UNIVERSITAS GUNADARMA

FAKULTAS TEKNOLOGI INDUSTRI

PENULISAN ILMIAH

PENGEMBANGAN NEURAL TRANSLATOR UNTUK QUERY PROGRAMING IN LOGIC MENGGUNAKAN ARSITEKTUR TEXT-TO-TEXT TRANSFORMER

DISUSUN OLEH:

Nama	**: Fernando Morientes Hutapea**
NPM	**: 50422570**
Program Studi	**: Informatika**
Pembimbing	**: Fikri Fadlillah, S.T., MMSI**

Jakarta
2025

PERNYATAAN ORIGINALITAS DAN PUBLIKASI

Saya yang bertanda tangan di bawah ini:

Nama	:	Fernando Morientes Hutapea
NPM	:	50422570
Judul Penulisan Ilmiah	:	Pengembangan Neural Translator untuk Query Programing in Logic menggunakan Arsitektur Text-To-Text Transformer
Tanggal Sidang	:	22 Agustus 2025
Tanggal Lulus	:	22 Agustus 2025

Menyatakan bahwa tulisan ini merupakan hasil karya saya sendiri dan dapat dipublikasikan sepenuhnya oleh Universitas Gunadarma. Segala kutipan dalam bentuk apa pun telah mengikuti kaidah dan etika ilmiah yang berlaku. Adapun isi dan tanggung jawab atas tulisan ini sepenuhnya menjadi tanggung jawab penulis, bukan Universitas Gunadarma.

Demikian pernyataan ini saya buat dengan sebenar-benarnya dan penuh kesadaran.

Jakarta, 22 Juli 2025

(Fernando Morientes Hutapea)

LEMBAR PENGESAHAN

Judul Penulisan Ilmiah	:	Pengembangan Neural Translator untuk Query Programing in Logic menggunakan Arsitektur Text-To-Text Transformer
Nama	:	Fernando Morientes Hutapea
NPM	:	50422570
Tanggal Sidang	:	22 Agustus 2025
Tanggal Lulus	:	22 Agustus 2025

Menyetujui,

Pembimbing	**Kepala Subbagian Sidang PI FTI**
(Dr.Fikri Fadlillah, S.T., MMSI)	(Dr. Achmad Fahrurozi, S.Si., M.Si.)

Ketua Program Studi

(Prof. Dr. Lintang Yuniar Banowosari, S.Kom., M.Sc.)

ABSTRAK

Fernando Morientes Hutapea, 50422570

PENGEMBANGAN NEURAL TRANSLATOR UNTUK QUERY PROGRAMING IN LOGIC MENGGUNAKAN ARSITEKTUR TEXT-TO-TEXT TRANSFORMER

Penulisan Ilmiah, Jurusan Informatika. Fakultas Teknologi Industri. Universitas Gunadarma, 2025.

Kata Kunci : Neural Translator, Transformer, T5, Prolog, Logika Formal, Halusinasi, Large Language Models, Ekstraksi Fakta, AI Simbolik.

(vi + 56 halaman)

Penelitian ini bertujuan mengembangkan model neural translator berbasis arsitektur Transformer, khususnya T5 (Text-to-Text Transfer Transformer), untuk menerjemahkan bahasa alami ke klausa logika formal yang dapat diproses oleh sistem berbasis aturan. Fokus utama penelitian ini adalah menguji efektivitas model dalam menghasilkan representasi formal yang valid secara sintaksis dan semantik. Penelitian ini menggunakan metode CRISP-DM, yang mencakup tahapan pemahaman bisnis, data, persiapan, pemodelan, evaluasi, dan penerapan, dengan data yang berasal dari dokumen perjanjian internasional yang diubah menjadi format Prolog.

Hasil evaluasi menunjukkan bahwa model yang dikembangkan mampu menghasilkan klausa logika dengan tingkat kecocokan tinggi (Exact Match/EM dan Execution Match/XM). Ini menunjukkan potensi besar untuk aplikasi seperti sistem tanya jawab berbasis logika, legaltech, dan AI simbolik. Penelitian ini membuka peluang penerapan pendekatan neuro-symbolic dalam aplikasi praktis yang memerlukan pemahaman dan penalaran berbasis logika formal.

KATA PENGANTAR

Puji dan Syukur dipanjatkan ke hadirat Allah SWT, atas berkah rahmat dan karunia-Nya, serta doa dan motivasi dari berbagai pihak sehingga dapat terselesaikannya tugas Penulisan Ilmiah yang berjudul "Pengembangan Neural Translator untuk Query Programing in Logic menggunakan Arsitektur Text-To-Text Transformer" dengan sebaik-baiknya.

Dalam segala kerendahan hati, perkenankanlah ucapan terima kasih ini dihaturkan atas dorongan dan bantuan yang diterima, sehingga dapat menyelesaikan Penulisan Ilmiah ini, ucapan terima kasih dihaturkan kepada:

1. Prof. Dr. Hj. E. S. Margianti, SE, MM., selaku rektor Universitas Gunadarma.
2. Prof. Dr. Ing. Adang Suhendra S.Si., SKom., MSc., selaku Dekan Fakultas Teknologi Industri Universitas Gunadarma.
3. Prof. Dr. Lintang Yuniar Banowosari, SKom., MSc., selaku Ketua Program Studi Informatika, Universitas Gunadarma.
4. Dr. Achmad Fahrurozi, S.Si, M.Si., selaku Kepala Subbagian Sidang Penulisan Ilmiah Fakultas Teknologi Industri Universitas Gunadarma.
5. Dr. Fikri Fadlillah, ST., MMSI., selaku dosen pembimbing yang telah banyak memberikan waktu dan sarannya selama pembuatan Penulisan Ilmiah ini.
6. Kedua orangtua, Bapak Saut Hutapea dan Ibu Roselyn PD.H ,.Ns,.MM.,MH, CAC, FISQua., berkat doa serta restu keduanya, maka Penulisan Ilmiah ini dapat terselesaikan.
7. Teman-teman Kelas 3IA24 serta pihak lainnya yang secara langsung ikut memberikan dukungan.

Penulisan Ilmiah ini diharapkan dapat bermanfaat bagi aktivitas akademis Program Studi Informatika Fakultas Teknologi Industri, Universitas Gunadarma, dan menyadari atas segala keterbatasan pengetahuan dan kemampuan diharapkan adanya kritik yang membangun dan semoga Penulisan Ilmiah ini dapat bermanfaat untuk orang banyak.

Jakarta, 22 Juli 2025

Fernando Morientes Hutapea

DAFTAR ISI

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi AI yang pesat dalam dua dekade terakhir telah mendorong integrasi antara kecerdasan buatan (*Artificial Intelligence/AI*) dan pemrosesan bahasa alami (*Natural Language Processing/NLP*) ke dalam berbagai bidang kehidupan. Salah satu implikasi signifikan dari perkembangan ini adalah meningkatnya kebutuhan akan sistem yang mampu menerjemahkan perintah dalam bahasa manusia ke dalam bentuk yang dapat dipahami oleh mesin, seperti query logika formal dalam sistem basis data atau sistem berbasis aturan.(Jurafsky & Martin, 2023)

