dim	learning_rate	batch_size	epochs	layers	kernel_regularizer	loss	Akurasi	F1 Score	Waktu Komputasi (detik)	fold
1	50	0.3	50	0.1	64	15	2	None	0.291	0.920	0.929	26.672	1
1	50	0.3	50	0.1	64	15	2	None	0.162	0.961	0.964	26.023	2
1	50	0.3	50	0.1	64	15	2	None	0.295	0.927	0.923	22.387	3
1	50	0.3	50	0.1	64	15	2	None	0.126	0.968	0.978	24.079	4
1	50	0.3	50	0.1	64	15	2	None	0.080	0.975	0.980	21.888	5
1	50	0.3	50	0.1	64	15	2	None	0.080	0.976	0.974	22.140	6
1	50	0.3	50	0.1	64	15	2	None	0.148	0.955	0.957	28.001	7
1	50	0.3	50	0.1	64	15	2	None	0.260	0.941	0.945	25.742	8
1	50	0.3	50	0.1	64	15	2	None	0.213	0.936	0.937	26.958	9
1	50	0.3	50	0.1	64	15	2	None	0.077	0.973	0.973	26.392	10
2	150	0.2	150	0.01	16	10	2	None	0.324	0.926	0.940	67.751	1
2	150	0.2	150	0.01	16	10	2	None	0.130	0.967	0.969	57.658	2
2	150	0.2	150	0.01	16	10	2	None	0.259	0.940	0.939	48.063	3

Tabel 4.6. Lanjuan

2	150	0.2	150	0.01	16	10	2	None	0.083	0.983	0.983	55.961	4
2	150	0.2	150	0.01	16	10	2	None	0.061	0.974	0.980	48.762	5
2	150	0.2	150	0.01	16	10	2	None	0.077	0.972	0.976	47.888	6
2	150	0.2	150	0.01	16	10	2	None	0.133	0.963	0.967	60.422	7
2	150	0.2	150	0.01	16	10	2	None	0.259	0.944	0.949	62.583	8
2	150	0.2	150	0.01	16	10	2	None	0.180	0.957	0.959	60.772	9
2	150	0.2	150	0.01	16	10	2	None	0.106	0.976	0.974	60.340	10
3	150	0.1	150	0.01	16	15	2	None	0.371	0.936	0.948	92.519	1
3	150	0.1	150	0.01	16	15	2	None	0.122	0.971	0.972	73.546	2
3	150	0.1	150	0.01	16	15	2	None	0.315	0.931	0.931	75.018	3
3	150	0.1	150	0.01	16	15	2	None	0.084	0.982	0.985	66.248	4
3	150	0.1	150	0.01	16	15	2	None	0.032	0.992	0.993	65.114	5
3	150	0.1	150	0.01	16	15	2	None	0.105	0.979	0.981	66.892	6
3	150	0.1	150	0.01	16	15	2	None	0.139	0.969	0.971	84.822	7
3	150	0.1	150	0.01	16	15	2	None	0.195	0.955	0.954	84.174	8
3	150	0.1	150	0.01	16	15	2	None	0.159	0.963	0.963	84.116	9
3	150	0.1	150	0.01	16	15	2	None	0.091	0.976	0.975	85.678	10

Tabel 4.6. Lanjutan

4	100	0.1	50	0.01	32	10	2	None	0.397	0.927	0.939	32.950	1
4	100	0.1	50	0.01	32	10	2	None	0.154	0.956	0.962	27.415	2
4	100	0.1	50	0.01	32	10	2	None	0.317	0.923	0.926	30.642	3
4	100	0.1	50	0.01	32	10	2	None	0.142	0.973	0.981	27.083	4
4	100	0.1	50	0.01	32	10	2	None	0.081	0.972	0.976	26.795	5
4	100	0.1	50	0.01	32	10	2	None	0.127	0.965	0.969	49.314	6
4	100	0.1	50	0.01	32	10	2	None	0.138	0.958	0.962	33.219	7
4	100	0.1	50	0.01	32	10	2	None	0.187	0.951	0.955	32.099	8
4	100	0.1	50	0.01	32	10	2	None	0.202	0.950	0.953	35.259	9
4	100	0.1	50	0.01	32	10	2	None	0.117	0.966	0.966	32.506	10

Pada tabel 4.6, disajikan hasil pengujian K-Fold Cross Validation dengan K=10 untuk keempat skema hyperparameter. Pengujian ini dilakukan untuk memperkuat validitas hasil, meningkatkan akurasi estimasi performa, dan memungkinkan perbandingan antar skema dengan lebih detail.

