SKRIPSI

IMPLEMENTASI *BERT* UNTUK IDENTIFIKASI OTOMATIS ENTITAS HUKUM DALAM DOKUMEN PUTUSAN PENGADILAN BAHASA INDONESIA

Oleh:

AHMAD ROSYIHUDDIN

200411100126

Dosen Pembimbing 1 : Firdaus Solihin, S.Kom., M.Kom.

Dosen Pembimbing 2 : Dr. Noor Ifada, S.T., MISD.

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNIK INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS TRUNOJOYO MADURA

2024

HALAMAN JUDUL

IMPLEMNTASI *BERT* UNTUK IDENTIFIKASI OTOMATIS ENTITAS HUKUM DALAM DOKUMEN PUTUSAN PENGADILAN BAHASA INDONESIA

SKRIPSI

Diajukan untuk Memenuhi Persyaratan Penyelesaian Studi Strata Satu (S1)
dan Memperoleh Gelar Sarjana Komputer (S.Kom)

di Universitas Trunojoyo Madura

Ahmad Rosyihuddin

(200411100126)

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNIK INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS TRUNOJOYO MADURA

2024

HALAMAN PENGESAHAN

IMPLEMNTASI *BERT* UNTUK IDENTIFIKASI OTOMATIS ENTITAS HUKUM DALAM DOKUMEN PUTUSAN PENGADILAN BAHASA INDONESIA

Skripsi diajukan sebagai salah satu syarat untuk penyelesaian Pendidikan Program Studi S1 Teknik Informatika Universitas Trunojoyo Madura

Oleh:

Ahmad Rosyihuddin

200411100126

Disetujui oleh Tim Penguji Skripsi : **Tanggal Sidang :17 Juli 2024**

Firdaus Solihin, S.Kom., M.Kom. **NIP. 19760627 200801 1 008**	_____	**(Pembimbing 1)**
Dr. Noor Ifada, S.T., MISD. **NIP. 19780317 200312 2 001**	_____	**(Pembimbing 2)**
Dr. Fika Hastarita Rachman, ST., M.Eng **NIP. 19830305 200604 2 002**	_____	**(Penguji 1)**
Dr. Indah Agustien Siradjuddin, S.Kom., M.Kom **NIP. 19780820 200212 2 001**	_____	**(Penguji 2)**
Ika Oktavia Suzanti, S.Kom., M.Cs **NIP. 19881018 201504 2 004**	_____	**(Penguji 3)**

Bangkalan,
Mengetahui,
Ketua Jurusan Teknik Informatika

Dr. Yeni Kustiyahningsih, S.Kom., M.Kom.
NIP. 19770921 200812 2 002

HALAMAN PERNYATAAN ORISINALITAS

Saya yang bertanda tangan di bawah ini, menyatakan bahwa skripsi saya dengan judul :

"IMPLEMNTASI *BERT* UNTUK IDENTIFIKASI OTOMATIS ENTITAS HUKUM DALAM DOKUMEN PUTUSAN PENGADILAN BAHASA INDONESIA"

1. Adalah asli, bukan merupakan karya pihak lain serta belum pernah diajukan untuk mendapatkan gelar akademik Sarjana Komputer baik di Universitas Trunojoyo Madura maupun di Perguruan Tinggi yang lain di Indonesia.
2. Tidak terdapat karya atau pendapat pihak lain yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis telah diacu dalam naskah ini dan disebutkan dalam referensi.

Apabila di kemudian hari terbukti skripsi ini sebagian atau seluruhnya merupakan hasil plagiasi atau terdapat hal-hal yang tidak sesuai degan pernyataan di atas, maka saya sanggup menerima sanksi akademis yang berlaku dengan segala akibat hukumnya sesuai peraturan Universitas Trunojoyo Madura dan atau peraturan perundang-undangan yang berlaku.

Bangkalan, 28 Juli 2024

Yang Menyatakan,

Ahmad Rosyihuddin

NIM. 200411100126

MOTTO / HALAMAN PERSEMBAHAN

“Jadi Orang Sabar itu berat karena hadiahnya Surga, kalau hadiah nya payung mending ikut jalan sehat. Wong Sabar Rejeki Lancar”

KATA PENGANTAR

Puji dan syukur kami panjatkan kehadirat Tuhan Yang Maha Esa atas segala rahmat dan karunia-Nya sehingga skripsi yang berjudul " Implementasi Bert Untuk Identifikasi Otomatis Entitas Hukum Dalam Dokumen Putusan Pengadilan Bahasa Indonesia" ini dapat diselesaikan tepat pada waktunya. Skripsi ini disusun sebagai salah satu syarat untuk menyelesaikan studi pada Program Sarjana di Fakultas Teknik, Universitas Trunojoyo Madura.

Penyusunan skripsi ini tidak terlepas dari bantuan dan dukungan berbagai pihak. Oleh karena itu, pada kesempatan ini, penulis ingin menyampaikan terima kasih yang sebesar-besarnya kepada:

1. Allah SWT yang telah memberikan kesempatan, kelancaran, dan kesehatan kepada Penulis dalam mengerjakan skripsi ini sehingga penelitian ini dapat diselesaikan dengan lancar.
2. Orang tua tercinta yang selalu memberikan doa, semangat, dan dukungan materiil maupun moral.
3. Bapak Firdaus Solihin, S.Kom., M.Kom., selaku dosen pembimbing satu, dan juga Ibu Dr. Noor Ifada, S.T., MISD., selakuk dosen pembimbing dua yang telah memberikan bimbingan, saran, dan motivasi yang sangat berharga dalam penyusunan skripsi ini.
4. Ibu Dr. Fika Hastarita Rachman, ST., M.Eng, selaku Ketua Program Studi Teknik Informatika di Fakultas Teknik Universitas Trunojoyo Madura yang telah memberikan fasilitas dan dukungan selama studi.
5. Seluruh dosen di Fakultas Teknik Universitas Trunojoyo Madura yang telah memberikan ilmu dan bantuan selama masa perkuliahan.
6. Sahabat dan rekan-rekan seperjuangan di Fakultas Teknik, Universitas Trunojoyo Madura, yang telah memberikan semangat dan bantuan selama masa studi dan penyusunan skripsi ini.
7. Semua pihak yang telah membantu baik secara langsung maupun tidak langsung dalam penyusunan skripsi ini, yang tidak dapat penulis sebutkan satu per satu. Bantuan dan dukungan mereka sangat berarti bagi penulis.

Penulis menyadari bahwa dalam penulisan skripsi ini masih terdapat kekurangan dan keterbatasan. Oleh karena itu, penulis mengharapkan kritik dan saran yang

membangun demi kesempurnaan skripsi ini di masa yang akan datang. Semoga skripsi ini dapat memberikan manfaat bagi pembaca dan pihak-pihak yang berkepentingan.

Demikian kata pengantar ini penulis sampaikan. Terima kasih.

Bangkalan, 24, Juli, 2024

Penulis

ABSTRAK

Dokumen putusan hukum adalah sumber informasi berharga untuk berbagai tujuan, seperti pemetaan data kasus dan analisis teks hukum. Namun, pemahaman dokumen ini memerlukan waktu karena struktur dan panjangnya yang kompleks. Oleh karena itu, diperlukan sistem otomatis untuk mengekstrak informasi penting. Proses ini, dikenal sebagai *Named Entity Recognition* (NER), bertujuan mengidentifikasi entitas tertentu dalam teks, seperti lokasi dan organisasi. Di Indonesia, telah dilakukan penelitian tentang ekstraksi entitas dari dokumen hukum, namun belum ada yang menggunakan model *Deep Learning* berbasis *Transformers* seperti *BERT*. Penelitian ini menggunakan model *BERT* berbahasa Indonesia, yaitu *indolem/indobert-base-uncased* dan *indobenchmark/indobert-base-p2,* yang dilatih pada dataset *IndoLEM* dan *IndoNLU*. Penelitian ini menggunakan dataset yang diambil melalui situs Direktori Putusan Mahkama Agung RI dengan cara *scraping*. Total jumlah data sebanyak 1000 dokumen. Hasil *scraping* kemudian dilakukan *preprocessing*, anotasi, pembagian data dengan menggunakan *5-Fold Cross Validation, modeling* serta evaluasi. Hasil pengujian menunjukkan bahwa model *Indolem/indobert-base-uncased* secara konsisten menunjukkan performa yang lebih baik dalam tugas NER pada bidang hukum dibandingkan dengan *Indobenchmark/indobert-base-p2,* dengan rata-rata nilai presisi, recall, dan F1-score masing-masing 90%, 88%, dan 89%. Sementara itu, *Indobenchmark/indobert-base-p2* mencatatkan nilai presisi, recall, dan F1-score masing-masing 88%, 88%, dan 84%.

Kata Kunci : Dokumen Putusan Hukum, *Named Entity Recognition (NER), BERT*

ABSTRACT

Legal decision documents are a valuable source of information for many purposes, such as case data mapping and legal text analysis. However, understanding these documents takes time due to their complex structure and length. Therefore, an automated system is needed to extract important information. This process, known as Named Entity Recognition (NER), aims to identify specific entities in the text, such as locations and organizations. In Indonesia, research has been conducted on entity extraction from legal documents, but none have used Transformers-based Deep Learning models such as BERT. This research uses Indonesian BERT models, namely indolem/indobert-base-uncased and indobenchmark/indobert-base-p2, which are trained on IndoLEM and IndoNLU datasets. This research uses datasets taken from the Supreme Court Decision Directory website by scraping. The total amount of data is 1000 documents. The scraping results are then subjected to preprocessing, annotation, data division using 5-Fold Cross Validation, modeling and evaluation. Test results show that the Indolem/indobert-base-uncased model consistently performs better in NER tasks in the legal field compared to Indobenchmark/indobert-base-p2, with average precision, recall, and F1-score values of 90%, 88%, and 89%, respectively. Meanwhile, Indobenchmark/indobert-base-p2 recorded precision, recall, and F1-score values of 88%, 88%, and 84%, respectively.

***Keywords**: Legal Decision Document, Named Entity Recognition (NER), BERT*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR KODE

BAB I PENDAHULUAN

1.1. Latar Belakang

Dokumen putusan hukum menjadi sumber informasi berharga yang dapat dimanfaatkan untuk berbagai keperluan, termasuk pemetaan data kasus berdasarkan pasal-pasal hukum dakwaan, lokasi kejadian, evaluasi kesamaan sejarah hukum, dan analisis data dokumen hukum [1]. Salah satu lembaga yang dapat memanfaatkan dokumen putusan hukum adalah Mahkamah Agung (MA). Mahkamah Agung (MA) sebagai lembaga pemegang kekuasaan kehakiman, memiliki tanggung jawab penting dalam menjalankan fungsi pengawasan yang melibatkan analisis berbagai kasus hukum yang bersumber dari dokumen putusan pengadilan [2]. Perkembangan dokumen putusan hukum di Indonesia terus mengalami peningkatan seiring dengan kompleksitas hukum dan jumlah kasus yang dihadapi oleh sistem peradilan. Menurut data yang tercatat pada situs Direktori Putusan Mahkamah Agung RI [3], per 10 November 2023, jumlah keseluruhan dokumen putusan hukum mencapai 8.234.123 dokumen, dengan penambahan sebanyak 255.957 dokumen dalam 3 bulan terakhir. Fenomena ini menunjukkan bahwa setiap peristiwa hukum yang terjadi mengakibatkan penambahan dokumen putusan, menciptakan tantangan tersendiri dalam pengelolaan dan analisis data hukum. Dokumen putusan hukum dapat diakses melalui format PDF dan disusun berdasarkan lembaga peradilan, tingkat perkara, dan jenis perkara. Namun, proses pemahaman dokumen membutuhkan waktu yang cukup lama karena struktur yang berbeda dan kompleksitas dalam jumlah halaman [4]. Oleh karena itu, diperlukan sebuah sistem yang mampu mengekstrak informasi penting secara otomatis untuk membantu pembaca memahami dan mencari informasi di dalam dokumen putusan hukum dengan input teks maupun file PDF. Proses pengenalan informasi ini, yang umumnya dikenal sebagai *Named Entity Recognition* (NER), yang bertujuan untuk mengidentifikasi informasi entitas dari dokumen atau teks yang termasuk dalam kategori yang telah ditetapkan, seperti lokasi, organisasi, dan sebagainya [4].

NER tidak hanya digunakan untuk mengekstraksi informasi, tetapi juga sangat penting untuk berbagai tugas pemrosesan bahasa alami, seperti menjawab pertanyaan, pencarian informasi, dan mesin penerjemah [5]. Untuk memperoleh

NER dari berbagai jenis seperti lokasi, organisasi, dan sebagainya [4], diperlukan data train yang telah di-anotasi. Proses anotasi menggunakan standar seperti anotasi *Part Of Speech* (POS), yang melibatkan anotasi *Inside-Outside* (IO), *Inside-Outside-Beginning* (IOB), *Inside-Outside-End* (IOE), *Inside-Outside-Beginning-End-Single* (IOBES), *Beginning-Inside* (BI), *Inside-End* (IE), dan *Beginning-Inside-End-Sigle* (BIES) untuk mengenali jenis entitas kata [6]. Dalam implementasi NER terdapat empat metode utama yang digunakan: *Rule-Base*, *Supervised Learning*, *Unsupervised Learning*, dan *Deep Learning* [5].

Di Indonesia, terdapat penelitian yang sudah mengimplementasikan mengenai ekstraksi entitas dari dokumen putusan hukum. Seperti yang dilakukan oleh Solihin & Budi [2]. Pada penelitian ini dilakukan ekstraksi entitas dengan *Rule-Base*, sehingga dalam proses penelitian tersebut dilakukan identifikasi ekstraksi struktur, tokenisasi, dan ekstraksi entitas. Dari penelitian tersebut menghasilkan rata-rata *Recall* adalah 0.82, rata-rata *Precision* adalah 0.96, dan rata-rata *F-Score* adalah 0.89. Penelitian serupa juga dilakukan oleh Nuranti & Yulianti [4] dengan mengekstraksi entitas dari dokumen putusan hukum. Dalam penelitiannya, Nuranti & Yulianti [4] menggunakan pendekatan *Bidirectional Long Short Term Memory* (Bi-LSTM) dan *Conditional Random Fields* (CRF) untuk mengenali 10 jenis badan hukum pada dokumen putusan pengadilan di Indonesia yaitu advokat, amar (hukuman), hakim, jaksa, organisasi, panitera, peraturan, putusan, tanggal, terlibat (orang). Penelitian ini menghasilkan bahwa kombinasi metode *Bi-LSTM* dan *CRF* mencapai nilai *F1* tertinggi yaitu 0,83, mengungguli metode *Deep Learning* lainnya seperti *CNN*, *LSTM*, dan *SVM*. Hasil penelitiannya menunjukkan bahwa model *Bi-LSTM* dan *CRF* ini dapat digunakan untuk mengidentifikasi informasi domain hukum yang relevan pada dokumen putusan pengadilan di Indonesia.

Saat ini, penelitian tentang implementasi NER dalam domain hukum telah dilakukan dengan berbagai metode di beberapa negara. Salah satu penelitian dilakukan menggunakan model pre-trained [5] yaitu model yang telah dilatih sebelumnya menggunakan dataset yang besar, untuk mengekstraksi informasi dari dokumen hukum yang ada di Brazil. Penelitian lain menggunakan metode *Rule-Based* dan deteksi tata letak dari setiap paragraf untuk mengatasi ambiguitas dalam

urutan pembacaan dan mengurangi gangguan dari hasil *Optical Character Recognition* (OCR), terutama pada dokumen hukum Amerika Serikat [7] yang memiliki tata letak yang kompleks.

Meskipun NER telah banyak diteliti untuk berbagai jenis dokumen di beberapa Negara [5][7], penerapan NER untuk ekstraksi informasi dari dokumen hukum berbahasa Indonesia masih terbatas [2]. Penelitian sebelumnya menunjukkan bahwa belum ada yang menggunakan *Transformers* untuk ekstraksi entitas dalam dokumen putusan hukum bahasa Indonesia. Dalam penelitian yang dilakukan oleh Berragan et al. [8], ekstraksi nama tempat dari artikel Wikipedia menunjukkan bahwa penggunaan model berbasis *Transformers* seperti *Bidirectional Encoder Representations from Transformers* (*BERT)* dapat meningkatkan kinerja NER. Penelitian yang dilakukan oleh Sun et al. [9] dengan menggunakan dataset domain biomedis dan membandingkan beberapa model yaitu *BioBERT-Softmax, BioBERT-CRF, BioBERT-BiLSTM-CRF*, dan *BioBERT-MRC*. Dari keempat metode tersebut, tiga diantaranya masih menggunakan metode konvensional, sedangkan satu metode mengadopsi pendekatan *Machine Reading Comprehension (MRC)*. Hasil penelitian menunjukkan bahwa *BioBERT-MRC* memberikan performa terbaik dengan nilai rata-rata F1 sebesar 92.70 [9].

BERT adalah terobosan terbaru dalam bidang *Natural Language Processing* (NLP). Berbeda dengan *transformer* yang dikembangkan untuk menyelesaikan masalah Penerjemahan Mesin Neural, *BERT* dirancang tidak hanya untuk menangani Penerjemahan Mesin Neural, tetapi juga untuk Tanya Jawab, Analisis Sentimen, Ringkasan Teks, dan berbagai tugas lainnya [10] . *BERT* merupakan model yang telah dilatih sebelumnya yang memungkinkan pemahaman bahasa sesuai dengan data yang telah dilatih sebelumnya. Namun, hal ini menjadi kelemahan *BERT* karena *BERT* dilatih dengan menggunakan korpus data, sehingga kosakata yang dipahami oleh model *BERT* terbatas pada data teks yang dilatih [10].

Secara umum, terdapat 2 jenis *BERT* yaitu *BERT monolingual* dan *BERT multilingual*. *BERT monolingual* adalah *BERT* yang dibangun untuk memahami satu bahasa, sedangkan *BERT multilingual* adalah *BERT* yang rancang untuk memahami banyak bahasa [10]. Salah satu contoh *BERT monolingual* dalam bahasa Indonesia adalah *IndoBERT*. *IndoBERT* terdapat 2 versi yaitu model pre-trained

yang dilatih dengan dataset *IndoLEM* [11] [10] dan model pre-trained yang dilatih dengan dataset *IndoNLU* [12] [10]. Menurut Sebastian et al. [10], tolak ukur evaluasi untuk *BERT* bahasa Indonesia menggunakan dataset *IndoLEM* dan *IndoNLU*.

Oleh karena itu, sebagai langkah inovatif, metode yang diusulkan dalam penelitian ini menggunakan *Deep Learning* berbasis *Transformers*, khususnya model *BERT* yang telah dilatih dengan menggunakan bahasa Indonesia. Hal ini karena dataset yang digunakan dalam penelitian ini adalah dokumen putusan hukum berbahasa Indonesia. Diharapkan bahwa metode ini dapat meningkatkan akurasi dan kinerja NER dalam konteks ekstraksi informasi dari teks putusan hukum berbahasa Indonesia.

1.2. Rumusan Masalah

1.2.1. Permasalahan

Penggunaan model *Deep Learning* berbasis *Transformers* telah umum digunakan untuk ekstraksi entitas, baik di bidang hukum maupun bidang lainnya. Meski begitu, di Indonesia, penelitian terkait ekstraksi entitas pada dokumen putusan hukum berbahasa Indonesia masih terbatas, dan metode yang digunakan juga cenderung terbatas.

1.2.2. Solusi Permasalahan

Untuk mengatasi permasalahan ini, penelitian mengusulkan solusi ekstraksi entitas dengan memanfaatkan *BERT*. Dalam konteks ini, eksplorasi dilakukan dengan menggunakan beberapa model bahasa Indonesia yang telah dilatih sebelumnya, termasuk *indolem/indobert-base-uncased* dan *indobenchmark/indobert-base-p2*. Kedua model pre-trained tersebut dipilih karena telah dilatih dengan menggunakan dua dataset yang dianggap sebagai tolak ukur evaluasi pemrosesan bahasa alami bahasa Indonesia [10]. Dengan melakukan perbandingan antara kedua model tersebut, diharapkan penelitian ini dapat mengidentifikasi model terbaik yang dapat meningkatkan akurasi dan kinerja NER dalam ekstraksi informasi dari dokumen putusan hukum berbahasa Indonesia. Pendekatan ini juga membuka peluang untuk lebih memahami kontribusi masing-masing model pre-trained terhadap tugas ekstraksi entitas di domain hukum.

1.2.3. Pertanyaan Penelitian

Berdasarkan permasalahan dan solusi yang diajukan, penelitian ini berupaya menjawab pertanyaan penelitian mengenai model pre-trained mana yang terbaik antara *Indolem/indobert-base-uncased* dan *Indobenchmark/indobert-base-p2* dalam melakukan ekstraksi entitas pada dokumen hukum berbahasa Indonesia.

1.3. Batasan Masalah

Penelitian ini memiliki batasan-batasan tertentu untuk fokus dan keterbatasan implementasi, antara lain:

1. Penelitian ini terbatas pada pengembangan dan penerapan NER pada teks dokumen putusan hukum dalam bahasa Indonesia.
2. Implementasi pre-trained *BERT* hanya akan dievaluasi dalam konteks pengenalan entitas pada dokumen putusan hukum.
3. Dokumen yang digunakan dalam penelitian ini, menggunakan dokumen putusan pidana sebagai objek analisis, dengan total sampel sebanyak 1000 dokumen yang diambil dari tahun putusan 2002 hingga 2019.
4. Pre-trained model Bahasa Indonesia yang digunakan terbatas pada dua model, yaitu *Indolem/indobert-base-uncased* dan *Indobenchmark/indobert-base-p2*.
5. Entitas yang dilibatkan dalam penelitian ini terdiri dari 12 entitas, yaitu nomor putusan, nama terdakwa, tindak pidana, pelanggaran hukum, tuntutan hukum, putusan hukum, tanggal putusan, hakim ketua, hakim anggota, panitera, penuntut umum, dan penasihat.

1.4. Tujuan dan Manfaat

1.4.1. Tujuan

Penelitian ini bertujuan untuk mengetahui model pre-trained terbaik yang dapat mengekstrak entitas pada dokumen putusan hukum bahasa Indonesia.

1.4.2. Manfaat

Hasil dari penelitian ini diharapkan memberikan manfaat sebagai berikut:

1. Kontribusi pada pengembangan teknologi NER khususnya dalam konteks hukum dan bahasa Indonesia.
2. Peningkatan efisiensi dalam analisis dan pengelolaan dokumen putusan hukum oleh Mahkamah Agung dan lembaga kehakiman lainnya.

3. Peningkatan pemahaman terhadap implementasi metode *Deep Learning*, terutama model pre-trained *IndoBERT*, dalam pengenalan entitas pada dokumen hukum bahasa Indonesia.

1.5. Sistematika Penulisan

Sistematika penulisan penulisan tugas akhir ini sebagai berikut:

Bab I mengawali penelitian ini dengan membahas aspek pendahuluan. Pada bab ini, diperkenalkan latar belakang permasalahan yang menjadi fokus penelitian, solusi yang diusulkan untuk menyelesaikan permasalahan tersebut, serta tujuan dan manfaat dari pelaksanaan penelitian ini. Selain itu, bab ini juga mencakup pembahasan mengenai batasan-batasan yang diterapkan dalam penelitian serta menjelaskan secara rinci sistematika penulisan tugas akhir.

Bab II merinci kajian pustaka yang membahas materi dan metode sebagai dasar teoritis dalam penelitian ini. Beberapa topik yang ditekankan dalam materi melibatkan aspek-aspek terkait seperti *BERT* untuk NER, proses anotasi, dan aspek-aspek lain yang relevan dalam konteks penelitian ini.

Bab III sebagai inti dari penelitian ini, membahas metode penelitian secara komprehensif. Mulai dari pemilihan dataset dengan menguraikan langkah-langkah pengumpulan data dan karakteristik dataset, penelitian juga mendetail kan implementasi model *BERT* untuk NER, proses anotasi data, evaluasi model, serta skenario uji coba.

Bab IV Bab ini menyajikan hasil penelitian yang diperoleh dari penerapan metode yang telah dibahas di bab sebelumnya. Hasil analisis data dan evaluasi performa model BERT untuk tugas NER akan diuraikan secara rinci.

Bab V Bab terakhir ini menyimpulkan keseluruhan hasil penelitian dan menyajikan kesimpulan utama yang dapat diambil. Selain itu, bab ini juga memberikan saran untuk penelitian lebih lanjut yang dapat dilakukan untuk mengembangkan atau menyempurnakan penelitian ini.

BAB II KAJIAN PUSTAKA

2.1. Dokumen Putusan Pengadilan

Dokumen putusan pengadilan dibuat setelah rangkaian persidangan selesai di Pengadilan Negeri. Secara umum, isi dokumen ini dapat mencakup 10 halaman atau bahkan lebih, tergantung pada kompleksitas dan kebutuhan yang muncul selama persidangan [2]. Terdapat beberapa jenis hukum di Indonesia salah satu nya adalah Hukum Pidana, yaitu hukum yang mengatur tentang perbuatan-perbuatan yang dilarang dan diancam dengan pidana tertentu bagi yang melakukan nya. Berikut adalah contoh potongan dari dokumen putusan pengadilan yang bisa dilihat dalam Gambar 2.1.

Direktori Putusan Mahkamah Agung Republik Indonesia
putusan.mahkamahagung.go.id

P U T U S A N

NOMOR : 90 / PID.Sus / 2018 / PN.Mjn

DEMI KEADILAN BERDASARKAN KETUHANAN YANG MAHA ESA

Pengadilan Negeri Majene yang mengadili perkara pidana dengan acara biasa pada tingkat pertama dengan susunan persidangan Majelis, telah menjatuhkan putusan sebagai berikut dalam perkara Terdakwa :

Nama Lengkap	:	ANDIKA SAPUTRA ALIAS DIKA BIN ABDUL RASID ;
Tempat lahir	:	Majene ;
Umur / Tanggal lahir	:	21 Tahun / 6 Juli 1997 ;
Jenis Kelamin	:	Laki-laki ;
Kebangsaan	:	Indonesia ;
Tempat tinggal	:	Lingkungan Passarang, Kelurahan Totoli, Kecamatan Banggae, Kabupaten Majene ;
Agama	:	Islam ;
Pekerjaan	:	- ;
Pendidikan	:	S L T A (Tamat) ;

Terdakwa dalam perkara ini ditahan dengan jenis tahanan Rutan oleh :

- Penyidik, sejak tanggal 1 Oktober 2018 sampai dengan tanggal 20 Oktober 2018 ;
- Perpanjanagan Penuntut Umum Kejaksaan Negeri Majene, sejak tanggal 21 Oktober 2018 sampai dengan tanggal 29 November 2018 ;
- Penuntut Umum Kejaksaan Negeri Majene, sejak tanggal 29 November

Gambar 2.1 Potongan beberapa bagian dari isi dokumen putusan pengadilan dalam bahasa Indonesia

Dalam dokumen putusan pengadilan beberapa entitas krusial dapat diidentifikasi. Entitas ini dapat mempermudah pencarian dan pemahaman terhadap informasi spesifik, termasuk nomor putusan, identitas terdakwa, tindak pidana, pelanggaran KUHP, serta elemen-elemen kunci lainnya yang merinci proses dan hasil keputusan pengadilan. Entitas yang diambil dari dokumen putusan hukum terdapat 11 entitas sesuai dengan penelitian [2] serta terdapat penambahan 1 entitas yaitu advokat sehingga total nya menjadi 12 entitas. Dari 12 entitas ini 11 diantara nya dipastikan

ada dalam sebuah dokumen, namun terdapat 1 entitas yang belum tentu ada pada sebuah dokumen yaitu entitas advokat. Daftar entitas bisa dilihat dalam Tabel 2.1.

Tabel 2.1 Daftar Entitas dan Keterangan

No	Entitas	Keterangan
1.	Nomor Putusan	Nomor Putusan dokumen
2.	Nama Terdakwa	Nama terdakwa
3.	Tindak Pidana	Kejahatan yang dilakukan
4.	Melanggar KUHP	Pelanggaran KUHP yang dilakukan
5.	Tuntutan Hukum	Jenis Hukuman yang diajukan
6.	Putusan Hukum	Jenis Hukuman Yang tetapkan
7.	Tanggal Putusan	Tanggal Keputusan
8.	Hakim Ketua	Nama Hakim Ketua
9.	Hakim Anggota	Nama Hakim Anggota
10.	Panitera	Nama Panitera
11.	Penuntut Umum	Nama Penuntut Umum
12.	Penasihat/Pengacara	Nama Penasihat/Pengacara

2.2. *Named Entity Recognition* (NER)

Named Entity Recognition (NER) adalah tugas dari ekstraksi informasi yang bertujuan untuk menemukan dan mengklasifikasikan entitas bernama dalam teks tidak terstruktur ke dalam kategori yang telah ditentukan [13]. Dalam konteks ini, NER digunakan untuk mengekstraksi informasi dari dokumen putusan hukum, mencakup entitas seperti nomor putusan, nama terdakwa, tanggal putusan, dan isi putusan hukum lainnya. Pada tahun 1995, NER pertama kali diperkenalkan dengan hanya mengidentifikasi tiga kategori, yaitu Entitas, Nama, dan Nomor [13].

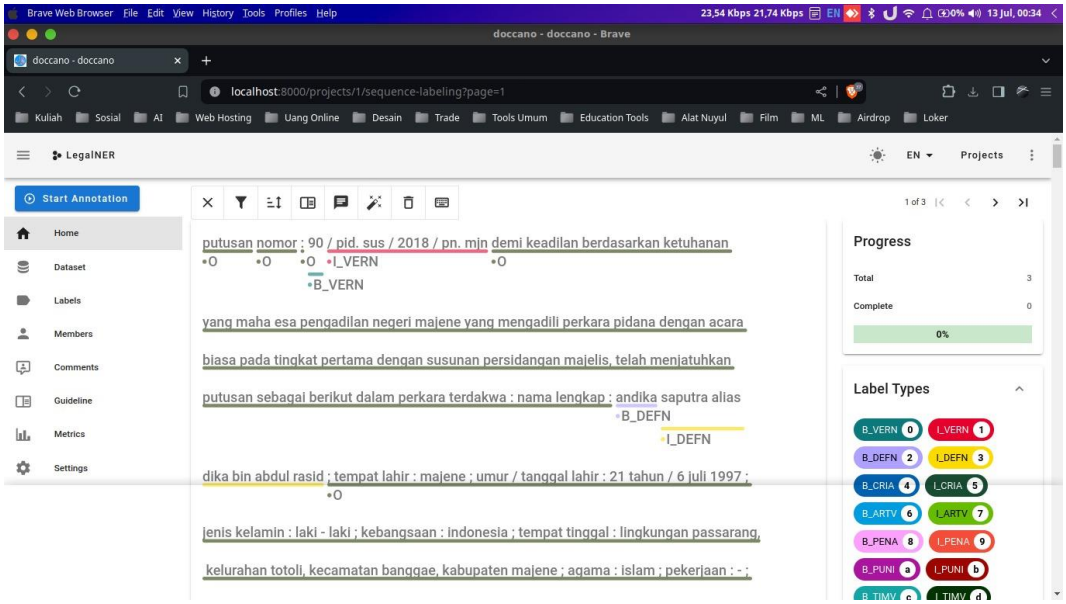
NER, merupakan suatu metode pembelajaran mesin yang sangat populer dan dianggap sebagai teknik dasar untuk banyak tugas *Natural Language Processing* (NLP). Sebelum tahun 2011, semua pekerjaan dalam NER bersifat spesifik domain dan dirancang untuk mengeksekusi tugas tertentu berdasarkan ontologi [13].

Pada awalnya sistem NER mengandalkan aturan yang dibuat oleh manusia, dan sistem NER yang berbasis pada aturan dianggap memerlukan waktu yang signifikan untuk direncanakan. Untuk mengatasi kendala ini, para peneliti mulai

mengembangkan sistem NER yang berfokus pada algoritma pembelajaran mesin. Mereka menerapkan berbagai metode pembelajaran, termasuk yang diawasi, semi ter awasi, dan tanpa pengawasan, sebagai solusi atas tantangan tersebut [14].

2.3. *Doccano*

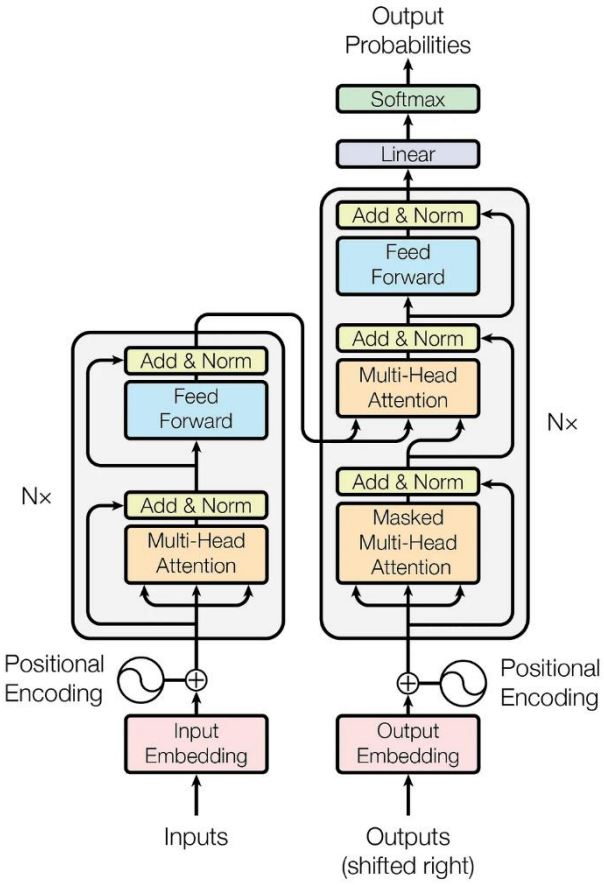
Doccano merupakan alat anotasi *open source* untuk membantu para praktisi *machine learning* dalam membuat dataset. *Doccano* dapat digunakan untuk anotasi banyak format data diantaranya klasifikasi teks, pelabelan urutan, NER, rangkuman teks, dan lain-lain [15]. Tampilan Doccano bisa dilihat pada Gambar 2.2.



Gambar 2.2 Tampilan *Doccano*

2.4. *Transformer*

Transformer adalah sebuah inovasi besar dalam domain NLP yang diperkenalkan oleh Vaswani et al. [16] pada tahun 2017. Model ini digunakan untuk pemodelan urutan *(Sequence Modeling)* dan transduksi *(Transduction).* Dibandingkan dengan pendekatan tradisional yang menggunakan rekurensi atau konvolusi, Transformer memperkenalkan suatu konsep baru yang hanya memanfaatkan *attention mechanism.* Dengan demikian, Transformer menjadi sangat efektif dalam menangani tugas-tugas pemrosesan bahasa alami, terjemahan mesin, dan masalah lain yang melibatkan data berurutan.



Gambar 2.3 Arsitektur *Transformer* [16]

Arsitektur *Transformer* terdiri dari dua komponen utama, yaitu *Encoder* dan *Decoder*. Kedua komponen ini terdiri dari lapisan-lapisan identik yang menggunakan dua komponen kunci, yaitu *Multi-Head Self-Attention* dan *Fully Connected Feed-Forward Network* [16]. *Attention mechanism* memungkinkan model untuk memberikan bobot yang berbeda pada bagian-bagian berbeda dari input, sehingga memungkinkan pemodelan konteks yang lebih baik. Arsitektur Transformer bisa dilihat pada Gambar 2.3.

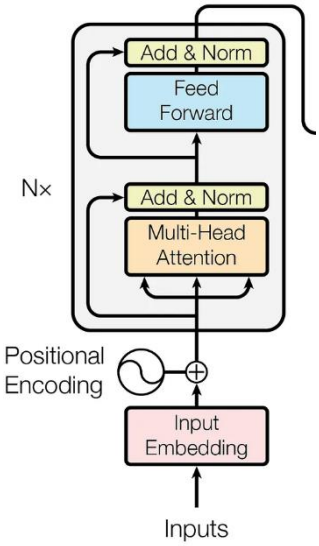
2.4.1. *Input Embedding*

Input embedding dalam model Transformer melibatkan beberapa langkah penting untuk mengonversi token input dan output menjadi vektor dengan dimensi d_{model}. Proses ini menggunakan embedding yang dipelajari (learned embeddings) untuk melakukan konversi tersebut. Selain itu, model ini juga menggunakan transformasi linear yang dipelajari (learned linear transformation) dan fungsi

softmax untuk mengubah output decoder menjadi probabilitas token berikutnya yang diprediksi [16].