Perkembangan kecerdasan buatan (AI) telah melalui tiga fase utama, yaitu *Rule-Based AI*, *Statistical AI*, dan *Generative AI*. Pada era awal (1950–1980an), AI dikembangkan menggunakan pendekatan rule-based, yaitu sistem pakar yang bekerja berdasarkan aturan logika "jika-maka" yang ditentukan secara manual oleh manusia, seperti sistem MYCIN untuk diagnosis medis. Memasuki era 1990an hingga 2010an, pendekatan statistik dan machine learning mulai mendominasi, di mana AI belajar dari data menggunakan algoritma seperti decision trees, support vector machines, dan neural networks. Revolusi besar terjadi dengan kemunculan deep learning dan arsitektur transformer, yang menjadi fondasi dari Generative AI di era 2020an. Teknologi ini memungkinkan AI tidak hanya mengenali pola, tetapi juga menghasilkan konten baru seperti teks, gambar, dan kode secara otonom, sebagaimana ditunjukkan oleh model seperti ChatGPT dan DALL·E. Perkembangan ini menunjukkan transisi dari sistem berbasis aturan kaku menuju sistem yang adaptif dan kreatif, meskipun tetap diiringi tantangan seperti kebutuhan komputasi tinggi dan isu etika.(Buchanan, B. G., & Shortliffe, E. H. 1984, Mitchell, T. M. 1997, Vaswani, A., et al. 2017, Bommasani, R., et al. 2021)

Large Language Model (LLM) adalah model kecerdasan buatan berskala besar yang dilatih pada sejumlah besar data teks untuk memahami dan menghasilkan bahasa alami secara kontekstual. Model ini biasanya dibangun menggunakan arsitektur *transformer*, seperti yang diperkenalkan oleh Vaswani et al. (2017), dan mampu melakukan berbagai tugas pemrosesan bahasa alami hanya dengan instruksi berupa *prompt*, tanpa pelatihan tambahan. Salah satu contoh paling dikenal adalah GPT-3 yang dikembangkan oleh OpenAI, yang menunjukkan bahwa LLM mampu menyelesaikan tugas-tugas kompleks seperti menjawab pertanyaan, menerjemahkan teks, menulis esai, hingga menghasilkan kode program (Brown et al., 2020). Meskipun memiliki kemampuan luar biasa, LLM juga menghadapi tantangan besar seperti potensi bias dari data pelatihan, kebutuhan komputasi yang tinggi, serta kesulitan dalam menjelaskan alasan di balik jawaban yang dihasilkan.(Bommasani et al., 2021).(Vaswani, A. et al. 2017, Brown, T. et al. 2020, Bommasani, R. et al. 2021)

Halusinasi merupakan kelemahan utama LLM yang belum dapat sepenuhnya diatasi. Meskipun banyak pendekatan empiris telah dikembangkan, secara teori LLM tidak mampu mempelajari semua fungsi yang dapat dihitung, sehingga halusinasi tidak bisa dihindari. Karena dunia nyata lebih kompleks dari dunia formal, halusinasi tetap terjadi pada LLM dunia nyata, terutama dalam tugas-tugas tertentu yang dibatasi oleh waktu komputasi. Studi ini juga mengevaluasi efektivitas mekanisme mitigasi dan implikasinya terhadap penggunaan LLM yang aman.(Ziwei Xu, Sanjay Jain, dan Mohan Kankanhalli, 2024)

Permasalahan halusinasi dapat diatasi dengan pendekatan yang mampu menjembatani fleksibilitas bahasa alami dengan struktur logika formal yang kaku. Salah satu solusi potensial adalah integrasi model generatif berbasis Transformer dengan teknik pemrograman simbolik atau pendekatan hybrid neuro-symbolic. Pendekatan ini diharapkan meningkatkan akurasi pemetaan semantik dan sintaksis, serta mengurangi kesalahan logika akibat halusinasi model. Penelitian ini difokuskan pada evaluasi efektivitas model

Transformer dalam menerjemahkan instruksi bahasa alami menjadi query logika formal secara konsisten dan andal.(Marcus, G, 2020)

Berdasarkan uraian di atas, penelitian ini bertujuan mengevaluasi dan mengembangkan pendekatan berbasis Transformer untuk menerjemahkan bahasa alami ke klausa logika formal. Fokus utama mencakup pengujian kemampuan model dalam memahami konteks semantik dan menghasilkan representasi formal yang dapat dieksekusi. Selain itu, penelitian juga menilai potensi integrasi pendekatan simbolik dalam mengurangi halusinasi serta meningkatkan keandalan sistem dalam konteks dunia nyata.(Ziwei Xu, Sanjay Jain, dan Mohan Kankanhalli, 2024)

1.2 Batasan Masalah

Berdasar latar belakang yang sudah dipaparkan, batasan masalah pada penelitian, terdiri dari:

1. Penelitian ini hanya melatih dan mengevaluasi model.
2. Model yang digunakan adalah model pre-trained dengan arsitektur Transformer.
3. Metrik yang digunakan hanya metrik Exact Match dan Execution Match.
4. Data yang digunakan berasal dari sumber dokumen resmi yang diperoleh melalui Treaty Room milik Kementerian Luar Negeri Republik Indonesia (Kemlu RI), Dataset terdiri dari total 4.744 data .

1.3 Tujuan Penelitian

Tujuan dari adanya penelitian ini adalah:

1. Mengembangkan model neural translator berbasis LLM untuk menerjemahkan bahasa alami ke dalam klausa logika formal.
2. Menerapkan reasoning berdasarkan klausa logika formal terhadap basis pengetahuan dalam bentuk prolog.

1.4 Metode Penelitian

Metode yang digunakan dalam penelitian ini menggunakan proses CRISP-DM yang terdiri dari:

1. Business Understanding

   Fase ini bertujuan untuk memperoleh pemahaman yang mendalam tentang tujuan dan permasalahan bisnis yang ingin diselesaikan melalui data mining. Pada tahap ini, tim proyek menentukan apa yang menjadi fokus analisis, mengidentifikasi tujuan utama, serta batasan dan kendala yang ada dalam proyek. Pemahaman yang kuat terhadap konteks bisnis sangat penting agar proyek data mining menghasilkan solusi yang relevan dan bermanfaat bagi organisasi. Keberhasilan dalam fase ini menjadi fondasi penting bagi langkah-langkah selanjutnya.

2. Data Understanding

   Dalam fase ini, data yang tersedia dikumpulkan dan diperiksa secara mendalam untuk memahami struktur, kualitas, dan potensi informasi yang terkandung di dalamnya. Proses ini mencakup eksplorasi awal terhadap data melalui analisis statistik dasar, visualisasi, serta identifikasi data yang hilang, outlier, atau anomali. Selain itu, kualitas dan kesesuaian data terhadap tujuan analisis juga dievaluasi. Hasil dari fase ini dapat menyebabkan revisi terhadap tujuan bisnis atau perencanaan ulang proyek jika ditemukan kekurangan signifikan pada data.

3. Data Preparation

   Tahap ini melibatkan kegiatan mempersiapkan data agar dapat digunakan dalam proses pemodelan. Persiapan mencakup pemilihan subset data yang relevan, pembersihan data dari kesalahan atau inkonsistensi, transformasi atau pembuatan

fitur baru, penggabungan data dari berbagai sumber, dan penyesuaian format data. Proses ini seringkali memakan waktu paling banyak dalam keseluruhan proyek data mining. Kualitas dataset akhir yang dihasilkan dalam fase ini sangat menentukan keberhasilan model yang akan dibangun pada tahap berikutnya.

4. Modeling

   Pada fase ini, teknik-teknik statistik dan machine learning digunakan untuk membangun model yang mampu menemukan pola atau hubungan dalam data. Langkah pertama adalah memilih metode pemodelan yang sesuai, kemudian merancang pengujian, membangun model, dan menyesuaikan parameter-parameter penting. Selanjutnya, model dievaluasi secara teknis untuk melihat performa dan akurasinya terhadap data uji. Fase ini sangat penting karena model yang dihasilkan menjadi dasar dalam pengambilan keputusan atau prediksi.