Berdasarkan hasil pengujian *K-Fold Cross Validation* dengan K=10, keempat skema hyperparameter menunjukkan performa yang berbeda dalam hal akurasi, efisiensi komputasi, dan waktu komputasi. Skema 1 (tanpa optimasi) menunjukkan keseimbangan antara akurasi (F1 Score 0.956) dan waktu komputasi (25.028 detik), namun memiliki loss yang relatif tinggi (0.173). Hal ini menunjukkan bahwa model bi-LSTM tanpa optimasi hyperparameter dapat mencapai akurasi yang cukup baik dengan waktu komputasi yang singkat, namun masih ada ruang untuk peningkatan dalam hal meminimalkan kesalahan prediksi. Skema 2 (dioptimalkan genetika) menunjukkan peningkatan signifikan dalam akurasi (F1 Score 0.964) dan loss (0.161) dibandingkan Skema 1. Namun, waktu komputasi Skema 2 (57.020 detik) dua kali lipat dibandingkan Skema 1. Hal ini menunjukkan bahwa optimasi hyperparameter dengan algoritma genetika dapat meningkatkan akurasi dan efisiensi model, namun dengan trade-off waktu komputasi yang lebih lama. Skema 3 (dioptimalkan genetika dengan penambahan parameter) menunjukkan performa terbaik dalam F1 Score (0.967) dan Loss (0.161). Namun, Skema 3 memiliki waktu komputasi yang jauh lebih lama (77.813 detik) dibandingkan Skema 1 dan Skema 2. Hal ini menunjukkan bahwa penambahan parameter dapat meningkatkan akurasi model, namun dengan trade-off waktu komputasi yang signifikan. Skema 4 (dioptimalkan genetika dengan

pengurangan parameter) menunjukkan performa yang sedikit lebih rendah dibandingkan Skema 2 (F1 Score 0.959, Loss 0.186). Namun, Skema 4 memiliki waktu komputasi yang lebih cepat (32.728 detik) dibandingkan Skema 2. Hal ini menunjukkan bahwa pengurangan parameter dapat meningkatkan efisiensi komputasi model, namun dengan trade-off akurasi yang sedikit menurun.

Pemilihan skema hyperparameter terbaik tergantung pada trade-off antara akurasi, efisiensi komputasi, dan waktu komputasi. Skema 1 cocok untuk situasi di mana waktu komputasi lebih penting daripada akurasi. Skema 2 cocok untuk situasi di mana akurasi dan efisiensi komputasi sama-sama penting. Skema 3 cocok untuk situasi di mana akurasi adalah yang terpenting, dan sumber daya komputasi yang memadai tersedia. Skema 4 cocok untuk situasi di mana efisiensi komputasi lebih penting daripada akurasi, dan peningkatan akurasi kecil dapat diabaikan. Penting untuk dicatat bahwa hasil ini diperoleh dengan dataset dan konfigurasi model tertentu. Pengujian dengan dataset dan konfigurasi yang berbeda dapat menghasilkan hasil yang berbeda.

Pengujian *K-Fold Cross Validation* dengan K=10 menunjukkan peran penting algoritma genetika dalam pengoptimasian skema *hyperparameter* model bi-LSTM. Algoritma ini terbukti efektif dalam meningkatkan akurasi dan efisiensi model, serta memungkinkan penemuan kombinasi hyperparameter yang optimal untuk kebutuhan spesifik. Skema 2, yang dioptimalkan dengan algoritma genetika, menunjukkan peningkatan signifikan dalam F1 Score dan Loss dibandingkan Skema 1 (tanpa optimasi). Hal ini menunjukkan bahwa algoritma genetika mampu menemukan kombinasi hyperparameter yang menghasilkan model yang lebih

akurat dan efisien. Skema 3 dan 4, yang juga dioptimalkan dengan algoritma genetika, menunjukkan performa terbaik dalam F1 Score dan Loss. Skema 3 memiliki F1 Score 0.967 dan Loss 0.161, sedangkan Skema 4 memiliki F1 Score 0.959 dan Loss 0.186. Performa Skema 3 dan 4 menunjukkan bahwa algoritma genetika mampu menemukan kombinasi hyperparameter yang menghasilkan model dengan akurasi yang sangat tinggi.