2.4.2. Encoder dan *Decoder Transformers*

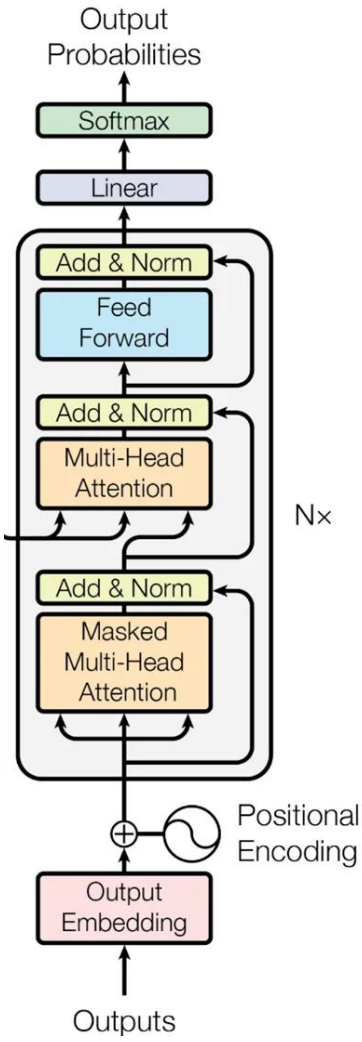
Encoder Transformer terdiri dari 6 lapisan identik, dan setiap lapisan memiliki dua sub-lapisan. Yang pertama adalah lapisan *Multi-Head Self-Attention* dan yang kedua adalah *Position-Wise Feed-Forward Network. Encoder* menggunakan koneksi residual disekitar kedua sub-lapisan tersebut, diikuti dengan lapisan normalisasi. Melalui mekanisme *Self-Attention*, setiap posisi dalam *encoder* dapat memfokuskan perhatian nya pada seluruh posisi dalam urutan input, sehingga dapat menangkap hubungan antar kata-kata yang berbeda [16]. Arsitektur *Encoder* bisa dilihat dalam Gambar 2.4.



Gambar 2.4 Arsitektur *Encoder* [16]

Selain itu, *Decoder Transformer* juga memiliki jumlah lapisan yang sama dengan *Encoder,* yaitu 6 lapisan. *Decoder* memiliki sub-lapisan ketiga selain dua lapisan *encoder* yang sudah ada. Lapisan ketiga berfungsi untuk menangani output *encoder*. Seperti pada *encoder*, *decoder* menerapkan koneksi residual di sekitar setiap sub-lapisan, diikuti oleh lapisan normalisasi. Sub-lapisan *self-attention* dalam lapisan *decoder* juga mengalami penyesuaian dengan menyembunyikan token setelah nya untuk mencegah model agar tidak memperhatikan posisi yang terjadi setelahnya. Dengan melakukan penyembunyian ini, bersamaan dengan perubahan posisi satu langkah pada vektor *embedding* output, dipastikan bahwa

ketika model membuat prediksi untuk suatu posisi i, model hanya bergantung pada informasi yang sudah ada pada posisi-posisi sebelumnya dalam urutan, dan tidak memperhatikan apa yang terjadi setelahnya [16]. Arsitektur dari *Decoder* bisa dilihat dalam Gambar 2.5.

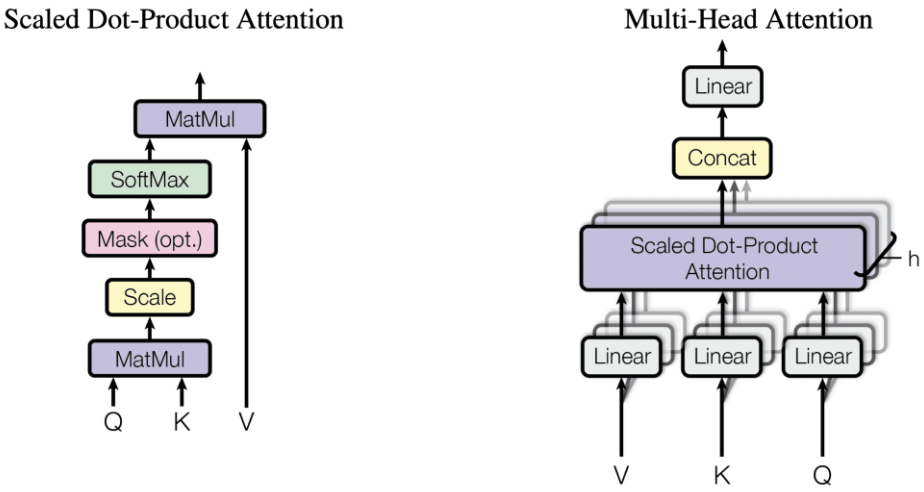


Gambar 2.5 Arsitektur Decoder [16]

2.4.3. *Attention Mechanism*

Attention Mechanism dalam *Transformer* memungkinkan model untuk memberikan bobot yang berbeda pada setiap elemen input, sehingga memungkinkan fokus pada informasi yang lebih relevan. Ini dilakukan melalui tiga matriks yaitu *Key*, *Query*, dan *Value*. Matriks *Query* digunakan untuk mengukur sejauh mana setiap elemen dalam urutan input cocok dengan elemen lainnya, yang diwakili oleh matriks *Key*. Hasilnya kemudian dinormalisasi untuk menghasilkan bobot melalui fungsi *softmax* dan digunakan untuk memberikan bobot pada matriks

Value [16]. *Attention mechanism* pada Transformers terdiri dari 2 komponen yaitu *Scale Dot-Product Attention* dan *Multi-Head Attention*. Arsitektur dari *attention mechanism* bisa dilihat dalam Gambar 2.*6*



Gambar 2.6 Arsitektur *Attention Mechanism* [16]

1. *Scaled Dot-Product Attention*

   Scaled Dot-Product Attention seperti yang digambarkan dalam Gambar 2.6, merupakan *attention mechanism* khusus yang digunakan dalam model *transformers*. Input terdiri dari *Query* dan *Key* dengan dimensi d_k, dan nilai dengan dimensi d_v. *Dot product* dari *Query* dengan semua *Key* dihitung, dibagi dengan $\sqrt{d_k}$, dan diterapkan fungsi *softmax* untuk mendapatkan bobot pada nilai-nilai. Dalam praktik nya, fungsi perhatian dihitung pada sekumpulan *query* secara simultan, dikemas bersama menjadi matriks Q. Kunci dan nilai juga dikemas bersama menjadi matriks K dan V [16]. Matriks output dihitung dengan menggunakan persamaan (2.1).

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \qquad (2.1)$$

   Keterangan:
   Q = Query
   K = Key
   V = Value
   d_k = Dimensi Key
   V = Value
   K^T = Key Transpose

Sedangkan untuk *softmax* dapat menggunakan persamaan (2.2).

$$Softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \quad (2.2)$$

Keterangan:
$Softmax(x_i)$ = Fungsi *Softmax* yang diterapkan pada vector x
x = Vektor input yang akan diubah menjadi probabilitas
i = Indeks dari elemen vektor x
e = Bilangan konstan Euler (2.71828…)
n = Jumlah elemen dalam vektor x
Σ = Simbol sigma yang menunjukkan penjumlahan

2. *Multi-Head Attention*

   Multi-Head Attention memungkinkan model untuk menghadiri informasi dari sub ruang representasi yang berbeda diposisi yang berbeda. Dengan menggunakan *h* kepala perhatian yang berjalan secara paralel, Transformer dapat menangkap informasi yang lebih kaya dibandingkan dengan perhatian tunggal [16]. *Attention mechanism* ini memainkan peran penting dalam memungkinkan Transformer untuk menangkap konteks yang kaya dari data *sequential*, yang telah terbukti efektif tidak hanya dalam tugas terjemahan mesin tetapi juga dalam berbagai aplikasi pemrosesan bahasa alami lainnya [16].

2.4.4. *Position-Wise Feed Forward Network*

Dalam arsitektur Transformer, terdapat *Position-wise Feed-Forward Network* (FFN) yang terdiri dari dua transformasi linear, dengan fungsi aktivasi *ReLU* diantara kedua transformasi tersebut. Jaringan ini diterapkan secara terpisah dan identik untuk setiap posisi dalam urutan, memungkinkan model untuk melakukan transformasi lebih lanjut pada data [16]. Secara matematis, operasi yang dilakukan oleh FFN pada input x dapat menggunakan persamaan (2.3).

$$FFN(x) = \max(0, xW_1 + b_1)W_2 + b_2 \quad (2.3)$$

Keterangan:
$W_1\ dan\ W_2$ = Matriks bobot
$b_1\ dan\ b_2$ = Bias

2.5. *BERT*

Bidirectional Encoder Representations from Transformers (BERT) adalah model yang telah dilatih secara *unsupervised* dengan dua tugas, yaitu *masked language model* dan *next sentence prediction*. Dalam *masked language model*, beberapa token secara acak di *Masking* (diganti dengan [MASK]), lalu BERT memprediksi token asli tersebut berdasarkan konteks kiri dan kanan. *Next sentence prediction* adalah tugas klasifikasi biner untuk memprediksi apakah dua kalimat yang diberikan saling berurutan dalam teks [17]. Untuk arsitektur BERT bisa dilihat dalam Gambar 2.7.

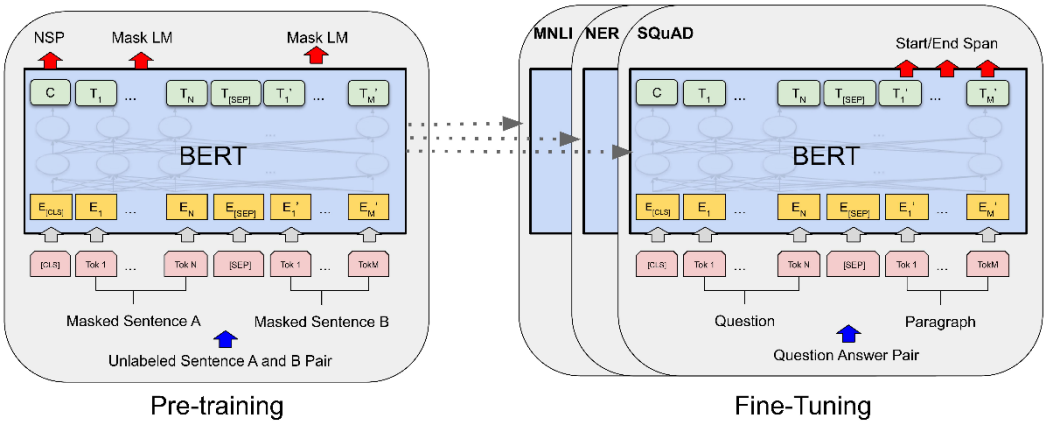
BERT menggunakan arsitektur *Transformers encoder* yang mendukung representasi *bidirectional*, berbeda dengan model sebelumnya seperti *ELMo* dan *OpenAI GPT* yang unidirectional. Meskipun *BERT* menggunakan arsitektur *Transformers encoder*, terdapat perbedaan antara *encoder BERT* dan *encoder Transformers* standar. Perbedaan ini terletak pada penggunaan fungsi aktivasi dalam proses *Feed Forward Network* (FFN), di mana *Transformers* menggunakan *ReLU* sebagai fungsi aktivasi, sedangkan *BERT* menggunakan *GELU* [17]. Perhitungan matematis *GELU* bisa dilihat dalam persamaan (2.4).

$$GELU(x) = x \cdot \frac{1}{2}\left[1 + \text{erf}(x/\sqrt{2})\right] \tag{2.4}$$

Keterangan:
x = data input yang akan di proses

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\, dt \tag{2.5}$$

Keterangan:
x = data input yang akan di proses
e = Konstanta Eular
dt = Diferensial



Gambar 2.7 Arsitektur BERT [17]

Layer *embedding* dalam BERT memainkan peran penting dalam menghasilkan representasi vektor untuk setiap token input. Proses ini melibatkan tiga jenis *embedding* yang berbeda: *Token Embeddings*, *Positional Embeddings*, dan *Segment Embeddings* [17]. Ketiga jenis embedding ini kemudian dijumlahkan untuk menghasilkan *embedding* akhir untuk setiap token. *Token Embeddings* mewakili representasi vektor dari setiap token. *Positional Embeddings* memberikan informasi tentang posisi setiap token dalam kalimat, menggunakan *positional embeddings* yang dipelajari, yang berarti model secara otomatis menyesuaikan representasi posisi berdasarkan data yang dilihat selama *pre-training* [16]. *Segment Embeddings* digunakan untuk membedakan antara kalimat yang berbeda dalam input yang sama, yang penting untuk tugas-tugas yang melibatkan beberapa kalimat [18]. Setelah ketiga jenis *embedding* ini dihitung untuk setiap token, mereka dijumlahkan untuk menghasilkan representasi akhir yang digunakan sebagai input untuk lapisan *encoder* BERT selanjutnya.

BERT juga telah dikembangkan dalam bahasa Indonesia, salah satunya adalah *Indolem/indobert-base-uncased* [11] dan *Indobenchmark/indobert-base-p2* [12]. *Indolem/indobert-base-uncased* adalah model BERT yang telah dilatih dengan data teks berbahasa Indonesia dan tidak mempertimbangkan kapitalisasi (*uncased*). Sementara itu, *Indobenchmark/indobert-base-p2* adalah model BERT lainnya yang juga dilatih dengan data teks berbahasa Indonesia dan telah melalui dua fase pelatihan (p2). Kedua model ini dapat digunakan untuk berbagai tugas pemrosesan bahasa alami dalam bahasa Indonesia, seperti klasifikasi teks, ekstraksi informasi, analisis sentimen, dan lainnya.

2.5.1. *Indolem/indobert-base-uncased*

indobert-base-uncased merupakan model bahasa yang dikembangkan khusus untuk memahami dan memproses bahasa Indonesia, berdasarkan arsitektur BERT yang populer. Model ini dirancang untuk mengatasi tantangan khusus dalam *natural language processing* (NLP) untuk bahasa Indonesia, yang mencakup kekurangan dataset yang dianotasi, ketersediaan sumber daya bahasa yang terbatas, dan kurangnya standarisasi sumber daya. *Indolem/indobert-base-uncased* dilatih menggunakan teknik *Masked Language Modeling* (MLM) pada dataset besar yang mencakup lebih dari 220 juta kata, yang dikumpulkan dari sumber-sumber seperti Wikipedia Indonesia, artikel berita dari Kompas dan Tempo, serta korpus web Indonesia [11]. Statistik dari dataset yang di sebutkan pada [11] bisa dilihat dalam Tabel 2.2.

Tabel 2.2 Statistik Dataset IndoLEM yang di sebutkan

Data	Jumlah Kata
Wikipedia Indonesia	74 Juta
Artikel berita (Kompas, Tempo, Liputan6)	55 Juta
Web Corpus Indoensia	90 Juta

Indobert-base-uncased menggunakan arsitektur transformers yang berasal dari BERT, serta mengadopsi konfigurasi model yang sama dengan BERT-base (Uncased). Detail konfigurasi model ini bisa dilihat dalam Tabel 2.3.

Tabel 2.3 Konfigurasi model *Indobert-base-uncased* [11]

Hyperparameter	Size
Embedding Size	768
Hidden Size	768
Intermediate size	3072
Number of attention heads	12
Number of hidden layers	12
Vocab size	31923

2.5.2. *Indobenchmark/indobert-base-p2*

Indobenchmark/indobert-base-p2 merupakan salah satu varian dari model *IndoBERT* yang dirancang khusus untuk memahami Bahasa Indonesia dengan lebih efektif. Model ini dilatih menggunakan dataset *Indo4B*, yang merupakan kumpulan data berskala besar dan bersih dari berbagai sumber publik seperti media sosial, blog, berita, dan situs web. *Indobenchmark/indobert-base-p2* dirancang untuk menangani berbagai tugas *Natural Language Understanding* (NLU) dalam bahasa Indonesia, mencakup 12 tugas yang beragam mulai dari klasifikasi kalimat tunggal hingga pelabelan urutan kalimat pasangan dengan tingkat kompleksitas yang berbeda-beda. Model ini merupakan bagian dari upaya untuk mengatasi keterbatasan sumber daya dan kemajuan penelitian dalam pemrosesan bahasa alami untuk bahasa Indonesia, yang merupakan salah satu bahasa yang paling banyak digunakan di internet [12]. Konfigurasi yang digunakan dalam model *Indobenchmark/indobert-base-p2* bisa dilihat dalam Tabel 2.4.

Tabel 2.4 Konfigurasi model *Indobenchmark/indobert-base-p2* [12]

Hyperparameter	Size
Embedding Size	768
Hidden Size	768
Intermediate size	3072
Number of attention heads	12
Number of hidden layers	12
Vocab size	30522

Indo4B terdiri dari sekitar 4 miliar kata dengan sekitar 250 juta kalimat. Dataset *Indo4B* mencakup kalimat Bahasa Indonesia formal, Bahasa Indonesia sehari-hari dan Bahasa Indonesia campuran dengan sumber yang berbeda-beda. Ukuran dataset *Indo4B* sebesar 23GB. Statistik dari dataset ini bisa dilihat dalam Tabel 2.5. Pada penelitian [12], dataset *Indo4B* juga dilakukan ujicoba membangdingkan dengan dataset *CC-ID* yang berukuran 180 GB (belum terkompres) dengan menggunakan model *fastText*. Hasil dari perbandingan ini dimenangkan oleh dataset *Indo4B*, sehingga penelitian pada [12] menyimpulkan bahwa meskipun dataset *Indo4b* jauh lebih kecil dari pada dataset *CC-ID* namun

dataset *Indo4B* memiliki variasi bahasa indonesia yang lebih banyak dan kualitas yang lebih baik di bandingkan dataset *CC-ID*.

Tabel 2.5 Statistik Dataset IndoNLU [12]

Dataset	Kalimat	Kata	Ukuran	Gaya	Sumber
OSCAR (Ortiz Suarez et al., 2019)	2,279,761,186	148,698,472	14.9 GB	mixed	OSCAR
CoNLLu Common Crawl (Ginter et al., 2017)	905,920,488	77,715,412	6.1 GB	mixed	LINDAT/CLARIAH-CZ
OpenSubtitles (Lison and Tiedemann, 2016)	105,061,204	25,255,662	664.8 MB	mixed	OPUS OpenSubtitles
Twitter Crawl2	115,205,737	11,605,310	597.5 MB	colloquial	Twitter
Wikipedia Dump1	76,263,857	4,768,444	528.1 MB	formal	Wikipedia
Wikipedia CoNLLu (Ginter et al., 2017)	62,373,352	4,461,162	423.2 MB	formal	LINDAT/CLARIAH-CZ
Twitter UI2 (Saputri et al., 2018)	16,637,641	1,423,212	88 MB	colloquial	Twitter
OPUS JW300 (Agic and Vulić, 2019)	8,002,490	586,911	52 MB	formal	OPUS
Tempo3	5,899,252	391,591	40.8 MB	formal	ILSP
Kompas3	3,671,715	220,555	25.5 MB	formal	ILSP
TED	1,483,786	111,759	9.9 MB	mixed	TED
BPPT	500,032	25,943	3.5 MB	formal	BPPT
Parallel Corpus	510,396	35,174	3.4 MB	formal	PAN Localization
TALPCo (Nomoto et al., 2018)	8,795	1,392	56.1 KB	formal	Tokyo University
Frog Storytelling (Moeljadi, 2012)	1,545	177	10.1 KB	mixed	Tokyo University
Total	**3,581,301,476**	**275,301,176**	**23.43 GB**		

2.5.3. *Masked Languge Model* (MLM)

Masked Language Modeling (MLM) adalah teknik pelatihan yang digunakan dalam model BERT untuk membangun representasi bidirectional yang dalam dari bahasa. Dalam MLM, sejumlah kecil token dalam sebuah urutan teks di *mask* atau di tutup secara acak, dan model dilatih untuk memprediksi token yang hilang ini. Misalnya, dalam satu kalimat, beberapa kata akan diganti dengan token [MASK], dan model harus memprediksi kata-kata asli berdasarkan konteks kata-kata lain di sekitarnya. Pendekatan ini memungkinkan model memahami konteks dari kedua arah (kiri dan kanan) secara simultan, berbeda dengan model bahasa tradisional yang hanya memprediksi kata dari kiri ke kanan atau sebaliknya [17].

Dengan menggunakan MLM, model BERT mampu menangkap hubungan yang lebih kompleks dalam teks dan memahami nuansa bahasa yang lebih baik. Namun, teknik ini menciptakan ketidakcocokan antara fase *pretrained* dan *fine-tuning* karena token [MASK] tidak muncul selama *fine-tuning*. Untuk mengatasi masalah ini, saat pelatihan, hanya 80% dari token yang dimasker digantikan oleh [MASK], sedangkan 10% diganti dengan token acak, dan 10% sisanya tetap tidak berubah. Hal ini membantu model beradaptasi dengan kondisi yang lebih mendekati penggunaan sebenarnya [17].

2.5.4. *Next Sentence Prediction* (NSP)

Next Sentence Prediction (NSP) adalah tugas pelatihan yang bertujuan untuk mengajarkan model memahami hubungan antara dua kalimat. Dalam NSP, model dilatih untuk memprediksi apakah dua kalimat dalam sebuah pasangan adalah kalimat yang berurutan dalam teks asli atau tidak. Dalam setiap contoh pelatihan, ada 50% kemungkinan bahwa kalimat kedua benar-benar mengikuti kalimat pertama (diberi label "IsNext") dan 50% kemungkinan bahwa kalimat kedua adalah kalimat acak dari korpus (diberi label "NotNext"). Tugas ini penting karena banyak aplikasi pemrosesan bahasa alami, seperti penjawab pertanyaan (QA) dan *natural language inference* (NLI), membutuhkan pemahaman yang baik tentang hubungan antar kalimat [17].

Dengan melatih model untuk tugas NSP, kemampuan model dalam menangkap hubungan semantik antar kalimat meningkat, yang pada gilirannya meningkatkan performa pada berbagai tugas hilir. Model yang dilatih dengan NSP dapat lebih baik dalam memahami konteks yang lebih luas dan membuat koneksi logis antara kalimat-kalimat dalam teks. Ini memberikan keunggulan yang signifikan dalam berbagai aplikasi, termasuk pemrosesan teks, analisis sentimen, dan pemahaman teks yang lebih mendalam. Kombinasi dari MLM dan NSP dalam pelatihan BERT menghasilkan model yang kuat dan fleksibel untuk berbagai tugas pemrosesan bahasa alami [17].

2.6. *BERT Tokenizer*

BERT Tokenizer adalah komponen penting dalam pemrosesan bahasa alami yang digunakan untuk memecah teks menjadi token-token yang lebih kecil agar dapat diproses oleh model *BERT* [19]. *BERT Tokenizer* menggunakan algoritma WordPiece untuk membuat kamus token, yang memilih unit subkata untuk kamus dengan tujuan memaksimalkan kemungkinan model bahasa [20]. Dalam proses tokenisasi ini terdapat 2 langkah yaitu pertama, teks dipecah menjadi kata-kata dengan memisahkan tanda baca dan spasi (*pre-tokenization*), dan kedua, setiap kata dipecah menjadi sub-kata atau token *WordPiece* [19]. Sub-kata yang bukan awal kata ditandai dengan simbol khusus (seperti "##" pada BERT) untuk menunjukkan bahwa sub-kata tersebut adalah lanjutan dari token sebelumnya. Proses tokenisasi *WordPiece* ini bisa dilihat dalam Tabel 2.6.

Tabel 2.6 Contoh Tokenisasi *WordPiece*

Teks Awal	john johanson’s
Pre-Tokenization	[john, johanson, ’, s]
Word Piece	[john, johan, ##son, ’, s]

Pada Tabel 2.6, teks awal dilakukan pre-tokenisasi dengan memisahkan setiap kata dan tanda baca. Selanjutnya, dilakukan proses *WordPiece* di mana token dipisahkan menjadi sub kata dengan dengan menambahkan tanda "##" pada token yang bukan merupakan awalan. Dalam contoh ini, kata “##son” merupakan lanjutan dari kata “johan”. Pendekatan ini memungkinkan *BERT* untuk menangani berbagai jenis kata, termasuk kata-kata yang tidak dikenal selama pelatihan, dengan memecahnya menjadi unit sub-kata yang dikenal [19].

2.7. Metrik Evaluasi

Metrik evaluasi digunakan untuk mengukur kinerja suatu model atau sistem dalam menyelesaikan tugas tertentu. Pemilihan metrik yang tepat sangat penting untuk memahami sejauh mana model atau sistem dapat memberikan hasil yang baik. Beberapa metrik evaluasi umum diantaranya, presisi, recall, dan f1-score. Terdapat istilah yang sering digunakan dalam metrik evaluasi yaitu:

1. *True Positive* (TP)

   True Positive merupakan kasus dimana model dapat memprediksi kelas positif dengan benar.

2. *False Positive* (FP)

   False Positive merupakan kasus dimana model salah dalam memprediksi kelas positif.

3. *True Negative* (TN)

   True Negative merupakan kasus dimana model dapat memprediksi kelas negatif dengan benar.

4. *False Negative* (FN)

   False Positive merupakan kasus dimana model salah dalam memprediksi kelas negatif.

2.7.1. Presisi

Presisi mengukur sejauh mana kesesuaian antara data yang diminta dengan prediksi yang diberikan oleh model. Nilai presisi yang tinggi mencerminkan tingkat rendahnya false positif [21]. Presisi dapat dihitung dengan persamaan (2.6).

$$Presisi = \frac{TP}{TP + FP} \tag{2.6}$$

Keterangan:
TP = *True Positive*
FP = *False Positive*

2.7.2. Recall

Recall mengukur sejauh mana pengamatan positif dapat diprediksi dengan tepat dari total pengamatan yang ada pada kelas yang sebenarnya. Nilai recall yang tinggi menandakan bahwa kemungkinan false negatif rendah [21]. Untuk mendapatkan nilai recall dapat menggunakan persamaan (2.7).

$$Recall = \frac{TP}{TP + FN} \tag{2.7}$$

Keterangan:
TP = *True Positive*
FP = *False Negative*

2.7.3. F1-Score

F1 Score adalah metrik evaluasi yang menggabungkan nilai presisi dan recall untuk memberikan gambaran menyeluruh tentang kinerja model. Ini dihitung dengan menggunakan rata-rata harmonik dari presisi dan recall, memberikan bobot lebih besar pada nilai yang lebih rendah [21]. Untuk mendapatkan nilai F1-Score dapat menggunakan persamaan (2.8).

$$F1\text{-}Score = 2 * \frac{Recall * Presisi}{Recall + Presisi} \quad (2.8)$$

2.8. Penelitian Terkait

Pada penelitian Solihin & Budi [2], melakukan penelitian tentang perekaman penegakan hukum berdasarkan dokumen putusan pengadilan menggunakan ekstraksi informasi berbasis aturan. Penelitian ini mengusulkan ekstraksi informasi berbasis aturan untuk dokumen putusan pengadilan untuk mengeksplorasi dan mengembangkan pencatatan penegakan hukum di Indonesia. Proses ini menggunakan tiga langkah, yaitu mengidentifikasi ekstraksi struktur, tokenisasi, dan mengekstraksi entitas. Dengan menggunakan 150 dataset dokumen putusan pengadilan, percobaan yang telah dilakukan berhasil mencapai nilai recall sebesar 0.82, presisi 0.96, dan f-score 0.89. Hasil ini mengindikasikan bahwa pendekatan ekstraksi informasi berbasis aturan dapat sukses diterapkan dalam konteks hukum, memberikan dukungan signifikan terhadap pengembangan catatan penegakan hukum di Indonesia.

Pada penelitian Wang et al. [5], mengusulkan sebuah pendekatan baru dengan menggabungkan beberapa model salah satu nya model pre-trained untuk pemahaman konteks dari dokumen hukum Brasil. Dataset yang digunakan dalam penelitian ini berasal dari *LeNER-Br* dengan total 10392 dokumen putusan hukum. Dari jumlah tersebut 9003 dokumen digunakan untuk pelatihan dan 1389 dokumen digunakan untuk pengujian. Dataset ini akan digunakan untuk melatih beberapa model diantaranya BERT(Dinamis)-IDCNN-CRF, BERT(Dinamis)-BiLSTM-CRF, BERT-IDCNN-CRF, BERT-BiLSTM-CRF, STM. Dari proses pelatihan menunjukkan bahwa STM lebih baik dan skor F1 adalah 93,23%, yang memberikan dasar penting untuk tugas-tugas NER [5].

Pada penelitian Nuranti & Yulianti [4], melakukan sebuah penelitian tentang Pengakuan Badan Hukum dalam Dokumen Putusan Pengadilan di Indonesia

Menggunakan Pendekatan Bi-LSTM dan CRF. Penelitian ini menguji efektivitas beberapa metode deep learning untuk mengenali sepuluh badan hukum dalam dokumen putusan pengadilan di Indonesia. Dalam tugas tersebut, pada penelitian ini ditemukan bahwa kombinasi metode Bi-LSTM dan CRF mencapai nilai F-1 tertinggi yaitu 0.83. Metode ini mengungguli metode deep learning lainnya (CNN, LSTM, LSTM+CRF, dan Bi-LSTM) sebesar 2 -12% dan metode machine learning (SVM end CRF) sebesar 1 - 76%. Hasil penelitian menunjukkan bahwa model ini dapat digunakan untuk mengidentifikasi informasi yang relevan tentang domain hukum di Indonesia.

Yang & Agrawal [7], melakukan penelitian tentang ekstraksi entitas kompleks dalam dokumen hukum. Dalam penelitian ini, diusulkan suatu sistem baru yang menggabungkan deteksi objek untuk Analisis Tata Letak Dokumen (DLA) dengan pembelajaran lemah ter awasi guna mengatasi tantangan mengekstraksi entitas kompleks yang terputus-putus dalam dokumen hukum. Objek tersebut kemudian akan diklasifikasikan dan dieksplorasi entitas nya dengan membangun aturan. Dataset yang digunakan dibagi menjadi 2 yaitu dataset dengan label Pseudo dan Gold, Dataset Pseudo berjumlah 4000 dokumen dengan menggunakan pendekatan yang diajukan (proposed approach), bukan anotasi manusia. Sedangkan yang Gold berjumlah 706 dokumen dengan anotasi yang dilakukan oleh tim ahli. Hasil eksperimen menunjukkan bahwa model yang dilatih hanya pada label Pseudo memiliki kinerja yang lebih baik dibandingkan dengan label Gold. Setelah diuji coba dengan kedua dataset tersebut, dilakukan pengujian dengan melatih model menggunakan dataset gabungan label Pseudo + Gold, dan hasilnya tetap sama, yaitu label Pseudo masih memberikan kinerja tertinggi yaitu presisi 0.655, recall 0.643, dan F1 Score 0.654. Hal ini menunjukkan bahwa model dengan label Pseudo dapat mengidentifikasi informasi yang ada pada label Gold [7].

Tabel 2.7 Penelitian Terkait NER Bidang Hukum

NO	Penelitian, Tahun	Permasalahan	Metode/Solusi	Hasil
1.	Solihin & Budi [2], 2018	Pencatatan Penegakan Hukum Berdasarkan	Rule-based Information Extraction	Telah Menghasilkan

		Dokumen Putusan Pengadilan di Indonesia		recall 0.82, presisi 0.96, F-Score 0.89.
2.	Wang et al. [5], 2020.	Ekstraksi Entitas dari Teks Hukum Brazil	Gabungan beberapa model diantaranya BERT(Dinamis)-IDCNN-CRF, BERT(Dinamis)-BiLSTM-CRF, BERT-IDCNN-CRF, BERT-BiLSTM-CRF, STM	STM merupakan model terbaik dibandingkan dengan model lain nya.
3.	Nuranti & Yulianti [4] , 2021	Ekstraksi Entitas Hukum dalam Dokumen Putusan Pengadilan di Indonesia	Menggunakan *CNN*, LSTM, Bi-LSTM, LSTM-CRF, Bi-LSTM-CRF.	Kombinasi metode Bi-LSTM dan CRF mencapai nilai F1-Score tertinggi yaitu 0.83.
4.	Yang & Agrawal [7], 2023	Mengekstraksi Entitas Kompleks dalam Dokumen Hukum Amerika	Rule-based Information Extraction	Dataset label pseudo dengan nilai presisi 0.665, recall 0.643, dan F1-Score 0.654.

Berdasarkan uraian penelitian Tabel 2.7, *NER* sudah digunakan dalam banyak penelitian, salah satu nya studi kasus Pencatatan Penegakan Hukum berdasarkan dokumen putusan pengadilan pada penelitian [2] dengan menerapkan metode rule based. Dari Tabel 2.7. Penggunaan metode rule based dapat diimplementasikan dalam domain hukum dan mendukung pengembangan catatan penegakan hukum di Indonesia. Penggunaan metode berbasis aturan juga pernah dilakukan pada

penelitian [7], namun pada penelitian [7] terdapat sistem baru yang menggabungkan deteksi objek untuk *Document Layout Analysis (DLA)* dengan pembelajaran lemah ter-awasi untuk mengatasi tantangan mengekstraksi entitas kompleks yang terputus-putus dalam dokumen hukum Amerika. Fokus penelitian [7] juga berbeda dari penelitian [2], penelitian [7] berfokus pada dataset yang telah dilakukan anotasi secara manual *(GOLD)* dan otomatis *(Pseudo)*, sehingga menghasilkan model terbaik dengan menggunakan dataset dengan label otomatis *(Pseudo)*.

Selain penelitian [2] terdapat juga penelitian ekstraksi entitas dalam Dokumen Putusan Pengadilan berbahasa Indonesia seperti pada penelitian [4] [5]. Kedua penelitian tersebut memiliki perbedaan dimana penelitian [4] menerapkan kombinasi metode *Bidirectional Long Short Term Memory (Bi-LSTM)* dan *Conditional Random Field (CRF)*. Sedangkan pada penelitian [5] menggabungkan *Impulse Detection Convolution Neural Network (IDCNN)* dan *Bi-LSTM*, penelitianini mengembangkan model pelabelan urutan yang dapat diskalakan yang diberi nama *Sequence Tagging Model (STM)*. Namun dari semua penelitian NER dalam dokumen putusan hukum di Indonesia belum ada yang menggunakan metode berbasis Transformers seperti pada penelitian [5].

Metode berbasis Transformers sudah banyak digunakan untuk tugas NER. Seperti Pada penelitian yang dilakukan oleh sun et al., menerapkan NER pada domain biomedis menggunakan *BERT*. Penelitian ini bertujuan untuk meningkatkan pemahaman mesin terhadap teks, memungkinkan mesin memperoleh lebih banyak pengetahuan sebelumnya melalui penggunaan query yang dirancang dengan baik, dan menghilangkan kebutuhan akan proses decoding seperti *Conditional Random Fields (CRF)*. Dalam penelitian ini, 6 dataset BioNER digunakan, termasuk *BC4CHEMD*, *BC5CDR-Chem*, *BC5CDR-Disease*, *NCBI-Disease*, *BC2GM*, dan *JNLPBA*. Penelitian membandingkan performa empat model, yaitu Bio*BERT-Softmax*, *BioBERT-CRF*, *BioBERT-BiLSTM-CRF*, dan *BioBERT-MRC*. Dari keempat model tersebut, tiga diantaranya masih menggunakan model konvensional, sedangkan satu model mengadopsi pendekatan *Machine Reading Comprehension (MRC)*. Hasil penelitian menunjukkan bahwa model *BioBERT-MRC* memberikan performa terbaik dengan nilai rata-rata F1 sebesar 92.70 untuk dataset *BC4CHEMD*, 93.92 untuk dataset *BC5CDR-Chem*,

87.56 untuk dataset *BC5CDR-Disease*, 89.39 untuk dataset *NCBI-Disease*, 85.11 untuk dataset *BC2GM*, dan 78.45 untuk dataset *JNLPBA* [9].

Luthfi et al. [22], melakukan penelitian yang bertujuan untuk mengembangkan teknik *Natural Language Processing* (NLP) canggih yang menggunakan NER berbasis BERT untuk mengidentifikasi dan mengotentikasi narator Hadits secara otomatis. Teknik NER yang diusulkan menambahkan pengklasifikasian feed-forward pada lapisan terakhir model BERT dilatih sebelumnya, dan solusi yang dihasilkan menerima skor F1 keseluruhan sebesar 99,63 persen dalam pengujian menggunakan Cahya/bert-base-indonesian-1.5G, serta skor F1 98,27 persen pada identifikasi narator Hadits menggunakan teks Hadits lain.

Khairunnisa et al. [23], melakukan penelitian dengan tujuan meningkatkan dataset bahasa Indonesia untuk ekstraksi entitas dengan menggunakan model *Long Short Term Memory (LSTM)* dan *Conditional Random Fields (CRF)* serta menerapkan model multi bahasa seperti *BERT* dan *XLM-roBERTa*. Dataset yang digunakan pada penelitian ini adalah dataset yang diterbitkan oleh Syaifuddin dan Nur-widiyantoro pada tahun 2016 dengan domain berita yang berisikan sekitar 2000 kalimat. Pada dataset tersebut Khairunnisa *et al.* menemukan bahwa terdapat ketidakkonesistensinan dalam dataset tersebut, sehingga akan dilakukan anotasi ulang oleh 3 orang penutur asli. Dari hasil penelitian ini mendapatkan nilai F1 terbaik sebesar 90.85 untuk model Bi-LSTM-CRF dan 94.90 untuk kombinasi IndoBERT dengan Bi-LSTM-CRF.

Tabel 2.8 Penelitian Terkait *BERT*

NO	Penelitian, Tahun	Permasalahan	Metode/solusi	Hasil
1.	Sun et al. [9], 2021	Pengenalan entitas biomedias dengan pendekatan *Machine Reading*	Menggunakan metode *BioBERT-Softmax, BioBERT-CRF, BioBERT-BiLSTM-CRF, dan BioBERT-MRC*	.Hasil penelitian menunjukkan bahwa BioBERT-MRC memberikan performa terbaik.