5. Evaluation

   Setelah model dibuat, perlu dilakukan evaluasi menyeluruh untuk memastikan bahwa model tersebut benar-benar menjawab permasalahan bisnis yang telah ditetapkan sebelumnya. Evaluasi dilakukan dari dua sudut pandang, yaitu teknis dan bisnis. Proses ini juga mencakup tinjauan terhadap keseluruhan alur pengerjaan proyek untuk mengidentifikasi hal-hal yang perlu diperbaiki atau diulang. Jika model sudah dianggap baik, keputusan selanjutnya adalah apakah hasil tersebut akan digunakan, disempurnakan, atau proyek perlu diulang.

6. Deployment

   Fase ini merupakan tahap akhir di mana hasil dari proyek data mining mulai digunakan dalam proses bisnis nyata. Model atau penemuan yang telah divalidasi diterapkan agar memberikan manfaat langsung bagi organisasi. Selain itu, dirancang

pula strategi pemantauan dan pemeliharaan model untuk memastikan bahwa model tetap relevan dan bekerja dengan baik dalam jangka waktu tertentu. Di akhir fase ini, laporan akhir dan dokumentasi lengkap dibuat sebagai referensi dan bahan evaluasi untuk proyek-proyek mendatang.

1.5 Sistematika Penulisan

Sistematika Penulisan dimuat dalam penelitian ini untuk memberikan gambaran terhadap kerangka laporan secara garis besar dengan menjelaskan beberapa bab, antara lain :

- **BAB 1 Pendahuluan**

Bab 1 ini menjelaskan perihal latar belakang dan juga masalah yang ingin dijelaskan dan diangkat dalam penelitian ini. Bab ini terdiri atas latar belakang, rumusan masalah, Batasan masalah, tujuan dan manfaat penelitian, dan sistematika penulisan.

- **BAB 2 Landasan Teori**

Bab 2 ini berisi tentang landasan teori dari penelitian yang dilakukan serta penjelasan mengenai metode apa yang digunakan. Bab ini juga menjelaskan mengenai penelitian terdahulu sebagai dasar dan acuan untuk teori-teori yang berhubungan dengan penelitian.

- **BAB 3 Analisis & Pembahasan**

Pada bab 3 ini akan menjelaskan proses CRISP-ML yang dilakukan dengan menjelaskan setiap bagian dan menampilkan hasil masing-masing tahap. Serta menampilkan hasil akhir dari penelitian yang dilakukan.

- **BAB 4 Penutup**

Bab 4 Penutup ini berisi mengenai Kesimpulan yang didapatkan dari penelitian dan saran yang dapat disampaikan mengenai hasil penelitian.

BAB II
LANDASAN TEORI

2.1 AI Simbolik

AI simbolik adalah pendekatan awal dalam pengembangan kecerdasan buatan yang memfokuskan pada representasi pengetahuan secara jelas dengan menggunakan simbol dan aturan logika. Metode ini berlandaskan pada logika matematika formal, yang memungkinkan mesin untuk melakukan penalaran yang dapat dipahami (explainable reasoning). Salah satu bentuk utama dari AI simbolik adalah logika proposisional, yang melibatkan proposisi-proposisi sederhana dan operator logika seperti AND, OR, dan NOT. Meskipun logika proposisional cukup berguna untuk pemodelan dasar, ia memiliki keterbatasan dalam hal ketidakmampuannya menangani kuantifikasi atau hubungan yang lebih kompleks antara objek. Untuk mengatasi keterbatasan tersebut, digunakan First-Order Logic (FOL), yang memperluas logika proposisional dengan menambahkan kuantor universal (∀) dan eksistensial (∃), serta fungsi dan relasi antar objek. FOL banyak digunakan dalam sistem AI untuk menyatakan pengetahuan tentang dunia nyata dalam bentuk fakta dan aturan.

Horn Clause adalah jenis aturan logika dalam First-Order Logic (FOL) yang terdiri dari satu kesimpulan (head) dan satu atau lebih premis (body). Secara formal, Horn Clause dinyatakan sebagai:

$$p_1 \land p_2 \land \ldots \land p_n \rightarrow q \quad (1)$$

Sedangkan dalam Prolog, Horn Clause ditulis sebagai:

$$q :- p_1, p_2, \ldots, p_n. \quad (2)$$

Horn Clause sangat efektif dalam inferensi karena strukturnya mendukung penerapan teknik penalaran seperti metode resolusi dan backward chaining. Hal ini menjadikannya sangat cocok untuk pengembangan sistem berbasis aturan.

Contoh penerapan dari AI simbolik ini dapat ditemukan pada bahasa pemrograman Prolog. Prolog menggunakan Horn Clause sebagai dasar dari sintaksis dan semantiknya. Dalam Prolog, pengetahuan diwakili dalam bentuk fakta dan aturan, dan penalaran dilakukan dengan menggunakan kueri. Keunggulan Prolog terletak pada sifat deklaratifnya serta kemampuan penelusuran yang efisien, sehingga sangat ideal untuk aplikasi seperti sistem tanya jawab, sistem pakar, serta sektor hukum dan kontraktual yang membutuhkan keterbukaan dalam penalaran logika.

Pendekatan ini sangat relevan dalam penelitian yang bertujuan untuk mengekstrak fakta dan aturan dari dokumen kontraktual, memungkinkan teks diubah menjadi format logis yang lebih mudah diproses.

2.2 Large Language Models

AI generatif adalah jenis kecerdasan buatan yang dirancang untuk menciptakan konten baru berdasarkan pola yang ditemukan dalam data pelatihan. Salah satu jenisnya yang paling menonjol adalah Large Language Models (LLM), seperti GPT-3, GPT-4, dan LLaMA, yang telah terbukti efektif dalam berbagai aplikasi pengolahan bahasa alami (NLP). LLM dapat digunakan untuk tugas-tugas seperti menjawab pertanyaan, merangkum dokumen, dan bahkan menghasilkan kode. Namun, LLM menghadapi masalah utama, yaitu fenomena hallucination, di mana model menghasilkan informasi yang tampak akurat secara bahasa tetapi tidak benar-benar sesuai dengan data atau kenyataan (Ji et al., 2023).

2.2.1 Generative Pretrained Transformer (GPT)

Generative Pretrained Transformer (GPT) adalah model bahasa besar yang dikembangkan oleh OpenAI dan telah menjadi metode utama dalam pengembangan kecerdasan buatan generatif berbasis teks. Model ini menggunakan arsitektur transformer unidirectional yang memproses input secara berurutan dari kiri ke kanan untuk memprediksi token berikutnya berdasarkan konteks yang ada.

Proses pelatihan model GPT terdiri dari dua tahap utama: pretraining dan fine-tuning. Pada tahap pretraining, model dilatih tanpa pengawasan menggunakan korpus teks besar untuk mempelajari pola umum bahasa. Setelah itu, fine-tuning dilakukan dengan pengawasan pada data spesifik yang digunakan untuk menyelesaikan tugas tertentu, seperti menjawab pertanyaan atau mengekstrak informasi. Sejak GPT-3, teknik in-context learning telah menggantikan fine-tuning, dengan memanfaatkan penyusunan prompt untuk memberikan konteks yang diperlukan dalam memandu keluaran model.

GPT mampu menangani berbagai tugas pemrosesan bahasa alami tanpa perlu penyesuaian tambahan pada arsitektur berkat kemampuannya memahami pola linguistik, struktur sintaksis, dan makna semantik. Versi terbaru, GPT-4, menunjukkan peningkatan besar dalam kemampuan reasoning, pemahaman konteks, serta penanganan instruksi yang lebih kompleks. Dalam penelitian ini, GPT digunakan sebagai model dasar untuk mengevaluasi strategi prompting dalam ekstraksi fakta dan aturan berbasis logika. Kinerja model sangat bergantung pada struktur dan kualitas prompt yang diberikan, yang akan dievaluasi melalui teknik zero-shot, few-shot, dan chain-of-thought prompting.