4.3. Diskusi Hasil dan *Threat to Validity*

Algoritma genetika terbukti efektif dalam mengoptimasi *hyperparameter* daripada menggunakan konfigurasi secara manual atau *default* dari *library bi-LSTM*. Hasil dari konfigurasi sudah diuji dengan komparasi awal, menunjukkan perbedaan nilai F1 Score, Loss dan Waktu komputasi antara skema *default* dan skema optimal, skema optimasi lebih baik daripada skema non-optimasi. Uji selanjutnya dengan *K Fold Cross Validation* dengan nilai K = 5 menunjukkan peningkatan secara rata-rata baik F1 Score, Loss, dan Waktu Komputasi, walau skema kedua memakan komputasi yang banyak dan loss yang rendah. Uji terakhir menggunakan K = 10, bahwasanya peningkatan secara rata-rata juga dialami oleh skema optimasi dibandingkan skema non-optimasi.

4.3.1. *Threat to Validity*

Penelitian untuk upaya penignkatan performa bi-LSTM dengan algoritma genetika ini mungkin mengalami banyak kesalahan teknis maupun non teknis, adapun hasil yang ditunjukkan merupakan sebenarnya dari urutan metodologis yang disampaikan. Namun demikian, pemilihan dataset, *pengcodingan*, alur

algoritma mungkin tidak sempurna, sehingga terdapat pernyataan dari penelitian ini bahwa :

1. *Generalizability* : Hasil penelitian ini diperoleh dengan menggunakan dataset dan konfigurasi model tertentu. Pengujian dengan dataset dan konfigurasi yang berbeda dapat menghasilkan hasil yang berbeda. Ukuran dataset yang digunakan dalam penelitian ini relatif kecil. Pengujian dengan dataset yang lebih besar dapat menghasilkan hasil yang lebih stabil. Dataset yang digunakan bisa saja *imballance* di mana *labelling* pada *tag* cenderung "O" karena kalimat yang panjang dalam suatu *sentence*.
2. *Internal validity*: Kemungkinan terdapat bias dalam proses pengumpulan data dan anotasi entitas. Hyperparameter yang dioptimalkan dengan algoritma genetika mungkin tidak optimal untuk semua dataset.
3. *Construct validity* : Metrik yang digunakan untuk mengevaluasi model, seperti F1 Score, Loss, dan Waktu Komputasi, mungkin tidak sepenuhnya mencerminkan performa model dalam situasi dunia nyata.
4. *External validity* : Hasil penelitian ini mungkin tidak dapat digeneralisasikan ke model lain atau tugas lain.

Adapun saran dalam mitigasi *threat to validity* yakni :

1. Uji coba dengan dataset dan konfigurasi yang berbeda.
2. Gunakan dataset yang lebih besar untuk pelatihan model.
3. Lakukan pengukuran yang lebih komprehensif untuk mengevaluasi model.
4. Replikasi penelitian dengan metode dan dataset yang berbeda.

Walau demikian peneliti juga menyediakan *replicable package* dalam lampiran yang tersedia sebagai bahan untuk uji lanjut dan koreksi bersama demi kemajuan penelitian dan secara general ilmu pengetahuan.

BAB V

PENUTUP

5.1. Kesimpulan

Penelitian ini menunjukkan bahwa algoritma genetika efektif dalam mengoptimalkan *hyperparameter* model Bi-LSTM untuk mendeteksi entitas lokasi dan waktu pada data teks berbahasa Indonesia tentang kebakaran hutan. Model Bi-LSTM yang dioptimalkan dengan algoritma genetika (Skema 2) mencapai F1 Score 0.964, yang lebih tinggi dibandingkan model Bi-LSTM *default* (Skema 1) dengan F1 Score 0.956. Namun model Bi-LSTM yang dioptimalkan membutuhkan waktu komputasi lebih lama (57.020 detik) dibandingkan model default (25.028 detik). Uji K-Fold Cross Validation dengan K=5 dan K=10 menunjukkan peningkatan rata-rata F1 Score dan Loss pada skema optimasi dibandingkan skema non-optimasi. Konsistensi dan keandalan algoritma yang dioptimasi melalui uji non silang dan uji silang dengan jumlah *fold* yang berbeda menunjukkan bahwa *hyperparameter* yang dioptimasi oleh algoritma genetika memiliki kecenderungan nilai performa yang baik walau memakan waktu yang lebih lama.