		Comprehension (MRC).		
2.	Luthfi et al [22], 2022	Pengenalan entitas untuk identifikasi perawi hadist otomatis	Menggunakan *BERT*	Menghasilkan nilai F1-Score sebesar 98.27 persen.
3.	Khairunnisa et al. [23], 2023	Peningkatan Dataset dan Transfer Multibahasa untuk Pengenalan Entitas pada berita bahasa indonesia.	Menggunakan BiLSTM, CRF, *IndoBERT* dan *XLM-RoBERTa*	Dari hasil penelitian ini Bi-LSTM-CRF mendapatkan nilai F1 terbaik sebesar 90.85

Berdasarkan uraian penelitian Tabel 2.8, *BERT* sudah digunakan dalam banyak penelitian untuk penerapan NER. BERT menunjukkan kinerja terbaik dibandingkan dengan metode lain, seperti yang dijelaskan dalam penelitian [9] dan [23], yang membandingkan beberapa model berbasis transformers. Penelitian tersebut juga mencatat bahwa model menggunakan pendekatan Machine Reading Comprehension (MRC), seperti BERT, mencapai kinerja terbaik dibandingkan dengan model konvensional seperti *Softmax*, CRF, dan BiLSTM-CRF, sebagaimana dijabarkan dalam penelitian [9].

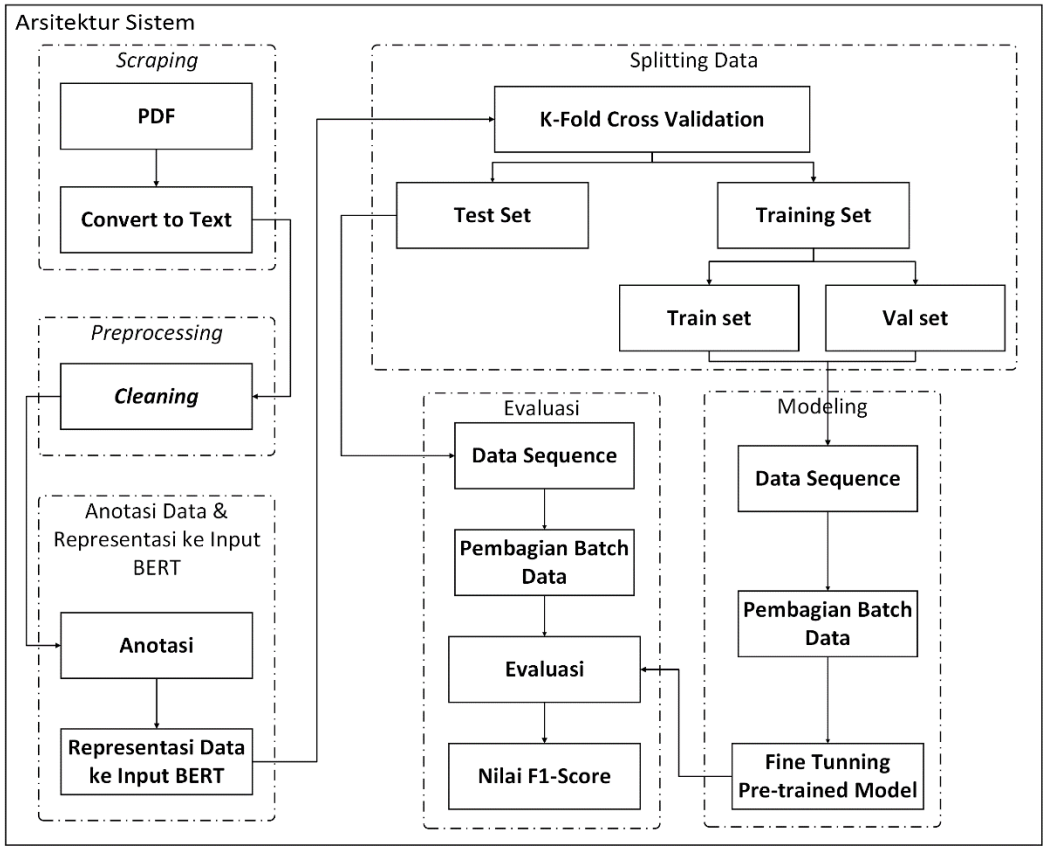
Penelitian lainnya, seperti [22], menggunakan model berbasis transformers, khususnya BERT, dan mencapai performa yang baik. Sebaliknya, penelitian [23] lebih berfokus pada peningkatan dataset yang kurang konsisten dengan melakukan anotasi ulang secara manual. Proses anotasi ulang tersebut diikuti oleh pelatihan beberapa model, dan hasilnya menunjukkan bahwa kombinasi BiLSTM-CRF dengan IndoBERT memberikan performa terbaik. Dari hasil penelitian

sebelumnya, dapat disimpulkan bahwa BERT dapat diandalkan untuk tugas NER dan mampu memberikan hasil yang memuaskan.

BAB III METODOLOGI PENELITIAN

3.1. Arsitektur Sistem

Desain sistem yang digunakan dijelaskan melalui gambar berikut ini. Pada bagian ini, akan diuraikan lebih detail mengenai struktur dan komponen-komponen yang terdapat dalam arsitektur sistem yang digunakan. Rincian lebih lanjut bisa dilihat dalam Gambar 3.1. Input yang digunakan dalam sistem ini nanti nya berupa dokumen maupun teks.



Gambar 3.1 Arsitektur Sistem

3.1.1. Scraping

Dalam tahap pertama dengan melakukan scraping untuk mendapatkan dokumen putusan hukum melalui website www.putusan3.mahkamahagung.go.id, proses ini dilakukan dengan menggunakan library Python *Beautiful Soup*. *Beautiful Soup* digunakan untuk mengekstrak informasi dari URL untuk mendapatkan file PDF dari dokumen putusan hukum serta mendownload nya, selanjutnya yaitu proses konversi pdf ke text, dalam hal ini dengan menggunakan library Python *PyPDF2*. Hasil konversi ke dalam bentuk teks ini disimpan terlebih dahulu ke

dalam bentuk list untuk setiap dokumen selanjutnya di disimpan dalam bentuk *Comma Separated Value* (CSV). Contoh dataset bisa dilihat dalam Tabel 3.1.

Tabel 3.1 Contoh Dokumen Putusan Pengadilan "NOMOR : 90 / PID. Sus / 2018 / PN. Mjn"

Text
Mahkamah Agung Republik Indonesia\nMahkamah Agung Republik Indonesia\nMahkamah Agung Republik Indonesia\nMahkamah Agung Republik Indonesia\nMahkamah Agung Republik Indonesia\nDirektori Putusan Mahkamah Agung Republik Indonesia\nputusan.mahkamahagung.go.id\nP U T U S A N\nNOMOR : 90 / PID.Sus / 2018 / PN.Mjn\nDEMI KEADILAN BERDASARKAN KETUHANAN YANG MAHA ESA\nPengadilan Negeri Majene yang mengadili perkara pidana dengan acara\nbiasa pada tingkat pertama dengan susunan persidangan Majelis, telah\nmenjatuhkan putusan sebagai berikut dalam perkara Terdakwa :\nNama Lengkap :ANDIKA SAPUTRA ALIAS DIKA BIN ABDUL\nRASID ;\nTempat lahir :Majene ;

3.1.2. Preprocessing

Dalam tahap preprocessing, data diolah untuk menghilangkan karakter-karakter yang tidak diperlukan dari hasil konversi pdf ke teks dan disiapkan untuk langkah selanjutnya. Dari hasil preprocessing yang dilakukan bisa dilihat dalam Tabel 3.2.

Tabel 3.2 Hasil *Preprocessing* untuk dokumen "NOMOR : 90 / PID. Sus / 2018 / PN. Mjn"

Sebelum Cleaning	Cleaning
Mahkamah Agung Republik Indonesia\nMahkamah Agung Republik Indonesia\nMahkamah Agung Republik Indonesia\nMahkamah Agung Republik Indonesia\nMahkamah Agung Republik Indonesia\nDirektori Putusan Mahkamah Agung Republik Indonesia\nputusan.mahkamahagung.go.id\nP	putusan nomor : 90 / pid. sus / 2018 / pn. mjn demi keadilan berdasarkan ketuhanan yang maha esa pengadilan negeri majene yang mengadili perkara pidana dengan acara biasa pada tingkat pertama dengan susunan persidangan majelis, telah

U T U S A N\nNOMOR : 90 / PID.Sus / 2018 / PN.Mjn\nDEMI KEADILAN BERDASARKAN KETUHANAN YANG MAHA ESA\nPengadilan Negeri Majene yang mengadili perkara pidana dengan acara\nbiasa pada tingkat pertama dengan susunan persidangan Majelis, telah\nmenjatuhkan putusan sebagai berikut dalam perkara Terdakwa :\nNama Lengkap :ANDIKA SAPUTRA ALIAS DIKA BIN ABDUL\nRASID ;\nTempat lahir :Majene ;	menjatuhkan putusan sebagai berikut dalam perkara terdakwa : nama lengkap : andika saputra alias dika bin abdul rasid ; tempat lahir : majane ;

3.1.3. Anotasi Data

Pada tahap ini, data hasil preprocessing dianotasi menggunakan teknik *Inside-Outside-Beginning* (IOB) sesuai dengan penelitian [4]. Teknik IOB merupakan metode anotasi yang digunakan untuk menandai entitas atau frasa tertentu dalam sebuah teks. Metode ini mengategorikan setiap kata dalam teks ke dalam tiga kategori utama: "B" (*Beginning*) digunakan untuk menandai awal dari suatu entitas atau frasa, "I" (*Inside*) digunakan untuk menandai bagian dalam dari suatu entitas atau frasa setelah kata pertama yang sudah ditandai sebagai awal ("B"), dan "O" (*Outside*) digunakan untuk menandai kata-kata diluar entitas atau frasa yang sedang dianotasi. Dalam penelitian ini label yang digunakan untuk anotasi sebanyak 24 label untuk entitas dan 1 label untuk bukan entitas yaitu ‘O’, label yang digunakan bisa dilihat dalam Tabel 3.3. Dengan label tersebut dapat dilakukan contoh anotasi dengan menggunakan IOB bisa dilihat dalam Tabel 3.4.

Tabel 3.3 Label Dataset

No	Label	Entitas
1.	B_VERN, I_VERN	Verdict Number (Nomor Putusan)
2.	B_DEFN, I_DEFN	Defendant Name (Nama Terdakwa)
3.	B_CRIA, I_CRIA	Critical Action (Tindak Pidana)
4.	B_ARTV, I_ARTV	Article Violation (Pelanggaran KUHP)

5.	B_PENA, I_PENA	Penalties (Tuntutan Hukum)
6.	B_PUNI, I_PUNI	Punishment (Putusan Hukum)
7.	B_TIMV, I_TIMV	Date of Verdict (Tanggal Putusan)
8.	B_JUDP, I_JUDP	Presiding Judge (Nama Hakim Ketua)
9.	B_JUDG, I_JUDG	Judge (Nama Hakim Anggota)
10.	B_REGI, I_REGI	Registrar (Nama Panitera)
11.	B_PROS, I_PROS	Prosecutor (Nama Penuntut Umum)
12.	B_ADVO, I_ADVO	Advocat (Nama Penasihat/Pengacara)

Tabel 3.4 Contoh Anotasi Dengan Menggunakan IOB untuk dokumen “NOMOR : 90 / PID. Sus / 2018 / PN. Mjn”

1	VERN	Text	putusan nomor : 90 / pid. sus / 2018 / pn. mjn demi
		Label	O O O B_VERN I_VERN I_VERN I_VERN I_VERN I_VERN I_VERN I_VERN I_VERN O
2	DEFN	Text	nama lengkap : andika saputra alias dika bin abdul rasid ;
		Label	O O O B_DEFN I_DEFN I_DEFN I_DEFN I_DEFN I_DEFN O
3	CRIA	Text	melakukan tindak pidana sebagaimana disebutkan dalam pertama yaitu “ dengan sengaja dan tanpa hak mendistribusikan dan / atau mentransmisikan dan / atau membuat dapat diaksesnya informasi elektronik dan / atau dokumen elektronik yang memiliki muatan yang melanggar kesusilaan ”
		Label	O O O O O O O O O B_CRIA I_CRIA O
4	ARTV	Text	sebagaimana diatur dan diancam pidana dalam pasal 27 ayat (1) jo pasal 45 ayat (1) uu ri no. 11 tahun 2008 jo

			uu ri no. 19 tahun 2016 tentang informasi transaksi elektronik (ite) ;
		Label	O O O O O O B_ARTV I_ARTV O
5	PENA	Text	pidana penjara selama 2 (dua) tahun
		Label	O O O B_PENA I_PENA I_PENA I_PENA I_PENA
6	PUNI	Text	pidana penjara selama 2 (dua) tahun ;
		Label	O O O B_PUNI I_PUNI I_PUNI I_PUNI I_PUNI O
7	TIMV	Text	pengadilan negeri majene pada hari rabu, tanggal 13 februari 2019
		Label	O O O O O O O B_TIMV I_TIMV I_TIMV
8	JUDP	Text	oleh kami mohammad fauzi salam, s. h., m. h., sebagai hakim ketua majelis
		Label	O O B_JUDP I_JUDP I_JUDP I_JUDP I_JUDP I_JUDP I_JUDP O O O O
9	JUDG	Text	saiful. hs, s. h, m. h., dan nona vivi sri dewi, s. h., masing - masing sebagai hakim anggota
		Label	B_JUDG I_JUDG I_JUDG I_JUDG I_JUDG I_JUDG O B_JUDG I_JUDG I_JUDG I_JUDG I_JUDG I_JUDG O O O O O O
10	REGI	Text	dibantu oleh andi m. syahrul k, s. h, m. h.,, panitera pengganti
		Label	O O B_REGI I_REGI I_REGI I_REGI I_REGI I_REGI I_REGI I_REGI O O
11	PROS	Text	dihadiri oleh andi asben awaluddin, s. h., m. h penuntut umum
		Label	O O B_PROS I_PROS I_PROS I_PROS I_PROS I_PROS I_PROS O O

12	ADVO	Text	terdakwa didampingi oleh penasihat hukum dewi suryaningsih, s. h sebagai advokat pada lembaga bantuan hukum
		Label	O O O O O B_ADVO I_ADVO I_ADVO I_ADVO O O O O

Proses anotasi ini melibatkan 12 entitas yang berbeda, sebagaimana bisa dilihat dalam Tabel 3.4. Proses anotasi dilakukan melalui dua cara, yaitu secara manual dan otomatis dengan menggunakan meta data. Proses manual melibatkan intervensi manusia untuk memberikan penandaan sesuai aturan IOB pada setiap kata dalam teks, yang kemudian diverifikasi oleh mahasiswa Hukum, dengan menggunakan Tools *Doccano*. Sementara itu, proses otomatis menggunakan metadata untuk memberikan penandaan secara otomatis pada beberapa entitas, dengan memanfaatkan informasi yang tersedia dalam metadata tersebut. Hasil dari kedua metode anotasi tersebut menjadikan sebuah teks di pisah menjadi kata yang bisa dilihat dalam Tabel 3.5. Dari hasil tersebut menunjukkan penandaan IOB untuk setiap kata. Secara total, terdapat 24 label yang mewakili entitas yang berhasil diidentifikasi dalam proses anotasi serta 1 label yang bukan merupakan sebuah entitas.

Tabel 3.5 Hasil Anotasi untuk dokumen “NOMOR : 90 / PID. Sus / 2018 / PN. Mjn”

Token	Hasil Anotasi
['putusan', 'nomor', ':', '90', '/', 'pid.', 'sus', '/', '2018', '/', 'pn.', 'mjn', 'demi', 'keadilan', 'berdasarkan', 'ketuhanan', 'yang', 'maha', 'esa', 'pengadilan', 'negeri', 'majene', 'yang', 'mengadili', 'perkara', 'pidana', 'dengan', 'acara', 'biasa', 'pada', 'tingkat', 'pertama', 'dengan', 'susunan', 'persidangan', 'majelis,', 'telah', 'menjatuhkan', 'putusan', 'sebagai', 'berikut', 'dalam',	['O', 'O', 'O', 'B_VERN', 'I_VERN', 'I_VERN', 'I_VERN', 'I_VERN', 'I_VERN', 'I_VERN', 'I_VERN', 'I_VERN', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'B_DEFN', 'I_DEFN', 'I_DEFN', 'I_DEFN', 'I_DEFN', 'I_DEFN', 'O', 'O', 'O', 'O', 'O', 'O']

'perkara', 'terdakwa', ':', 'nama', 'lengkap', ':', 'andika', 'saputra', 'alias', 'dika', 'bin', 'abdul', 'rasid', ';', 'tempat', 'lahir', ':', 'majene', ';']	

3.1.4. Representasi Data ke Input BERT

Dalam tahap ini, data yang telah dianotasi dalam bentuk token akan diproses dan digabungkan menjadi satu string untuk setiap kalimat. Penentuan kalimat dilakukan berdasarkan keberadaan titik koma sebagai batas pembentukan sebuah kalimat. Data dalam bentuk token diambil dari Tabel 3.5. Hasil dari representasi data ke input BERT dapat dilihat dalam Tabel 3.6.

Tabel 3.6 Hasil Representasi data ke input BERT untuk dokumen "NOMOR : 90 / PID. Sus / 2018 / PN. Mjn"

Text	Label
putusan nomor : 90 / pid. sus / 2018 / pn. mjn demi keadilan berdasarkan ketuhanan yang maha esa pengadilan negeri majene yang mengadili perkara pidana dengan acara biasa pada tingkat pertama dengan susunan persidangan majelis, telah menjatuhkan putusan sebagai berikut dalam perkara terdakwa : nama lengkap : andika saputra alias dika bin abdul rasid ;	O O O B_VERN I_VERN I_VERN I_VERN I_VERN I_VERN I_VERN I_VERN I_VERN O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O B_DEFN I_DEFN I_DEFN I_DEFN I_DEFN I_DEFN O
Tempat lahir : majene ;	O O O O O

Setelah direpresentasikan dengan input *BERT*, yaitu dengan mengubah teks kembali menjadi bentuk string, diperoleh sebanyak 137.238 baris. Selanjutnya, dilakukan penghapusan kalimat yang tidak mengandung entitas khusus atau hanya memiliki label O. Langkah ini dilakukan untuk meminimalisir ketidakseimbangan antara data yang memiliki entitas khusus dan data yang hanya memiliki label O, karena tidak semua kalimat mengandung entitas khusus. Dari proses penghapusan

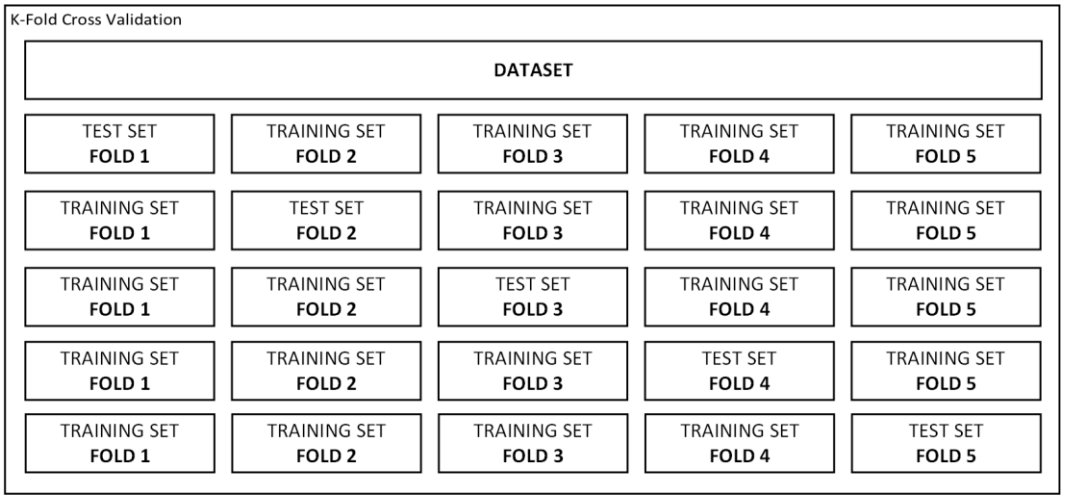
ini, dihasilkan sebanyak 12.024 baris. Hasil dari proses penghapusan kalimat yang tidak mengandung entitas khusus bisa dilihat dalam Tabel 3.7.

Tabel 3.7 Hasil penghapusan kalimat yang tidak memiliki entitas khusus untuk dokumen "NOMOR : 90 / PID. Sus / 2018 / PN. Mjn"

Text	Label
putusan nomor : 90 / pid. sus / 2018 / pn. mjn demi keadilan berdasarkan ketuhanan yang maha esa pengadilan negeri majene yang mengadili perkara pidana dengan acara biasa pada tingkat pertama dengan susunan persidangan majelis, telah menjatuhkan putusan sebagai berikut dalam perkara terdakwa : nama lengkap : andika saputra alias dika bin abdul rasid ;	O O O B_VERN I_VERN I_VERN I_VERN I_VERN I_VERN I_VERN I_VERN I_VERN O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O B_DEFN I_DEFN I_DEFN I_DEFN I_DEFN I_DEFN O

3.1.5. Splitting Dataset

Pada tahap ini, dataset akan dibagi menjadi training set, test set dan validation set. Validation set di ambil dari training set dengan persentase 90% untuk training set dan 10 % untuk validation set. Untuk pembagian training set dan test set akan dilakukan dengan menggunakan metode *K-Fold Cross-Validation* dengan nilai K sebesar 5. Pemilihan K=5 umum digunakan dalam penelitian. Dalam *K-Fold Cross-Validation*, data training dibagi menjadi lima subset atau fold yang memiliki ukuran yang sama. Selama proses training, satu fold dijadikan data test, sementara empat fold lainnya digunakan sebagai data training. Proses ini diulangi sebanyak lima kali, sehingga setiap fold berperan sebagai data test satu kali. Proses 5-*Fold Cross Validation* bisa dilihat dalam Gambar 3.2.



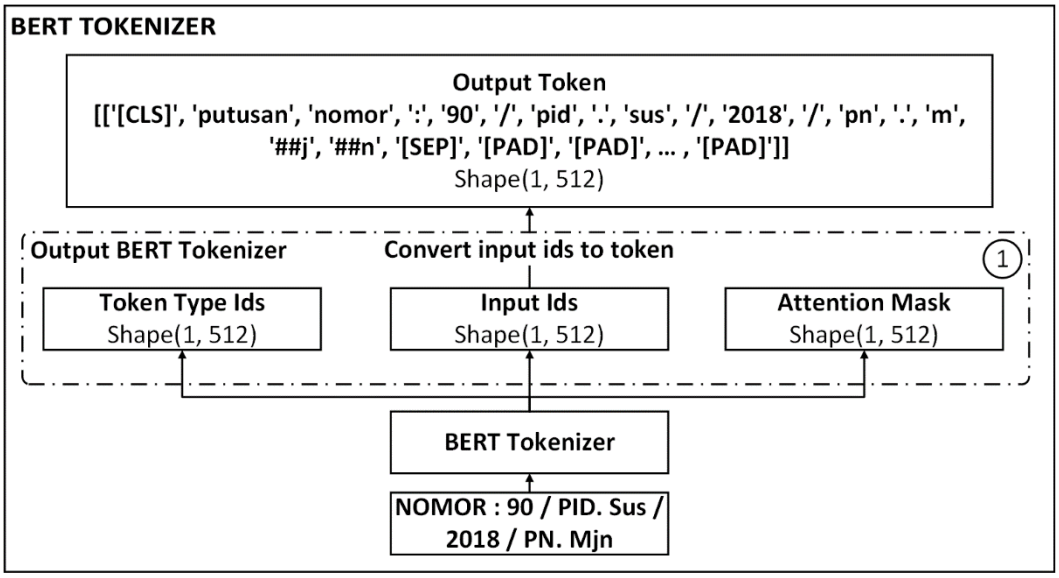
Gambar 3.2 Proses 5-*Fold Cross Validation*

Tujuan dari proses ini adalah untuk meningkatkan reliabilitas dan generalisasi model, menghindari overfitting, serta memastikan bahwa model dapat diuji dengan berbagai variasi data. Selain itu, metode *K-Fold Cross-Validation* juga memungkinkan untuk mengukur performa model dengan lebih akurat karena setiap titik data memiliki kesempatan untuk muncul dalam training set dan test set.

3.1.6. Modeling

3.1.6.1. Data Sequence

Pada proses DataSequence, teks akan diubah menjadi representasi numerik menggunakan *BERT Tokenizer*, sementara label akan diubah menjadi ID numerik menggunakan fungsi align_label. Proses transformasi teks ke representasi numerik menggunakan *BERT Tokenizer* ditunjukkan pada Gambar 3.3 dengan contoh input "NOMOR: 90/PID. Sus/2018/PN. Mjn". *BERT Tokenizer* menghasilkan tiga output: Input IDs, Token Type IDs, dan *Attention Mask*. Setiap output memiliki peran masing-masing dalam input ke model BERT. Input IDs digunakan untuk *word embedding*, *Attention Mask* digunakan untuk memperhatikan token yang penting dan mengabaikan token yang tidak relevan seperti '[PAD]', sementara Token Type IDs digunakan untuk memahami hubungan antar kalimat sehingga model dapat memproses dua kalimat secara langsung. Namun, dalam penelitian ini, *Token Type IDs* tidak dimanfaatkan. Hasil dari *BERT Tokenizer* ini menghasilkan dimensi (1, 512) dimana 1 merupakan jumlah data dan 512 merupakan jumlah token maksimal dari *BERT Tokenizer*. Output dari *BERT Tokenizer* bisa dilihat dalam Tabel 3.8.

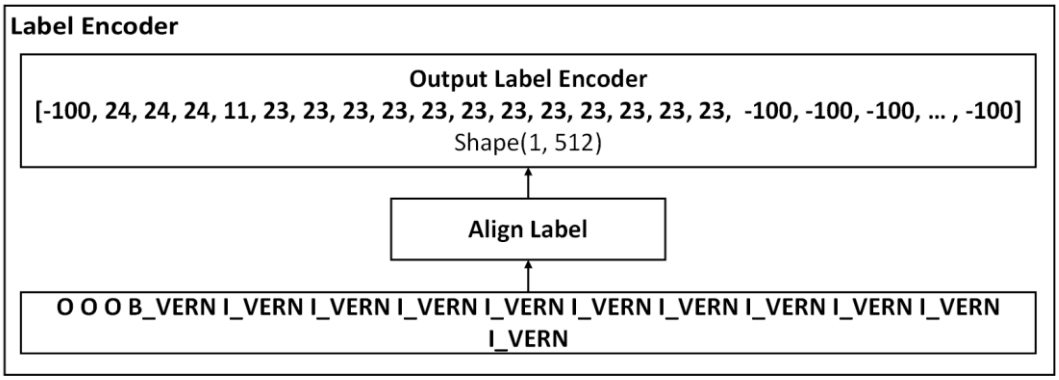


Gambar 3.3 BERT Tokenizer

Tabel 3.8 Hasil BERT Tokenizer

Text	Output	Nilai
Putusan Nomor : 90 / PID. Sus / 2018 / PN. Mjn	*input_ids*	[[3, 7050, 3286, 30, 5754, 19, 4200, 18, 3451, 19, 6062, 19, 6237, 18, 55, 954, 935, 4, 0, 0, 0,…]]
	attention_mask	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0,…]

Proses perubahan label menjadi representasi numerik bisa dilihat dalam Gambar 3.4. Contoh label yang digunakan adalah label dari teks yang telah digunakan sebagai contoh sebelumnya. Hasilnya adalah ID label yang telah dibuat menggunakan fungsi Align Label.



Gambar 3.4 Align Label

3.1.6.2. Pembagian Batch Data

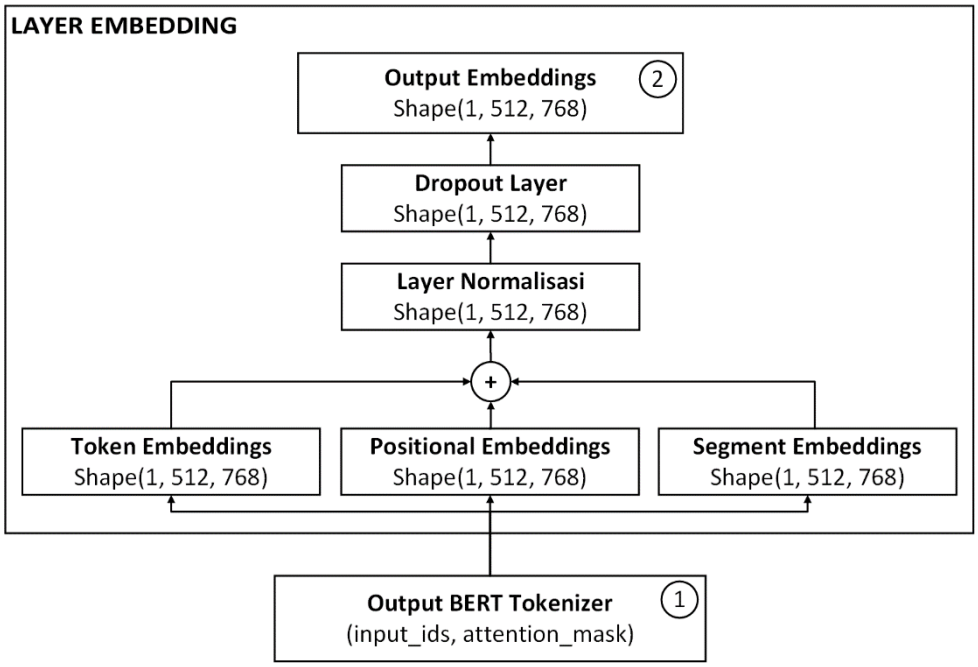
Pada tahap ini, data hasil proses sequence selanjutnya dibagi menjadi beberapa batch data. Pembagian batch data berguna untuk mengatur jumlah data yang akan diproses setiap iterasi. Parameter yang digunakan terkait jumlah batch telah ditentukan dalam skenario penelitian. Setelah proses pembagian batch data selesai selanjutnya data dapat di proses untuk dilakukan fine tuning model.

3.1.6.3. Fine Tuning Model

Pada tahap fine tuning model *BERT*, terdapat beberapa proses yang dilakukan. Berikut adalah tahap-tahap yang dilakukan:

1. *Embedding Layer*

Embedding layer merupakan tahap awal pada proses fine tuning BERT, di tahap ini teks yang telah di pecah menjadi token pada tahap tokenisasi menggunakan BERT Tokenizer selanjutnya di proses pada Embedding layer yang menghasilkan 3 buah nilai diantaranya *Token Embedding*, *Positional Encoding*, dan *Segment Embedding*, namun dalam proses NER *Segment Embedding* tidak digunakan. Contoh proses ini bisa dilihat dalam Gambar 3.5. Ketiga jenis embedding ini memiliki dimensi yang sama, yaitu (1, 512, 768), dimana angka satu merupakan jumlah data, angka 512 merupakan jumlah token, dan angka 768 merupakan dimensi dari representasi dari setiap token. Ketiga embedding ini digabungkan untuk menghasilkan Output Embedding yang akan menjadi input untuk *Encoder Layer*. Output Embedding ini merupakan representasi numerik dari teks yang siap untuk diproses lebih lanjut oleh model BERT.

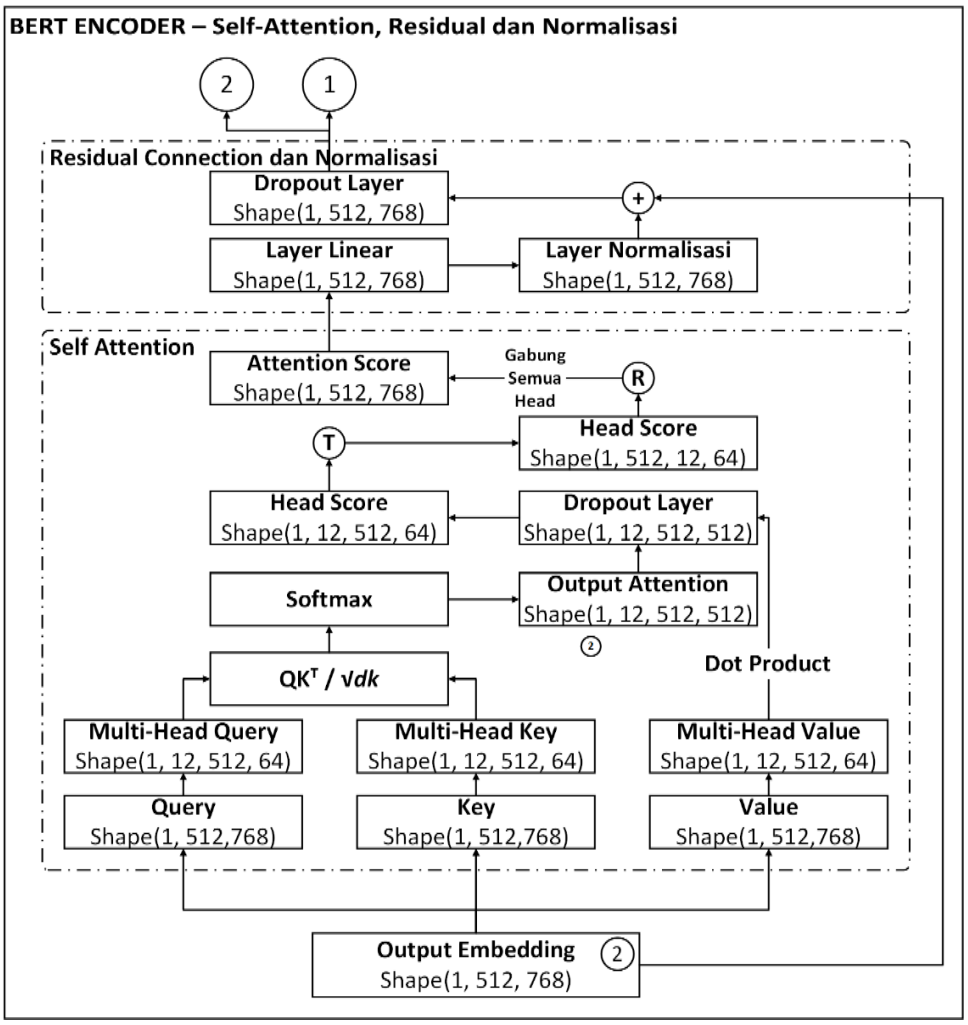


Gambar 3.5 Contoh *Embedding Layer*

2. *Encoder Layer*

Pada *Encoder Layer*, terdapat beberapa sub *layer* yaitu *Self-Attention*, *Feed Forward Network*, *Residual Connection* dan Normalisasi. *Encoder* layer menerima input dari hasil embedding yang telah ditokenisasi dan diubah menjadi vektor dengan dimensi (1, 512, 768). Vektor ini kemudian diproses menggunakan *Self-Attention*, yang memungkinkan model untuk memfokuskan pada setiap bagian dari input dengan membuat tiga komponen yaitu *Query*, *Key*, dan *Value*. Dari 3 komponen tersebut masing-masing dibagi dengan *head attention* yaitu 12 sehingga menghasilkan shape (1, 12, 512, 64) dimana 1 merupakan jumlah data, 12 merupakan jumlah *head attention*, 512 merupakan jumlah token, dan 64 merupakan hasil pembagian representasi token yang awalnya 768 dengan *head attention*. Setelah itu, dilakukan penerapan persamaan (2.1) dengan melakukan *dot product* antara *Query* dan *Key* yang telah ditransposisikan, dan hasilnya dibagi dengan akar kuadrat dari dimensi *Key* untuk mendapatkan skor *attention*. Skor ini kemudian dinormalisasi menggunakan fungsi *softmax* untuk mendapatkan distribusi probabilitas yang menunjukkan seberapa penting setiap kata terhadap kata lain dalam kalimat. Fungsi *softmax* dapat dihitung dengan persamaan (2.2). Output dari *Self-Attention* kemudian diolah melalui *Residual Connection*, Normalisasi dan

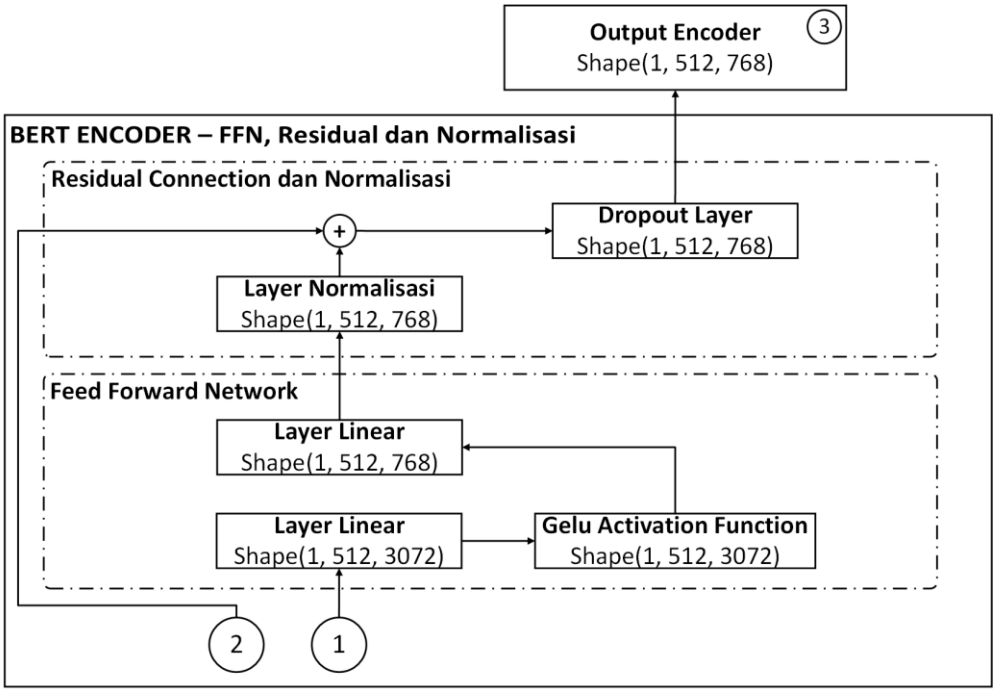
Dropout sehingga menghasilkan dimensi (1, 512, 768). Contoh alur proses dari *Self-Attention*, *Residual Connection* dan Normalisasi ini bisa dilihat dalam Gambar 3.6.