2.2.2 LLaMA (Large Language Model Meta AI)

LLaMA (Large Language Model Meta AI) adalah serangkaian model bahasa besar yang dikembangkan oleh Meta AI dengan tujuan untuk menciptakan model yang lebih efisien, terbuka, dan dapat bersaing dengan model komersial seperti GPT. Berbeda dengan banyak LLM komersial yang bersifat tertutup, LLaMA dirancang sebagai model sumber terbuka untuk mendukung riset dan kolaborasi dalam bidang kecerdasan buatan.

Menggunakan arsitektur transformer yang serupa dengan GPT, LLaMA dioptimalkan untuk efisiensi dan kinerja meskipun menggunakan jumlah parameter yang lebih kecil. Sebagai contoh, LLaMA-2 dengan 13 miliar parameter dapat bersaing dengan GPT-3 yang memiliki 175 miliar parameter dalam tugas-tugas bahasa alami, seperti klasifikasi teks, rangkuman, dan penjawaban pertanyaan. Model ini dilatih dengan data terbuka yang besar dan dirancang agar dapat dijalankan pada perangkat keras dengan sumber daya terbatas, seperti satu GPU riset.

Keunggulan LLaMA terletak pada fleksibilitasnya dalam eksperimen dan penyesuaian untuk berbagai pendekatan fine-tuning, seperti Low-Rank Adaptation (LoRA) dan Instruction Tuning. Selain itu, LLaMA menjadi dasar pengembangan model lain seperti Alpaca, Vicuna, dan OpenLLaMA, yang masing-masing disesuaikan untuk berbagai tujuan spesifik, seperti dialog, penalaran, dan penjelasan berbasis instruksi.

Dalam penelitian ini, LLaMA digunakan sebagai model acuan untuk mengevaluasi strategi prompting, dengan fokus pada pendekatan open-source yang dapat dikustomisasi. LLaMA memberikan kontribusi signifikan dalam perbandingan performa dan efisiensi ekstraksi fakta serta aturan logis dari dokumen kontraktual menggunakan teknik prompting.

2.2.3 Mistral

Mistral merupakan keluarga model bahasa besar open-source yang dikembangkan oleh perusahaan AI Mistral yang berbasis di Prancis. Model ini didesain untuk memberikan performa tinggi sambil mempertahankan efisiensi arsitektur yang optimal, menjadikannya salah satu pesaing utama dalam ekosistem model bahasa terbuka. Mistral menawarkan model dengan parameter yang lebih kecil tetapi tetap mampu memberikan akurasi dan kemampuan reasoning yang setara dengan model yang lebih besar, seperti GPT-3.5 dan LLaMA-2.

Mistral-7B adalah salah satu varian yang paling terkenal, yakni model decoder-only yang dilatih dengan teknik Group Query Attention (GQA), serta menggunakan sliding window attention untuk memproses konteks input yang panjang dengan efisien. Teknik ini memungkinkan model untuk memanfaatkan lebih banyak konteks tanpa meningkatkan kompleksitas komputasi yang signifikan.

Model ini dilatih menggunakan dataset besar yang mencakup dokumen multibahasa, kode sumber, serta data dari berbagai bidang ilmu dan sosial. Hal ini memungkinkan Mistral untuk mengadaptasi dan bekerja di berbagai domain, serta menyelesaikan tugas-tugas seperti ekstraksi informasi, klasifikasi, pemrosesan instruksi, dan pemodelan dialog.

Keunggulan utama Mistral terletak pada efisiensinya dan kemampuannya untuk disesuaikan secara lokal. Dalam penelitian ini, Mistral dianggap sebagai alternatif strategis untuk mengevaluasi keefektifan prompting, terutama karena kemampuannya dalam menafsirkan instruksi secara eksplisit dan mendukung pengujian reasoning berbasis chain-of-thought.

Dengan ukuran parameter yang kompak namun tetap mempertahankan performa yang tinggi, Mistral menunjukkan potensi besar sebagai model pilihan dalam lingkungan

yang memiliki keterbatasan sumber daya, sekaligus mendukung validitas strategi prompting pada model non-komersial.

2.2.4 Text-to-Text Transformer

Text-to-Text Transformer adalah paradigma dalam pemrosesan bahasa alami yang mengubah berbagai jenis tugas linguistik menjadi format input dan output berbasis teks. Pendekatan ini dikenal luas melalui model T5 (Text-to-Text Transfer Transformer) yang dikembangkan oleh Google Research dan menjadi konsep penting dalam pengembangan model generatif.

Sementara pendekatan tradisional sering membedakan berbagai jenis tugas seperti klasifikasi, ekstraksi, dan generasi teks, pendekatan text-to-text menyatukan semuanya ke dalam format seragam, yakni teks masuk dan teks keluar. Sebagai contoh, tugas ekstraksi informasi akan diformulasikan sebagai "Ekstrak entitas dari teks berikut: [...]", dan tugas klasifikasi sentimen akan dirumuskan sebagai "Tentukan sentimen dari kalimat berikut: [...]". Pendekatan ini memberikan fleksibilitas tinggi dalam arsitektur model dan memudahkan adaptasi terhadap berbagai jenis instruksi.

Model T5 menggunakan arsitektur encoder-decoder berbasis Transformer dan dilatih dengan teknik pretraining menggunakan dataset besar seperti Colossal Clean Crawled Corpus (C4). Selama pretraining, model dilatih untuk memprediksi bagian teks yang hilang melalui teknik span corruption. Setelah itu, model T5 dapat di-fine-tune untuk berbagai tugas NLP dengan format yang konsisten.

Pendekatan ini menjadi landasan bagi berbagai model lanjutan seperti FLAN-T5, mT5 (versi multibahasa), dan UL2, yang menggabungkan kemampuan encoder-decoder dengan format multitugas. Dalam penelitian ini, text-to-text transformer digunakan untuk merancang prompt berbasis instruksi yang jelas dan generik. Format ini ideal untuk

menyusun berbagai strategi prompting seperti zero-shot, few-shot, dan chain-of-thought, karena semua skenario dapat disajikan dalam bentuk input-output teks secara langsung.

Keunggulan utama dari pendekatan text-to-text adalah keseragaman arsitektur, kemampuan untuk mentransfer pembelajaran antar tugas, serta adaptasi yang mudah terhadap instruksi yang lebih kompleks dan eksplisit. Oleh karena itu, text-to-text transformer memberikan dasar yang solid dalam penelitian berbasis prompting pada dokumen kontraktual.

2.3 Permasalahan Large Language Models (LLM)

Large Language Models (LLM) telah membuktikan kemampuannya yang luar biasa dalam menyelesaikan berbagai tugas pemrosesan bahasa alami. Namun, di balik kemampuan tersebut, LLM masih memiliki beberapa masalah mendasar yang tidak bisa diabaikan. Permasalahan ini mencakup berbagai aspek, mulai dari teknis, akurasi, keandalan, hingga masalah etis terkait dengan informasi yang dihasilkan.

Kelemahan-kelemahan tersebut umumnya disebabkan oleh sifat dasar LLM yang mengandalkan pendekatan statistik-probabilistik, keterbatasan dalam data pelatihan, serta kurangnya mekanisme penalaran eksplisit yang dapat diaudit dengan jelas. Akibatnya, model bisa menghasilkan output yang tidak tepat, bias, atau tidak sesuai dengan konteks yang diinginkan.