Efisiensi waktu baik upaya pencarian dan percobaan evaluasi terhadap *hyperparameter* yang dioptimasi membutuhkan waktu yang lama dan komputasi yang besar. Untuk uji coba skema dengan jumlah populasi yang banyak memakan waktu yang banyak. Selanjutnya untuk pengujian model baik implementasi

langsung maupun uji secara *cross validation* juga memakan waktu yang banyak, beban komputasi semakin berat.

Penelitian terhadap upaya peningkatan performa algoritma genetika ini juga menunjukkan bahwa perbedaan *F1 Score, Loss*, dan waktu komputasi antara *hyperparameter default* dan yang dioptimasi tidak memiliki selisih yang signifikan, hanya berbeda 0.2-0.4 saja. Hal ini menandakan bahwa upaya pencarian algoritma genetika bisa saja tidak *worth it* dengan waktu dan penggunaan sumberdaya komputasi.

5.1. Saran

Saran utama dari penelitian ini adalah melakukan uji dengan dataset yang berbeda dan dengan penggunaan *sentence* yang seimbang. Hal ini penting untuk mencegah model *overfitting* dalam mempelajari data. Suatu *sentence* yang terdiri dari BI yang lebih sedikit dibanding O dalam proporsi jumlah kata dapat mempengaruhi pembelajaran model.

Algoritma dalam tugas *named entity recognition* sangatlah beragam, ke depannya dapat menggunakan algoritma geneika pada algoritma lainnya misal CNN, Transformer, CRF, GPT dan lain sebagainya.

Keterbatasan jumlah epoch yang besar menjadi pertimbangan dalam penelitian ini. Walau algoritma genetika secara eksperimental berhasil mencari nilai terbaik dari kombinasi *hyperparameter*, namun *hyperparameter* jumlah epoch hanya sedikit dengan kombinasi 5,10,15 dapat menjadi kekurangan dari

pembelajaran model Bi-LSTM. Permasalahan sumberdaya komputasi dipertimbangkan jika ingin membangun epoch lebih dari 15.

Waktu yang dibutuhkan dalam pencarian optimasi ini sangat lama. Peneliti atau pengguna model boleh tak menggunakan algoritma yang dioptimasi atau tidak perlu mencari algoritma optimasi jika sumberdaya komputasi terbatas. Namun juga perlu mempertimbangkan performa dasar dari algoritma secara *defaultnya*.

Konsekuensi dari keterbatasan komputasi yang besar dan membutuhkan waktu yang lama adalah jumlah eksperimen yang dilakukan hanya satu kali, terutama untuk menguji skema yang memiliki banyak populasi. Satu populasi bisa berisi banyak proses komputasi pembelajaran Bi-LSTM. Sehingga untuk saran, penelitian selanjutnya dapat menguji atau mengembangkan kembali dengan mempertimbangkan cara atau metode komputasi lainnya, misal melalui *paralel computing* atau HPC.

Performa dalam penelitian ini mengutamakan F1-Score sebagai matriks perfroma yang umum dalam NER, walau demikian, analisis performa lanjutan perlu dilakukan seperti *confussion matrics*.

Algoritma optimasi sangat banyak, baik berbasis evolusi maupun heuristik. Peneliti menyarankan untuk mencoba algoritma lain, baik pengembangan algoritma genetika *(GA)* maupun algoritma lain seperti *PSO (Particle Swarm Optimization), Differential Evolution (DE), dan Firefly Algorithm (FA)*.

5.2. Replication Package

Replication package yang tersedia adalah dataset, kode pemrograman dalam bahasa Python, dan hasil analisis atau hasil uji sebagai pembanding. Segala *replication package* tersedia di lampiran.

DAFTAR PUSTAKA

PUSTAKA BUKU

Burgess, J., Baym, N. K, 2022, Twitter: A Biography, NYU Press, New York

Nanda Saputra, M.Pd., Nurul Aida Fitri, M.Pd.,2020, Teori dan Aplikasi Bahasa Indonesia, Yayasan Penerbit Muhammad Zaini, Pidie Aceh

Michalski, R. S., Carbonell, J. G., & Mitchell, T. M, 2013, Machine learning: An artificial intelligence approach, Springer Science & Business Media, Palo Alto

Kochenderfer, M. J., & Wheeler, T. A, 2019, Algorithms for optimization, Mit Press, Massachusetts

Diwekar, U. M, 2020, Introduction to applied optimization (Vol. 22), edisi 2, Springer Nature, Clarendon Hills

Hluck, G, 2019, 12 Genetic Algorithms, The handbook of applied expert systems, 12(4), 2-3

Hochreiter, S., & Schmidhuber, J, 1997, Long short-term memory, Neural computation, 9(8), 1735-1780.