Gambar 3.6 Contoh Proses *Self-Attention*, *Residual Connection* dan Normalisasi

Proses selanjutnya adalah melalui Feed Forward Network dengan linear layer pertama yang menghasilkan dimensi (1, 512, 3072), di mana 3072 merupakan panjang weight dari linear layer pertama pada Feed Forward Network. Kemudian, hasilnya diproses menggunakan fungsi aktivasi GELU yang dapat dihitung dengan persamaan (2.4), menghasilkan dimensi yang sama. Setelah itu, hasil tersebut masuk ke linear layer lagi dan dikembalikan ke dimensi (1, 512, 768). Hasil dari Feed Forward Network ini selanjutnya diproses melalui Residual Connection dan Normalisasi. Berbeda dengan Residual Connection pertama, Residual Connection kedua ini ditambahkan dengan hasil Residual Connection pertama. Setelah itu, output tersebut dinormalisasi melalui layer normalisasi. Output akhir dari BERT

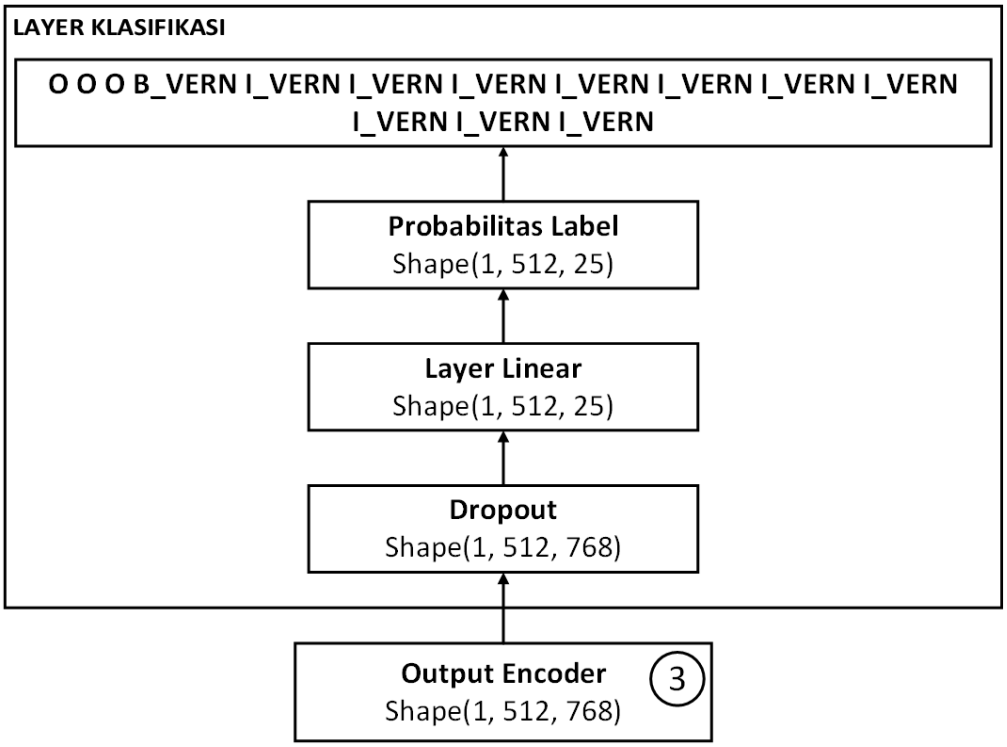
Encoder kemudian akan masuk ke dalam layer klasifikasi untuk prediksi label. Contoh proses Feed Forward Network, Residual Connection, dan Normalisasi ini bisa dilihat dalam Gambar 3.7.



Gambar 3.7 Contoh proses *Feed Forward Network*, *Residual Connection* dan Normalisasi

3. Layer Klasifikasi

Dalam tahap ini, hasil dari *encoder* diteruskan ke layer klasifikasi. Penambahan Layer klasifikasi berperan dalam mengidentifikasi fitur atau perpaduan tertentu dalam representasi yang dihasilkan oleh *encoder*, sehingga dapat menentukan label entitas. Contoh alur ini bisa dilihat dalam Gambar 3.8. Output dari *encoder* dengan dimensi (1, 512, 768) diolah melalui *Dropout* dan menghasilkan dimensi yang sama, kemudian dilanjutkan ke *linear* layer yang melakukan transformasi representasi vector dari 768 menjadi 25 yaitu sebanyak jumlah label. Hasilnya adalah dimensi (1, 512, 25), dimana 1 adalah jumlah data, 512 adalah jumlah token, dan 25 adalah jumlah label. Hasil akhir dari layer ini, berupa probabilitas, digunakan untuk menentukan label yang paling mungkin untuk setiap segmen dari data input.



Gambar 3.8 Contoh Layer Klasifikasi

3.1.7. Evaluasi

Dalam tahap evaluasi model BERT untuk tugas NER pada dokumen putusan hukum, F1-Score menjadi metrik utama yang digunakan untuk mengukur performa dari model. F1-Score dipilih karena mencakup kedua aspek penting dalam evaluasi model, yaitu presisi dan recall. F1-Score dihitung sebagai rata-rata harmonik dari presisi dan recall, yang memberikan perhatian yang seimbang terhadap keduanya.

Fokus utama dalam perhitungan F1-Score adalah kinerja model dalam mengidentifikasi entitas hukum, seperti nama terdakwa, nomor putusan, tanggal putusan, hakim, panitera, dan sebagainya. Proses perhitungan nya dilakukan dengan mempertimbangkan setiap entitas atau token yang diprediksi oleh model. Dengan demikian, F1-Score memberikan gambaran menyeluruh tentang kemampuan model dalam mengenali entitas-entitas yang penting dalam konteks dokumen putusan hukum, memungkinkan evaluasi yang lebih mendalam terhadap kinerja model tersebut.

3.2. Dataset

Dataset dalam penelitian ini diperoleh dari dokumen putusan hukum yang diambil melalui situs Direktori Putusan Mahkamah Agung RI www.putusan3.mahkamahagung.go.id. Dokumen yang diambil melibatkan klasifikasi Pidana, dari tahun putusan 2002 hingga 2019 dengan jumlah total 1100 dokumen dengan rata-rata 10 halaman. Pengambilan dokumen dilakukan melalui proses "*scraping*" menggunakan library Python Beautiful Soup. Dalam proses ini, Beautiful Soup digunakan untuk mengekstrak informasi dari struktur HTML halaman web, dengan memasukkan URL dari situs putusan hukum tersebut ke dalam script Python yang dikembangkan menggunakan Beautiful Soup. Proses "*scraping*" ini membantu mendapatkan dokumen dalam format PDF, yang selanjutnya dilakukan tahap konversi dari PDF ke bentuk teks yang tersimpan dalam bentuk CSV.

Sebelum memasuki tahap *preprocessing*, dari total 1100 dokumen [2] yang disimpan dalam bentuk CSV, dilakukan proses pemilihan dataset dengan kriteria jumlah kalimat minimal lebih dari 40. Penentuan kalimat dilakukan dengan menggunakan titik koma sebagai pembatas dalam pembentukan sebuah kalimat. Sehingga, diperoleh total 1000 dokumen yang memenuhi kriteria tersebut. Untuk dapat diproses oleh BERT, 1000 dokumen tersebut direpresentasikan dalam format yang siap diproses oleh BERT, yaitu dengan memecah dokumen menjadi tiap kalimat, sehingga menghasilkan 12.024 baris. Pada setiap baris nya, terdapat rata-rata 84 token.

Entitas yang diambil dalam dokumen putusan hukum berjumlah 12 entitas, yaitu: nomor putusan, nama terdakwa, tindak pidana, melanggar KUHP, tuntutan hukum, putusan hukum, tanggal putusan, hakim ketua, hakim anggota, panitera, penuntut umum, dan penasihat. Contoh entitas yang terdapat pada dokumen putusan hukum bisa dilihat dalam Tabel 3.9. Untuk sampel dataset bisa dilihat dalam Tabel 3.10.

Tabel 3.9 Contoh Entitas dengan Label

Text Dokumen Putusan	Label
90 / PID.sus / 2018 / PN.Mjn	(VERN) Nomor Putusan
Andika Saputra Alias Dika bin Abdul	(DEFN) Nama Terdakwa

Dengan sengaja dan tanpa hak mendistribusikan dan / atau mentransmisikan dan / atau membuat dapat diaksesnya Informasi Elektronik dan / atau Dokumen Elektronik yang memiliki muatan yang melanggar kesusilaan	(CRIA) Tindak Pidana
Pasal 27 ayat (1) Jo Pasal 45 ayat (1) UU RI No. 11 Tahun 2008 Jo UU RI No. 19 Tahun 2016 Tentang Informasi Transaksi Elektronik (ITE)	(ARTV) Pelanggaran KUHP
2 (dua) tahun,	(PENI) Tuntutan Hukum
2 (dua) tahun	(PUNI) Putusan Hukum
13 Februari 2019	(TIMV) Tanggal Putusan
Mohammad Fauzi Salam, S.H., M.H.,	(JUDP) Hakim Ketua
Saiful. HS, S.H, M.H., dan Nona Vivi Sri Dewi, S.H.,	(JUDG) Hakim Anggota
Andi M. Syahrul K, S.H, M.H.,,	(REGI) Panitera
Andi Asben Awaluddin, S. H., M.H	(PROS) Penuntut Umum
Dewi Suryaningsih, S. H	(ADVO) Pengacara

Tabel 3.10 Sampel Dataset

NO	Text	Labels
1.	PUTUSAN . NOMOR : 1974 / Pid . Sus / 2012 / PN . JKT . BAR . DEMI KEADILAN BERDASARKAN KETUHANAN YANG MAHA ESA . Pengadilan Negeri Jakarta Barat yang memeriksa dan mengadili perkara - perkara pidana dalam peradilan tingkat pertama dengan acara biasa telah menjatuhkan putusan sebagai berikut atas nama terdakwa : AGNES TRI AHADI Als ANAS , lahir di Jakarta , umur / tanggal , 30 tahun / 8 Agustus 1982 , jenis kelamin laki - laki , kebangsaan Indonesia , tempat tinggal Jl . Kamp . Bali Gg . II . No . 5 Kel . Kampung Bali Kec . Tanah Abang Jakarta Pusat Terdakwa ditahan sejak tanggal 27 Juli 2012 sampai sekarang ;	O O O O B_VERN I_VERN I_VERN I_VERN I_VERN I_VERN I_VERN I_VERN I_VERN I_VERN I_VERN I_VERN I_VERN I_VERN O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O B_DEFN I_DEFN I_DEFN I_DEFN I_DEFN I_DEFN O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O
2.	Setelah mendengar tuntutan pidana dari Jaksa Penuntut Umum tertanggal 9 Januari 2013 , yang pada pokoknya agar Pengadilan Negeri Jakarta Barat berkenan memutuskan sebagai berikut : MENUNTUT : 1 Menyatakan terdakwa AGNES TRI AHADI Als AGNES telah terbukti secara sah dan meyakinkan bersalah melakukan tindak pidana Narkotika memiliki , menyimpan , menguasai , atau menyediakan Narkotika golongan I bukan	O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O B_DEFN I_DEFN I_DEFN I_DEFN I_DEFN O O O O O O O O O O B_CRIA O O O O O O O O O O O O O O O O O O O O O O B_ARTV I_ARTV I_ARTV I_ARTV I_ARTV I_ARTV I_ARTV I_ARTV I_ARTV

	tanaman sebagaimana didakwakan dalam dakwaan kedua yaitu melanggar ketentuan unsure pasal 112 ayat (1) UURI No . 35 tahun 2009 tentang Narkotika ;	I_ARTV I_ARTV I_ARTV I_ARTV I_ARTV O
3.	2 Menjatuhkan pidana terhadap terdakwa DERRY SUSANTO Als YANTO , dengan pidana penjara selama 5 (lima) tahun dan 3 (tiga) bulan dikurangi selama terdakwa berada dalam tahanan sementara dan pidana dendasebesar Rp . 800 . 0 . 0 , 0 , - (delapan ratus ribu rupiah) subsidair 4 (empat) bulan 1penjara ;	O O O O O B_DEFN I_DEFN I_DEFN I_DEFN O O O O O B_PENA I_PENA I_PENA I_PENA I_PENA I_PENA I_PENA I_PENA I_PENA I_PENA I_PENA O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O
4.	Menimbang , bahwa terdakwa didakwa dengan surat dakwaan Jaksa Penuntut Umum melanggar pasal 114 ayat (1) UU RI No . 35 tahun 2009 , jo pasal 112 ayat (1) UU RI No . 35 tahun 2009 , tentang Narkotika adalah sebagai berikut : Dakwaan ;	O O O O O O O O O O O O B_ARTV I_ARTV O O O O O O
5.	Kesatu : Bahwa ia terdakwa Agnes Tri Ahadi Als Anas , pada hari Selasa tanggal 24 Juli 2012 , sekitar pukul 19 . 0 wib atau setidak - tidaknya pada suatu waktu masih bulan Juli 2012 bertempat di Jl . KS . Tubun No . 34 Kel . Kota Bambu Selatan Kec . Palmerah Jakarta Barat , tepatnya didepan Klinik Citra , atau setidak - tidaknya pada suatu tempat tertentu yang masih termasuk dalam daerah Hukum Pengadilan Negeri Jakarta Barat yang berwenang memeriksa dan mengadili perkara ini , tanpa hak atau melawan hukum menawarkan untuk dijual , menjual , membeli , menerima , menjadi perantara dalam jual beli , menukar atau menyerahkan Narkotika Golongan I jenis Putaw , perbuatan mana terdakwa lakukan dengan cara sebagai berikut : 2 Pada awalnya hari Selasa tanggal 24 Juli 2012 , sekira pukul 17 . 30 wib , terdakwa tidak senghaja bertemu dengan temannya bernama : Sayuti (DPO) di Jl . Jati Baru Tanah Abang Jakarta Pusat kemudian Sayuti minta tolong pada terdakwa , dan memberikan uang sebesar Rp . 100 . 0 (seratus ribu rupiah) untuk dibelikan barang berupa Putaw sebanyak 2 (dua) paket ;	O O O O O B_DEFN I_DEFN I_DEFN I_DEFN I_DEFN O B_DEFN O O O O O O O O O O O O O B_DEFN O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O

3.3. Skenario Pengujian

Tolak ukur evaluasi untuk BERT Indonesia dengan menggunakan dataset *IndoLEM* dan *IndoNLU*. 2 dataset tersebut telah digunakan untuk melatih model, yaitu *Indolem/indobert-base-uncased* dengan dataset *IndoLEM* dan Indobenchmark*/indobert-base-p2* dengan dataset *IndoNLU.* Sehingga, skenario pengujian dilakukan untuk mencari performa model terbaik antara *Indolem/indobert-base-uncased* dan *Indobenchmark/indobert-base-p2*, dengan

mempertimbangkan F1-Score yang diperoleh dari setiap model. Hyper parameter yang digunakan pada penelitian ini mengacu pada penelitian [24] yang bisa dilihat dalam Tabel 3.11.

Tabel 3.11 Skenario Pengujian

Model	**Training Batch Size** [24]	**Test Batch Size** [24]	**Learning Rate** [24]	**Epoch** [24]
Indolem/indobert-base-uncased	4	2	0.00001	3
Indobenchmark/indobert-base-p2	4	2	0.00001	3

BAB IV HASIL DAN PEMBAHASAN

Bab IV memuat hasil dari penerapan sistem dengan menggunakan metode yang telah dipilih. Hasil penerapan meliputi lingkungan uji coba, hasil dari percobaan pengujian, serta evaluasi dan analisis dari hasil pengujian.

4.1. Lingkungan Uji Coba

Bagian ini akan menguraikan detail tentang spesifikasi perangkat yang digunakan dalam proses pengujian model NER, baik itu perangkat keras maupun perangkat lunak. Rincian mengenai perangkat keras dan perangkat lunak tersedia pada Tabel 4.1dan pada Tabel 4.2.

Tabel 4.1 Kebutuhan Sistem Training

No	Kebutuhan	Jenis
1.	Sistem Operasi	Linux
2.	CPU	Intel(R) Xeon(R)
3.	GPU	Tesla P100-PCIE-16GB
4.	RAM	30 GB
5.	Bahasa Pemrograman	Python

Tabel 4.2 Lingkungan perangkat lunak yang digunakan

NO	Perangkat Lunak	Versi	Fungsi
1.	Transformers	4.39.3	Digunakan untuk memuat dan menggunakan model NLP, serta melakukan tokenisasi teks.
2.	PyTorch	2.1.2	Digunakan untuk membangun dan melatih model neural networks.
3.	Tqdm	4.66.1	Digunakan untuk membuat bar kemajuan (progress bar) yang menampilkan kemajuan iterasi dalam proses pengolahan data atau pelatihan model.
4.	Sklearn	1.2.2	Digunakan untuk menghitung metrik evaluasi seperti presisi, recall, dan F-score.

5.	Pandas	2.2.1	Digunakan untuk manipulasi dan analisis data tabular menggunakan struktur data DataFrame.
6.	Numpy	1.26.4	Digunakan untuk komputasi numerik dan operasi matematika pada data numerik.
7	Doccano	1.8.4	Digunakan untuk anotasi dataset

4.2. Tahapan Pembuatan Program

Bagian ini akan menjelaskan proses dari pembuatan program yang di butuhkan dalam membuat sistem. Program yang di buat ini di jalankan dengan menggunakan virtual machine dari Kaggle.

4.2.1. Scraping

Proses Scraping merupakan proses untuk mendapatkan data dengan mengambil dokumen pdf yang terdapat pada laman putusan pengadilan, yang selanjutnya di lakukan konversi ke dalam text yang di simpan dalam bentuk csv. Berikut adalah code untuk melakukan scraping pada laman putusan pengadilan.

4.2.1.1. *Install* dan Impor *Library*

Langkah awal dalam proses scraping adalah menginstal *library* PyPDF2. *Library* ini digunakan untuk mengonversi dokumen PDF menjadi teks yang kemudian disimpan dalam format CSV. Kode instalasi *library* bisa dilihat dalam Kode 4.1. Setelah proses instalasi selesai, langkah berikutnya adalah mengimpor library yang diperlukan dalam proses scraping. *Library* yang diimpor bisa dilihat dalam Kode 4.2.

Kode 4.1 *Install PyPDF2*

```
1.  !pip install PyPDF2
```

Kode 4.2 Impor Library untuk Scraping

```
2.  from bs4 import BeautifulSoup
3.  import requests
4.  import PyPDF2
5.  import re
6.  import pandas as pd
7.  import zipfile
8.  import os
9.  import nltk
10. import re
11. from nltk.tokenize import word_tokenize
12. nltk.download('punkt')
```

4.2.1.2. Fungsi scraping_putusan()

Pada bagian ini, dibuat sebuah fungsi bernama scraping_putusan() yang memiliki dua parameter, yaitu URL dan filename. URL adalah alamat web dari putusan pengadilan yang akan dilakukan scraping, dan filename adalah nama file hasil *scraping* yang disimpan dalam format CSV. Kode fungsi scraping_putusan() bisa dilihat dalam Kode 4.3. Fungsi ini dirancang untuk mengekstrak URL dari setiap konten di halaman awal pencarian situs web putusan pengadilan. Tujuannya adalah untuk memungkinkan navigasi ke halsaman selanjutnya yang memberikan informasi apakah dokumen putusan pengadilan tersedia atau tidak. Tugas pengecekan ini dilakukan dengan menggunakan fungsi get_data(). Selain itu, fungsi ini juga mengambil data tanggal yang terdapat pada setiap konten halaman awal.

Kode 4.3 Fungsi scraping_putusan()

```
13. def scraping_putusan(url, filename):
14.   # Mengirim permintaan GET ke URL
15.   response = requests.get(url)
16.
17.   # Memeriksa apakah permintaan berhasil
18.   if response.status_code == 200:
19.
20.     # Parsing halaman web dengan BeautifulSoup
21.     soup = BeautifulSoup(response.text, 'html.parser')
22.
23.     # Mencari elemen-elemen yang mengandung teks putusan hukum
24.     putusan_hukum = soup.find_all('div', class_='spost clearfix')
25.
26.     putusan_text = []
27.     putusan_filename = []
28.     for i in putusan_hukum:
29.       # mengambil semua elemen tag <strong>
30.       tag = i.find_all('strong')
31.       for y in tag:
32.         # Mencari url untuk mengakses halaman berikutnya
33.         if y.find('a'):
34.           get_data_text = get_data(y.find('a').get('href'))
35.           if get_data_text:
36.             # pengambilan data seperti tanggal, text hasil konversi serta
    nama file jika terdapat filenya
37.             tanggal = re.findall(r'\d{2}-\d{2}-\d{4}',i.get_text())
38.             putusan_text.append([get_data_text[0]] + tanggal +
    [get_data_text[1]])
39.             putusan_filename.append(get_data_text[0])
40.
41.     # mengembalikan seluruh data yang di dapat beserta dengan link next
    button
42.     buttons = soup.find_all('ul', class_='pagination justify-content-
    center')
43.     for button in buttons:
44.       tag_a = button.find_all('a')
45.       for link in tag_a:
46.         if link.text.strip() == 'Next':
47.           return (putusan_filename, putusan_text, link.get('href'))
48.
49.   else:
50.       print("Gagal mengakses halaman web. Kode status:",
    response.status_code)
```

Penjelasan Kode:

1. Definisi Fungsi

   Pada baris nomor 13 dibuat sebuah fungsi “scraping_putusan” yang menerima dua parameter yaitu "url" yang berisikan url dari halaman web putusan pengedilan, "filename" yang berisikan nama file yang akan digunakan untuk menyimpan hasil scraping.

2. Mengirim Permintaan GET

   Pada baris nomor 15, fungsi mengirim permintaan GET ke URL yang diberikan menggunakan “requests.get”.

3. Memeriksa Status Permintaan

   Pada baris nomor 18, fungsi memeriksa apakah permintaan berhasil dengan memeriksa “response.status_code”.

4. Parsing Halaman Web

   Pada baris nomor 21, halaman web diparsing menggunakan “BeautifulSoup” untuk memudahkan pengambilan elemen HTML.

5. Mencari Elemen Teks Putusan Hukum

   Pada baris nomor 24, fungsi mencari elemen-elemen yang mengandung teks putusan hukum dengan mencari “div” dengan kelas “spost clearfix”.

6. Inisialisasi List

   Pada baris nomor 26 dan 27, dua list kosong diinisialisasi untuk menyimpan teks putusan dan nama file.

7. Iterasi Melalui Elemen Putusan Hukum

   Pada baris nomor 28 hingga 39, dilakukan iterasi melalui elemen-elemen yang mengandung teks putusan hukum:

   - Pada baris nomor 30, mengambil semua elemen tag “strong”.
   - Pada baris nomor 33, memeriksa apakah “strong” mengandung tag “a”.
   - Pada baris nomor 34, mengambil URL halaman berikutnya dari tag “a” dan memanggil fungsi “get_data” untuk mengambil teks.
   - Pada baris nomor 37, menggunakan regex untuk menemukan tanggal dalam teks.
   - Pada baris nomor 38 dan 39, menambahkan teks putusan dan nama file ke list.

8. Mengambil Link Tombol Next

   Pada baris nomor 42 hingga 47, fungsi mencari elemen “ul” dengan kelas “pagination justify-content-center” untuk menemukan tombol “Next”:

   - Pada baris nomor 44, mencari semua tag “a” dalam elemen “ul.
   - Pada baris nomor 46, memeriksa apakah teks dari tag “a” adalah “Next” dan mengembalikan hasil scraping beserta URL tombol “Next”.

9. Penanganan Kesalahan

   Pada baris nomor 50, jika permintaan GET gagal, fungsi mencetak pesan kesalahan dengan kode status.

4.2.1.3. Fungsi get_data()

Pada bagian ini, dibuat sebuah fungsi bernama get_data() dengan satu parameter yaitu URL. Fungsi ini digunakan dalam fungsi sebelumnya, yaitu scraping_putusan(). URL yang diterima adalah URL halaman yang berisi informasi mengenai dokumen putusan. Fungsi ini dirancang untuk memeriksa apakah dokumen putusan pengadilan tersedia atau tidak di halaman tersebut. Jika ada, fungsi akan mengunduhnya dan memanggil fungsi read_pdf() untuk melakukan konversi dari file PDF ke teks. Fungsi ini mengembalikan hasil konversi dari file PDF beserta nama file yang telah diunduh. Kode dari fungsi get_data() bisa dilihat dalam Kode 4.4.

Kode 4.4 Fungsi get_data()

```
51. def get_data(url):
52.   # Mengirim permintaan GET ke URL
53.   response = requests.get(url)
54.
55.   # Memeriksa apakah permintaan berhasil
56.   if response.status_code == 200:
57.     # Parsing halaman web dengan BeautifulSoup
58.     soup = BeautifulSoup(response.text, 'html.parser')
59.     box_lampiran = soup.find_all('ul', class_='portfolio-meta
    nobottommargin')[1]
60.
61.     # Cek Link PDF
62.     if box_lampiran.find_all('a'):
63.       link_pdf = box_lampiran.find_all('a')[1]
64.       response_pdf = requests.get(link_pdf.get('href'))
65.
66.       # Download PDF
67.       if response_pdf.status_code == 200:
68.         # Menyimpan file PDF yang diunduh ke sistem lokal
69.         pdf_name = link_pdf.text.strip().replace('/', '_')
70.         with open(pdf_name , 'wb') as pdf_file:
71.             pdf_file.write(response_pdf.content)
72.
73.         print(f"File PDF '{pdf_name}' berhasil diunduh.")
74.         return (pdf_name, read_pdf(pdf_name))
75.
76.       else:
77.           print("Gagal mengunduh file PDF. Kode status:",
    response.status_code)
78.           return False
79.
```

```
80.     else:
81.       print('Link Download Tidak Tersedia')
82.       return False
```

Penjelasan Kode:

1. Definisi Fungsi

   Pada baris nomor 51 dibuat sebuah fungsi “get_data” yang menerima satu parameter yaitu url dari halaman web putusan pengadilan.

2. Mengirim Permintaan GET

   Pada baris nomor 53, fungsi mengirim permintaan GET ke URL yang diberikan menggunakan “requests.get”.

3. Memeriksa Status Permintaan

   Pada baris nomor 56, fungsi memeriksa apakah permintaan berhasil dengan memeriksa “response.status_code”.

4. Parsing Halaman Web

   Pada baris nomor 58, halaman web diparsing menggunakan “BeautifulSoup” untuk memudahkan pengambilan elemen HTML.

5. Mencari Elemen Lampiran

   Pada baris nomor 59, fungsi mencari elemen “ul” dengan kelas “portfolio-meta nobottommargin” pada posisi kedua dalam halaman.

6. Memeriksa Ketersediaan Link PDF

   Pada baris nomor 62 hingga 78, fungsi memeriksa apakah terdapat link “a” dalam elemen “box_lampiran”:

   - Pada baris nomor 63, mengambil link PDF kedua dalam elemen “box_lampiran”.
   - Pada baris nomor 64, mengirim permintaan GET ke URL dari link PDF.

7. Mengunduh PDF

   Pada baris nomor 67 hingga 74, jika permintaan GET untuk PDF berhasil, fungsi mengunduh dan menyimpan file PDF:

   - Pada baris nomor 69, menentukan nama file PDF dengan mengganti karakter “slash (/)” menjadi underscore (“_”).
   - Pada baris nomor 70 dan 71, menyimpan file PDF ke sistem lokal.
   - Pada baris nomor 73, mencetak pesan berhasil dan mengembalikan nama file PDF dan hasil pembacaan file menggunakan “read_pdf”.

8. Penanganan Kesalahan Pengunduhan PDF

   Pada baris nomor 77, jika permintaan GET untuk PDF gagal, fungsi mencetak pesan kesalahan dengan kode status.

9. Penanganan Ketidaktersediaan Link PDF

   Pada baris nomor 81, jika link PDF tidak tersedia, fungsi mencetak pesan "Link Download Tidak Tersedia" dan mengembalikan “False”.

4.2.1.4. Fungsi read_pdf()

Pada bagian ini, dibuat sebuah fungsi bernama read_pdf() dengan satu parameter yaitu file_pdf. Parameter ini berisi lokasi dari file PDF yang telah diunduh, yang dilakukan pada fungsi sebelumnya, yaitu get_data(). Tujuan dari fungsi ini adalah mengonversi file PDF ke teks menggunakan *library PyPDF2*. Dalam fungsi ini, terdapat pemanggilan fungsi lain yaitu clean_text yang berfungsi untuk menghapus karakter-karakter yang tidak diperlukan. Fungsi ini mengembalikan sebuah teks berupa string hasil konversi menggunakan *library PyPDF2*. Kode fungsi ini bisa dilihat dalam Kode 4.5.

Kode 4.5 Fungsi read_pdf()

```
83. def read_pdf(file_pdf):
84.   try:
85.     pdf_text = ''
86.     pdf_file = open(file_pdf, 'rb')
87.     pdf_reader = PyPDF2.PdfReader(pdf_file)
88.
89.     for page_num in range(len(pdf_reader.pages)):
90.         page = pdf_reader.pages[page_num]
91.         text = clean_text(page.extract_text())
92.         pdf_text += text
93.
94.     pdf_file.close()
95.     return pdf_text.strip()
96.
97.   except requests.exceptions.RequestException as e:
98.     print("Error:", e)
```

Penjelasan Kode:

1. Definisi Fungsi

   Pada baris nomor 83 dibuat sebuah fungsi “read_pdf” yang menerima satu parameter yaitu file_pdf" untuk nama file PDF atau path lokasi PDF yang akan dibaca.

2. Blok Try-Except

   Pada baris nomor 84, digunakan blok “try-except” untuk menangani kesalahan selama proses membaca file PDF.

3. Inisialisasi Variabel

   Pada baris nomor 85, variabel “pdf_text” diinisialisasi sebagai string kosong untuk menyimpan teks yang diekstrak dari PDF.

4. Membuka File PDF

   Pada baris nomor 86, file PDF dibuka dalam mode baca biner (“'rb'“).

5. Inisialisasi PDF Reader

   Pada baris nomor 87, “PyPDF2.PdfReader” digunakan untuk membaca konten dari file PDF.

6. Iterasi Melalui Halaman PDF

   Pada baris nomor 89 hingga 92, dilakukan iterasi melalui setiap halaman dalam PDF:

   - Pada baris nomor 90, mengambil halaman saat ini.
   - Pada baris nomor 91, mengekstrak teks dari halaman dan membersihkannya menggunakan fungsi “clean_text”.
   - Pada baris nomor 92, menambahkan teks yang diekstrak ke variabel “pdf_text”:

7. Menutup File PDF

   Pada baris nomor 94, file PDF ditutup setelah selesai dibaca.

8. Mengembalikan Teks yang Diekstrak

   Pada baris nomor 95, teks yang diekstrak dari PDF dikembalikan dengan menghapus spasi ganda yang berada di awal dan akhir kalimat.

9. Penanganan Kesalahan

   Pada baris nomor 97, blok “except” menangkap kesalahan “requests.exceptions.RequestException” dan mencetak pesan kesalahan.

4.2.1.5. Fungsi zip_dokumen()

Pada bagian ini, dibuat sebuah fungsi bernama zip_dokumen() dengan dua parameter yaitu filename dan listData. Filename adalah nama dari file dari hasil kompres format zip, sedangkan listData berisi semua file yang telah diunduh. Tujuan dari fungsi ini adalah untuk mengompres seluruh file ke dalam format zip, sehingga memudahkan dalam mengunduh seluruh file tersebut. Kode fungsi ini bisa dilihat dalam Kode 4.6.

Kode 4.6 Fungsi zip_dokumen()

```
99. def zip_dokumen(filename,listData):
100.     # Buka file zip untuk penulisan
101.     with zipfile.ZipFile(filename, "w", zipfile.ZIP_DEFLATED) as zipf:
102.         for file in listData:
103.             # Path lengkap ke file yang akan dimasukkan ke dalam zip
104.             file_path = os.path.join('/content', file)
105.
106.             # Masukkan file ke dalam zip dengan nama yang sama
107.             zipf.write(file_path, os.path.basename(file_path))
108.
109.     print(f"File {filename} berhasil diciptakan.")
```

Penjelasan Kode:

1. Definisi Fungsi

   Pada baris nomor 99 dibuat sebuah fungsi “zip_dokumen” yang menerima dua parameter yaitu "filename" yang berisikan nama file zip yang akan dibuat, "listData" yang berisikan daftar nama file yang akan dimasukkan ke dalam file zip.

2. Membuka File Zip untuk Penulisan

   Pada baris nomor 101, menggunakan “with” statement untuk membuka file zip dalam mode penulisan (“"w"“) dengan kompresi “ZIP_DEFLATED”.

3. Iterasi Melalui Daftar File

   Pada baris nomor 102, dilakukan iterasi melalui setiap file dalam “listData”.

4. Path Lengkap ke File

   Pada baris nomor 104, menentukan path lengkap ke file yang akan dimasukkan ke dalam zip dengan menggabungkan direktori “/content” dan nama file.

5. Menambahkan File ke dalam Zip

   Pada baris nomor 107, menambahkan file ke dalam file zip dengan nama yang sama seperti file aslinya menggunakan “zipf.write”.

6. Mencetak Pesan Sukses

   Pada baris nomor 109, setelah semua file berhasil dimasukkan ke dalam file zip, fungsi mencetak pesan sukses.

4.2.1.6. Fungsi main()

Pada bagian ini, dibuat sebuah fungsi bernama main() dengan satu parameter yaitu num_page yang berisi jumlah halaman yang ingin diakses. Fungsi ini merupakan fungsi utama dan fungsi awal yang akan menjalankan semua fungsi

sebelumnya. Tujuan dari fungsi ini adalah menjalankan fungsi-fungsi sebelumnya dan menyimpan semua hasil yang diperoleh dari fungsi-fungsi tersebut dalam bentuk CSV setelah dikonversi, serta memanggil fungsi zip_dokumen() untuk mengompres file-file yang telah diunduh. Kode dari fungsi main() bisa dilihat dalam Kode 4.7.

Kode 4.7 Fungsi main()

```
110.    def main(num_page):
111.      url =
    'https://putusan3.mahkamahagung.go.id/search.html?q=&jenis_doc=putusan&cat=d
    92c02366ae91966e4cdbe6279fc36eb&jd=&tp=0&court=&t_put=&t_reg=&t_upl=&t_pr='
112.      filename = 'Dataset Putusan Hukum v2.csv'
113.      putusan_file = []
114.      putusan_text = []
115.      next_button = url
116.
117.      for page in range(1, num_page+1):
118.        print(f"============== Page {page} ==============")
119.        putusan_file_new, putusan_text_new, next_button_new =
    scraping_putusan(next_button,filename)
120.        putusan_file += putusan_file_new
121.        putusan_text += putusan_text_new
122.        next_button = next_button_new
123.
124.      print(f'Berhasil mengunduh sebanyak {len(putusan_text)} File')
125.
126.      # Menyimpan Datset Dalam Bentuk CSV
127.      df = pd.DataFrame(putusan_text, columns=['Nama File','Tanggal
    Register', 'Tanggal Putus', 'Tanggal Upload', 'Dokumen'])
128.      df.to_csv(filename, index=False)
129.      print('Berhasil menyimpan ke dalam file Dataset Putusan Hukum v2.csv')
130.
131.      zip_dokumen('Data Dokumen.zip', putusan_file)
132.
133.    main(10)
```

Penjelasan Kode:

1. Definisi Fungsi

   Pada baris nomor 110 dibuat sebuah fungsi “main” yang menerima satu parameter yaitu "num_page" yang berisi Jumlah halaman yang akan di-scrape.

2. Inisialisasi URL dan Nama File

   Pada baris nomor 111 dan 112, inisialisasi url untuk halaman web putusan pengadilan dan nama file CSV untuk menyimpan hasil scraping.

3. Inisialisasi List dan URL Tombol Next

   Pada baris nomor 113 hingga 115, dibuat list kosong untuk menyimpan nama file dan teks putusan serta URL tombol next.

4. Iterasi Melalui Halaman

   Pada baris nomor 117 hingga 122, dilakukan iterasi melalui setiap halaman sesuai dengan jumlah halaman yang diberikan:

- Pada baris nomor 118, mencetak informasi halaman saat ini.
- Pada baris nomor 119, memanggil fungsi “scraping_putusan” untuk halaman saat ini dan menyimpan hasilnya.
- Pada baris nomor 120 dan 121, menambahkan hasil baru ke list yang sudah ada.
- Pada baris nomor 122, memperbarui URL tombol next.

5. Mencetak Jumlah File yang Diunduh

   Pada baris nomor 124, mencetak jumlah file yang berhasil diunduh.