Memahami berbagai masalah ini menjadi hal yang sangat penting, terutama saat LLM digunakan dalam sistem yang memerlukan ketelitian, akuntabilitas, dan tanggung jawab informasi. Oleh karena itu, subbab ini akan membahas tiga isu utama yang sering muncul saat menggunakan LLM, yaitu halusinasi, bias, dan beberapa keterbatasan lainnya terkait dengan representasi pengetahuan serta kontrol terhadap hasil keluaran.

2.3.1 Halusinasi

Dalam konteks LLM, halusinasi merujuk pada kejadian di mana model menghasilkan informasi yang keliru, menyesatkan, atau bahkan sepenuhnya salah, meskipun disajikan dengan cara yang meyakinkan. Hal ini disebabkan oleh ketergantungan model pada pola statistik dalam data pelatihan, alih-alih pada pemahaman yang sesungguhnya mengenai kebenaran. Dalam aplikasi seperti sistem ekstraksi fakta atau chatbot hukum, halusinasi dapat menimbulkan dampak serius, seperti penyebaran informasi yang salah atau kesalahan dalam pengambilan keputusan. Oleh karena itu, mendeteksi dan mengurangi halusinasi menjadi aspek yang sangat penting dalam pengembangan LLM.

2.3.2 Bias

Bias dalam LLM terjadi ketika model mencerminkan atau memperburuk ketimpangan sosial, budaya, atau gender yang ada dalam data pelatihan. Hal ini bisa muncul dalam bentuk stereotip, ketidakadilan dalam representasi, hingga diskriminasi yang tidak langsung. Meskipun LLM tidak memiliki keinginan atau preferensi, bias yang ada dalam data pelatihan dapat menghasilkan keputusan yang tidak adil atau tidak etis. Penggunaan LLM dalam konteks yang sensitif, seperti dalam perekrutan atau layanan publik, perlu memperhitungkan potensi bias tersebut dan menerapkan strategi untuk menguranginya.

2.3.3 Kurangnya Transparansi dan Interpretabilitas

Salah satu kesulitan utama dalam penggunaan LLM adalah kurangnya transparansi mengenai bagaimana model mengambil keputusan. Arsitektur transformer besar yang digunakan dalam LLM berfungsi sebagai "kotak hitam" (black box), di mana proses internalnya sangat sulit dipahami oleh manusia. Akibatnya, pengguna atau peneliti kesulitan untuk melacak logika di balik hasil keluaran model. Keterbatasan dalam interpretabilitas ini menjadi hambatan dalam menilai keandalan model, melakukan audit etis, serta memenuhi persyaratan regulasi seperti GDPR yang mengharuskan penjelasan atas keputusan otomatis.

2.3.4 Konsumsi Sumber Daya dan Isu Keberlanjutan

Pelatihan dan operasionalisasi LLM membutuhkan sumber daya komputasi yang sangat besar, yang berpengaruh langsung terhadap konsumsi energi dan jejak karbon yang dihasilkan. Berdasarkan beberapa studi, pelatihan satu model LLM bisa menghasilkan emisi karbon setara dengan beberapa penerbangan transatlantik. Ini menimbulkan pertanyaan mendalam tentang keberlanjutan jangka panjang dan dampak lingkungan dari perkembangan teknologi AI. Oleh karena itu, pendekatan yang lebih ramah lingkungan perlu dipertimbangkan, seperti model yang lebih efisien, kompresi model, atau penggunaan energi terbarukan.

2.3.5 Risiko Penyalahgunaan dan Tantangan Etika

LLM dapat disalahgunakan untuk tujuan yang merugikan, seperti penyebaran disinformasi, pembuatan deepfake teks, atau rekayasa sosial. Terutama dengan adanya model yang tersedia dalam bentuk open source, risiko tersebut semakin meningkat, karena akses tanpa kontrol yang memadai menjadi lebih mudah. Masalah etika juga timbul terkait privasi data, perlindungan hak kekayaan intelektual, dan pembatasan penggunaan dalam bidang-bidang sensitif seperti pendidikan dan hukum. Untuk itu, sangat penting adanya regulasi dan kerangka etika yang jelas dalam pengembangan dan distribusi LLM.

2.4 Penjanjian Internasional

Perjanjian internasional adalah dokumen hukum yang mengikat antara dua negara atau lebih, disusun secara tertulis, dan diatur sesuai dengan hukum internasional. Dalam penelitian ini, perjanjian internasional berfungsi sebagai sumber utama data yang akan dianalisis dan dikonversi menjadi fakta serta aturan berbasis logika Prolog. Dokumen-dokumen ini diperoleh dari sumber resmi yang telah dipublikasikan, seperti Treaty Room,

dan mencakup berbagai topik, termasuk kerja sama bilateral, nota kesepahaman (MoU), dan klausul hukum kontraktual lainnya.

Perjanjian internasional umumnya ditulis dalam bahasa formal dengan struktur yang relatif konsisten, yang meliputi bagian-bagian seperti pendahuluan (preambule), tujuan kerja sama, ruang lingkup kegiatan, mekanisme pelaksanaan, pendanaan, dan ketentuan penutupan. Struktur tersebut menjadikannya sangat cocok untuk diekstraksi menggunakan pendekatan berbasis logika simbolik.

Dalam studi ini, sebanyak 476 dokumen perjanjian internasional digunakan sebagai data sumber. Setiap dokumen diekstraksi untuk mengambil bagian-bagian pentingnya, kemudian dikonversi menjadi representasi simbolik dengan menggunakan klausa Horn yang diproses di dalam sistem logika Prolog. Proses ini melibatkan penggunaan teknik prompting khusus agar Large Language Model (LLM) dapat memahami dan mengonversi isi dokumen menjadi struktur yang valid baik secara sintaksis maupun semantik dalam Prolog.

Dengan menggunakan perjanjian internasional sebagai objek ekstraksi, penelitian ini juga menggambarkan potensi LLM untuk memahami teks hukum dan mengubahnya menjadi format formal yang dapat diproses oleh mesin. Ini membuka kemungkinan baru untuk penerapan AI simbolik dalam legaltech, pengarsipan hukum, serta sistem pendukung keputusan kebijakan luar negeri.

2.5 Studi Pendahuluan: Strategi Prompting

Prompting adalah teknik yang digunakan untuk mengarahkan perilaku model bahasa besar (LLM) dalam menyelesaikan tugas tertentu hanya dengan teks instruksi. Berbagai strategi prompting telah dikembangkan untuk meningkatkan performa dan akurasi model, termasuk:

- **Zero-Shot Prompting**
  Model diberi instruksi tanpa diberikan contoh terlebih dahulu. Contoh: “Identifikasi kewajiban dalam teks berikut.”

- **Few-Shot Prompting**
  Beberapa contoh input-output diberikan untuk membantu model mengenali pola. Ini adalah fokus utama dalam penelitian ini (Brown et al., 2020).

- **Chain-of-Thought Prompting**
  Model diminta untuk menalar secara bertahap sebelum memberikan jawaban akhir. Teknik ini sangat bermanfaat untuk tugas yang membutuhkan logika bertingkat (Wei et al., 2022).

- **Self-Consistency Prompting**
  Model menghasilkan beberapa hasil melalui sampling, kemudian memilih jawaban mayoritas yang paling konsisten (Wang et al., 2022).

- **Instruction Prompting**
  Menggunakan instruksi eksplisit dengan gaya direktif, seperti yang dilakukan pada InstructGPT dan ChatGPT (Ouyang et al., 2022).

- **ReACT (Reasoning and Acting)**
  Teknik yang menggabungkan penalaran dan aksi secara interaktif, sangat cocok untuk sistem multi-langkah atau berbasis agen (Yao et al., 2022).