Sudipa, I. G. I., Udayana, I. P. A. E. D., Rizal, A. A., Kharisma, P. I., Indriyani, T., Asana, I. M. D. P., ... & Rachman, A, 2023, METODE PENELITIAN BIDANG ILMU INFORMATIKA (Teori & Referensi Berbasis Studi Kasus). PT. Sonpedia Publishing Indonesia, Jambi

Hasibuan, P. D., & Zainal, A, 2007, Metodologi Penelitian pada Bidang Ilmu Komputer dan Teknologi Informasi; Konsep, Teknik, dan Aplikasi, Fasilkom UI, Depok

Fanani, A. A., & Swara, S. E. (2023). Matematika Optimasi: Metode dan Penerapan. Universitas Brawijaya Press.

Kochenderfer, M. J., & Wheeler, T. A. (2019). Algorithms for optimization. Mit Press.

Putra, I. M. S. (2018). Penerapan Algoritma Genetika Dan Implementasi Dalam MATLAB. vol, 53, 1689-1699.

PUSTAKA MAJALAH, JURNAL ILMIAH ATAU PROSIDING

Pasai, M., 2020, Dampak kebakaran hutan dan penegakan hukum, Jurnal pahlawan, 3(1), 36-46

H. N. Salsabila, A. F. Sahitya, and P. Mahyatar, 2020, Spatio-temporal pattern analysis of forest fire event in South Kalimantan using integration remote sensing data and GIS for forest fire disaster mitigation, IOP Conference Series: Earth and Environmental Science, vol. 540, p. 012011, Aug. 2020, doi: https://doi.org/10.1088/1755-1315/540/1/012011.

Fatayat, F., & Risanto, J 2020, Model Sistem Deteksi Dini Kebakaran Hutan Dan Lahan (Karhutla) Berbasis Android Di Kabupaten Pelalawan, Simtika, 3(3), 19-25

Putra, A. A., & Kurniawan, R, 2020, Bidirectional LSTM-CNNs Untuk Ekstraksi Entity Lokasi Kebakaran Pada Berita Online Berbahasa Indonesia, In Seminar Nasional Official Statistics (Vol. 2020, No. 1, pp. 319-327)

PALLAVI, B. G., KUMAR, E. R., KARNATI, R., & KUMAR, R. A., 2022, LSTM Based Named Entity Chunking and Entity Extraction, 2022 First International Conference on Artificial Intelligence Trends and Pattern Recognition (ICAITPR), pp. 1-4, IEEE

Budi, I., & Suryono, R. R., 2023, Application of named entity recognition method for Indonesian datasets: a review. Bulletin of Electrical Engineering and Informatics, 12(2), 969-978.

Kumar, P., Batra, S., & Raman, B., 2021, Deep neural network hyper-parameter tuning through twofold genetic approach. Soft Computing, 25, 8747-8771

Sun, Y., Xue, B., Zhang, M., Yen, G. G., & Lv, J., 2020, Automatically designing CNN architectures using the genetic algorithm for image classification. IEEE transactions on cybernetics, 50(9), 3840-3854

Erden, C., 2023, Genetic algorithm-based hyperparameter optimization of deep learning models for PM2. 5 time-series prediction. International Journal of Environmental Science and Technology, 20(3), 2959-2982

Haq, R., Zhang, X., Khan, W., & Feng, Z., 2023, Urdu named entity recognition system using deep learning approaches. The Computer Journal, 66(8), 1856-1869

Majumder, A., Paul, A., & Banerjee, A., 2022, Deep learning-based approach using word and character embedding for named entity recognition from Hindi-English tweets. In Applications of Networks, Sensors and Autonomous Systems Analytics: Proceedings of ICANSAA 2020 (pp. 237-243). Springer Singapore

Alzaidi, B. S., Abushark, Y., & Khan, A. I., 2022, Arabic Location Named Entity Recognition for Tweets using a Deep Learning Approach. International Journal of Advanced Computer Science and Applications, 13(12)

Pérez-Díez, I., Pérez-Moraga, R., López-Cerdán, A., Salinas-Serrano, J. M., & la Iglesia-Vayá, M. D. (2021). De-identifying Spanish medical texts-named entity recognition applied to radiology reports. Journal of Biomedical Semantics, 12, 1-13.