6. Menyimpan Dataset dalam Bentuk CSV

   Pada baris nomor 127 hingga 129, membuat DataFrame dari teks putusan dan menyimpannya dalam bentuk CSV:

   - Pada baris nomor 127, membuat DataFrame dengan kolom yang sesuai.
   - Pada baris nomor 128, menyimpan DataFrame ke file CSV.
   - Pada baris nomor 129, mencetak pesan sukses.

7. Membuat File Zip dari Dokumen

   Pada baris nomor 131, memanggil fungsi “zip_dokumen” untuk membuat file zip dari dokumen yang diunduh.

8. Memanggil Fungsi “main”

   Pada baris nomor 133, fungsi “main” dipanggil dengan argumen 10, yang berarti melakukan scraping pada 10 halaman.

4.2.2. Preprocessing

Pada tahap preprocessing, dilakukan saat melakukan konversi dari PDF ke teks selama proses scraping. Berikut adalah kode preprocessing yang dilakukan dan hasilnya bisa dilihat dalam Tabel 3.2: Pada bagian ini, dibuat fungsi clean_text() untuk membersihkan teks hasil konversi dari PDF ke teks. Pembersihan dilakukan dengan menghapus kalimat "Mahkamah Agung Republik Indonesia\n" yang merupakan watermark dari dokumen putusan pengadilan, kalimat “Direktori Putusan Mahkama Agung Republik Indonesia\nputusan.mahkamaagung.go.id\n” yang merupakan header dan dokumen, serta karakter-karakter lain yang tidak berguna seperti uf0fc, uf0a7, uf0a8, uf0b7. Kode dari fungsi ini bisa dilihat dalam Kode 4.8.

Kode 4.8 Fungsi clean_text()

```
134.    def clean_text(text):
135.        text = text.replace("Mahkamah Agung Republik Indonesia\nMahkamah
    Agung Republik Indonesia\nMahkamah Agung Republik Indonesia\nMahkamah Agung
    Republik Indonesia\nMahkamah Agung Republik Indonesia\nDirektori Putusan
    Mahkamah Agung Republik Indonesia\nputusan.mahkamahagung.go.id\n", "")
136.        text = text.replace("\nDisclaimer\nKepaniteraan Mahkamah Agung
    Republik Indonesia berusaha untuk selalu mencantumkan informasi paling kini
    dan akurat sebagai bentuk komitmen Mahkamah Agung untuk pelayanan publik,
    transparansi dan akuntabilitas\npelaksanaan fungsi peradilan. Namun dalam
    hal-hal tertentu masih dimungkinkan terjadi permasalahan teknis terkait
    dengan akurasi dan keterkinian informasi yang kami sajikan, hal mana akan
    terus kami perbaiki dari waktu kewaktu.\nDalam hal Anda menemukan inakurasi
    informasi yang termuat pada situs ini atau informasi yang seharusnya ada,
    namun belum tersedia, maka harap segera hubungi Kepaniteraan Mahkamah Agung
    RI melalui :\nEmail : kepaniteraan@mahkamahagung.go.id", "")
137.        text = text.replace("Telp : 021-384 3348 (ext.318)", "")
138.        text = text.replace('P U T U S A N', 'PUTUSAN').replace('T erdakwa',
    'Terdakwa').replace('T empat', 'Tempat').replace('T ahun', 'Tahun')
139.        text = text.replace('P  E  N  E  T  A  P  A  N',
    'PENETAPAN').replace('J u m l a h', 'Jumlah').replace('\n', ' ')
140.        text = re.sub(r'\nHalaman \d+ dari \d+ .*', '', text)
141.        text = re.sub(r'Halaman \d+ dari \d+ .*', '', text)
142.        text = re.sub(r'\nHal. \d+ dari \d+ .*', '', text)
143.        text = re.sub(r'Hal. \d+ dari \d+ .*', '', text)
144.        text = re.sub(r' +|[\uf0fc\uf0a7\uf0a8\uf0b7]', ' ', text)
145.        text = re.sub(r'[\u2026]+|\.{3,}', '', text)
146.        return text.strip()
```

Penjelasan Kode:

1. Definisi Fungsi

   Pada baris nomor 134 dibuat sebuah fungsi “clean_text” yang menerima satu parameter yaitu Teks yang akan dibersihkan.

2. Menghapus Teks Berulang

   Pada baris nomor 135, menghapus teks header dan watermark dokumen yang mencantumkan nama Mahkamah Agung dan Direktori Putusan. Contoh teks header dan watermark bisa dilihat pada Gambar 4.1.

3. Menghapus Disclaimer

   Pada baris nomor 136, menghapus teks disclaimer panjang yang terdapat pada footer dokumen. Contoh teks disclaimer panjang bisa dilihat pada Gambar 4.2.

4. Menghapus Informasi Kontak

   Pada baris nomor 137, menghapus teks informasi kontak yang terdapat pada footer dokumen. Contoh teks informasi kontak bisa dilihat pada Gambar 4.2

5. Memperbaiki Format Teks

   Pada baris nomor 138 dan 139, memperbaiki format teks dengan mengganti spasi yang berlebih dan mengganti huruf besar menjadi kecil.

6. Menghapus Nomor Halaman

   Pada baris nomor 140 hingga 143, menghapus teks nomor halaman. Dalam hal ini terdapat beberapa kondisi halaman di sesuaikan dengan format dokumen. Contoh dokumen untuk teks halaman bisa dilihat pada Gambar 4.2:

   - Pada baris nomor 140, menghapus format "Halaman x dari y" yang juga terdapat karakter “\n”.
   - Pada baris nomor 141, menghapus format "Halaman x dari y" tanpa karakter “\n”.
   - Pada baris nomor 142, menghapus format "Hal. x dari y" yang juga terdapat karakter “\n”.
   - Pada baris nomor 143, menghapus format "Hal. x dari y"tanpa karakter “\n”.

7. Menghapus Spasi Ekstra dan Karakter Khusus

   Pada baris nomor 144, menghapus spasi ekstra dan karakter yang tidak diperlukan seperti “uf0fc”.

8. Menghapus Elipsis

   Pada baris nomor 145, menghapus satu atau lebih karakter elipsis (…) atau titik berturut-turut dengan string kosong.

9. Mengembalikan Teks yang Bersih

   Pada baris nomor 146, teks yang sudah dibersihkan dikembalikan dengan menghapus spasi ekstra di awal dan akhir.


Header
Direktori Putusan Mahkamah Agung Republik Indonesia
putusan.mahkamahagung.go.id

P U T U S A N

NOMOR : 90 / PID.Sus / 2018 / PN.Mjn

DEMI KEADILAN BERDASARKAN KETUHANAN YANG MAHA ESA

Pengadilan Negeri Majene yang mengadili perkara pidana dengan acara biasa pada tingkat pertama dengan susunan persidangan Majelis, telah menjatuhkan putusan sebagai berikut dalam perkara Terdakwa :

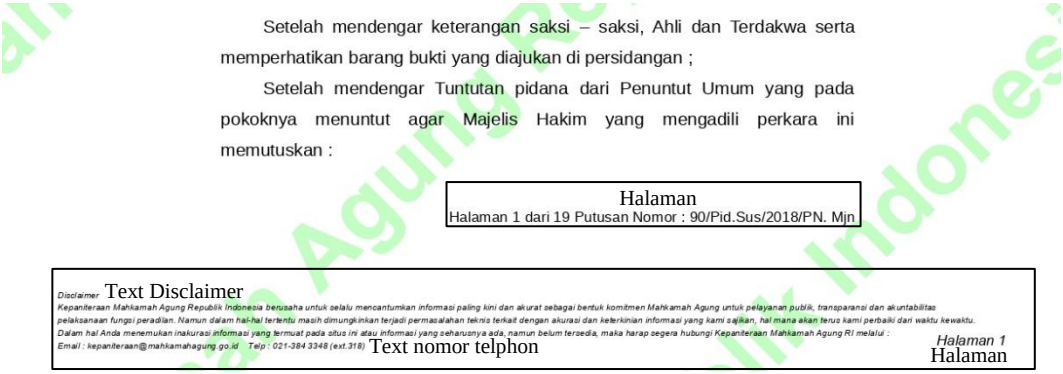
Nama Lengkap	:	ANDIKA SAPUTRA ALIAS DIKA BIN ABDUL RASID ;
Tempat lahir	:	Majene ;
Umur / Tanggal lahir	:	21 Tahun / 6 Juli 1997 ;
Jenis Kelamin	:	Laki-laki ;
Kebangsaan	:	Indonesia ;
Tempat tinggal	:	Lingkungan Passarang, Kelurahan Totoli, Kecamatan Banggae, Kabupaten Majene ;
Agama	:	Islam ;
Pekerjaan	:	- ;
Pendidikan	:	S L T A (Tamat) ;

Terdakwa dalam perkara ini ditahan dengan jenis tahanan Rutan oleh :

- Penyidik, sejak tanggal 1 Oktober 2018 sampai dengan tanggal 20 Oktober 2018 ;

Watermark

Gambar 4.1 Contoh Header Dokumen



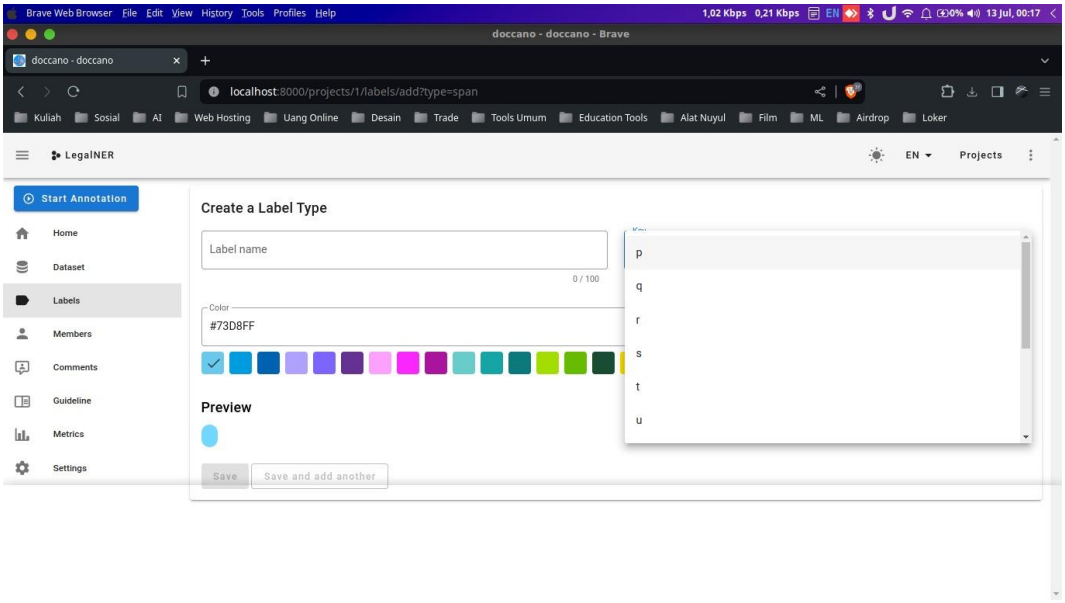
Gambar 4.2 Contoh Footer Dokumen

4.2.3. Anotasi

Pada bagian ini, proses anotasi telah dilakukan oleh mahasiswa hukum, dengan menggunakan teknik anotasi Inside-Outside-Beginning (IOB). Proses anotasi dilakukan dengan menggunakan tools bernama *Doccano* dengan beberapa tahapan diantaranya:

1. Set Label

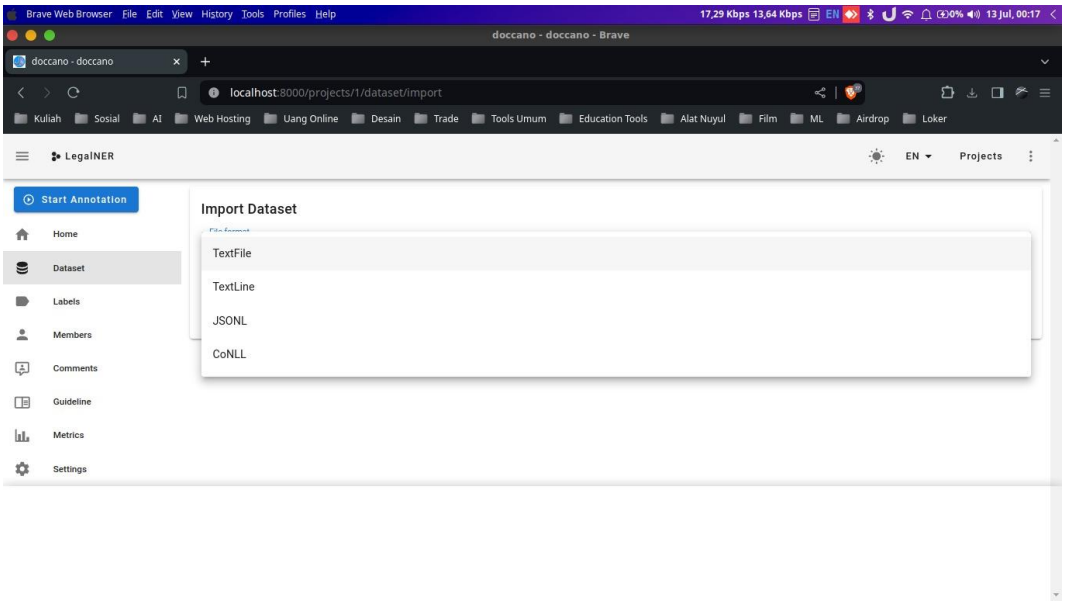
   Label yang digunakan pada proses anotasi bisa dilihat dalam Tabel 3.3, dengan tambahan satu label yaitu label “O” yang menunjukkan bahwa teks tersebut bukan sebuah entitas. Setiap label di masukkan satu persatu dengan memberikan ID pada setiap label. Proses ini bisa dilihat pada Gambar 4.3.



Gambar 4.3 Set Label pada Doccano

2. Muat Dataset

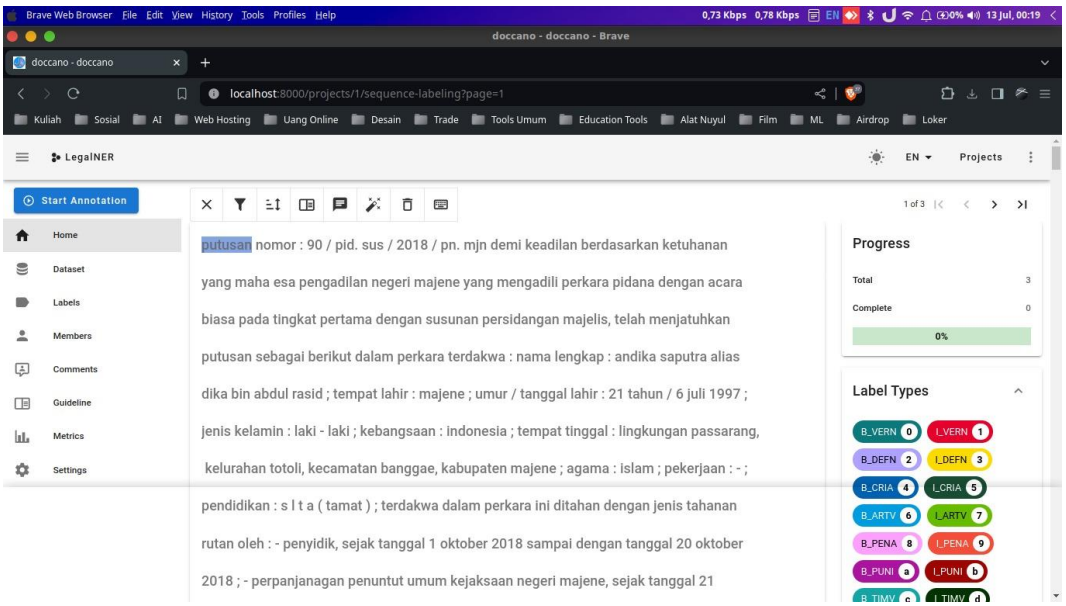
Hasil preprossing yang telah di simpan selanjutnya dimuat ke dalam *Doccano* pada *sidebar* Dataset. Proses ini bisa dilihat pada Gambar 4.4.



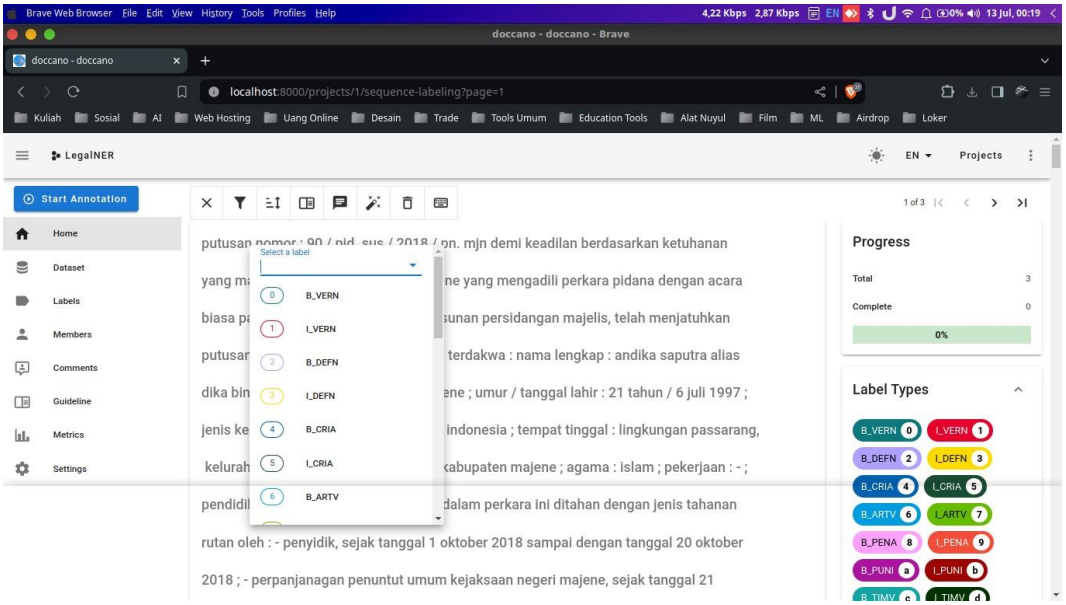
Gambar 4.4 Memuat Dataset pada Doccano

3. Proses Anotasi

Proses anotasi ini dilakukan dengan cara blok tiap token dan pilih label yang digunakan. Proses ini bisa dilihat pada Gambar 4.5 dan Gambar 4.6.



Gambar 4.5 Pemilihan token untuk di anotasi pada *Doccano*



Gambar 4.6 Pilih Label untuk token yang di pilih pada *Doccano*

4.2.4. Representasi Data ke Input BERT

Pada bagian ini, merupakan proses lanjutan setelah melakukan anotasi, yaitu mengubah data token menjadi kalimat dalam bentuk string dengan dipotong per kalimat. Pemotongan ini dilakukan berdasarkan adanya titik koma. Proses representasi data ini dilakukan melalui beberapa tahapan sebagai berikut:

4.2.4.1. Impor *Library*

Tahapan pertama ini dilakukan dengan mengimpor beberapa *library* yang dibutuhkan di antaranya *tqdm*, *pandas*, *gdown*, dan *re*. Kode untuk impor *library* ini bisa dilihat dalam Kode 4.9.

Kode 4.9 Impor *library*

```
147.    from tqdm import tqdm
148.    import pandas as pd
149.    import gdown
150.    import re
```

4.2.4.2. Mengunduh Dataset dari Google Drive

Tahap selanjutnya adalah mengunduh dataset yang telah disimpan di Google Drive menggunakan *library gdown*. Cara menggunakannya adalah dengan memanggil gdown.download() dengan parameter pertama sebagai URL dari Google Drive dan parameter kedua sebagai nama file beserta formatnya. Serta parameter terakhir yaitu quite dapat diatur untuk menampilkan output proses

pengunduhan atau tidak. Jika ingin menampilkan, maka set quite menjadi False. Kode untuk proses ini bisa dilihat dalam Kode 4.10.

Kode 4.10 Mengunduh Dataset

```
151.    gdown.download(f'https://drive.google.com/uc?id=1lmhKijLUDFW4gT1FtXwz3TD
    4jOGIc4Pk', 'Dataset Pidana 1000.json', quiet=False)
```

4.2.4.1. Load Dataset

Setelah proses pengunduhan selesai selanjutnya memuat data ke dalam variabel df dengan menggunakan *library pandas* yaitu *read_json* karena file yang di buka berformat JSON. Kode untuk memuat dataset ini bisa dilihat dalam Kode 4.11.

Kode 4.11 Load dataset

```
152.    df = pd.read_json('Dataset Pidana 1000.json')
```

4.2.4.2. Fungsi convertBERT()

Selanjutnya merupakan pembuatan fungsi yang bernama convertBERT() dengan beberapa parameter. Pertama, dataset, yang merupakan dataset yang telah dimuat pada tahap sebelumnya dan disimpan dalam variabel "df". Parameter berikutnya adalah nama kolom dari dataset, dengan default value "text" untuk kolom_text dan "tag" untuk kolom_tag. Fungsi ini bertujuan untuk mengubah dataset yang awalnya berupa token menjadi kalimat dalam bentuk string dengan memotong kalimat setiap kali ditemukan karakter titik koma. Kode untuk proses ini bisa dilihat dalam Kode 4.12 dan hasil dari proses convertBERT ini bisa dilihat dalam Tabel 3.6.

Kode 4.12 Fungsi convertBERT()

```
153.    def convertBERT(dataset, kolom_text='text', kolom_tag='text-tags'):
154.      new_dataset = pd.DataFrame(columns = ["text", "labels"])
155.      for i, row in tqdm(dataset.iterrows(), total=len(dataset),
    desc="Converting to BERT format"):
156.        text = ""
157.        tag = ""
158.        for index, (word, l) in enumerate(zip(row[kolom_text],
    row[kolom_tag])):
159.          if word != ';':
160.            text += word + " "
161.            tag += str(l) + " "
162.          else:
163.            text += word + " "
164.            tag += l + " "
165.            # Membuat DataFrame baru dari text dan tag
166.            data = pd.DataFrame({"text": [text.strip()], "labels":
    [tag.strip()]})
167.
```

```
168.            # Menambahkan DataFrame baru ke list atau dataset yang ada
169.            new_dataset = pd.concat([new_dataset, data], ignore_index=True)
170.
171.            text = ""
172.            tag = ""
173.
174.        return new_dataset
175.
176.    bert_format = convertBERT(df)
177.
178.    bert_format.to_csv("Dataset Pidana (BERT Format).csv", index=False)
```

Penjelasan Kode:

1. Definisi Fungsi

   Pada baris nomor 153 dibuat sebuah fungsi *convertBERT* menerima tiga parameter diantaranya yaitu “dataset” yang berisikan dataset yang akan diubah, “kolom_text” yang berisikan nama kolom dalam dataset yang berisi teks (default: 'text'), “kolom_tag” yang berisikan nama kolom dalam dataset yang berisi tag (default: 'text-tags').

2. Inisialisasi DataFrame Baru

   Pada baris nomor 154, dibuat DataFrame kosong dengan dua kolom "text" dan "labels".

3. Iterasi Melalui Dataset

   Pada baris nomor 155, digunakan *tqdm* untuk iterasi dengan progress bar dan *iterrows()* digunakan untuk iterasi baris demi baris dari dataset.

4. Inisialisasi Variabel Text dan Tag

   Pada baris nomor 156 dan 157, variabel “text” dan “tag” digunakan untuk menyimpan teks dan tag sementara.

5. Iterasi Melalui Kata dan Tag

   Pada baris nomor 158, digunakan *zip()* untuk mengiterasi kata (“word”) dan tag (“l”) secara bersamaan dari kolom yang sesuai.

6. Kondisi Untuk Memisahkan Berdasarkan Titik Koma

   Pada baris nomor 159 hingga 172, dilakukan pengecekan apakah “word” adalah titik koma (“;”):

   - Jika “word” bukan titik koma (“;”):
     - pada baris nomor 160 dan 161, menambahkan “word” dan “tag” ke variabel “text” dan “tag” dengan spasi.
   - Jika “word” adalah titik koma:

- Pada baris nomor 163 dan 164, menambahkan “word” dan “tag” ke variabel “text” dan “tag” dengan spasi.
- Pada baris nomor 166, membuat DataFrame baru “data” dengan “text” dan “tag” yang sudah terkumpul
- Pada baris nomor 169, menggabungkan DataFrame baru “data” ke “new_dataset”.
- Pada baris nomor 171 dan 172, mengosongkan variabel “text” dan “tag” untuk kalimat berikutnya:

7. Mengembalikan DataFrame Baru

   Pada baris nomor 174, mengembalikan “new_dataset” yang berisi teks dan tag dalam format yang sesuai untuk model BERT.

8. Menggunakan Fungsi dan Menyimpan Dataset

   Pada baris nomor 176 dan 178, memanggil fungsi “convertBERT” dengan dataset “df” dan menyimpan hasil dataset yang sudah diubah ke dalam file CSV dengan nama "Dataset Pidana (BERT Format).csv" tanpa indeks.

4.2.4.3. Fungsi remove()

Pada tahap ini, dibuat fungsi bernama remove() dengan parameter dataset, yang merupakan hasil dari fungsi convertBERT() sebelumnya. Tujuan dari fungsi ini adalah menghapus setiap kalimat yang tidak memiliki label entitas, seperti yang ditunjukkan pada Tabel 3.6 baris 2, dimana semua label hanya terdiri dari label "O" dan tidak ada label entitas lainnya. Sehingga dilakukan penghapusan untuk baris tersebut. Hasil nya bisa dilihat dalam Tabel 3.7 dan kode fungsi ini bisa dilihat dalam Kode 4.13.

Kode 4.13 Fungsi removeO()

```
179.    def removeO(dataset):
180.        indices_to_remove = []
181.        for i, (text, label) in tqdm(dataset.iterrows(), total=len(dataset),
    desc="Removing O"):
182.            if set(label.split()) == {'O'}:
183.                indices_to_remove.append(i)
184.
185.        dataset = dataset.drop(indices_to_remove).reset_index(drop=True)
186.        return dataset
187.
188.    remove_O = removeO(bert_format)
189.
190.    remove_O.to_csv("Dataset Pidana (BERT Format) Without O.csv",
    index=False)
```

Penjelasan Kode Fungsi Remove O:

1. Definisi Fungsi

   Fungsi “removeO” menerima parameter “dataset”, yang merupakan DataFrame berisi data teks dan label yang ingin diolah.

2. Inisialisasi Daftar Indeks untuk Dihapus

   Pada baris nomor 180, “indices_to_remove” diinisialisasi sebagai daftar kosong yang akan menyimpan indeks dari baris yang perlu dihapus.

3. Iterasi Melalui Dataset

   Pada baris nomor 181 hingga 183, dilakukan iterasi melalui setiap baris dalam dataset menggunakan “tqdm” untuk menunjukkan progress bar, berikut proses yang dilakukan:

   - Fungsi “iterrows()” digunakan untuk mendapatkan indeks “i”, teks “text”, dan label “label” untuk setiap baris.
   - Jika set label hanya berisi 'O', maka indeks “i” dari baris tersebut ditambahkan ke “indices_to_remove”.

4. Hapus Baris dengan Label 'O'

   Pada baris nomor 185, dataset di-drop berdasarkan “indices_to_remove” dan indeks dataset di-reset menggunakan “reset_index(drop=True)”.

5. Kembalikan Dataset

   Pada baris nomor 186, dataset yang telah dihapus baris dengan label 'O' dikembalikan sebagai hasil fungsi.

6. Panggil Fungsi “removeO” dan Simpan Dataset

   Pada baris nomor 188 hingga 190, fungsi “removeO” dipanggil dengan “bert_format” sebagai parameter, dan hasilnya disimpan dalam variabel “remove_O”dengan nama "Dataset Pidana (BERT Format) Without O.csv".

4.2.5. Modeling

4.2.5.1. *Instal* dan Impor *Library*

Tahap pertama dalam proses pemodelan sama seperti porses sebelumnya, yaitu dengan menginstal dan mengimpor *library* yang dibutuhkan, seperti *library transformers, dataloader,* dan *gdown*. Kode untuk *install library* bisa dilihat dalam Kode 4.14.

Kode 4.14 *Install Library* untuk Modeling

```
!pip install transformers
!pip install dataloader
!pip install gdown
```

Setelah instalasi selesai, langkah selanjutnya adalah mengimpor library yang dibutuhkan. Kode untuk mengimpor *library* bisa dilihat dalam Kode 4.15.

Kode 4.15 Impor Library untuk Modeling

```
from transformers import BertTokenizerFast
from transformers import BertForTokenClassification
from torch.utils.data import DataLoader
from sklearn.metrics import precision_recall_fscore_support
from sklearn.metrics import classification_report
from tqdm import tqdm
from sklearn.model_selection import KFold
from statistics import mean
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
import torch
import gdown
import warnings
import os
import numpy as np
warnings.filterwarnings('ignore')
os.environ['TOKENIZERS_PARALLELISM'] = 'true'
```

4.2.5.2. Mengunduh dan Memuat Dataset

Tahap selanjutnya adalah mengunduh dataset yang telah disimpan di Google Drive dengan menggunakan *library gdown*. Kode untuk mengunduh dataset bisa dilihat dalam Kode 4.16.

Kode 4.16 Mengunduh Dataset

```
gdown.download(f'https://drive.google.com/uc?id=1pgdVCs7CVOzyA6n6i3dY9xxDUi2dMqvm', 'Dataset Pidana 1000 Without O (BERT Format).csv', quiet=False)
```

Setelah proses pengunduhan selesai, dataset dimuat dan disimpan dalam variabel *df* dengan menggunakan *library pandas*. Kode memuat dataset ini dapat diliat pada Kode 4.17.

Kode 4.17 Memuat Dataset

```
df = pd.read_csv('Dataset Pidana 1000 Without O (BERT Format).csv')
df
```

4.2.5.3. Inisialisasi Label

Tahap selanjutnya adalah menginisialisasi label dengan mengambil seluruh label unik yang ada di dataset dan menyimpannya dalam variabel. Terdapat dua variabel yang digunakan, yaitu labels_to_ids dan ids_to_labels. Fungsi masing-

masing variabel ini adalah untuk menyimpan label, namun terdapat perbedaan. Variabel labels_to_ids digunakan untuk menyimpan Id dari label, di mana label merupakan key dan Id merupakan valuenya. Sedangkan ids_to_labels merupakan kebalikannya, di mana keynya adalah Id dan value-nya adalah label. Variabel labels_to_ids ini digunakan saat melakukan training maupun evaluasi, karena data awalnya berupa teks dan mesin hanya bisa membaca angka, sehingga diperlukan transformasi dari teks ke angka. Sementara ids_to_labels digunakan saat melihat hasil prediksi, karena model menghasilkan sebuah probabilitas indeks label, sehingga dibutuhkan key yang merupakan Id label. Kode untuk inisialisasi label bisa dilihat dalam Kode 4.18.

Kode 4.18 Inisialisasi Label

```
215.  # Split labels based on whitespace and turn them into a list
216.  labels = [i.split() for i in df['labels'].values.tolist()]
217.
218.  # Check how many labels are there in the dataset
219.  unique_labels = set()
220.
221.  for lb in labels:
222.    [unique_labels.add(i) for i in lb if i not in unique_labels]
223.
224.  print(unique_labels)
225.
226.  # Map each label into its id representation and vice versa
227.  labels_to_ids = {k: v for v, k in enumerate(sorted(unique_labels))}
228.  ids_to_labels = {v: k for v, k in enumerate(sorted(unique_labels))}
229.  print(labels_to_ids)
```

Penjelasan Kode:

1. Mengambil semua label yang ada dalam Dataset

   Pada baris nomor 215 hingga 216, label dalam kolom 'labels' dari dataframe “df” dipecah berdasarkan spasi dan di simpan dalam list “labels”

2. Membuat Data Label Unik

   Pada baris nomor 219 hingga 224, dilakukan pemeriksaan dan penambahan label yang unik:

   - Pada baris nomor 219, inisialisasi set “unique_labels” untuk menyimpan label unik.
   - Pada baris nomor 221 hingga 222, dilakukan iterasi melalui setiap label dalam “labels” dan menambahkan label unik ke dalam set “unique_labels”.
   - Pada baris nomor 224, mencetak label unik.

3. Memetakan Setiap Label ke dalam Representasi ID dan Sebaliknya

   Pada baris nomor 226 hingga 228, setiap label dipetakan ke dalam representasi ID dengan nama variabel “labels_to_ids” dan juga setiap ID di petakan ke dalam representasi Label.

4. Mencetak Kamus Label ke ID

   Pada baris nomor 229, mencetak kamus “labels_to_ids”.

4.2.5.4. Inisialisasi Model dan BERT Tokenizer

Tahap selanjutnya yaitu inisialisasi model BERT dengan menggunakan arsitektur BertForTokenClassification dari PyTorch. Kode inisialisasi model bisa dilihat dalam Kode 4.19.

Kode 4.19 Inisialisasi Model

```
230.    model = BertForTokenClassification.from_pretrained('indolem/indobert-
    base-uncased', num_labels=len(unique_labels))
```

Untuk inisialisasi BERT Tokenizer dengan menggunakan arsitektur BertTokenizerFast. Kode inisialisasi BERT Tokenizer bisa dilihat dalam Kode 4.20.

Kode 4.20 Inisialisasi BERT Tokenizer

```
231.    tokenizer = BertTokenizerFast.from_pretrained('indolem/indobert-base-
    uncased')
```

4.2.5.5. Data Sequence

Pada tahap ini, dibuat sebuah *class* dengan nama DataSequence. *Class* ini berfungsi sebagai transformasi data dari teks ke bentuk numerik. Selain *class* DataSequence terdapat satu fungsi yang bernama align_label yang berperan dalam mentransformasi label dari teks ke bentuk numerik. Proses transformasi teks menggunakan BERT Tokenizer yang telah di inisialisasi sebelumnya. Kode class DataSequence ini bisa dilihat dalam Kode 4.21.

Kode 4.21 Class DataSequence dan Fungsi Align_label

```
232.    def align_label(texts, labels, tokenizer, labels_to_ids):
233.        tokenized_inputs = tokenizer(texts, padding='max_length',
    max_length=512, truncation=True)
234.
235.        word_ids = tokenized_inputs.word_ids()
236.
237.        previous_word_idx = None
238.        label_ids = []
239.
240.        for word_idx in word_ids:
241.
242.            if word_idx is None:
```

```
243.                 label_ids.append(-100)
244.
245.             elif word_idx != previous_word_idx:
246.                 try:
247.                     label_ids.append(labels_to_ids[labels[word_idx]])
248.                 except:
249.                     label_ids.append(-100)
250.             else:
251.                 try:
252.                     label_ids.append(labels_to_ids[labels[word_idx]])
253.                 except:
254.                     label_ids.append(-100)
255.             previous_word_idx = word_idx
256.
257.         return label_ids
258.
259.     class DataSequence(torch.utils.data.Dataset):
260.
261.         def __init__(self, df):
262.
263.             lb = [i.split() for i in df['labels'].values.tolist()]
264.             txt = df['text'].values.tolist()
265.             self.texts = [tokenizer(str(i),
266.                                     padding='max_length', max_length = 512,
    truncation=True, return_tensors="pt") for i in txt]
267.             self.labels = [align_label(i,j, tokenizer, labels_to_ids) for
    i,j in zip(txt, lb)]
268.
269.         def __len__(self):
270.
271.             return len(self.labels)
272.
273.         def get_batch_data(self, idx):
274.
275.             return self.texts[idx]
276.
277.         def get_batch_labels(self, idx):
278.
279.             return torch.LongTensor(self.labels[idx])
280.
281.         def __getitem__(self, idx):
282.
283.             batch_data = self.get_batch_data(idx)
284.             batch_labels = self.get_batch_labels(idx)
285.
286.             return batch_data, batch_labels
```

Penjelasan Kode Fungsi “align_label”:

1. Definisi Fungsi

   Pada baris nomor 232, fungsi “align_label” menerima empat parameter yaitu "texts" yang berisikan teks yang akan dilakukan tokenisasi, "labels" yang berisikan label yang sesuai dengan teks, "tokenizer" yang berisikan tokenizer yang digunakan untuk memproses teks, "labels_to_ids" yang berisikan kamus yang memetakan label ke ID.

2. Tokenisasi Teks

   Pada baris nomor 233, dilakukan tokenisasi menggunakan “tokenizer” dengan padding dan panjang maksimal 512. Hal ini dilakukan untuk mendapatkan ID dari setiap token.

3. Mengambil ID token

   Pada baris nomor 235, mengambil ID setiap token dari hasil tokenisasi oleh tokenizer.

4. Inisialisasi Variabel

   Pada baris nomor 237 hingga 238, inisialisasi variabel “previous_word_idx” dan “label_ids”.

5. Proses Penyelarasan Label

   Pada baris nomor 240 hingga 255, dilakukan proses penyelarasan label dengan token berdasarkan id dari token:

   - Pada baris nomor 242 hingga 243, jika “word_idx” adalah “None”, tambahkan “-100” ke “label_ids”.
   - Pada baris nomor 245 hingga 254, jika “word_idx” berbeda dengan “previous_word_idx”, tambahkan label yang sesuai dari “labels_to_ids” atau “-100” jika terjadi kesalahan.