- **Retrieval-Augmented Prompting (RAP)**
  Menggabungkan hasil pencarian dari basis data vektor atau dokumen eksternal ke dalam konteks prompt (Lewis et al., 2020).

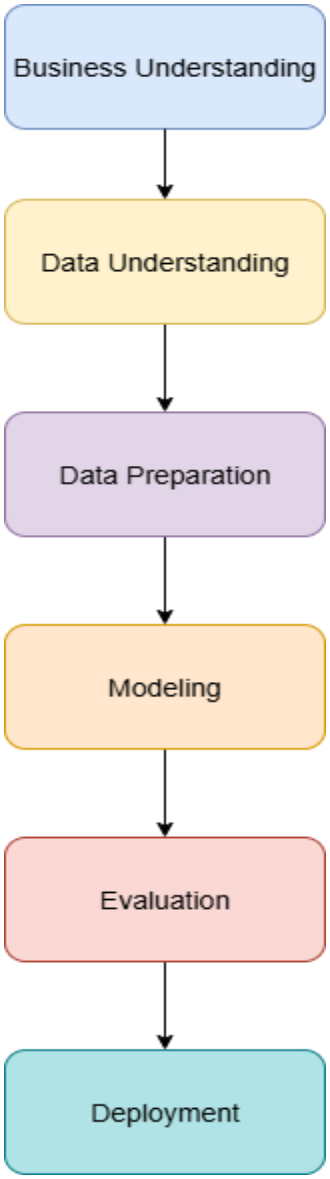
Berdasarkan studi literatur, strategi prompting yang tepat dapat meningkatkan hasil ekstraksi fakta secara signifikan, terutama dalam konteks dokumen hukum yang kompleks dan penuh makna.

2.6 Penutup

Bab ini menguraikan landasan teori dan konsep-konsep utama yang mendasari penelitian ini, termasuk pendekatan AI simbolik dan generatif, logika formal, serta strategi prompting pada LLM. Memahami konsep-konsep ini merupakan langkah awal yang krusial dalam merancang serta mengaplikasikan metode ekstraksi fakta dan aturan dari dokumen kontraktual, yang akan dibahas lebih lanjut pada Bab III.

BAB III
ANALISIS & PEMBAHASAN

3.1 Tahapan Penelitian Berdasarkan CRISP-DM



Gambar 3.1 Crisp DM

3.2 Business Understanding

Salah satu permasalahan utama yang sering muncul dalam pemanfaatan Large Language Model (LLM) adalah fenomena halusinasi (*hallucination*), yaitu ketika model menghasilkan informasi yang terdengar meyakinkan namun tidak didasarkan pada fakta yang benar. Meskipun model mampu menyusun jawaban yang secara linguistik tampak benar dan koheren, informasi tersebut kerap kali tidak dapat diverifikasi kebenarannya karena tidak bersumber dari pengetahuan aktual. Halusinasi menjadi ancaman serius, khususnya pada penerapan LLM dalam konteks yang menuntut akurasi tinggi seperti sistem tanya jawab berbasis pengetahuan, pembuatan ringkasan ilmiah, maupun penerjemahan kalimat ke dalam bahasa logika formal seperti Prolog.

Halusinasi terjadi karena LLM hanya mengandalkan representasi statistik internal yang diperoleh dari proses pelatihan terhadap korpus besar, tanpa memiliki akses langsung ke sumber informasi eksternal atau terkini. Dengan demikian, ketika model dihadapkan pada pertanyaan faktual atau membutuhkan pengetahuan spesifik, ia cenderung menghasilkan jawaban berdasarkan "dugaan terlatih" alih-alih fakta yang sebenarnya. Hal ini tentu berbahaya, karena pengguna dapat menerima informasi yang salah seolah-olah itu benar.

3.3 Data Understanding

Pada tahap ini, dilakukan pemahaman menyeluruh terhadap struktur dan karakteristik data yang digunakan dalam penelitian. Tujuannya adalah untuk memastikan bahwa data yang tersedia sesuai dengan kebutuhan proses analisis serta layak untuk digunakan dalam pelatihan model.

3.3.1 Sumber Data

Data yang digunakan dalam penelitian ini berasal dari dokumen resmi yang diperoleh melalui Treaty Room milik Kementerian Luar Negeri Republik Indonesia (Kemlu RI). Ada 476 dokumen perjanjian internasional digunakan sebagai data sumber,dan pengambilan data menggunakan data sekunder.Dengan jumlah dataset terdiri dari total 4.744 data.Treaty Room merupakan salah satu fasilitas yang menyimpan berbagai dokumen perjanjian internasional, nota diplomatik, serta dokumen hukum lainnya yang bersifat publik dan legal-formal. Dokumen yang dikumpulkan mencerminkan berbagai bentuk perjanjian bilateral maupun multilateral yang melibatkan Indonesia sebagai pihak utama.

3.3.2 Karakteristik Data

Setiap dokumen yang digunakan memiliki struktur yang relatif seragam, terdiri dari tiga bagian utama:

- **Judul Dokumen**: Bagian ini menunjukkan jenis dan topik dari perjanjian, misalnya "Perjanjian Ekstradisi antara Republik Indonesia dan Negara X".
- **Pembukaan (Preambule)**: Merupakan bagian naratif yang menjelaskan latar belakang, pertimbangan hukum, dan maksud dari perjanjian.
- **Isi atau Pasal (Articles)**: Terdiri dari kumpulan pasal-pasal yang mengatur substansi perjanjian. Setiap pasal memiliki struktur sendiri dan memuat ketentuan hukum yang bersifat mengikat.

Karakteristik tersebut membuat data ini kaya akan struktur bahasa formal dan legal, sehingga relevan untuk digunakan dalam tugas seperti ekstraksi informasi, segmentasi dokumen, atau penerjemahan ke bahasa logika formal (misalnya: Prolog).

3.3.3 Wujud Data

Wujud awal dari data yang digunakan dalam penelitian ini adalah dokumen fisik berupa naskah perjanjian internasional yang diperoleh dari Treaty Room. Dokumen-dokumen tersebut kemudian dipindai dan diekstraksi isinya melalui proses digitalisasi, baik secara manual maupun semi-otomatis. Hasil ekstraksi ini dikonversi ke dalam format teks terstruktur dan disusun dalam bentuk file CSV sebagai format akhir yang digunakan untuk pelatihan model.

File CSV tersebut memiliki tiga kolom utama: question, answer, dan context. Kolom question berisi kalimat interogatif dalam bahasa alami, answer berisi klausa logika Prolog yang menjadi jawaban, sedangkan context menyimpan fakta-fakta Prolog yang relevan sebagai dasar inferensi.

Setiap baris data mewakili satu pasangan pertanyaan dan jawaban yang dapat digunakan untuk pelatihan model *sequence-to-sequence*. Format CSV dipilih karena sederhana dan kompatibel dengan berbagai pustaka pemrosesan data dalam Python seperti pandas dan datasets dari HuggingFace. dan kompatibel dengan berbagai pustaka pemrosesan data dalam Python seperti pandas dan datasets dari HuggingFace.

3.3.4 Jumlah Data

Dataset terdiri dari total 4.744 data yang dibagi menjadi tiga bagian utama: 3.795 untuk training, 474 untuk validation, dan 475 untuk test. Pembagian ini memastikan model dapat dilatih dan divalidasi secara optimal dengan proporsi data yang seimbang. Distribusi ini penting agar model memiliki eksposur memadai terhadap variasi data selama pelatihan dan pengujian. Jumlah data pada masing-masing bagian juga menjadi acuan dalam penentuan parameter seperti batch_size, jumlah epoch, dan frekuensi evaluasi. Dengan pembagian yang proporsional, proses fine-tuning dapat berlangsung efisien sambil tetap menjaga akurasi dan

kemampuan generalisasi model terhadap data baru. Informasi ini menjadi dasar dalam penyusunan pipeline preprocessing dan pelatihan model.