Lin, J. C. W., Shao, Y., Zhang, J., & Yun, U., 2020, Enhanced sequence labeling based on latent variable conditional random fields. Neurocomputing, 403, 431-440.

Song, B., Li, F., Liu, Y., & Zeng, X. (2021). Deep learning methods for biomedical named entity recognition: a survey and qualitative comparison. Briefings in Bioinformatics, 22(6), bbab282.

Li, J., Sun, A., Han, J., & Li, C. (2020). A survey on deep learning for named entity recognition. IEEE transactions on knowledge and data engineering, 34(1), 50-70.

Jehangir, B., Radhakrishnan, S., & Agarwal, R. 2023. A survey on Named Entity Recognition—datasets, tools, and methodologies. Natural Language Processing Journal, 3, 100017.

Eligüzel, N., Çetinkaya, C., & Dereli, T. 2022. Application of named entity recognition on tweets during earthquake disaster: a deep learning-based approach. Soft Computing, 26(1), 395-421.

Xu, A., & Wang, C., 2022, March. The Study of NER Methods Based on Bi-LSTM+ CRF Model. In CIBDA 2022; 3rd International Conference on Computer Information and Big Data Applications (pp. 1-8). VDE

Ronran, C., & Lee, S., 2020, February. Effect of character and word features in bidirectional lstm-crf for ner. In 2020 IEEE International Conference on Big Data and Smart Computing (BigComp) (pp. 613-616). IEEE

Wang, J., Xu, W., Fu, X., Xu, G., & Wu, Y., 2020. ASTRAL: adversarial trained LSTM-CNN for named entity recognition. Knowledge-Based Systems, 197, 105842

Cho, M., Ha, J., Park, C., & Park, S., 2020. Combinatorial feature embedding based on CNN and LSTM for biomedical named entity recognition. Journal of biomedical informatics, 103, 103381

Halim, A. H., Ismail, I., & Das, S, 2021, Performance assessment of the metaheuristic optimization algorithms: an exhaustive review. Artificial Intelligence Review, 54, 2323-2409.

Bischl, B., Binder, M., Lang, M., Pielok, T., Richter, J., Coors, S., ... & Lindauer, M, 2023, Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 13(2), e1484.

Aufa, B. Z., Suyanto, S., & Arifianto, A, 2020, Hyperparameter setting of LSTM-based language model using grey wolf optimizer. In 2020 international conference on data science and its applications (ICoDSA) (pp. 1-5). IEEE.

Thoppil, N. M., Vasu, V., & Rao, C. S. P, 2022, Bayesian optimization LSTM/bi-LSTM network with self-optimized structure and hyperparameters for remaining useful life estimation of lathe spindle unit. Journal of Computing and Information Science in Engineering, 22(2), 021012.

Tuerxun, W., Xu, C., Guo, H., Guo, L., Zeng, N., & Cheng, Z, 2022, An ultra-short-term wind speed prediction model using LSTM based on modified tuna swarm optimization and successive variational mode decomposition. Energy Science & Engineering, 10(8), 3001-3022.

Shahid, F., Zameer, A., & Muneeb, M, 2021, A novel genetic LSTM model for wind power forecast. Energy, 223, 120069.

Raji, I. D., Bello-Salau, H., Umoh, I. J., Onumanyi, A. J., Adegboye, M. A., & Salawudeen, A. T. (2022). Simple deterministic selection-based genetic algorithm for hyperparameter tuning of machine learning models. Applied Sciences, 12(3), 1186.

Lee, S., Kim, J., Kang, H., Kang, D. Y., & Park, J, 2021, Genetic algorithm based deep learning neural network structure and hyperparameter optimization. Applied Sciences, 11(2), 744.

Wang, Z., & Sobey, A, 2020, A comparative review between Genetic Algorithm use in composite optimisation and the state-of-the-art in evolutionary computation. Composite Structures, 233, 111739.

Gen, M., & Lin, L., 2023, Genetic algorithms and their applications. In Springer handbook of engineering statistics (pp. 635-674). London: Springer London.

Sohail, A, 2023, Genetic algorithms in the fields of artificial intelligence and data sciences. Annals of Data Science, 10(4), 1007-1018.