6. Mengembalikan Label yang Sudah Diselaraskan

   Pada baris nomor 257, kembalikan “label_ids” yang sudah diselaraskan.

Penjelasan Kode *Class* “DataSequence”:

1. Definisi *Class*

   Pada baris nomor 259, dibuat kelas “DataSequence” yang merupakan subclass dari “torch.utils.data.Dataset”.

2. Inisialisasi Kelas

   Pada baris nomor 261 hingga 267, inisialisasi kelas dilakukan:

   - Pada baris nomor 263, label dipecah berdasarkan spasi dan diubah menjadi sebuah list.
   - Pada baris nomor 264, teks diubah menjadi sebuah list.
   - Pada baris nomor 265 hingga 266, teks dilakukan tokenisasi menggunakan “tokenizer” dengan padding dan panjang maksimal 512.
   - Pada baris nomor 267, label diselaraskan dengan hasil tokenisasi menggunakan fungsi “align_label”.

3. Mengembalikan Panjang Dataset

   Pada baris nomor 269 hingga 271, sebuah methode untuk mengembalikan panjang dataset.

4. Mengambil Batch Data

   Pada baris nomor 273 hingga 275, sebuah methode untuk mengambil batch data berdasarkan indeks yang diberikan.

5. Mengambil Batch Label

   Pada baris nomor 277 hingga 279, sebuah methode untuk mengambil batch label berdasarkan indeks yang diberikan.

6. Mengambil Item

   Pada baris nomor 281 hingga 286, sebuah methode untuk mengambil item data dan label berdasarkan indeks yang diberikan.

4.2.5.6. Fungsi train loop()

Pada tahap ini dibuat sebuah fungsi dengan nama train_loop() yang berfungsi untuk memanggil class DataSequence, membagi data menjadi beberapa batch dan juga *fine tuning* model. Pembagian batch data ini dilakukan dengan menggunakan library DataLoader dengan paramter batch yang telah di tentukan pada skenario pengujian yang bisa dilihat dalam Kode 4.22 baris 10 dan 11. Proses *fine tuning* ini terdiri beberapa proses utama yaitu training dan validasi. Adapun alur proses model dilakukan beberapa proses diantaranya *Embedding Layer*, *Encoder Layer*, dan Layer Klasifikasi.

1. *Embedding Layer*

   Pada *Embedding Layer*, hasil dari BERT Tokenizer yang telah di proses sebelumnya dengan menggunakan *class DataSequence*, dibagi menjadi tiga bagian yaitu *word embedding*, *position embedding*, dan *token type embedding*. Namun, dalam tugas NER ini, token type embedding tidak digunakan. Masing-masing dimensi nya ada (1, 512, 768) dimana 1 adalah jumlah data, 512 adalah jumlah token, dan 768 adalah vector representasi tiap token. Hasil dari *word embeddings* dan *positional embedding* kemudian digabungkan dengan menjumlahkan kedua matriks tersebut. Dimensi dari hasil penggabungan ini tetap sama seperti sebelumnya, yaitu (1, 512, 768). Hasil

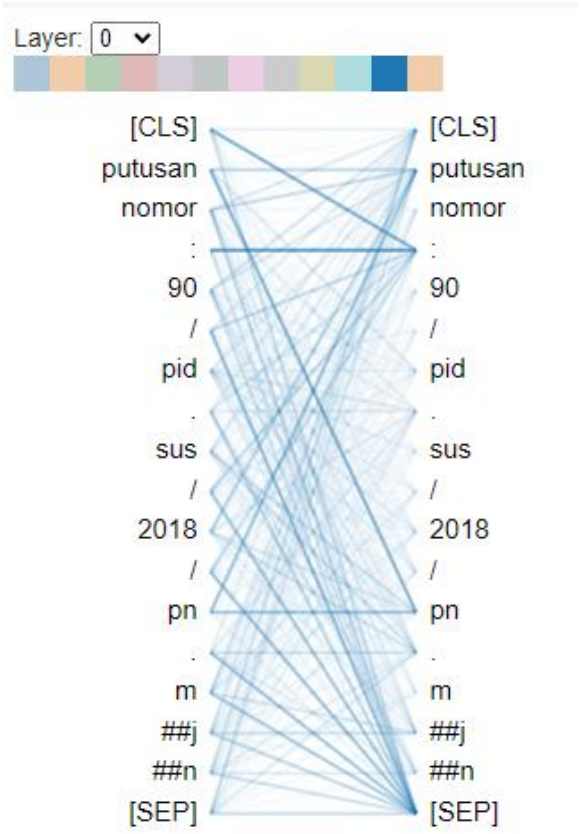
dari penggabungan, kemudian masuk ke dalam layer *Norm Embeddings* yang menghasilkan dimensi yang tetap sama seperti sebelumnya. Setalah masuk ke dalam layer Norm, proses dilanjutkan ke dropout embeddings, yang menghasilkan dimensi yang tetap sama seperti sebelumnya. Output dari dropout embeddings juga merupakan output akhir dari *Embeddings Layer* yang nantinya dilanjutkan pada proses *Encoder Layer*.

2. *Encoder Layer*

   Pada *encoder* layer, terdapat beberapa proses yang dilakukan, diantaranya *self attention*, *feed forwd network* dan 2 *residual connection* dan *normalisasi*.

   a. *Self Attention*

      Pada tahap ini dilakukan perhitungan attention pada setiap token engan membagi output embedding menjadi tiga bagian, yaitu *query*, *key*, dan *value*. Masing-masing bagian memiliki dimensi (1, 512, 768). Selanjutnya hasil dari query, key dan value dibagi untuk representasinya sehingga dihasilkan dimensi (1, 12, 512, 64) dimana 1 adalah jumlah data, 12 adalah jumlah head attention, 512 adalah jumlah token dan 64 adalah hasil pembagian antara vector representasi token dengan jumlah head, yaitu 12. Selanjutnya dilakukan perhitungan attention menggunakan persamaan (2.1), yang menghasilkan dimensi (1, 12, 512, 512). Dimensi ini berubah dari 64 menjadi 512, di mana 512 ini merupakan jumlah token. Hasil perhitungan attention kemudian dimasukkan ke dalam layer dropout, yang dalam implementasinya disebut sebagai dropout 1 attention yang menghasilkan dimensi yang tetap sama. Dan visualisasi untuk setiap token bisa dilihat dalam Gambar 4.7. dimana semakin tebal garis nya mengartikan semakin tinggi nilai *attention* nya.



Gambar 4.7 Visualisasi Attention

Hasil dropout kemudian dilanjutkan dengan persamaan (2.1) yaitu dengan *dot product* dari *value* yang menghasilkan dimensi (1, 12, 512, 64). Dari hasil dot product ini, kemudian dilakukan penggabungan head attention dengan cara *transpose* dan *reshape* yang menghasilkan dimensi awal yaitu (1, 512, 768).

b. *Residual Connection* dan *Layer Norm*

Pada *residual connection* dan normalisasi yang pertama ini, hasil dari *self attention* dimasukkan ke dalam layer linear yang dalam implementasinya disebut sebagai *attention-dense*, yang menghasilkan dimensi tetap sama dengan sebelumnya.

Setelah dilakukan proses *attention-dense* kemudian dilanjutkan ke dalam proses normalisasi menggunakan layerNorm yang dalam implementasinya disebut dengan *attention-LayerNorm*.mProses selanjutnya adalah *residual connection* yang dalam implementasinya disebut *residual connection (attention),* dengan menambahkan *output embeddings* yang awal dengan hasil dari *attention-LayerNorm*. Porses

selanjutnya adalah masuk ke dalam dropout ke dua dalam implementasinya disebut dengan *dropout 2 attention*.

c. *Feed Forward Network*

Pada proses selanjutnya, yaitu Feed Forward Network dengan terdapat beberapa proses diantaranya layer linear yang dalam impelementasinya disebut sebagai *intermediate-dense*. Proses ini menghasilkan dimensi yang berbeda dengan sebelumnya yaitu (1, 512, 3072) dimana 3072 merupakan ukuran dari *Feed Forward Network*. Proses selanjutnya kemudian dilakukan perhitungan fungsi aktivasi gelu dengan menggunakan persamaan (2.4). Dalam implementasinya proses ini disebut sebagai *intermediate-gelu*. Proses ini menghasilkan dimensi yang tetap sama dengan sebelumnya. Proses selanjutnya yaitu dengan memasukkan hasil intermediate-gelu ke dalam layer linear, dalam implementasinya disebut sebagai *output-dense* yang menghasilkan dimensi berbeda dengan sebelumnya yaitu (1, 512, 768).

d. *Residual Connection* dan Normalisasi

Setelah proses *Feed Forward Network*, dilanjutkan proses selanjutnya yaitu residual connection dan normalisasi yang kedua. Proses pertama akan masuk ke dalam layer normalisasi, dalam implementasinya disebut dengan *LayerNorm (output-LayerNorm)*, dengan menghasilkan dimensi yang sama dengan sebelumnya. Proses selanjutnya yaitu residual connection, dalam implementasinya disebut dengan *residual connection ffn*, dengan menambahkan hasil LayerNorm (output-LayerNorm) dengan output dari *self-attention*. Proses selanjutnya yaitu dengan memasukkan ke dalam dropout yang dalam implementasinya disebut dengan *output dropout (output-dropout)*, hasil dropout ini merupakan hasil akhir dari *encoder* BERT, dengan dimensi (1, 512, 768). Hasil akhir *encoder* ini selanjutnya di proses dalam layer klasifikasi.

3. Layer Klasifikasi

Pada tahap ini, dilakukan klasifikasi berdasarkan hasil *encoder* dengan dilakukan beberapa proses diantaranya dropout, dengan menghasilkan dimensi (1, 512, 768). Hasil dari dropout kemudian masuk kedalam layer

linear yang dalam implementasinya disebut dengan classifier. Hasil dari classifier ini juga merupakan hasil akhir yang berupa probabilitas label dengan dimensi (1, 512, 25) dimana 25 merupakan jumlah label.

Pada proses yang di jelaskan sebelumnya, merupakan alur model. Proses fine tuning sendriri terdiri dari training dan validasi dengan jumlah epoch yang di tentukan yang bisa dilihat dalam Kode 4.22. Setelah proses training dan validasi selesai sesuai dengan epoch yang di tentukan, fungsi ini menyimpan model hasil training yang ada pada Kode 4.22 baris 93.

Kode 4.22 Fungsi train_loop()

```
287.    def train_loop(model, df, batch_size, num_epochs, learning_rate,
    fold_num):
288.        use_cuda = torch.cuda.is_available()
289.        device = torch.device("cuda" if use_cuda else "cpu")
290.
291.        df_train, df_val = np.split(df.sample(frac=1,
    random_state=42),[int(.9 * len(df))])
292.
293.        train_dataset = DataSequence(df_train)
294.        val_dataset = DataSequence(df_val)
295.
296.        train_loader = DataLoader(train_dataset, num_workers=4,
    batch_size=batch_size)
297.        val_loader = DataLoader(val_dataset, num_workers=4,
    batch_size=batch_size)
298.
299.        optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)
300.
301.        if use_cuda:
302.            model = model.cuda()
303.
304.        for epoch_num in range(num_epochs):
305.            print(f"\nFold {fold_num + 1} - Epoch {epoch_num + 1}")
306.
307.            total_loss_train = 0
308.            all_true_labels_train = []
309.            all_predicted_labels_train = []
310.
311.            model.train()
312.
313.            for train_data, train_label in tqdm(train_loader):
314.                train_label = train_label.to(device)
315.
316.                mask = train_data['attention_mask'].squeeze(1).to(device)
317.                input_id = train_data['input_ids'].squeeze(1).to(device)
318.
319.                optimizer.zero_grad()
320.
321.                loss, logits = model(input_id, attention_mask=mask,
    labels=train_label, return_dict=False)
322.
323.                for i in range(logits.shape[0]):
324.
325.                    logits_clean = logits[i][train_label[i] != -100]
326.                    label_clean = train_label[i][train_label[i] != -100]
327.
328.                    predictions = logits_clean.argmax(dim=1)
329.                    total_loss_train += loss.item()
330.
331.                    label_convert_train = [ids_to_labels[label] for label in
    label_clean.cpu().numpy() if label != -100]
332.                    convert_pred_train = [ids_to_labels[label] for label in
    predictions.cpu().numpy() if label != -100]
333.
334.                    all_true_labels_train.extend(label_convert_train)
335.                    all_predicted_labels_train.extend(convert_pred_train)
336.
337.                loss.backward()
338.                optimizer.step()
339.
```

```
340.            model.eval()
341.
342.            total_loss_val = 0
343.            all_true_labels_val = []
344.            all_predicted_labels_val = []
345.            class_names = ['B_ADVO', 'B_ARTV', 'B_CRIA', 'B_DEFN', 'B_JUDG',
    'B_JUDP', 'B_PENA', 'B_PROS', 'B_PUNI', 'B_REGI', 'B_TIMV', 'B_VERN',
    'I_ADVO', 'I_ARTV', 'I_CRIA', 'I_DEFN', 'I_JUDG', 'I_JUDP', 'I_PENA',
    'I_PROS', 'I_PUNI', 'I_REGI', 'I_TIMV', 'I_VERN', 'O']
346.
347.            for val_data, val_label in val_loader:
348.                val_label = val_label.to(device)
349.                mask = val_data['attention_mask'].squeeze(1).to(device)
350.                input_id = val_data['input_ids'].squeeze(1).to(device)
351.
352.                loss, logits = model(input_id, attention_mask=mask,
    labels=val_label, return_dict=False)
353.
354.                for i in range(logits.shape[0]):
355.                    logits_clean = logits[i][val_label[i] != -100]
356.                    label_clean = val_label[i][val_label[i] != -100]
357.
358.                    predictions = logits_clean.argmax(dim=1)
359.                    total_loss_val += loss.item()
360.
361.                    label_convert = [ids_to_labels[label] for label in
    label_clean.cpu().numpy() if label != -100]
362.                    convert_pred = [ids_to_labels[label] for label in
    predictions.cpu().numpy() if label != -100]
363.
364.                    all_true_labels_val.extend(label_convert)
365.                    all_predicted_labels_val.extend(convert_pred)
366.
367.            # Calculate F1-Score
368.            matrik_evaluasi_train =
    classification_report(y_pred=all_predicted_labels_train,
    y_true=all_true_labels_train, output_dict=True)['macro avg']
369.            matrik_evaluasi_val =
    classification_report(y_pred=all_predicted_labels_val,
    y_true=all_true_labels_val, output_dict=True)['macro avg']
370.            print(f'Epochs: {epoch_num + 1} | Train Loss: {total_loss_train
    / len(df_train): .3f} | Train F1-Score: {matrik_evaluasi_train["f1-score"]:
    .3f} | Val Loss: {total_loss_val / len(df_val): .3f} | Val F1-Score:
    {matrik_evaluasi_val["f1-score"]: .3f}')
371.
372.
373.        # Simpan model setiap fold
374.        torch.save(model.state_dict(), f'indoBERT-indoLEM-Fold-{fold_num +
    1}.pth')
375.        return model
```

Penjelasan Kode:

1. Definisi Fungsi

   Pada baris nomor 287, fungsi “train_loop” menerima beberapa parameter yaitu "model" yang berisikan model pretrain yang digunakan, "df" yang berisikan DataFrame yang berisi data train, "batch_size" yang berisikan ukuran batch yang digunakan selama proses training, "num_epochs" yang berisikan jumlah epoch pada proses training, "learning_rate" yang berisikan besar learning rate yang digunakan, "fold_num" yang berisikan jumlah Fold yang digunakan untuk splitting dataset.

2. Inisialisasi CUDA

   Pada baris nomor 288 hingga 289, dilakukan pengecekan ketersediaan CUDA (GPU).

3. Pembagian Data Train dan Validasi

   Pada baris nomor 291, DataFrame “df” dibagi menjadi data train (“df_train”) dan data validasi (“df_val”) menggunakan “np.split”.

4. Inisialisasi Dataset dan DataLoader

   - Pada baris nomor 293 hingga 294, transformasi data ke bentuk numerik dengan menggunakan objek “DataSequence” untuk “train_dataset” dan “val_dataset” menggunakan data yang telah dibagi sebelumnya.
   - “DataLoader” digunakan untuk memuat data dalam batch selama training (“train_loader” untuk data train dan “val_loader” untuk data validasi).

5. Optimizer

   Pada baris nomor 299, “Adam” optimizer digunakan untuk mengoptimalkan parameter model.

6. Pindahkan Model ke GPU

   Pada baris nomor 301 hingga 302, jika CUDA tersedia, model dipindahkan ke GPU.

7. Loop Pelatihan (Epoch)

   Pada baris nomor 304 hingga 370, dilakukan loop untuk setiap epoch:

   - **Mode Training** (baris 25 hingga 52):
     a. Model diatur ke mode pelatihan (“model.train()”).
     b. Dilakukan iterasi berdasarkan “train_loader” untuk setiap batch data dan label, dengan melakukan predict data train melalui model pada baris 35. Model akan memberikan attention pada setiap token dengan menggunakan persamaan (2.1).
     c. Loss dihitung dengan memanggil model dan optimizer (“optimizer.zero_grad()”, “loss.backward()”, dan “optimizer.step()”).
     d. Selama iterasi, label dan prediksi yang benar disimpan untuk perhitungan metrik.
   - **Mode Evaluasi** (baris 54 hingga 79):
     a. Model diatur ke mode evaluasi (“model.eval()”).
     b. Dilakukan iterasi nerdasarkan “val_loader” untuk setiap batch data dan label validasi. Proses ini sama dengan proses mode train sebelumnya

namun dalam mode evaluasi ini dropout dan layer normaliasasi tidak di aktifkan.

c. Loss dihitung dengan memanggil model.

d. Metrik seperti F1-score dihitung untuk data validasi.

- **Logging** (baris 82 hingga 84):

a. Outputkan loss dan F1-score untuk setiap epoch.

8. Simpan Model

Pada baris nomor 374, model disimpan setelah selesai pelatihan untuk setiap lipatan.

9. Kembalikan Model

Pada baris nomor 375, model yang telah dilatih dikembalikan sebagai hasil dari fungsi.

4.2.5.7. Fungsi kfold_cross_validation()

Pada tahap ini, dibuat sebuah fungsi bernama kfold_cross_validation() yang digunakan untuk membagi dataset menjadi 5 fold. Namun, terdapat perbedaan dalam penggunaan fungsi ini antara model IndoBERT (IndoLEM) dan IndoBERT (IndoNLU). Fungsi kfold_cross_validation untuk model IndoBERT (IndoLEM) bisa dilihat dalam Kode 4.23. Dalam fungsi ini, dataset dibagi menjadi 5 fold dan di simpan dalam bentuk csv. Setelah dataset dibagi menjadi 5 fold, fungsi train_loop() dipanggil untuk melakukan proses training berdasarkan dataset fold pertama. Setelah proses train_loop() selesai, dilakukan evaluasi dengan memanggil fungsi evaluasi yang bisa dilihat dalam Kode 4.27. Hasil evaluasi ini mengembalikan nilai berupa precison recall dan f1-score. Proses ini dilanjutkan dari fold pertama hingga fold terakhir.

Kode 4.23 Fungsi kfold model *Indolem/indobert-base-uncased*

```
376.    def kfold_cross_validation(model, df, num_folds=5, batch_size_train=4,
    batch_size_test=2, num_epochs=3, learning_rate=5e-3):
377.        kf = KFold(n_splits=num_folds, shuffle=True)
378.        use_cuda = torch.cuda.is_available()
379.        device = torch.device("cuda" if use_cuda else "cpu")
380.
381.        precision_result = {}
382.        recall_result = {}
383.        f1_score_result = {}
384.
385.        for fold_num, (train_index, test_index) in enumerate(kf.split(df)):
386.            df_train, df_test = df.iloc[train_index], df.iloc[test_index]
387.            df_train.to_csv(f"Training set Fold-{fold_num + 1}.csv",
    index=False)
388.            df_test.to_csv(f"Test set Fold-{fold_num + 1}.csv", index=False)
389.
390.            model_copy = train_loop(model, df_train, batch_size_train,
    num_epochs, learning_rate, fold_num)
```

```
391.
392.            print(f'\n========== Evaluasi Model ===========')
393.            evaluasi_result = evaluate(model_copy, df_test, batch_size_test)
394.            precision_result[fold_num + 1] = evaluasi_result['precision']
395.            recall_result[fold_num + 1] = evaluasi_result['recall']
396.            f1_score_result[fold_num + 1] = evaluasi_result['f1-score']
397.
398.            print(f'\n======= Uji Coba dengan Data Baru ========')
399.            evaluate_one_text(model_copy, 'PUTUSAN . NOMOR : 187 / Pid . Sus
    / 2014 / PN . JKT . TIM . DEMI KEADILAN BERDASARKAN KETUHANAN YANG MAHA ESA
    . ')
400.            print()
401.            evaluate_one_text(model_copy, 'MENUNTUT : 1 Menyatakan terdakwa
    AGNES TRI AHADI Als AGNES telah terbukti secara sah dan meyakinkan bersalah
    melakukan tindak pidana Narkotika memiliki , menyimpan , menguasai , atau
    menyediakan Narkotika golongan I bukan tanaman sebagaimana didakwakan dalam
    dakwaan kedua yaitu melanggar ketentuan unsure pasal 112 ayat ( 1 ) UURI No
    . 35 tahun 2009 tentang Narkotika ; ')
402.            print()
403.            evaluate_one_text(model_copy, 'PUTUSAN Nomor 77/Pid.B/2023/PN
    Jkt.Pst DEMI KEADILAN BERDASARKAN KETUHANAN YANG MAHA ESA Pengadilan Negeri
    Jakarta Pusat yang mengadili perkara pidana dengan acara pemeriksaan biasa
    dalam tingkat pertama menjatuhkan putusan sebagai berikut dalam perkara
    Terdakwa : 1.      Nama lengkap   : Arif Bin Santung')
404.
405.        # Sort hasil evaluasi dari yang terbesar ke terkecil
406.        precision_sorted = sorted(precision_result.items(), key=lambda x:
    x[1], reverse=True)
407.        max_precision_key, max_precision_value = precision_sorted[0]
408.
409.        recall_sorted = sorted(recall_result.items(), key=lambda x: x[1],
    reverse=True)
410.        max_recall_key, max_recall_value = recall_sorted[0]
411.
412.        f1_sorted = sorted(f1_score_result.items(), key=lambda x: x[1],
    reverse=True)
413.        max_f1_key, max_f1_value = f1_sorted[0]
414.
415.        # Rata-Rata Evaluasi
416.        precision_avg = np.mean(tuple(precision_result.values()))
417.        recall_avg = np.mean(tuple(recall_result.values()))
418.        f1_avg = np.mean(tuple(f1_score_result.values()))
419.
420.        print('\n========================= RESULT
    ============================')
421.        print(f'Average Precision : {precision_avg:.3f} | Average Recall :
    {recall_avg:.3f} | Average F1-Score : {f1_avg:.3f}')
422.        print(f'Model dengan Precision Tertinggi : Fold {max_precision_key}
    -> Precision : {max_precision_value:.3f}')
423.        print(f'Model dengan Recall Tertinggi : Fold {max_recall_key} ->
    Recall : {max_recall_value:.3f}')
424.        print(f'Model dengan F1-Score Tertinggi : Fold {max_f1_key} -> F1-
    Score : {max_f1_value:.3f}')
```

Penjelasan Kode untuk model *Indolem/indobert-base-uncased*:

1. Definisi Fungsi

   Pada baris nomor 376 dibuat sebuah fungsi “kfold_cross_validation” yang menerima beberapa parameter:

   - “model” = Model pretrain yang digunakan.
   - “df” = DataFrame yang berisi data untuk k-fold cross-validation.
   - “num_folds” = Jumlah folds untuk k-fold cross-validation (default = 5).
   - “batch_size_train” = Ukuran batch untuk data train (default = 4).
   - “batch_size_test” = Ukuran batch untuk data test (default = 2).
   - “num_epochs” = Jumlah epoch untuk pelatihan (default = 3).

- “learning_rate” = Learning rate untuk optimizer (default = 5e-3).

2. Inisialisasi K-Fold

   Pada baris nomor 377, KFold dari scikit-learn diinisialisasi dengan jumlah fold “num_folds” dan parameter “shuffle=True” untuk mengacak data sebelum dibagi menjadi beberapa fold.

3. Pengecekan CUDA

   Pada baris nomor 378 hingga 379, fungsi memeriksa ketersediaan CUDA dan menentukan perangkat (“device”) yang akan digunakan untuk proses training.

4. Inisialisasi Hasil Evaluasi

   Pada baris nomor 381 hingga 383, tiga dictionary (“precision_result”, “recall_result”, “f1_score_result”) didefinisikan untuk menyimpan hasil evaluasi precision, recall, dan F1-score dari setiap fold.

5. Loop K-Fold

   Pada baris 385 hingga 404, terdapat loop utama untuk k-fold cross-validation. Pada setiap iterasi:

   - Pada baris 386, data train (“df_train”) dan data test (“df_test”) dibagi berdasarkan indeks yang dihasilkan oleh KFold.
   - Pada baris 387 dan 388, data train dan data test disimpan dalam bentuk CSV pada setiap fold.
   - Pada baris 390, model dilatih dengan memanggil fungsi “train_loop” menggunakan data train.
   - Pada baris 392 hingga 396, model yang telah dilatih dievaluasi menggunakan data test dengan memanggil fungsi “evaluate”, dan hasilnya disimpan dalam “precision_result”, “recall_result”, dan “f1_score_result”.
   - Pada baris 398 hingga 403, model diuji dengan beberapa contoh teks menggunakan fungsi “evaluate_one_text” untuk menunjukkan performa model pada data baru.

6. Mengurutkan Hasil Evaluasi

   Pada baris 405 hingga 413, hasil evaluasi diurutkan dari yang terbesar ke terkecil untuk precision, recall, dan F1-score. Nilai tertinggi dan fold yang sesuai akan di simpan.

7. Menghitung Rata-Rata Evaluasi

   Pada baris 415 hingga 418, rata-rata dari precision, recall, dan F1-score dari semua lipatan dihitung menggunakan “np.mean”.

8. Menampilkan Hasil

   Pada baris 420 hingga 424, hasil evaluasi dicetak, termasuk rata-rata precision, recall, dan F1-score, serta lipatan dengan precision, recall, dan F1-score tertinggi.

Sedangkan fungsi kfold_cross_validation untuk model *Indobenchmark/indobert-base-p2* tidak lagi pembagian dataset menjadi5 fold secara langsung, melainkan menggunakan dataset yang telah dibagi menjadi 5 fold oleh fungsi yang digunakan untuk model *Indolem/indobert-base-uncased*, yang di simpan dalam bentuk csv sebelumnya. Kode fungsi kfold_cross_validation untuk model *Indobenchmark/indobert-base-p2* ini bisa dilihat dalam Kode 4.24.

Kode 4.24 Fungsi kfold model *Indobenchmark/indobert-base-p2*

```
425.    def kfold_cross_validation(model, path_train_folds, path_test_folds,
    batch_size_train=4, batch_size_test=2, num_epochs=3, learning_rate=5e-3):
426.        use_cuda = torch.cuda.is_available()
427.        device = torch.device("cuda" if use_cuda else "cpu")
428.
429.        precision_result = {}
430.        recall_result = {}
431.        f1_score_result = {}
432.
433.        for fold_num, (train_fold, test_fold) in
    enumerate(zip(os.listdir(path_train_folds), os.listdir(path_test_folds))):
434.            df_train, df_test =
    pd.read_csv(f'{path_test_folds}/{train_fold}'),
    pd.read_csv(f'{path_test_folds}/{test_fold}')
435.
436.            model_copy = train_loop(model, df_train, batch_size_train,
    num_epochs, learning_rate, fold_num)
437.
438.            print(f'\n========== Evaluasi Model ===========')
439.            evaluasi_result = evaluate(model_copy, df_test, batch_size_test)
440.            precision_result[fold_num + 1] = evaluasi_result['precision']
441.            recall_result[fold_num + 1] = evaluasi_result['recall']
442.            f1_score_result[fold_num + 1] = evaluasi_result['f1-score']
443.
444.            print(f'\n======= Uji Coba dengan Data Baru ========')
445.            evaluate_one_text(model_copy, 'PUTUSAN . NOMOR : 187 / Pid . Sus
    / 2014 / PN . JKT . TIM . DEMI KEADILAN BERDASARKAN KETUHANAN YANG MAHA ESA
    . ')
446.            print()
447.            evaluate_one_text(model_copy, 'MENUNTUT : 1 Menyatakan terdakwa
    AGNES TRI AHADI Als AGNES telah terbukti secara sah dan meyakinkan bersalah
    melakukan tindak pidana Narkotika memiliki , menyimpan , menguasai , atau
    menyediakan Narkotika golongan I bukan tanaman sebagaimana didakwakan dalam
    dakwaan kedua yaitu melanggar ketentuan unsure pasal 112 ayat ( 1 ) UURI No
    . 35 tahun 2009 tentang Narkotika ; ')
448.            print()
449.            evaluate_one_text(model_copy, 'PUTUSAN Nomor 77/Pid.B/2023/PN
    Jkt.Pst DEMI KEADILAN BERDASARKAN KETUHANAN YANG MAHA ESA Pengadilan Negeri
    Jakarta Pusat yang mengadili perkara pidana dengan acara pemeriksaan biasa
    dalam tingkat pertama menjatuhkan putusan sebagai berikut dalam perkara
    Terdakwa : 1.      Nama lengkap   : Arif Bin Santung')
450.
451.        # Sort hasil evaluasi dari yang terbesar ke terkecil
```

```
452.        precision_sorted = sorted(precision_result.items(), key=lambda x:
    x[1], reverse=True)
453.        max_precision_key, max_precision_value = precision_sorted[0]
454.
455.        recall_sorted = sorted(recall_result.items(), key=lambda x: x[1],
    reverse=True)
456.        max_recall_key, max_recall_value = recall_sorted[0]
457.
458.        f1_sorted = sorted(f1_score_result.items(), key=lambda x: x[1],
    reverse=True)
459.        max_f1_key, max_f1_value = f1_sorted[0]
460.
461.        # Rata-Rata Evaluasi
462.        precision_avg = np.mean(tuple(precision_result.values()))
463.        recall_avg = np.mean(tuple(recall_result.values()))
464.        f1_avg = np.mean(tuple(f1_score_result.values()))
465.
466.        print('\n======================== RESULT
    ==============================')
467.        print(f'Average Precision : {precision_avg:.3f} | Average Recall :
    {recall_avg:.3f} | Average F1-Score : {f1_avg:.3f}')
468.        print(f'Model dengan Precision Tertinggi : Fold {max_precision_key}
    -> Precision : {max_precision_value:.3f}')
469.        print(f'Model dengan Recall Tertinggi : Fold {max_recall_key} ->
    Recall : {max_recall_value:.3f}')
470.    print(f'Model dengan F1-Score Tertinggi : Fold {max_f1_key} -> F1-Score
    : {max_f1_value:.3f}')
```

Penjelasan Kode untuk model *Indobenchmark/indobert-base-p2*:

1. Definisi Fungsi

   Pada baris nomor 425 dibuat sebuah fungsi “kfold_cross_validation” yang menerima beberapa parameter:

   - “model” = Model pretraind yang digunakan.
   - “path_train_folds” = Path direktori yang berisi data train untuk setiap fold.
   - “path_test_folds” = Path direktori yang berisi data test untuk setiap fold.
   - “batch_size_train” = Ukuran batch untuk data train (default = 4).
   - “batch_size_test” = Ukuran batch untuk data test (default = 2).
   - “num_epochs” = Jumlah epoch untuk pelatihan (default = 3).
   - “learning_rate” = Learning rate untuk optimizer (default = 5e-3).

2. Pengecekan CUDA

   Pada baris nomor 426 hingga 427, fungsi memeriksa ketersediaan CUDA dan menentukan perangkat (“device”) yang akan digunakan untuk proses training.

3. Inisialisasi Hasil Evaluasi

   Pada baris nomor 429 hingga 431, tiga dictionary (“precision_result”, “recall_result”, “f1_score_result”) didefinisikan untuk menyimpan hasil evaluasi precision, recall, dan F1-score dari setiap lipatan (fold) k-fold cross-validation.

4. Loop K-Fold

   Pada baris 433 hingga 449, terdapat loop utama untuk k-fold cross-validation. Pada setiap iterasi:

   - "train_fold" dan "test_fold" diperoleh dari enumerasi list file dalam direktori "path_train_folds" dan "path_test_folds".
   - Data train ("df_train") dan data test ("df_test") dibaca dari file CSV menggunakan "pd.read_csv".
   - Model dilatih dengan memanggil "train_loop" menggunakan data train.
   - Model yang telah dilatih dievaluasi menggunakan data test dengan memanggil "evaluate", dan hasilnya disimpan dalam "precision_result", "recall_result", dan "f1_score_result".
   - Model diuji dengan beberapa contoh teks menggunakan "evaluate_one_text" untuk menunjukkan performa model pada data baru.

5. Mengurutkan Hasil Evaluasi

   Pada baris 451 hingga 459, hasil evaluasi diurutkan dari yang terbesar ke terkecil untuk precision, recall, dan F1-score. Nilai tertinggi pada fold yang sesuai akan di simpan.

6. Menghitung Rata-Rata Evaluasi

   Pada baris 461 hingga 464, menghitung rata-rata dari precision, recall, dan F1-score dari semua fold.

7. Menampilkan Hasil

   Pada baris 466 hingga 470, hasil evaluasi dicetak, termasuk rata-rata precision, recall, dan F1-score, serta fold dengan precision, recall, dan F1-score tertinggi.

4.2.5.8. Inisialisasi Parameter dan Training Model

Pada tahap ini, dilakukan inisialisasi parameter seperti learning rate, epoch, batch size, dan jumlah fold. Parameter-parameter inisialisasi ini kemudian dimasukkan ke dalam fungsi kfold_cross_validation() yang telah dibuat sebelumnya. Terdapat sedikit perbedaan dalam inisialisasi parameter antara model *Indolem/indobert-base-uncased* dan Indobenchmark*/indobert-base-p2*, di mana model Indobenchmark*/indobert-base-p2* memerlukan lokasi penyimpanan dataset yang telah dibagi oleh fungsi kfold_cross_validation untuk model *Indolem/indobert-base-uncased*. Kode untuk inisialisasi paramter model

Indolem/indobert-base-uncased bisa dilihat dalam Kode 4.25, sementara kode untuk inisialisasi parameter model *Indobenchmark/indobert-base-p2* bisa dilihat dalam Kode 4.26.

Kode 4.25 Insialisasi Parameter dan Training Model *Indolem/indobert-base-uncased*

```
    LEARNING_RATE = 0.00001
    EPOCHS = 3
    BATCH_SIZE_TRAIN = 4
    BATCH_SIZE_TEST = 2
    FOLDS = 5

    # Training
    kfold_cross_validation(model, df, num_folds=FOLDS,
batch_size_train=BATCH_SIZE_TRAIN, batch_size_test=BATCH_SIZE_TEST,
num_epochs=EPOCHS, learning_rate=LEARNING_RATE)
```

Kode 4.26 Insialisasi Parameter dan Training Model *Indobenchmark/indobert-base-p2*

```
    LEARNING_RATE = 0.00001
    EPOCHS = 1
    BATCH_SIZE_TRAIN = 4
    BATCH_SIZE_TEST = 2
    PATH_TRAIN_FOLDS = "/kaggle/input/5-fold-dataset/Train Fold"
    PATH_TEST_FOLDS = "/kaggle/input/5-fold-dataset/Train Fold"

    # Training
    kfold_cross_validation(model, path_train_folds=PATH_TRAIN_FOLDS,
path_test_folds=PATH_TEST_FOLDS, batch_size_train=BATCH_SIZE_TRAIN,
batch_size_test=BATCH_SIZE_TEST, num_epochs=EPOCHS,
learning_rate=LEARNING_RATE)
```

4.2.6. Evaluasi

4.2.6.1. Fungsi Evaluate

Tahap ini merupakan pembuatan fungsi evaluasi yang digunakan pada tahap *fine tuning* model. Proses evaluasi ini hampir dengan proses fine tuning yaitu dengan memanggil *class* DataSequence untuk transformsi data dari teks ke bentuk numerik serta membagi data menjadi beberapa batch data dengan menggunakan DataLoader. Kode fungsi evaluasi ini bisa dilihat dalam Kode 4.27.