3.4 Data Preparation

Tahap *Data Preparation* merupakan langkah penting dalam proses penelitian ini karena data yang digunakan berasal dari dokumen hukum formal yang tidak secara langsung dapat diproses oleh model bahasa. Oleh karena itu, diperlukan sejumlah tahapan untuk mengolah data mentah menjadi bentuk yang siap digunakan dalam pelatihan model. Proses ini mencakup pembersihan teks, ekstraksi fakta dan aturan logika, serta penyusunan dataset dalam format yang sesuai dengan kebutuhan model.

3.4.1 Pembersihan Data

Tahap awal yang dilakukan dalam proses data preparation adalah pembersihan data untuk memastikan bahwa data yang digunakan dalam pelatihan model memiliki integritas dan konsistensi yang tinggi. Proses ini bertujuan untuk menghilangkan potensi gangguan yang dapat memengaruhi performa model, seperti data yang tidak lengkap, duplikat, atau mengandung karakter tidak valid. Fokus utama pembersihan berada pada tiga kolom inti dalam dataset, yaitu context, question, dan answer, yang masing-masing memiliki peran penting dalam membentuk pasangan input-output bagi model.

Setiap baris diperiksa secara menyeluruh untuk memastikan bahwa ketiga elemen tersebut tersedia dan dalam kondisi layak, termasuk melakukan penyesuaian format teks seperti menghapus karakter newline yang tidak diinginkan.

Berikut pseudocode dari proses pembersihan:

ALGORITMA 3.4.1 Pembersihan Data

INPUT: Dataset mentah (CSV)

OUTPUT: Dataset bersih

1. START
2. Buang baris yang memiliki nilai kosong pada context, question, atau answer
3. Hapus baris yang duplikat
4. Ganti karakter newline (\n) dengan spasi
5. Simpan dataset bersih ke variabel dataframe
6. END

Pseudocode ini menggambarkan alur pembersihan data yang dilakukan sebelum digunakan dalam proses pelatihan model. Proses diawali dengan memuat dataset mentah yang berisi kolom context, question, dan answer. Baris-baris data kemudian diperiksa untuk mendeteksi dan menghapus nilai kosong yang dapat menyebabkan error saat pelatihan. Selain itu, baris duplikat juga dihapus guna menghindari redundansi yang dapat mengganggu generalisasi model. Setelah itu, karakter newline (\n) pada setiap kolom diganti dengan spasi untuk menjaga keseragaman format teks. Dataset yang telah dibersihkan kemudian disimpan ke dalam sebuah variabel DataFrame sebagai output akhir dari proses ini. Langkah-langkah ini bertujuan memastikan bahwa data memiliki kualitas yang memadai dan dalam format yang konsisten agar dapat ditokenisasi secara efisien pada tahap berikutnya.

Proses ini diimplementasikan sebagai berikut:

```
df = pd.read_csv("qa.csv")

df.dropna(subset=["question", "context", "answer"],
inplace=True)

df.drop_duplicates(inplace=True)

df["question"] = df["question"].str.replace("\n", "
").str.strip()

df["context"] = df["context"].str.replace("\n", " ").str.strip()
```

Kode ini digunakan untuk melakukan proses pembersihan awal terhadap dataset yang berisi tiga kolom utama: question, context, dan answer. Langkah pertama dimulai dengan membaca dataset dari file qa.csv menggunakan pustaka pandas, dan menyimpannya dalam sebuah objek DataFrame. Setelah dataset dimuat, langkah selanjutnya adalah menghapus baris-baris yang memiliki nilai kosong (missing values) pada salah satu dari ketiga kolom utama, karena data yang tidak lengkap dapat mengganggu proses pelatihan model. Kemudian, dilakukan penghapusan baris duplikat untuk menghindari redundansi informasi yang dapat menyebabkan bias dalam model. Selanjutnya, setiap teks pada kolom question, context, dan answer diproses dengan mengganti karakter newline (\n) menjadi spasi biasa guna menjaga keseragaman format teks. Setelah itu, setiap teks juga dibersihkan dari spasi kosong di awal dan akhir string dengan menggunakan fungsi str.strip(). Langkah-langkah ini secara keseluruhan bertujuan untuk memastikan bahwa data berada dalam kondisi yang bersih, seragam, dan siap untuk digunakan dalam tahap tokenisasi dan pelatihan model lebih lanjut.

Setelah proses pembersihan, data dibagi menjadi tiga subset, yaitu:

- **Training set**: 3.795 data (80%)

- **Validation set**: 474 data (10%)
- **Test set**: 475 data (10%)

Pembagian ini dilakukan secara acak menggunakan train_test_split dari scikit-learn, dan kemudian dikonversi ke dalam format DatasetDict dari HuggingFace agar dapat diproses langsung oleh pipeline model.

```
train_df, temp_df = train_test_split(df, test_size=0.2, random_state=42)
val_df, test_df = train_test_split(temp_df, test_size=0.5, random_state=42)
```

Setelah proses pembersihan selesai, dataset kemudian dibagi menjadi tiga subset utama: training, validation, dan test. Proses pembagian ini dilakukan dalam dua tahap dengan menggunakan fungsi train_test_split dari pustaka scikit-learn. Pada tahap pertama, 80% dari data dialokasikan sebagai data pelatihan (train_df), sementara 20% sisanya disimpan dalam temp_df. Tahap kedua membagi temp_df secara seimbang menjadi data validasi (val_df) dan data pengujian (test_df), masing-masing sebesar 10% dari total data. Penggunaan parameter random_state=42 bertujuan untuk menjamin reprodusibilitas hasil pembagian. Strategi ini memastikan bahwa model dilatih pada data yang beragam, divalidasi secara objektif selama pelatihan, dan diuji pada data yang benar-benar baru untuk mengukur performa akhirnya secara adil dan akurat.

3.4.2 Ekstraksi fakta & Aturan

Tahapan ini bertujuan untuk mentransformasi informasi hukum yang terkandung dalam kolom context menjadi bentuk representasi simbolik berupa fakta dan aturan logika Prolog. Informasi yang sebelumnya berbentuk narasi atau tabel hukum dikonversi menjadi struktur logika yang dapat diproses oleh sistem reasoning. Proses ini mencakup identifikasi entitas, relasi antarentitas, serta penarikan inferensi berdasarkan atribut tertentu.

Ekstraksi dilakukan secara semi-manual dengan mempertimbangkan struktur kalimat dan pola yang berulang dalam data hukum. Tujuannya adalah agar jawaban yang dihasilkan model nantinya tidak hanya bersifat teks biasa, melainkan mengikuti sintaks logika Prolog yang dapat diuji secara semantik. Format answer dalam dataset telah disusun sedemikian rupa agar memuat fakta logika (seperti penandatangan(indonesia, menteri_luar_negeri)) maupun aturan (seperti masa_berlaku(perjanjian123, 5_tahun)), yang keduanya dapat diproses oleh mesin inferensi logika formal.

Sebagai contoh nyata dari dataset, berikut salah satu baris data:

- Context: "penandatangan(indonesia, menteri_luar_negeri).
  masa_berlaku(perjanjian123, 5_tahun)."
- Question: "Siapa yang menandatangani perjanjian tersebut?"
- Answer: "penandatangan(indonesia, menteri_luar_negeri)."