Kara, A., 2021. Multi-step influenza outbreak forecasting using deep LSTM network and genetic algorithm. Expert Systems with Applications, 180, 115153

Palconit, M. G. B., Almero, V. J. D., Rosales, M. A., Sybingco, E., Bandala, A. A., Vicerra, R. R. P., & Dadios, E. P., 2020, November. Towards Tracking: Investigation of Genetic Algorithm and LSTM as Fish Trajectory Predictors

in Turbid Water. In 2020 IEEE REGION 10 CONFERENCE (TENCON) (pp. 744-749). IEEE

Zhang, W., Yang, K., Yu, N., Cheng, T., & Liu, J., 2020, December. Daily milk yield prediction of dairy cows based on the GA-LSTM algorithm. In 2020 15th IEEE International Conference on Signal Processing (ICSP) (Vol. 1, pp. 664-668). IEEE

Li, W., Zang, C., Liu, D., & Zeng, P., 2020, October. Short-term Load Forecasting of Long-short Term Memory Neural Network Based on Genetic Algorithm. In 2020 IEEE 4th Conference on Energy Internet and Energy System Integration (EI2) (pp. 2518-2522). IEEE

Ashok, D. M., Ghanshyam, A. N., Salim, S. S., Mazahir, D. B., & Thakare, B. S., 2020, June. Sarcasm detection using genetic optimization on LSTM with CNN. In 2020 International Conference for Emerging Technology (INCET) (pp. 1-4). IEEE

Rosalina, R., Auzar, A., & Hermandra, H., 2020. Penggunaan Bahasa Slang di Media Sosial Twitter. JURNAL TUAH: Pendidikan dan Pengajaran Bahasa, 2(1), 77-84.

Prayudi, S., & Nasution, W., 2020. Ragam bahasa dalam media sosial Twitter. Jurnal Metamorfosa, 8(2), 269-280.

Li, J., Sun, A., Han, J., & Li, C., 2020, A survey on deep learning for named entity recognition. IEEE Transactions on Knowledge and Data Engineering, 34(1), 50-70

Yadav, V., & Bethard, S., 2019, A survey on recent advances in named entity recognition from deep learning models. arXiv preprint arXiv:1910.11470

Ralph Grishman and Beth Sundheim, 1996, Message understanding conference-6: A brief history. In COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics, volume 1

Erik F Tjong Kim Sang, 2002, Introduction to the conll-2002 shared task: language-independent named entity recognition, proceedings of the 6th conference on natural language learning. August, 31:1–4

Erik F Tjong Kim Sang and Fien De Meulder, 2003, Introduction to the conll-2003 shared task: Language-independent named entity recognition. In Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4, pages 142–147. Association for Computational Linguistics

Gali, K., Surana, H., Vaidya, A., Shishtla, P. M., & Sharma, D. M., 2008, Aggregating machine learning and rule based heuristics for named entity

recognition. In Proceedings of the IJCNLP-08 Workshop on Named Entity Recognition for South and South East Asian Languages,

Khaled Shaalan. 2014. A survey of arabic named entity recognition and classification. Computational Linguistics, 40(2):469–510,

Darina Benikova, Chris Biemann, and Marc Reznicek. 2014. Nosta-d named entity annotation for german: Guidelines and dataset. In LREC, pages 2524–2531

Jakub Piskorski, Lidia Pivovarova, Jan Snajder, Josef Steinberger, Roman Yangarber, et al, 2017, T ˚ he first crosslingual challenge on recognition, normalization and matching of named entities in slavic languages. In Proceedings of the 6th Workshop on Balto-Slavic Natural Language Processing. Association for Computational Linguistics

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. Lingvisticae Investigationes, 30(1):3–26.

Ronan Collobert, Jason Weston, L ́eon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa, 2011, Natural language processing (almost) from scratch. Journal of Machine Learning Research, 12(Aug):2493–2537.