Kode 4.27 Fungsi Evaluate

```
    def evaluate(model, df_test, batch_size_test):

        test_dataset = DataSequence(df_test)

        test_dataloader = DataLoader(test_dataset, num_workers=4,
batch_size=batch_size_test)

        use_cuda = torch.cuda.is_available()
        device = torch.device("cuda" if use_cuda else "cpu")

        if use_cuda:
            model = model.cuda()

        all_true_labels = []
        all_predicted_labels = []

        for test_data, test_label in test_dataloader:
```

```
504.             test_label = test_label.to(device)
505.             mask = test_data['attention_mask'].squeeze(1).to(device)
506.             input_id = test_data['input_ids'].squeeze(1).to(device)
507.
508.             loss, logits = model(input_id, attention_mask=mask,
    labels=test_label, return_dict=False)
509.
510.             for i in range(logits.shape[0]):
511.                 logits_clean = logits[i][test_label[i] != -100]
512.                 label_clean = test_label[i][test_label[i] != -100]
513.
514.                 predictions = logits_clean.argmax(dim=1)
515.
516.                 label_convert = [ids_to_labels[label] for label in
    label_clean.cpu().numpy() if label != -100]
517.                 convert_pred = [ids_to_labels[label] for label in
    predictions.cpu().numpy() if label != -100]
518.
519.                 all_true_labels.extend(label_convert)
520.                 all_predicted_labels.extend(convert_pred)
521.
522.         # Calculate F1-Score
523.         view_report = classification_report(y_pred=all_predicted_labels,
    y_true=all_true_labels)
524.         matiks_evaluasi = classification_report(y_pred=all_predicted_labels,
    y_true=all_true_labels, output_dict=True)['weighted avg']
525.         print(view_report)
526.         print(f'Precision: {matiks_evaluasi["precision"]:.3f} | Recall:
    {matiks_evaluasi["recall"]:.3f} | F1-Score: {matiks_evaluasi["f1-
    score"]:.3f}')
527.
528.         return matiks_evaluasi
```

Penjelasan Kode:

1. Definisi Fungsi

   Pada baris nomor 488 dibuat sebuah fungsi “evaluate” yang menerima tiga parameter:

   - “model” = Model yang akan dievaluasi.
   - “df_test” = DataFrame yang berisi data test.
   - “batch_size_test” = Ukuran batch untuk data test.

2. Pembuatan Dataset Test

   Pada baris nomor 490, dataset test dibuat menggunakan kelas “DataSequence” dengan “df_test” sebagai parameter.

3. Pembuatan DataLoader

   Pada baris nomor 492, DataLoader dibuat untuk dataset test dengan parameter “num_workers” bernilai 4 dan “batch_size” bernilai sesuai dengan yang di tentukan.

4. Pengecekan CUDA

   Pada baris nomor 494 hingga 495, fungsi memeriksa ketersediaan CUDA dan menentukan perangkat (“device”) yang akan digunakan untuk evaluasi.

5. Pemindahan Model ke CUDA (Jika Tersedia)

   Pada baris 497 hingga 498, model dipindahkan ke CUDA jika tersedia.

6. Inisialisasi List untuk Label Asli dan Prediksi

   Pada baris 500 hingga 501, dua list (“all_true_labels” dan “all_predicted_labels”) diinisialisasi untuk menyimpan label asli dan prediksi.

7. Loop Melalui DataLoader Uji

   Pada baris 503 hingga 520, terdapat loop utama untuk iterasi berdasarkan DataLoader test:

   - Pada baris 504 hingga 506, data dan label dipindahkan ke perangkat (“device”).
   - Pada baris 508, model dievaluasi pada batch data test dan menghasilkan “loss” dan “logits”.
   - Pada baris 510 hingga 517, untuk setiap prediksi dalam batch:
     a. Pada baris 511 hingga 512, “logits” dan “label” yang relevan (bukan -100) diambil.
     b. Pada baris 514, prediksi dihasilkan dengan mengambil nilai tertinggi pada setiap vector dari “logits”.
     c. Pada baris 516 hingga 517, label dan prediksi diubah menjadi bentuk asli (bukan indeks) dan ditambahkan ke list “all_true_labels” dan “all_predicted_labels”.

8. Menghitung dan Menampilkan Laporan Evaluasi

   Pada baris 522 hingga 526, laporan evaluasi dihitung menggunakan “classification_report” dari scikit-learn:

   - Pada baris 523, hasil evaluasi hanya utntuk di cetak.
   - Pada baris 524, hasil evaluasi dihitung dan digunakan sebagai nilai yang di kembalikan sehingga terdapat parameter “output_dict=True”.
   - Pada baris 525 hingga 526, laporan evaluasi dicetak dengan metrik precision, recall, dan F1-score.

9. Mengembalikan Metrik Evaluasi

   Pada baris 528, fungsi mengembalikan dictionary metrik evaluasi yang berisi precision, recall, dan F1-score.

4.2.6.2. Fungsi Evaluate One Text

Kode 4. 28 Fungsi Evaluate One Text

```
529.    def align_word_ids(texts):
530.
531.        tokenized_inputs = tokenizer(texts, padding='max_length',
    max_length=512, truncation=True)
532.
533.        word_ids = tokenized_inputs.word_ids()
534.
535.        previous_word_idx = None
536.        label_ids = []
537.
538.        for word_idx in word_ids:
539.
540.            if word_idx is None:
541.                label_ids.append(-100)
542.
543.            elif word_idx != previous_word_idx:
544.                try:
545.                    label_ids.append(1)
546.                except:
547.                    label_ids.append(-100)
548.            else:
549.                try:
550.                    label_ids.append(1 if label_all_tokens else -100)
551.                except:
552.                    label_ids.append(-100)
553.            previous_word_idx = word_idx
554.
555.        return label_ids
556.
557.
558.    def evaluate_one_text(model, sentence):
559.
560.
561.        use_cuda = torch.cuda.is_available()
562.        device = torch.device("cuda" if use_cuda else "cpu")
563.
564.        if use_cuda:
565.            model = model.cuda()
566.
567.        text = tokenizer(sentence, padding='max_length', max_length = 512,
    truncation=True, return_tensors="pt")
568.
569.        mask = text['attention_mask'].to(device)
570.        input_id = text['input_ids'].to(device)
571.        label_ids =
    torch.Tensor(align_word_ids(sentence)).unsqueeze(0).to(device)
572.
573.        logits = model(input_id, mask, None)
574.        logits_clean = logits[0][label_ids != -100]
575.
576.        predictions = logits_clean.argmax(dim=1).tolist()
577.        prediction_label = [ids_to_labels[i] for i in predictions]
578.        print(sentence)
579.        print(prediction_label)
```

Penjelasan Kode fungsi align_word_ids:

1. Definisi Fungsi

   Fungsi “align_word_ids” menerima parameter “texts”, yang merupakan teks input yang akan di-tokenisasi.

2. Tokenisasi Teks

   Pada baris nomor 531, teks di-tokenisasi menggunakan “tokenizer” dengan padding, panjang maksimum 512.

3. Ambil Word IDs

Pada baris nomor 533, didapatkan “word_ids” dari tokenisasi.

4. Inisialisasi Variabel

   Pada baris nomor 535 hingga 536, variabel “previous_word_idx” diinisialisasi sebagai “None” dan “label_ids” diinisialisasi sebagai daftar kosong.

5. Loop Melalui Word IDs

   Pada baris nomor 538 hingga 553, dilakukan loop melalui setiap “word_idx” di dalam “word_ids”:

   - Jika “word_idx” adalah “None”, maka append -100 ke “label_ids”.
   - Jika “word_idx” tidak sama dengan “previous_word_idx”, maka append 1 ke “label_ids” (dengan penanganan kesalahan).
   - Jika “word_idx” sama dengan “previous_word_idx”, maka append 1 atau -100 ke “label_ids” tergantung pada nilai “label_all_tokens” (dengan penanganan kesalahan).
   - Set “previous_word_idx” menjadi “word_idx”.

6. Kembalikan “label_ids”

   Pada baris nomor 555, “label_ids” dikembalikan sebagai hasil fungsi.

Penjelasan Kode Evaluate One Text

1. Definisi Fungsi

   Fungsi “evaluate_one_text” menerima parameter “model” dan “sentence”, yang merupakan model yang akan dievaluasi dan kalimat yang akan dievaluasi.

2. Inisialisasi CUDA

   Pada baris nomor 561 hingga 562, dilakukan pengecekan ketersediaan CUDA (GPU) dan penetapan perangkat (“device”).

3. Pindahkan Model ke GPU

   Pada baris nomor 564 hingga 565, jika CUDA tersedia, model dipindahkan ke GPU.

4. Tokenisasi Teks

   Pada baris nomor 567, kalimat di-tokenisasi menggunakan “tokenizer” dengan padding, panjang maksimum 512, dan pemangkasan (“truncation”).

Hasil tokenisasi dikembalikan dalam bentuk tensor PyTorch (“return_tensors="pt"“).

5. Pindahkan Input ke GPU

   Pada baris nomor 569 hingga 570, “attention_mask” dan “input_ids” dari hasil tokenisasi dipindahkan ke perangkat (“device”).

6. Dapatkan “label_ids”

   Pada baris nomor 571, “label_ids” dihasilkan dengan memanggil fungsi “align_word_ids” dan hasilnya dikonversi ke tensor PyTorch, kemudian dipindahkan ke perangkat.

7. Dapatkan Logits dari Model

   Pada baris nomor 573, model dipanggil dengan “input_id” dan “mask” untuk mendapatkan logits.

8. Bersihkan Logits

   Pada baris nomor 574, logits dibersihkan dengan mengabaikan posisi yang memiliki label -100 pada “label_ids”.

9. Dapatkan Prediksi

   Pada baris nomor 576, prediksi didapatkan dengan mengambil argmax dari “logits_clean” dan dikonversi ke daftar.

10. Konversi Prediksi ke Label

    Pada baris nomor 577, prediksi dikonversi ke label menggunakan “ids_to_labels”.

11. Cetak Kalimat dan Prediksi

    Pada baris nomor 578 hingga 579, kalimat dan label prediksi dicetak.

4.3. Hasil Skenario Pengujian

Pada skenario pengujian, dilakukan dengan menggunakan beberapa parameter, di antaranya batch size, epoch, dan learning rate, dengan dua model pre-trained yaitu *Indolem/indobert-base-uncased* dan *Indobenchmark/indobert-base-p2*. Skenario ini bisa dilihat dalam Tabel 3.11. Pengujian ini juga dilakukan dengan menggunakan 5 fold. Berikut adalah hasil skenario pengujian pada setiap fold:

4.3.1. Hasil Pengujian Fold 1

Pada pengujian Fold 1, model *Indolem/indobert-base-uncased* menunjukkan performa yang lebih baik dibandingkan dengan model *Indobenchmark/indobert-base-p2*. Hal ini terlihat dari rata-rata presisi, recall, dan F1-Score yang lebih tinggi terdapat pada model *Indolem/indobert-base-uncased*, sebagaimana ditunjukkan dalam Tabel 4.3. Model *Indolem/indobert-base-uncased* mencapai nilai tertinggi untuk presisi, recall, dan F1-Score pada beberapa label tertentu, sementara model *Indobenchmark/indobert-base-p2* memiliki nilai yang lebih rendah pada metrik evaluasi yang sama.

Tabel 4.3 Hasil Pengujian Fold 1

NO	Label	*Indolem/indobert-base-uncased*				*Indobenchmark/indobert-base-p2*			
		Presisi	**Recall**	**F1-Score**	**Support**	**Presisi**	**Recall**	**F1-Score**	**Support**
1.	B_ADVO	0,743	**0,891**	**0,810**	156	**0,815**	0,744	0,778	**160**
2.	I_ADVO	0,729	**0,938**	**0,821**	563	**0,766**	0,840	0,801	**576**
3.	B_ARTV	**0,839**	**0,611**	**0,707**	**581**	0,791	0,538	0,640	576
4.	I_ARTV	**0,859**	**0,917**	**0,887**	**5626**	**0,859**	0,872	0,865	5556
5.	B_CRIA	**0,887**	0,693	0,778	192	0,848	**0,729**	**0,784**	**207**
6.	I_CRIA	0,803	0,597	0,685	**704**	**0,841**	**0,581**	**0,687**	699
7.	B_DEFN	**0,861**	**0,922**	**0,890**	3911	0,810	0,893	0,850	**4121**
8.	I_DEFN	**0,896**	**0,942**	**0,918**	7894	0,887	0,898	0,893	**8428**
9.	B_JUDG	**0,923**	**0,942**	**0,932**	291	0,859	0,878	0,868	**304**
10.	I_JUDG	**0,958**	**0,958**	**0,958**	1203	0,936	0,950	0,943	**1226**
11.	B_JUDP	**0,948**	**0,948**	**0,948**	153	0,933	0,944	0,939	**162**
12.	I_JUDP	**0,972**	**0,981**	**0,977**	645	0,941	0,970	0,955	**662**
13.	B_PENA	**0,728**	**0,526**	**0,610**	234	0,693	0,491	0,575	234
14.	I_PENA	**0,871**	0,801	**0,834**	**2459**	0,739	**0,865**	0,797	2423
15.	B_PROS	**0,739**	**0,185**	**0,296**	92	0,590	0,247	0,348	**93**
16.	I_PROS	**0,777**	**0,850**	**0,812**	507	0,760	0,783	0,771	**525**
17.	B_PUNI	0,779	**0,785**	**0,782**	270	**0,855**	0,696	0,767	270
18.	I_PUNI	**0,926**	**0,894**	**0,909**	**3382**	0,925	0,875	0,899	3321
19.	B_REGI	**0,644**	**0,376**	**0,475**	**101**	0,500	0,310	0,383	100
20.	I_REGI	0,697	**0,919**	**0,792**	457	**0,717**	0,818	0,764	**472**
21.	B_TIMV	**0,988**	**0,953**	**0,970**	86	0,975	0,919	0,946	86
22.	I_TIMV	**0,970**	**0,970**	**0,970**	199	0,955	0,960	0,957	199
23.	B_VERN	0,879	**0,960**	**0,918**	**448**	**0,914**	0,907	0,910	420
24.	I_VERN	**0,938**	**0,950**	**0,944**	**3466**	0,942	0,930	0,936	3452
25.	O	**0,984**	**0,980**	**0,982**	165083	0,980	0,978	0,979	**165330**
Rata-Rata/ Total		**0,854**	**0,820**	**0,824**	198703	0,833	0,785	0,802	**199602**

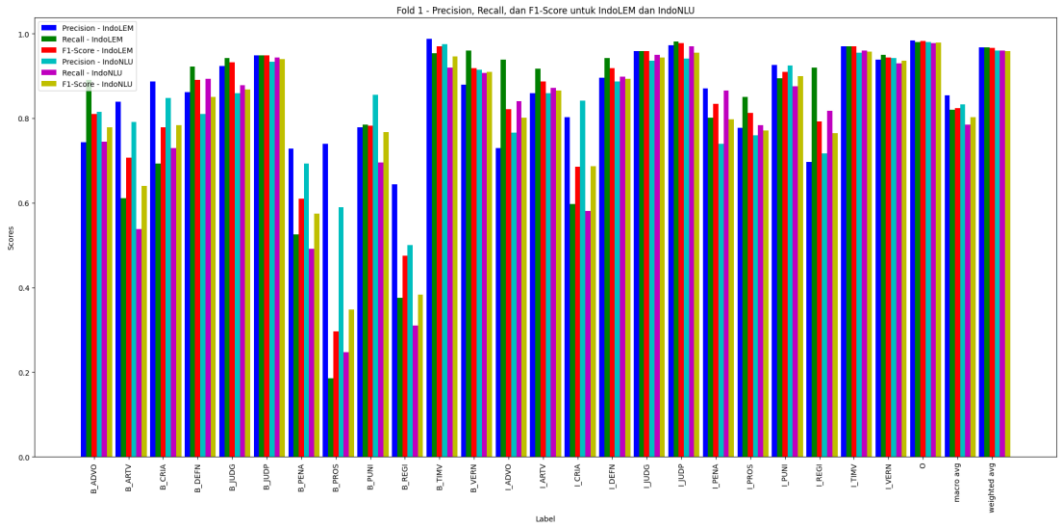
Pada Tabel 4.3 dan juga Gambar 4.8, terlihat bahwa beberapa label menunjukkan performa yang lebih unggul pada masing-masing model. Hal ini dapat dilihat dalam Tabel 4.4, pada model *Indolem/indobert-base-uncased*, label B_TIMV memiliki nilai presisi tertinggi sebesar 0,988, sedangkan pada model *Indobenchmark/indobert-base-p2*, label B_TIMV juga memiliki nilai presisi

tertinggi namun sedikit lebih rendah yaitu 0,975. Hal serupa terjadi pada nilai recall tertinggi dimana label I_JUDP memperoleh nilai 0,970 untuk kedua model. Begitu pula dengan nilai F1-Score tertinggi yang didominasi oleh label I_JUDP dengan nilai 0,977 untuk *Indolem/indobert-base-uncased* dan 0,955 untuk *Indobenchmark/indobert-base-p2*.

Tabel 4.4 Nilai label tertinggi dan terendah Fold 1

Metrik Evaluasi	***Indolem/indobert-base-uncased***		***Indobenchmark/indobert-base-p2***	
	Nilai	**Label**	**Nilai**	**Label**
	Nilai Tertinggi		**Nilai Tertinggi**	
Presisi	0,988	B_TIMV	0,975	B_TIMV
Recall	0,970	I_JUDP	0,970	I_JUDP
F1-Score	0,977	I_JUDP	0,955	I_JUDP
	Nilai Terendah		**Nilai Terendah**	
Presisi	0,644	B_REGI	0,500	B_REGI
Recall	0,185	B_PROS	0,247	B_PROS
F1-Score	0,296	B_PROS	0,348	B_PROS

Namun, terdapat pula beberapa label yang menunjukkan performa rendah pada kedua model. Seperti label B_REGI memiliki nilai presisi terendah dengan 0,644 pada model *Indolem/indobert-base-uncased* dan 0,500 pada model *Indobenchmark/indobert-base-p2*. Sementara itu, label B_PROS menunjukkan nilai recall terendah dengan 0,185 pada model *Indolem/indobert-base-uncased* dan 0,247 pada model *Indobenchmark/indobert-base-p2*. Untuk nilai F1-Score terendah, label B_PROS memperoleh nilai 0,296 pada model *Indolem/indobert-base-uncased* dan 0,348 pada model *Indobenchmark/indobert-base-p2*.



Gambar 4.8 Diagram Hasil Fold 1

4.3.2. Hasil Pengujian Fold 2

Pada pengujian Fold 2, model *Indolem/indobert-base-uncased* kembali menunjukkan performa yang lebih unggul dibandingkan dengan model *Indobenchmark/indobert-base-p2*. Hal dapat dilihat dari rata-rata nilai presisi, recall, dan F1-Score yang lebih tinggi terdapat pada model *Indolem/indobert-base-uncased*, sebagaimana terlihat dalam Tabel 4.5.

Tabel 4.5 Hasil Pengujian Fold 2

NO	Label	*Indolem/indobert-base-uncased*				*Indobenchmark/indobert-base-p2*			
		Presisi	**Recall**	**F1-Score**	**Support**	**Presisi**	**Recall**	**F1-Score**	**Support**
1.	B_ADVO	0,867	0,867	0,867	188	**0,880**	**0,884**	**0,882**	**199**
2.	I_ADVO	0,770	**0,954**	0,852	634	**0,909**	0,914	**0,912**	**654**
3.	B_ARTV	0,812	**0,709**	**0,757**	**592**	**0,850**	0,627	0,722	587
4.	I_ARTV	**0,929**	**0,911**	**0,920**	**5809**	0,907	0,906	0,907	5748
5.	B_CRIA	0,813	**0,804**	0,809	184	**0,843**	0,789	**0,815**	**190**
6.	I_CRIA	0,797	**0,677**	**0,732**	733	**0,866**	0,624	0,725	**744**
7.	B_DEFN	0,931	**0,959**	**0,945**	4076	**0,909**	0,932	0,921	**4257**
8.	I_DEFN	**0,926**	**0,970**	**0,948**	8389	0,922	0,954	0,937	**8945**
9.	B_JUDG	**0,952**	0,901	0,926	332	0,932	**0,932**	**0,932**	**340**
10.	I_JUDG	**0,979**	0,895	0,935	1612	0,957	**0,962**	**0,960**	**1634**
11.	B_JUDP	**0,929**	**0,963**	**0,945**	162	0,928	0,954	0,941	**175**
12.	I_JUDP	**0,973**	**0,973**	**0,973**	865	0,968	0,961	0,964	**875**
13.	B_PENA	0,792	**0,600**	**0,683**	285	**0,825**	0,557	0,665	**287**
14.	I_PENA	**0,896**	**0,922**	**0,909**	**2817**	0,883	0,858	0,870	2769
15.	B_PROS	**0,705**	**0,433**	**0,537**	127	0,571	0,153	0,241	**131**
16.	I_PROS	**0,772**	**0,922**	**0,840**	667	0,748	0,774	0,761	**691**
17.	B_PUNI	0,824	**0,794**	**0,808**	247	**0,856**	0,745	0,797	247
18.	I_PUNI	**0,947**	0,955	**0,951**	**2644**	0,918	**0,963**	0,940	2602
19.	B_REGI	**0,693**	**0,511**	**0,588**	137	0,433	0,183	0,257	**142**
20.	I_REGI	**0,819**	**0,854**	**0,836**	555	0,757	0,727	0,742	**571**
21.	B_TIMV	0,935	**0,871**	**0,902**	116	**0,934**	0,853	0,892	116
22.	I_TIMV	0,904	0,884	**0,894**	277	**0,906**	**0,874**	0,890	277
23.	B_VERN	**0,912**	0,927	**0,919**	**400**	0,895	**0,940**	0,917	382
24.	I_VERN	0,943	**0,967**	**0,955**	**3221**	**0,955**	0,942	0,949	3201
25.	O	**0,989**	**0,986**	**0,988**	166659	0,984	0,985	0,985	**166938**
Rata-Rata/ Total		**0,872**	**0,848**	**0,857**	201728	0,861	0,800	0,821	**202702**

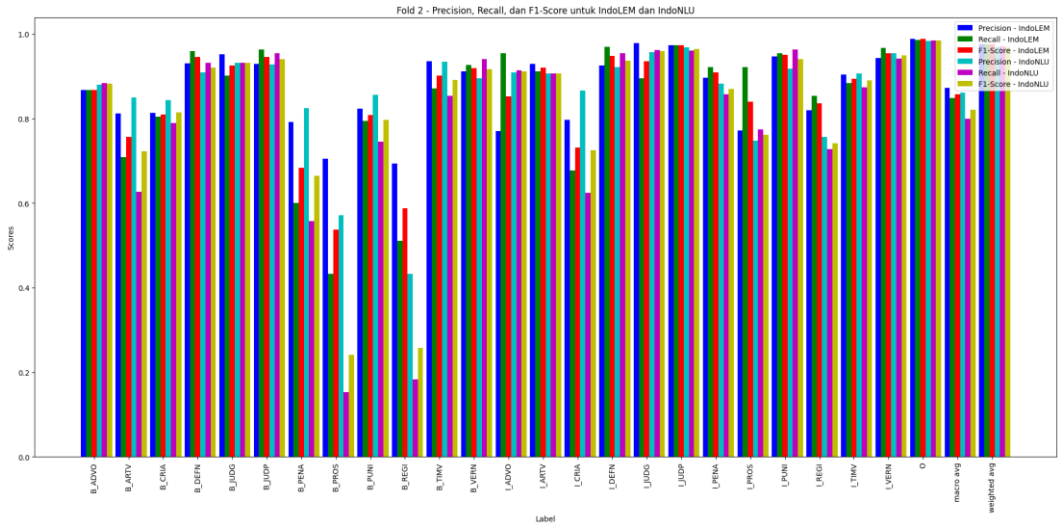
Dilihat dari Tabel 4.5 dan juga Gambar 4.9, terlihat bahwa beberapa label menunjukkan performa yang lebih unggul pada masing-masing model. Hal ini dapat dilihat dalam Tabel 4.6, pada model *Indolem/indobert-base-uncased*, label I_JUDG mencapai nilai presisi tertinggi dengan 0,979, sedangkan pada model *Indobenchmark/indobert-base-p2*, label I_JUDG juga memiliki nilai presisi tertinggi namun sedikit lebih rendah yaitu 0,973. Hal yang serupa terjadi pada nilai recall tertinggi dimana label I_JUDP memperoleh nilai 0,973 untuk *Indolem/indobert-base-uncased* dan 0,970 untuk *Indobenchmark/indobert-base-*

p2. Begitu pula dengan nilai F1-Score tertinggi yang didominasi oleh label I_JUDP dengan nilai 0,977 untuk *Indolem/indobert-base-uncased* dan 0,964 untuk *Indobenchmark/indobert-base-p2*.

Tabel 4.6 Nilai label tertinggi dan terendah Fold 2

Metrik Evaluasi	***Indolem/indobert-base-uncased***		***Indobenchmark/indobert-base-p2***	
	Nilai	**Label**	**Nilai**	**Label**
	Nilai Tertinggi		**Nilai Tertinggi**	
Presisi	0,979	I_JUDG	0,968	I_JUDP
Recall	0,973	I_JUDP	0,963	I_TIMV
F1-Score	0,973	I_JUDP	0,964	I_JUDP
	Nilai Terendah		**Nilai Terendah**	
Presisi	0,693	B_REGI	0,433	B_REGI
Recall	0,433	B_PROS	0,153	B_PROS
F1-Score	0,537	B_PROS	0,241	B_PROS

Namun demikian, terdapat beberapa label yang menunjukkan performa rendah pada kedua model. Seperti label B_REGI memiliki nilai presisi terendah dengan 0,693 pada model *Indolem/indobert-base-uncased* dan 0,433 pada model *Indobenchmark/indobert-base-p2*. Sementara itu, label B_PROS menunjukkan nilai recall terendah dengan 0,433 pada model *Indolem/indobert-base-uncased* dan 0,571 pada model *Indobenchmark/indobert-base-p2*. Untuk nilai F1-Score terendah, label B_PROS memperoleh nilai 0,537 pada model *Indolem/indobert-base-uncased* dan 0,665 pada model *Indobenchmark/indobert-base-p2*.



Gambar 4.9 Diagram Hasil Fold 2

4.3.3. Hasil Pengujian Fold 3

Pada pengujian Fold 3, model *Indolem/indobert-base-uncased* terus menunjukkan keunggulannya atas model *Indobenchmark/indobert-base-p2*. Pada Tabel 4.7 menunjukkan bahwa rata-rata presisi, recall, dan F1-Score dari *Indolem/indobert-base-uncased* lebih tinggi dibandingkan dengan *Indobenchmark/indobert-base-p2*. Pada pengujian ini, model *Indolem/indobert-base-uncased* sekali lagi mencapai nilai tertinggi pada beberapa metrik evaluasi dibandingkan dengan *Indobenchmark/indobert-base-p2*.

Tabel 4.7 Hasil Pengujian Fold 3

NO	Label	*Indolem/indobert-base-uncased*				*Indobenchmark/indobert-base-p2*			
		Presisi	**Recall**	**F1-Score**	**Support**	**Presisi**	**Recall**	**F1-Score**	**Support**
1.	B_ADVO	**0,973**	**0,929**	**0,950**	154	0,910	0,916	0,913	**155**
2.	I_ADVO	**0,910**	**0,964**	**0,936**	473	0,857	0,924	0,889	**485**
3.	B_ARTV	**0,916**	**0,760**	**0,831**	**558**	0,877	0,726	0,794	551
4.	I_ARTV	**0,932**	**0,956**	**0,944**	**5280**	0,907	0,938	0,922	5207
5.	B_CRIA	0,875	**0,866**	**0,870**	186	**0,882**	0,788	0,832	**189**
6.	I_CRIA	**0,911**	**0,753**	**0,824**	676	0,884	0,630	0,736	**678**
7.	B_DEFN	**0,981**	**0,963**	**0,972**	4330	0,865	0,957	0,909	**4517**
8.	I_DEFN	**0,962**	**0,980**	**0,971**	8489	0,903	0,959	0,930	**9064**
9.	B_JUDG	**0,978**	**0,978**	**0,978**	319	0,963	0,957	0,960	**327**
10.	I_JUDG	**0,979**	**0,991**	**0,985**	1372	0,929	**0,991**	0,959	**1393**
11.	B_JUDP	**0,976**	**0,976**	**0,976**	166	0,933	0,971	0,951	**171**
12.	I_JUDP	**0,989**	0,993	**0,991**	711	**0,989**	**0,984**	0,986	**731**
13.	B_PENA	**0,872**	**0,798**	**0,834**	248	0,859	0,690	0,765	248
14.	I_PENA	**0,938**	**0,960**	**0,949**	**2751**	0,897	0,886	0,892	2703
15.	B_PROS	0,725	**0,687**	**0,705**	**115**	**0,944**	0,298	0,453	114
16.	I_PROS	**0,879**	**0,884**	**0,882**	553	0,822	0,811	0,816	**587**
17.	B_PUNI	**0,884**	**0,877**	**0,880**	235	0,866	0,873	0,869	**236**
18.	I_PUNI	**0,976**	0,935	0,955	**2428**	0,937	**0,979**	**0,957**	2395
19.	B_REGI	**0,793**	**0,645**	**0,711**	107	0,726	0,421	0,533	107
20.	I_REGI	**0,929**	0,833	**0,878**	503	0,765	**0,885**	0,820	**529**
21.	B_TIMV	0,921	**0,894**	**0,907**	**104**	**0,935**	0,861	0,897	101
22.	I_TIMV	0,950	0,898	0,923	254	**0,951**	**0,925**	**0,938**	254
23.	B_VERN	0,926	**0,957**	**0,941**	**483**	**0,930**	0,896	0,913	461
24.	I_VERN	0,965	**0,980**	**0,973**	**3626**	**0,971**	0,936	0,953	3613
25.	O	**0,993**	**0,993**	**0,993**	166811	0,989	0,984	0,987	**167098**
Rata-Rata/ Total		**0,925**	**0,898**	**0,910**	200932	0,900	0,847	0,863	**201914**

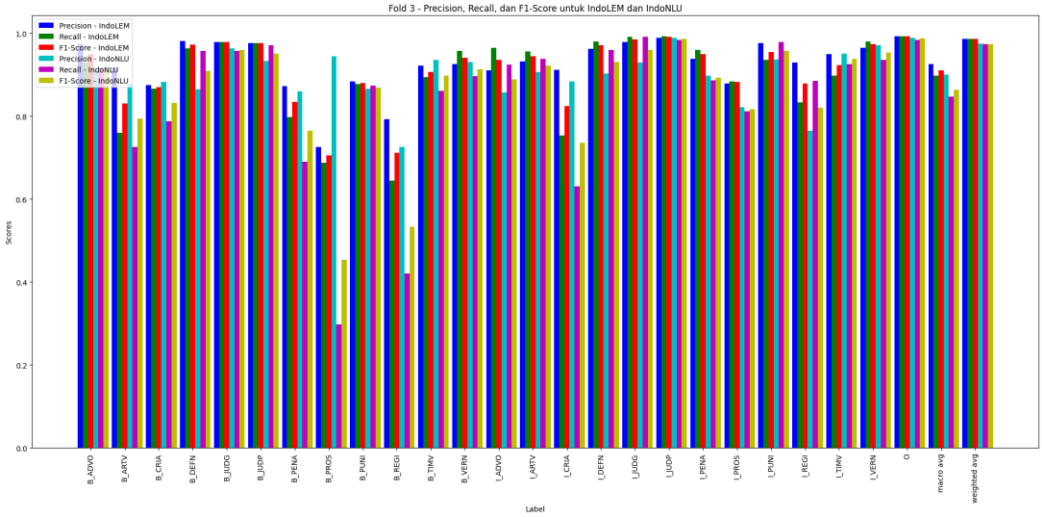
Pada Tabel 4.7 dan juga Gambar 4.10, terlihat bahwa beberapa label menunjukkan performa yang lebih unggul pada masing-masing model. Hal ini dapat dilihat dalam Tabel 4.8, model *Indolem/indobert-base-uncased* mencapai presisi tertinggi sebesar 0,989 untuk label I_JUDP, sedangkan *Indobenchmark/indobert-base-p2* mencapai presisi tertinggi 0,989 untuk label yang sama. Hal ini juga terjadi pada recall tertinggi dimana model *Indolem/indobert-*

base-uncased memperoleh nilai 0,993 untuk label I_JUDP, sementara *Indobenchmark/indobert-base-p2* mencapai 0,984. F1-Score tertinggi juga dicapai oleh label I_JUDP dengan nilai 0,991 untuk *Indolem/indobert-base-uncased* dan 0,986 untuk *Indobenchmark/indobert-base-p2*.

Tabel 4.8 Nilai label tertinggi dan terendah Fold 3

Metrik Evaluasi	***Indolem/indobert-base-uncased***		***Indobenchmark/indobert-base-p2***	
	Nilai	**Label**	**Nilai**	**Label**
	Nilai Tertinggi		**Nilai Tertinggi**	
Presisi	0,989	I_JUDP	0,989	I_JUDP
Recall	0,993	I_JUDP	0,991	I_JUDG
F1-Score	0,991	I_JUDP	0,986	I_JUDP
	Nilai Terendah		**Nilai Terendah**	
Presisi	0,725	B_PROS	0,726	B_REGI
Recall	0,645	B_REGI	0,298	B_PROS
F1-Score	0,705	B_PROS	0,453	B_PROS

Namun demikian, terdapat beberapa label yang menunjukkan performa rendah pada kedua model. Seperti label B_PROS dan B_REGI mengalami tantangan dalam klasifikasi. Pada model *Indolem/indobert-base-uncased*, label B_PROS memiliki nilai presisi, recall, dan F1-Score terendah berturut-turut sebesar 0,725, 0,298, dan 0,453. Sedangkan pada model *Indobenchmark/indobert-base-p2*, label B_REGI memiliki nilai presisi terendah dengan 0,726, recall terendah dengan 0,421, dan F1-Score terendah dengan 0,533.



Gambar 4.10 Diagram Hasil Fold 3

4.3.4. Hasil Pengujian Fold 4

Pada pengujian Fold 4, model *Indolem/indobert-base-uncased* juga menunjukkan performa yang lebih baik dibandingkan dengan model *Indobenchmark/indobert-base-p2*. Tabel 4.9 menunjukkan bahwa rata-rata presisi, recall, dan F1-Score dari *Indolem/indobert-base-uncased* lebih tinggi daripada *Indobenchmark/indobert-base-p2*. Seperti pada fold-fold sebelumnya, model *Indolem/indobert-base-uncased* mencapai nilai tertinggi pada beberapa metrik evaluasi dibandingkan dengan model *Indobenchmark/indobert-base-p2*.

Tabel 4.9 Hasil Pengujian Fold 4

NO	Label	*Indolem/indobert-base-uncased*				*Indobenchmark/indobert-base-p2*			
		Presisi	**Recall**	**F1-Score**	**Support**	**Presisi**	**Recall**	**F1-Score**	**Support**
1.	B_ADVO	0,895	**0,919**	**0,907**	186	**0,920**	0,892	0,906	**194**
2.	I_ADVO	**0,946**	0,950	**0,948**	576	0,783	**0,959**	0,862	**581**
3.	B_ARTV	0,863	**0,793**	**0,827**	**550**	**0,945**	0,665	0,781	543
4.	I_ARTV	**0,942**	0,966	**0,954**	**5156**	0,871	**0,976**	0,921	5081
5.	B_CRIA	0,933	**0,933**	**0,933**	180	**0,992**	0,642	0,780	**190**
6.	I_CRIA	0,966	**0,881**	**0,921**	**639**	**0,980**	0,620	0,760	637
7.	B_DEFN	**0,976**	**0,978**	**0,977**	4441	0,963	0,953	0,958	**4641**
8.	I_DEFN	**0,966**	**0,990**	**0,978**	8176	0,965	0,953	0,959	**8676**
9.	B_JUDG	**0,975**	0,965	**0,970**	370	0,951	**0,971**	0,961	**377**
10.	I_JUDG	0,977	**0,994**	**0,985**	1636	**0,980**	0,983	0,982	**1688**
11.	B_JUDP	**0,984**	0,973	0,978	184	0,974	**0,985**	**0,979**	**194**
12.	I_JUDP	0,980	0,994	0,987	873	**0,983**	0,991	0,987	**880**
13.	B_PENA	0,814	**0,845**	**0,829**	233	**0,821**	0,708	0,760	233
14.	I_PENA	**0,959**	0,939	**0,949**	**2267**	0,887	**0,957**	0,921	2233
15.	B_PROS	0,671	**0,750**	**0,708**	136	**0,813**	0,449	0,578	136
16.	I_PROS	0,880	**0,945**	**0,911**	712	**0,889**	0,829	0,858	**736**
17.	B_PUNI	**0,847**	**0,917**	**0,881**	242	0,763	0,665	0,711	242
18.	I_PUNI	**0,936**	**0,985**	**0,960**	2610	0,928	0,872	0,899	**2562**
19.	B_REGI	**0,695**	**0,724**	**0,709**	123	0,667	0,496	0,569	**125**
20.	I_REGI	**0,870**	**0,914**	**0,891**	593	0,811	0,844	0,827	**615**
21.	B_TIMV	0,932	**0,948**	**0,940**	115	**0,955**	0,922	0,938	115
22.	I_TIMV	**0,962**	**0,937**	**0,949**	300	0,961	0,901	0,930	**302**
23.	B_VERN	**0,937**	**0,939**	**0,938**	**429**	0,924	0,924	0,924	409
24.	I_VERN	**0,972**	**0,945**	**0,958**	**3319**	0,971	0,934	0,952	3312
25.	O	**0,996**	**0,993**	**0,994**	177389	0,989	0,990	0,990	**177576**
Rata-Rata/ Jumlah		**0,915**	**0,925**	**0,919**	211435	0,908	0,843	0,868	**212278**

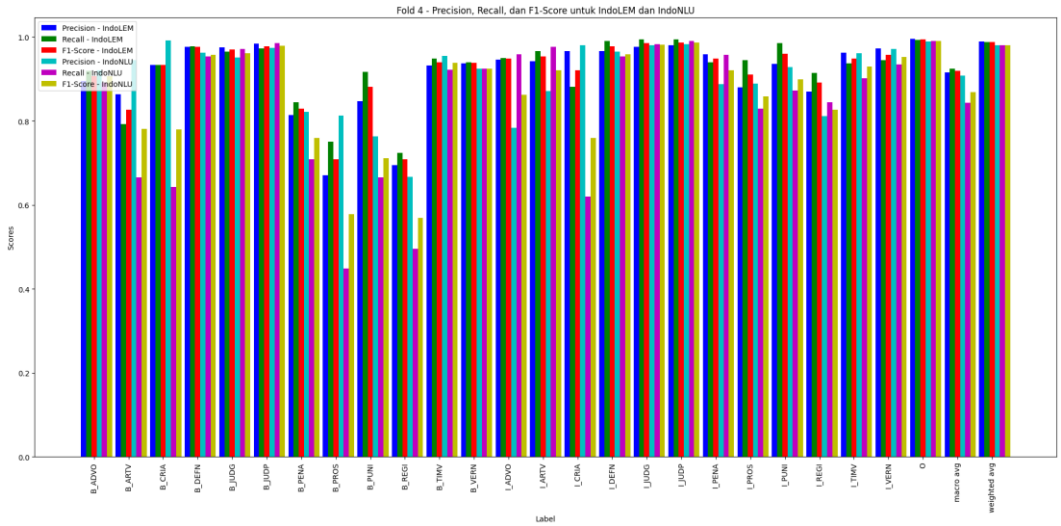
Pada Tabel 4.9 dan juga Gambar 4.11, terlihat bahwa beberapa label menunjukkan performa yang lebih unggul pada masing-masing model. Hal ini dapat dilihat dalam Tabel 4.10, model *Indolem/indobert-base-uncased* mencapai presisi tertinggi sebesar 0,984 untuk label B_JUDP, sedangkan

Indobenchmark/indobert-base-p2 mencapai presisi tertinggi 0,983 untuk label I_JUDP. Hal ini juga terjadi pada recall tertinggi dimana model *Indolem/indobert-base-uncased* memperoleh nilai 0,994 untuk label I_JUDG, sementara *Indobenchmark/indobert-base-p2* mencapai 0,991 untuk label I_JUDP. F1-Score tertinggi juga dicapai oleh label I_JUDP dengan nilai 0,987 untuk kedua model.