Khan, W., Daud, A., Nasir, J. A., & Amjad, T., 2016, A survey on the state-of-the-art machine learning models in the context of NLP. Kuwait Journal of Science, 43(4)

Jurafsky, D., & James, H., 2000, Speech and language processing an introduction to natural language processing, computational linguistics, and speech. 176 Negation and speculation detection in the medical domain and review texts

Sharma, A., Amrita, Chakraborty, S., & Kumar, S., 2022, Named entity recognition in natural language processing: A systematic review. In Proceedings of Second Doctoral Symposium on Computational Intelligence: DoSCI 2021 (pp. 817-828). Springer Singapore

Zhang, C., & Lu, Y., 2021, Study on artificial intelligence: The state of the art and future prospects. Journal of Industrial Information Integration, 23, 100224

Soydaner, D., 2020, A comparison of optimization algorithms for deep learning. International Journal of Pattern Recognition and Artificial Intelligence, 34(13), 2052013

Lambora, A., Gupta, K., & Chopra, K., 2019, February. Genetic algorithm-A literature review. In 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon) (pp. 380-384). IEEE.

Islam, M., Chen, G., & Jin, S., 2019, An overview of neural network. American Journal of Neural Networks and Applications, 5(1), 7-11

Alshemali, B., & Kalita, J., 2020, Improving the reliability of deep neural networks in NLP: A review. Knowledge-Based Systems, 191, 105210

Nosouhian, S., Nosouhian, F., & Khoshouei, A. K., 2021, A review of recurrent neural network architecture for sequence learning: Comparison between LSTM and GRU

Gers, F. A., Schmidhuber, J., & Cummins, F., 2000, Learning to forget: Continual prediction with LSTM. Neural Computation, 12(10), 2451-2471

Gorgolis, N., Hatzilygeroudis, I., Istenes, Z., & Gyenne, L. G., 2019, July. Hyperparameter optimization of LSTM network models through genetic algorithm. In 2019 10th International Conference on Information, Intelligence, Systems and Applications (IISA) (pp. 1-4). IEEE.

Tarwani, K. M., & Edem, S., 2017, Survey on recurrent neural network in natural language processing. Int. J. Eng. Trends Technol, 48(6), 301-304

Zhang, Y., Liu, Q., & Song, L., 2018, Sentence-state LSTM for text representation. arXiv preprint arXiv:1805.02474

Wang, W., Tong, M., & Yu, M, 2020, Blood glucose prediction with VMD and LSTM optimized by improved particle swarm optimization. IEEE Access, 8, 217908-217916.

Katoch, S., Chauhan, S. S., & Kumar, V, 2021, A review on genetic algorithm: past, present, and future. Multimedia tools and applications, 80, 8091-8126.

Anguita, D., Ghelardoni, L., Ghio, A., Oneto, L., & Ridella, S. (2012, April). The'K'in K-fold Cross Validation. In ESANN (Vol. 102, pp. 441-446).

Lobo, F.G., Lima, C.F. (2007). Adaptive Population Sizing Schemes in Genetic Algorithms. In: Lobo, F.G., Lima, C.F., Michalewicz, Z. (eds) Parameter Setting in Evolutionary Algorithms. Studies in Computational Intelligence, vol 54. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-69432-8_9

Eiben, Å. E., Hinterding, R., & Michalewicz, Z. (1999). Parameter control in evolutionary algorithms. IEEE Transactions on Evolutionary Computation, 3(2), 124-141.

Goldberg, D. E. (2006). Genetic Algorithms in Search, Optimization, and Machine Learning. Addison Wesley Longman.

Lobo, F. G., & Lima, C. F. (2005). Adaptive population sizing schemes in genetic algorithms. Proceedings of the Genetic and Evolutionary Computation Conference.

Reeves, C. R. (1993). Using genetic algorithms with small populations. In Proceedings of the Fifth International Conference on Genetic Algorithms, pp. 92-99. San Francisco.

Dodge, J., Ilharco, G., Schwartz, R., Farhadi, A., Hajishirzi, H., & Smith, N. (2020). Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. arXiv preprint arXiv:2002.06305.

PUSTAKA LAPORAN PENELITIAN

Sun, Y., & Wang, H, 2018, Learning Temporal Structures of Random Patterns. arXiv preprint arXiv:1805.10827.

PUSTAKA ELEKTRONIK

Stokel-Walker, C. (2023). Why is Twitter becoming X?.

Utami, E.; Istiyanto, J.E.; Hartati, S.; Marsono; Ashari, A., 25 November 2009, Developing Transliteration Pattern of Latin Character Text Document Algorithm Based on Linguistics Knowledge of Writing Javanese Script, http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5417267

Olah, C., 2015, Understanding LSTM Networks. Diakses pada 28 Januari 2020, dari https://colah.github.io/posts/2015-08-Understanding-LSTMs.