Tabel 4.10 Nilai label tertinggi dan terendah Fold 4

Metrik Evaluasi	***Indolem/indobert-base-uncased***		***Indobenchmark/indobert-base-p2***	
	Nilai	**Label**	**Nilai**	**Label**
	Nilai Tertinggi		**Nilai Tertinggi**	
Presisi	0,984	B_JUDP	0,983	I_JUDP
Recall	0,994	I_JUDG	0,991	I_JUDP
F1-Score	0,987	I_JUDP	0,987	I_JUDP
	Nilai Terendah		**Nilai Terendah**	
Presisi	0,671	B_PROS	0,667	B_REGI
Recall	0,724	B_REGI	0,449	B_PROS
F1-Score	0,708	B_PROS	0,569	B_REGI

Namun demikian, terdapat beberapa label yang menunjukkan performa rendah pada kedua model. Seperti label B_PROS dan B_REGI mengalami tantangan dalam klasifikasi. Pada model *Indolem/indobert-base-uncased*, label B_PROS memiliki nilai presisi, recall, dan F1-Score terendah berturut-turut sebesar 0,671, 0,750, dan 0,578. Sedangkan pada model *Indobenchmark/indobert-base-p2*, label B_REGI memiliki nilai presisi terendah dengan 0,667, recall terendah dengan 0,496, dan F1-Score terendah dengan 0,569.



Gambar 4.11 Diagram Hasil Fold 4

4.3.5. Hasil Pengujian Fold 5

Pada pengujian Fold 5, model *Indolem/indobert-base-uncased* kembali menunjukkan performa yang lebih baik dibandingkan dengan model *Indobenchmark/indobert-base-p2*. Tabel 4.11 menunjukkan bahwa rata-rata presisi, recall, dan F1-Score dari *Indolem/indobert-base-uncased* lebih tinggi dibandingkan dengan *Indobenchmark/indobert-base-p2*. Pada pengujian ini, model *Indolem/indobert-base-uncased* sekali lagi mencapai nilai tertinggi pada beberapa metrik evaluasi dibandingkan dengan *Indobenchmark/indobert-base-p2*.

Tabel 4.11 Hasil Pengujian Fold 5

NO	Label	*Indolem/indobert-base-uncased*				*Indobenchmark/indobert-base-p2*			
		Presisi	**Recall**	**F1-Score**	**Support**	**Presisi**	**Recall**	**F1-Score**	**Support**
1.	B_ADVO	**0,941**	**0,968**	**0,954**	247	0,864	0,940	0,900	**250**
2.	I_ADVO	**0,974**	**0,975**	**0,974**	875	0,773	0,987	0,867	**893**
3.	B_ARTV	**0,917**	**0,859**	**0,887**	**567**	0,886	0,717	0,793	565
4.	I_ARTV	**0,976**	**0,978**	**0,977**	**5468**	0,940	0,905	0,923	5394
5.	B_CRIA	**0,940**	**0,954**	**0,947**	196	0,906	0,774	0,835	**199**
6.	I_CRIA	0,838	**0,979**	**0,903**	610	**0,873**	0,657	0,750	**615**
7.	B_DEFN	**0,982**	**0,990**	**0,986**	4339	0,953	0,968	0,960	**4586**
8.	I_DEFN	**0,989**	**0,994**	**0,991**	8200	0,955	0,972	0,963	**8813**
9.	B_JUDG	**0,988**	**0,980**	**0,984**	347	0,971	0,938	0,954	**354**
10.	I_JUDG	0,990	**0,995**	**0,993**	1552	**0,992**	0,897	0,942	**1580**
11.	B_JUDP	**0,982**	**0,976**	**0,979**	167	0,966	0,977	0,972	**176**
12.	I_JUDP	0,977	**0,992**	**0,985**	774	**0,981**	0,974	0,977	**795**
13.	B_PENA	**0,861**	**0,840**	**0,850**	257	0,825	0,733	0,776	**258**
14.	I_PENA	**0,973**	**0,972**	**0,972**	**2482**	0,947	0,913	0,930	2446
15.	B_PROS	**0,754**	**0,772**	**0,763**	127	0,678	0,469	0,555	**130**
16.	I_PROS	**0,937**	**0,911**	**0,924**	617	0,837	0,859	0,848	**633**
17.	B_PUNI	0,892	**0,882**	**0,887**	271	**0,908**	0,804	0,853	271
18.	I_PUNI	**0,963**	**0,994**	**0,978**	**2957**	0,946	0,967	0,956	2909
19.	B_REGI	0,805	**0,829**	**0,817**	129	**0,826**	0,438	0,573	**130**
20.	I_REGI	**0,904**	**0,956**	**0,929**	569	0,838	0,883	0,860	**582**
21.	B_TIMV	0,929	**0,897**	**0,912**	116	**0,933**	0,836	0,882	116
22.	I_TIMV	0,957	**0,908**	**0,932**	**293**	**0,961**	0,839	0,896	292
23.	B_VERN	0,936	**0,953**	**0,945**	**385**	**0,946**	0,926	0,936	363
24.	I_VERN	**0,960**	**0,978**	**0,969**	**2877**	0,959	0,960	0,959	2863
25.	O	**0,997**	**0,995**	**0,996**	172268	0,989	0,991	0,990	**172731**
Rata-Rata/ Jumlah		**0,934**	**0,941**	**0,937**	206690	0,906	0,853	0,874	**207944**

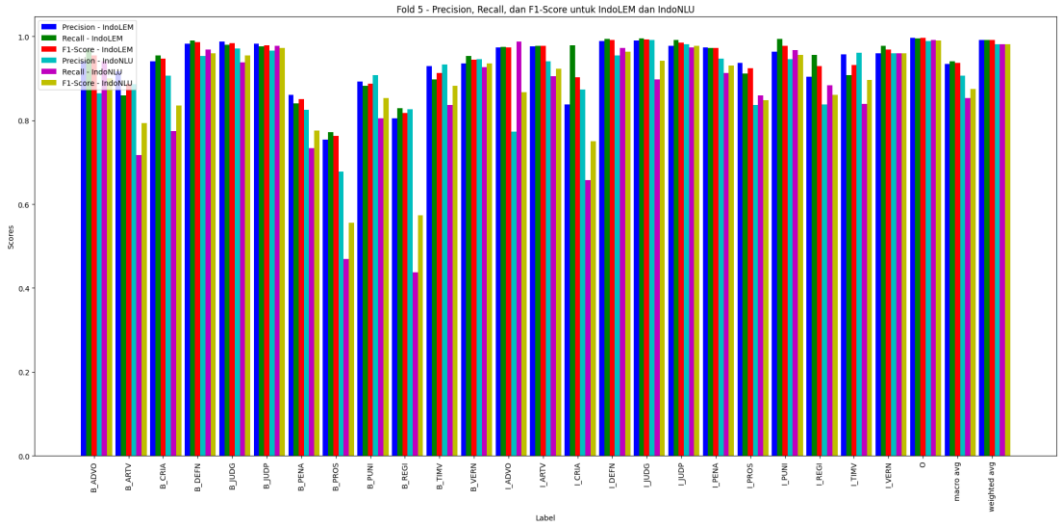
Pada Tabel 4.11 dan juga Gambar 4.12, terlihat bahwa beberapa label menunjukkan performa yang lebih unggul pada masing-masing model. Hal ini dapat dilihat dalam Tabel 4.12, model *Indolem/indobert-base-uncased* mencapai presisi tertinggi sebesar 0,990 untuk label I_JUDG, sedangkan *Indobenchmark/indobert-base-p2* mencapai presisi tertinggi 0,992 untuk label yang sama. Hal ini juga terjadi pada recall tertinggi dimana model *Indolem/indobert-*

base-uncased memperoleh nilai 0,995 untuk label I_JUDG, sementara *Indobenchmark/indobert-base-p2* mencapai 0,994 untuk label I_PUNI. F1-Score tertinggi juga dicapai oleh label I_JUDG dengan nilai 0,993 untuk *Indolem/indobert-base-uncased* dan 0,991 untuk label I_DEFN pada *Indobenchmark/indobert-base-p2*.

Tabel 4.12 Nilai label tertinggi dan terendah Fold 5

Metrik Evaluasi	***Indolem/indobert-base-uncased***		***Indobenchmark/indobert-base-p2***	
	Nilai	**Label**	**Nilai**	**Label**
	Nilai Tertinggi		**Nilai Tertinggi**	
Presisi	0,990	I_JUDG	0,992	I_JUDG
Recall	0,995	I_JUDG	0,987	I_ADVO
F1-Score	0,993	I_JUDG	0,985	I_JUDP
	Nilai Terendah		**Nilai Terendah**	
Presisi	0,754	B_PROS	0,678	B_PROS
Recall	0,772	B_PROS	0,438	B_REGI
F1-Score	0,763	B_PROS	0,555	B_REGI

Namun demikian, terdapat beberapa label yang menunjukkan performa rendah pada kedua model. Seperti label B_PROS dan B_REGI yang mengalami tantangan dalam klasifikasi. Pada model *Indolem/indobert-base-uncased*, label B_PROS memiliki nilai presisi, recall, dan F1-Score terendah berturut-turut sebesar 0,754, 0,772, dan 0,555. Sedangkan pada model *Indobenchmark/indobert-base-p2*, label B_REGI memiliki nilai presisi terendah dengan 0,678, recall terendah dengan 0,438, dan F1-Score terendah dengan 0,573.



Gambar 4.12 Diagram Hasil Fold 5

4.3.6. Analisis Hasil Pengujian

Berdasarkan hasil pengujian pada 5 fold yang dilakukan, model *Indolem/indobert-base-uncased* secara konsisten menunjukkan performa yang lebih baik dibandingkan dengan model *Indobenchmark/indobert-base-p2*. Pada Tabel 4.13, Rata-rata presisi, recall, dan F1-score dari model *Indolem/indobert-base-uncased* lebih tinggi pada setiap fold, dengan presisi tertinggi dicapai pada fold ke-5 0.934, recall tertinggi pada fold ke-5 0.941, dan F1-score tertinggi pada fold ke-5 0.937. Sebaliknya, model *Indobenchmark/indobert-base-p2* menunjukkan presisi tertinggi pada fold ke-4 0.908, recall tertinggi pada fold ke-5 0,853, dan F1-score tertinggi pada fold ke-5 0,874. Hal ini menandakan bahwa model *Indolem/indobert-base-uncased* konsisten dalam menghasilkan prediksi yang lebih akurat dan konsisten di setiap fold pengujian.

Tabel 4.13 Rangkuman Hasil Rata-Rata Pengujian Semua Fold

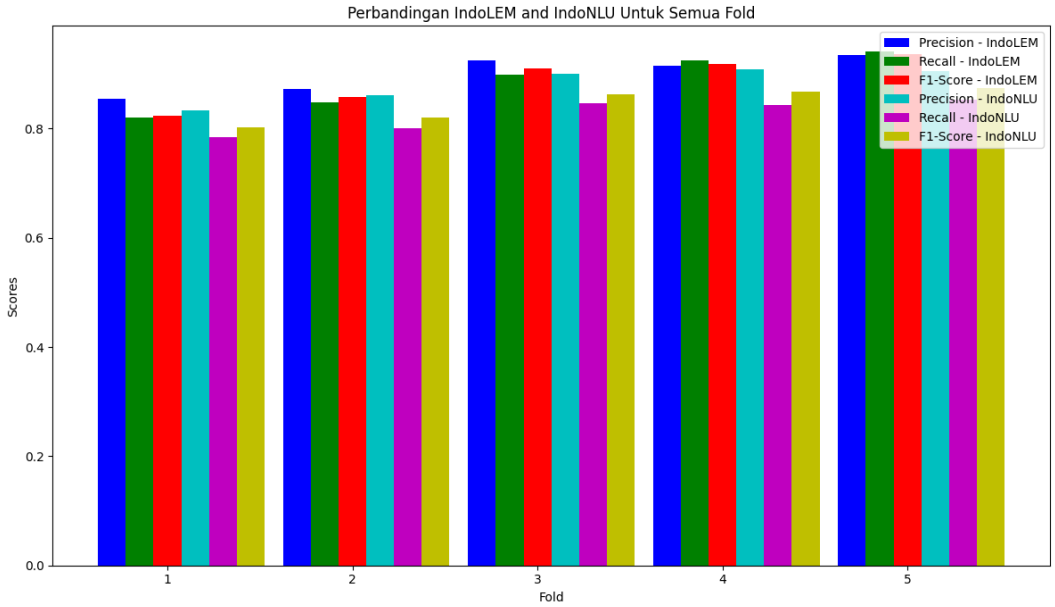
Fold	*Indolem/indobert-base-uncased*			*Indobenchmark/indobert-base-p2*		
	Presisi	**Recall**	**F1-Score**	**Presisi**	**Recall**	**F1-Score**
1	0,854	0,820	0,824	0,833	0,785	0,802
2	0,872	0,848	0,857	0,861	0,800	0,821
3	0,925	0,898	0,910	0,900	0,847	0,863
4	0,915	0,925	0,919	0,908	0,843	0,868
5	**0,934**	**0,941**	**0,937**	0,906	0,853	0,874
Rata- Rata	**0,900**	**0,886**	**0,890**	0,882	0,826	0,845

Dari aspek presisi, recall, dan F1-Score, model *Indolem/indobert-base-uncased* secara konsisten mencatatkan nilai yang lebih tinggi dalam sebagian besar label dibandingkan dengan *Indobenchmark/indobert-base-p2*. Misalnya, label I_JUDG dan I_JUDP merupakan label-label yang konsisten mencatatkan performa tertinggi pada kedua model, dengan nilai presisi, recall, dan F1-Score yang signifikan, dapat dilihat Tabel 4.3 hingga Tabel 4.12. Hal ini menunjukkan bahwa *Indolem/indobert-base-uncased* mampu mengatasi variasi dan kompleksitas tugas NER dalam bidang hukum bahasa Indonesia dengan lebih baik, hal ini disebabkan oleh jumlah token vocab yang lebih besar yaitu 31.923 dibandingkan dengan *Indobenchmark/indobert-base-p2* yaitu 30.522, seperti yang dijelaskan pada informasi konfigurasi model pada Tabel 2.3 dan Tabel 2.4. Dengan jumlah vocab yang lebih besar, model akan lebih sedikit memecah token menjadi subtoken. Semakin banyak token yang di pecah menjadi sub token, semakin berkurang juga

arti dari token tersebut. Hal ini dapat dilihat pada kolom support Tabel 4.3 hingga Tabel 4.11, dimana jumlah support pada model *Indobenchmark/indobert-base-p2* lebih banyak, yang artinya model *Indobenchmark/indobert-base-p2* memecahkan lebih banyak token dibandingkan dengan model *Indolem/indobert-base-uncased*. Selain itu, dataset yang digunakan oleh model *Indolem/indobert-base-uncased* mencakup lebih banyak variasi Bahasa Indonesia dan memiliki kualitas teks bahasa yang lebih baik dibandingkan dengan dataset yang digunakan oleh model *Indobenchmark/indobert-base-p2,* walaupun jumlah dataset yang digunakan pada model *Indobenchmark/indobert-base-p2* lebih besar. Seperti pada penelitian [12] yang melakukan perbandingan dataset berukuran 23 GB dengan 180 GB.

Namun demikian, terdapat beberapa label seperti B_PROS dan B_REGI yang menunjukkan performa rendah pada kedua model. Label-label ini mencatatkan nilai presisi, recall, dan F1-Score yang terendah, menunjukkan bahwa model-model ini menghadapi kesulitan dalam mengklasifikasikan pola-pola yang tepat untuk label-label tersebut. Hal ini disebabkan karena proses preprocessing pada tokenisasi masih kurang sesuai dengan hasil tokenisasi dari BERT sehingga mengakibatkan perbedaan label yang dihasilkan antara dataset dan proses penyelarasan label menggunakan tokenisasi BERT. Peningkatan performa pada label-label ini dapat menjadi fokus untuk pengembangan dan peningkatan model-model ini di masa mendatang, mungkin dengan pengoptimalan lebih lanjut terhadap data training yang spesifik atau penyesuaian parameter model.

Secara keseluruhan, hasil pengujian menunjukkan bahwa *Indolem/indobert-base-uncased* memiliki keunggulan dalam mengatasi tugas NER dalam bidang hukum Bahasa Indonesia dibandingkan dengan *Indobenchmark/indobert-base-p2*. Hal ini bisa dilihat pada Gambar 4.13.



Gambar 4.13 Hasil Diagram Semua Fold

BAB V PENUTUP

5.1. Kesimpulan

Berdasarkan hasil pengujian pada lima fold, model *Indolem/indobert-base-uncased* secara konsisten menunjukkan performa yang lebih baik dibandingkan dengan model *Indobenchmark/indobert-base-p2* dalam tugas *Named Entity Recognition* (NER) pada bidang hukum bahasa Indonesia. Model *Indolem/indobert-base-uncased* mencatatkan rata-rata presisi, recall, dan F1-score yang lebih tinggi, yaitu 90%, 88%, dan 89%, dibandingkan dengan *Indobenchmark/indobert-base-p2* yang memiliki nilai 88%, 82%, dan 84%. Performa *Indolem/indobert-base-uncased* yang unggul ini disebabkan oleh jumlah token vocab yang lebih besar dan juga dataset yang digunakan oleh model *Indolem/indobert-base-uncased* lebih mencakup variasi Bahasa Indonesia dan kualitas teks yang lebih baik.

5.2. Saran

Berdasarkan analisis hasil pengujian yang telah dilakukan, berikut adalah beberapa saran untuk perbaikan dan pengembangan lebih lanjut dalam pengembangan tugas NER dalam bidang hukum Bahasa Indonesia:

1. Pengembangan Metode

   Mengembangkan metode lain dengan vocab yang lebih besar dan lebih representatif terhadap variasi bahasa Indonesia, seperti *Indolem/indobert-base-uncased* namun versi yang lebih besar (*Large)*. Atau bisa menggunakan metode sebelum BERT yaitu *OpenAI GPT* dan *ELMo* yang juga metode berbasis transformers.

2. Peningkatan Kualitas Data

   Perlu dilakukan peningkatan dan perbaikan kualitas dataset, khususnya untuk label-label yang menunjukkan performa rendah seperti B_PROS dan B_REGI. Dataset yang lebih besar dan lebih bervariasi dapat membantu model mengenali pola dengan lebih baik.

3. Pengoptimalan Hyperparameter

   Penelitian lebih lanjut dapat difokuskan pada pengoptimalan hyperparameter untuk kedua model. Hal ini bisa dilakukan melalui teknik hyperparameter tuning untuk mencari konfigurasi terbaik yang dapat

meningkatkan performa model, terutama untuk label-label dengan performa rendah.

Dengan menerapkan saran-saran di atas, diharapkan performa model dalam tugas NER pada bidang hukum Bahasa Indonesia dapat ditingkatkan secara signifikan.

REFERENSI

[1] F. Solihin, I. Budi, R. F. Aji, and E. Makarim, "Advancement of information extraction use in legal documents," *Int. Rev. Law, Comput. Technol.*, vol. 35, no. 3, pp. 322–351, 2021, doi: 10.1080/13600869.2021.1964225.

[2] F. Solihin and I. Budi, "Recording of law enforcement based on court decision document using rule-based information extraction," *2018 Int. Conf. Adv. Comput. Sci. Inf. Syst. ICACSIS 2018*, pp. 349–354, 2019, doi: 10.1109/ICACSIS.2018.8618187.

[3] "Direktori Putusan." https://putusan3.mahkamahagung.go.id/ (accessed Mar. 30, 2024).

[4] E. Qadri Nuranti and E. Yulianti, "Legal Entity Recognition in Indonesian Court Decision Documents Using Bi-LSTM and CRF Approaches," *2020 Int. Conf. Adv. Comput. Sci. Inf. Syst.*, 2020, doi: 10.1109/ICACSIS51025.2020.9263157.

[5] Z. Wang, Y. Wu, P. Lei, and C. Peng, "Named Entity Recognition Method of Brazilian Legal Text based on pre-training model," *J. Phys. Conf. Ser.*, vol. 1550, no. 3, 2020, doi: 10.1088/1742-6596/1550/3/032149.

[6] T. Gelar, A. Nanda, and A. Bakhrun, "Serverless Named Entity Recognition untuk Teks Instruksional Pertanian Kota," *J. Tek. Inform. dan Sist. Inf.*, vol. 8, no. 3, pp. 597–606, 2022, doi: 10.28932/jutisi.v8i3.5447.

[7] H. W. Yang and A. Agrawal, *Extracting Complex Named Entities in Legal Documents via Weakly Supervised Object Detection*, vol. 1, no. 1. Association for Computing Machinery, 2023. doi: 10.1145/3539618.3591852.

[8] C. Berragan, A. Singleton, A. Calafiore, and J. Morley, "Transformer based named entity recognition for place name extraction from unstructured text," *Int. J. Geogr. Inf. Sci.*, vol. 37, no. 4, pp. 747–766, 2023, doi: 10.1080/13658816.2022.2133125.

[9] C. Sun, Z. Yang, L. Wang, Y. Zhang, H. Lin, and J. Wang, "Biomedical named entity recognition using BERT in the machine reading comprehension framework," *J. Biomed. Inform.*, vol. 118, no. April, p. 103799, 2021, doi: 10.1016/j.jbi.2021.103799.

[10] D. Sebastian, H. D. Purnomo, and I. Sembiring, “BERT for Natural Language Processing in Bahasa Indonesia,” *2022 2nd Int. Conf. Intell. Cybern. Technol. Appl. ICICyTA 2022*, pp. 204–209, 2022, doi: 10.1109/ICICyTA57421.2022.10038230.

[11] F. Koto, A. Rahimi, J. H. Lau, and T. Baldwin, “IndoLEM and IndoBERT: A Benchmark Dataset and Pre-trained Language Model for Indonesian NLP,” *COLING 2020 - 28th Int. Conf. Comput. Linguist. Proc. Conf.*, pp. 757–770, 2020, doi: 10.18653/v1/2020.coling-main.66.

[12] B. Wilie *et al.*, “IndoNLU: Benchmark and Resources for Evaluating Indonesian Natural Language Understanding,” pp. 843–857, 2020, [Online]. Available: http://arxiv.org/abs/2009.05387

[13] P. Bose, S. Srinivasan, W. C. Sleeman, J. Palta, R. Kapoor, and P. Ghosh, “A survey on recent named entity recognition and relationship extraction techniques on clinical texts,” *Appl. Sci.*, vol. 11, no. 18, 2021, doi: 10.3390/app11188319.

[14] V. Naik, P. Patel, and R. Kannan, “Legal Entity Extraction: An Experimental Study of NER Approach for Legal Documents,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 14, no. 3, pp. 775–783, 2023, doi: 10.14569/IJACSA.2023.0140389.

[15] “Understanding Your Textual Data Using Doccano | by Ori Cohen | Towards Data Science.” https://archive.ph/wZ71A#selection-859.8-859.46 (accessed Jul. 18, 2024).

[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, and L. Jones, “Attention Is All You Need,” no. Nips, 2017.

[17] M. C. Kenton, L. Kristina, and J. Devlin, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” no. Mlm, 1953.

[18] R. van der Goot, M. Müller-Eberstein, and B. Plank, “Frustratingly Easy Performance Improvements for Low-resource Setups: A Tale on BERT and Segment Embeddings,” *2022 Lang. Resour. Eval. Conf. Lr. 2022*, no. June, pp. 1418–1427, 2022.

[19] X. Song, A. Salcianu, Y. Song, D. Dopson, and D. Zhou, “Fast WordPiece Tokenization,” *EMNLP 2021 - 2021 Conf. Empir. Methods Nat. Lang.*

Process. Proc., pp. 2089–2103, 2021, doi: 10.18653/v1/2021.emnlp-main.160.

[20] A. Nayak, H. Timmapathini, K. Ponnalagu, and V. Gopalan Venkoparao, "Domain adaptation challenges of BERT in tokenization and sub-word representations of Out-of-Vocabulary words," pp. 1–5, 2020, doi: 10.18653/v1/2020.insights-1.1.

[21] Ernianti Hasibuan and Elmo Allistair Heriyanto, "Analisis Sentimen Pada Ulasan Aplikasi Amazon Shopping Di Google Play Store Menggunakan Naive Bayes Classifier," *J. Tek. dan Sci.*, vol. 1, no. 3, pp. 13–24, 2022, doi: 10.56127/jts.v1i3.434.

[22] E. T. Luthfi, Z. I. M. Yusoh, and B. M. Aboobaider, "BERT based Named Entity Recognition for Automated Hadith Narrator Identification," *Int. J. Adv. Comput. Sci. Appl.*, vol. 13, no. 1, pp. 604–611, 2022, doi: 10.14569/IJACSA.2022.0130173.

[23] S. O. Khairunnisa, Z. Chen, and M. Komachi, "Dataset Enhancement and Multilingual Transfer for Named Entity Recognition in the Indonesian Language," vol. 22, no. 6, 2023, doi: 10.1145/3592854.

[24] D. H. Fudholi, A. Zahra, S. Rani, S. N. Huda, I. V. Paputungan, and Z. Zukhri, "BERT-based tourism named entity recognition: making use of social media for travel recommendations," *PeerJ Comput. Sci.*, vol. 9, 2023, doi: 10.7717/PEERJ-CS.1731.

LAMPIRAN

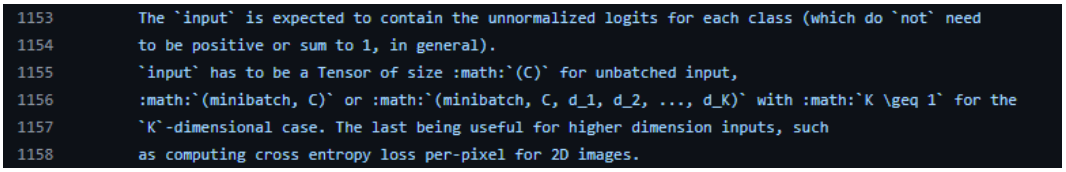
1. Layer Klasifikasi Proses Training

Dalam layer klasifikasi pada Gambar 3.8 terdapat 2 sub layer yaitu *Dropout* dan *Linear Layer*, dimana seharusnya dalam layer klasifikasi terdapat sebuah fungsi aktivasi untuk perhitungan Loss. Namun, dalam arsitektur model *BertForTokenClassification* tidak di jelaskan secara eksplit penggunaan fungsi aktivasi dalam layer klasifikasi. Hal ini di karenakan dalam *BertForTokenClassification* perhitungan Loss menggunakan *CrossEntropy*. Bukti penggunakan *CrossEntropy* ini dapat dilihat pada laman https://github.com/huggingface/transformers/blob/main/src/transformers/models/bert/modeling_bert.py, atau dapat dilihat pada Lampiran Gambar 1.1 pada baris 1908 dan 1909.

```
1901            sequence_output = outputs[0]
1902
1903            sequence_output = self.dropout(sequence_output)
1904            logits = self.classifier(sequence_output)
1905
1906            loss = None
1907            if labels is not None:
1908                loss_fct = CrossEntropyLoss()
1909                loss = loss_fct(logits.view(-1, self.num_labels), labels.view(-1))
1910
1911            if not return_dict:
1912                output = (logits,) + outputs[2:]
1913                return ((loss,) + output) if loss is not None else output
1914
1915            return TokenClassifierOutput(
1916                loss=loss,
1917                logits=logits,
1918                hidden_states=outputs.hidden_states,
1919                attentions=outputs.attentions,
1920            )
1921
```

Lampiran Gambar 1.1 Bukti Pengunaan *CrossEntropy*

Dalam PyTorch CrossEntropy membutuhkan sebuah inputan Logits yang belum di normalisasi, dan output dari layer terkahir (linear layer) merupakan logits yang belum dinormalisasi, hal ini dapat dilihat pada Lampiran Gambar 1.1 baris 1904. Bukti bahwa CrossEntropy membutuhkan sebuah logits yang belum dinormalisasi dapat dilihat pada laman https://github.com/pytorch/pytorch/blob/main/torch/nn/modules/loss.py, atau dapat dilihat pada Lampiran Gambar 1.2.



```
1153        The `input` is expected to contain the unnormalized logits for each class (which do `not` need
1154        to be positive or sum to 1, in general).
1155        `input` has to be a Tensor of size :math:`(C)` for unbatched input,
1156        :math:`(minibatch, C)` or :math:`(minibatch, C, d_1, d_2, ..., d_K)` with :math:`K \geq 1` for the
1157        `K`-dimensional case. The last being useful for higher dimension inputs, such
1158        as computing cross entropy loss per-pixel for 2D images.
```

Lampiran Gambar 1.2 Bukti *CrossEntropy* menggunakan Input Logits

2. Klasifikasi Evaluasi

   Pada tahap evaluasi model, terdapat 2 pendekatan yang dapat digunakan untuk mendapatkan label prediksi diantaranya:

   1. Menggunakan argmax (nilai tertinggi dari representasi vector)

      Contoh pendekatan argmax dapat dilihat pada Lampiran Kode 1.1.

      Lampiran Kode 1.1 Pendekatan argmax

```
# Looping berdasarkan jumlah batch data
for i in range(logits.shape[0]):
  # mengambil semua nilai kecuali index label nya -100
  logits_clean = logits[i][label[i] != -100]
  print(f"Shape Logits Clean : {logits_clean.shape}")

  # mengambil semua label kecuali -100
  label_clean = label[i][label[i] != -100]
  print(f"Shape Label Clean : {label_clean.shape}")
  print(f"Label Clean : {label_clean}")

  # mencari nilai maximum pada dimensi index 1
  predictions = logits_clean.argmax(dim=1)
  print(f"Shape Predictions : {predictions.shape}")
  print(f"Label Predictions : {predictions}")

  true_label_convert = [ids_to_labels[label] for label in label_clean.cpu().numpy() if label != -100]
  print(f"Label Actual : {true_label_convert}")
  pred_label_convert = [ids_to_labels[label] for label in predictions.cpu().numpy() if label != -100]
  print(f"Label Predict : {pred_label_convert}")

  print(f"Panjang Label Apakah sama? {len(true_label_convert)==len(pred_label_convert)}")
```

   2. Menggunakan Probabilitas dalam hal ini menggunakan Softmax

      Contoh pendekatan probabilitas Softmax dapat dilihat pada Lampiran Kode 1.2.

      Lampiran Kode 1.2 Pendekatan Probabilitas Softmax

```
import torch.nn.functional as F

# Looping berdasarkan jumlah batch data
for i in range(logits.shape[0]):
    # Mengambil semua nilai kecuali index label nya -100
    logits_clean = logits[i][label[i] != -100]
    print(f"Shape Logits Clean : {logits_clean.shape}")

    # Mengambil semua label kecuali -100
    label_clean = label[i][label[i] != -100]
    print(f"Shape Label Clean : {label_clean.shape}")
    print(f"Label Clean : {label_clean}")

    # Menerapkan softmax pada logits untuk mendapatkan probabilitas
    probabilities = F.softmax(logits_clean, dim=1)
    predictions = probabilities.argmax(dim=1)
    print(f"Shape Predictions : {predictions.shape}")
    print(f"Label Predictions : {predictions}")

```

```
    # Konversi indeks ke label
    true_label_convert = [ids_to_labels[label] for label in label_clean.cpu().numpy() if label != -100]
    print(f"Label Actual : {true_label_convert}")
    pred_label_convert = [ids_to_labels[label] for label in predictions.cpu().numpy() if label != -100]
    print(f"Label Predict : {pred_label_convert}")

    # Pengecekan panjang label yang sesuai
    print(f"Panjang Label Apakah sama? {len(true_label_convert) == len(pred_label_convert)}")
```

3. Pengaruh BertTokenizer terhadap Dataset

BERT membutuhkan input sebuah teks yang nantinya akan di tokenisasi sendiri menggunakan BERT Tokenizer. BERT Tokenizer menggunakan algoritma WordPiece sehingga terdapat perbedaan antara hasil tokenisasi dari anotasi dan Bert Tokenizer. Hal ini dapat dibuktikan dengan uji coba menggunakan Lampiran Kode 1.3.

Lampiran Kode 1.3 Analisis Tokenisasi BERT

```
def align_label_Example(texts, labels, tokenizer, labels_to_ids):
    print(f"Text : {texts}")
    tokenized_inputs = tokenizer(texts, padding='max_length', max_length=512, truncation=True)
    token = tokenizer.convert_ids_to_tokens(tokenized_inputs.input_ids)
    clear_token = [i for i in token if i not in ['[CLS]', '[SEP]', '[PAD]']]
    print(f"Hasil Tokenisasi BERT : {clear_token}")
    print(f"Jumlah Token Hasil Tokenisasi BERT: {len(clear_token)}")
    print()
    print(f"Labels : {labels}")
    print(f"Jumlah Label : {len(labels)}")

    word_ids = tokenized_inputs.word_ids()
    # print(f"Splitting Biasa : {texts.split()}")
    # print(f"Word Ids : {word_ids}")

    previous_word_idx = None
    label_ids = []

    for word_idx in word_ids:

        if word_idx is None:
            label_ids.append(-100)

        elif word_idx != previous_word_idx:
            try:
                label_ids.append(labels_to_ids[labels[word_idx]])
            except:
                label_ids.append(-100)
        else:
            try:
                label_ids.append(labels_to_ids[labels[word_idx]])
            except:
                label_ids.append(-100)
        previous_word_idx = word_idx

    clear_label = [i for i in label_ids if i != -100]

    return clear_label, token

txt = df_pros["text"][134]
lb = df_pros["labels"][134].split()
labels_to_ids =  {'B_ADVO': 0, 'B_ARTV': 1, 'B_CRIA': 2, 'B_DEFN': 3, 'B_JUDG': 4, 'B_JUDP': 5, 'B_PENA': 6, 'B_PROS': 7, 'B_PUNI': 8, 'B_REGI': 9, 'B_TIMV': 10, 'B_VERN': 11, 'I_ADVO': 12, 'I_ARTV': 13, 'I_CRIA': 14, 'I_DEFN': 15, 'I_JUDG': 16, 'I_JUDP': 17, 'I_PENA': 18, 'I_PROS': 19, 'I_PUNI': 20, 'I_REGI': 21, 'I_TIMV': 22, 'I_VERN': 23, 'O': 24}
label_cv, token = align_label_Example(txt,lb, tokenizer, labels_to_ids)
```

DAFTAR RIWAYAT HIDUP

Data Pribadi :

Nama	: Ahmad Rosyihuddin
Jenis Kelamin	: Laki-Laki
Tempat, Tanggal Lahir	: Gresik, 05 Agustus 2001
Alamat	: Dusun Ngawen RT 002 RW 001
Desa	: Ngawen
Kecamatan	: Sidayu
Kabupaten	: Gresik
Agama	: Islam
Status Pernikahan	: Belum Menikah
Kewarganegaraan	: Indonesia
Program Studi	: Teknik Informatika
E-mail	: rosyihuddin.dev@gmail.com
No. Handphone	: -

Riwayat Pendidikan :

No.	Instansi Pendidikan	Periode Tempuh
1	MI Kanjeng Sepuh I Ngawen	2008 – 2014
2	MTs. Kanjeng Sepuh Sidayu	2014 – 2017
3	SMK Assa'adah Bungah	2017 – 2020
4	Universitas Trunojoyo Madura	2020 – 2024