Pada contoh ini, informasi dalam context menyatakan fakta-fakta hukum mengenai pihak dan masa berlaku suatu perjanjian, sementara answer memuat representasi logika Prolog yang diturunkan dari fakta tersebut. Model akan dilatih untuk memahami konteks semacam ini dan menghasilkan klausa Prolog yang sesuai berdasarkan pertanyaan yang diberikan.

Contoh representasi fakta dan aturan yang diekstrak:

```
penandatangan(indonesia, menteri_luar_negeri).
masa_berlaku(perjanjian123, 5_tahun).
```

Berikut pseudocode proses ekstraksi fakta dan aturan:

ALGORITMA 3.4.2 Ekstraksi fakta & Aturan

Input: Teks hukum mentah dari dokumen perjanjian internasional (berupa kalimat naratif atau tabel), serta daftar entitas dan relasi yang telah didefinisikan sebelumnya.

Output: Fakta Prolog yang disimpan ke kolom context dan aturan/inferensi logika Prolog yang disimpan ke kolom answer.

1. START
2. Ambil kalimat atau tabel dari dokumen hukum
3. Identifikasi entitas penting (misal: negara, pejabat, perjanjian)
4. Tentukan relasi antarentitas (misal: penandatangan, masa_berlaku)
5. Ubah ke format fakta Prolog: relasi(arg1, arg2)
6. Simpan sebagai bagian dari kolom context atau answer
7. END

Proses ekstraksi fakta dan aturan dalam penelitian ini disusun secara sistematis untuk mengubah informasi dalam bentuk teks naratif hukum menjadi representasi simbolik berbasis logika Prolog. Tahapan diawali dengan perintah START sebagai penanda dimulainya proses ekstraksi. Dokumen hukum kemudian dibaca, baik dalam bentuk kalimat maupun tabel, yang kemudian dianalisis untuk mengidentifikasi entitas penting seperti negara, pejabat pemerintah, atau nama perjanjian. Entitas tersebut berfungsi sebagai argumen dalam struktur fakta.

Langkah selanjutnya adalah menentukan relasi yang menghubungkan antar entitas, misalnya relasi "penandatangan" antara negara dan pejabat. Setelah relasi dan entitas

dikenali, sistem membentuk representasi logika dalam format Prolog dengan struktur relasi(arg1, arg2). Fakta dan aturan yang dihasilkan kemudian disimpan ke dalam kolom context (untuk fakta) dan answer (untuk aturan atau inferensi) dalam dataset.

Proses ini diakhiri dengan penanda END sebagai akhir dari proses ekstraksi simbolik. Dengan tahapan ini, data mentah yang sebelumnya bersifat naratif diubah menjadi data logika yang terstruktur dan siap digunakan untuk pelatihan model berbasis reasoning.

Contoh implementasi sederhana (semi-manual):

```
context_text = "Indonesia diwakili oleh Menteri Luar Negeri"
if "Indonesia" in context_text and "Menteri Luar Negeri" in
context_text:
   fakta = "penandatangan(indonesia, menteri_luar_negeri)."
```

Kode tersebut digunakan untuk mengekstraksi informasi faktual dari sebuah kalimat yang memiliki struktur tertentu. Dalam hal ini, sistem mengecek keberadaan dua entitas penting, yaitu “Indonesia” dan “Menteri Luar Negeri”, dalam potongan teks hukum. Jika keduanya ditemukan, maka dibentuklah sebuah klausa Prolog penandatangan(indonesia, menteri_luar_negeri)., yang menyatakan hubungan antar entitas. Pendekatan ini merepresentasikan ekstraksi semi-manual berbasis aturan eksplisit dan digunakan untuk membangun basis data simbolik dari dokumen hukum. Proses semacam ini sangat berguna pada tahap awal penyusunan dataset simbolik untuk melatih model *text-to-logic* berbasis LLM.

3.4.3 Membangun Dataset

Setelah proses pembersihan dan ekstraksi fakta dilakukan, langkah selanjutnya adalah menyusun dataset dalam format yang kompatibel dengan proses pelatihan model *language model*. Dataset disusun dalam bentuk *structured table* dengan tiga atribut utama, yaitu:

- question: kalimat interogatif dalam bahasa alami
- context: kumpulan fakta Prolog yang relevan sebagai dasar inferensi
- answer: klausa Prolog sebagai jawaban yang diturunkan dari konteks

Dataset ini digunakan dalam arsitektur *sequence-to-sequence*, di mana model menerima pasangan question dan context sebagai input dan menghasilkan answer sebagai output.

Berikut struktur format dari masing-masing baris data:

```
{
"question": "Siapa yang menandatangani perjanjian?",
"context": "penandatangan(indonesia, menteri_luar_negeri).
masa_berlaku(perjanjian123, 5_tahun).",
"answer": "penandatangan(indonesia, menteri_luar_negeri)."
}
```

Dataset disimpan dalam format CSV agar mudah dimuat dan diolah dengan pustaka pandas dan datasets dari HuggingFace. Setelah dibersihkan, dataset kemudian dikonversi ke dalam struktur DatasetDict yang berisi tiga bagian utama:

- train: untuk proses pelatihan model
- validation: untuk evaluasi selama pelatihan
- test: untuk pengujian akhir performa model

Pembagian ini memastikan setiap subset memiliki proporsi dan distribusi yang seimbang agar model dapat belajar secara efektif dan mampu menggeneralisasi data yang belum pernah dilihat sebelumnya.

Berikut pseudocode untuk menyusun dataset:

ALGORITMA 3.4.3 Membangun Dataset

INPUT: Dataset hasil pembersihan dan ekstraksi fakta (CSV)
OUTPUT: Dataset terstruktur dalam format DatasetDict (train, validation, test)

1. START
2. Baca file CSV menggunakan pustaka pandas
3. Pisahkan kolom question, context, dan answer sebagai atribut utama
4. Bagi dataset menjadi 80% training, 10% validation, dan 10% test
5. Konversi masing-masing bagian ke format Dataset dari HuggingFace
6. Gabungkan dalam struktur DatasetDict
7. Simpan dataset ke disk
8. END

Pseudocode yang disajikan pada subbab ini menggambarkan langkah-langkah sistematis dalam menyusun dataset hasil ekstraksi dan pembersihan agar siap digunakan untuk pelatihan model berbasis HuggingFace. Proses diawali dengan membaca file CSV yang telah dibersihkan dan memuat tiga kolom utama, yaitu question, context, dan answer.

Kolom-kolom ini merupakan representasi dari pertanyaan bahasa alami, fakta hukum dalam bentuk Prolog, serta jawaban yang berbentuk klausa logika Prolog. Selanjutnya, dataset dibagi menjadi tiga bagian dengan rasio 80% untuk data pelatihan (training), 10% untuk validasi (validation), dan 10% untuk pengujian (test). Pembagian ini dilakukan secara acak namun terkontrol untuk menjaga distribusi data tetap merata. Masing-masing subset kemudian dikonversi ke dalam format Dataset dari pustaka HuggingFace agar dapat diproses langsung oleh pipeline Trainer. Semua subset tersebut kemudian digabungkan dalam satu struktur DatasetDict dan disimpan ke disk, sehingga bisa diakses kembali saat proses pelatihan berlangsung tanpa perlu melakukan preprocessing ulang. Prosedur ini memastikan bahwa data telah tersusun rapi, memiliki format standar, dan kompatibel dengan kebutuhan pelatihan model *sequence-to-sequence*.

Implementasi kode Python:

```
from datasets import Dataset, DatasetDict
from sklearn.model_selection import train_test_split
import pandas as pd
# 1. Membaca dataset dari file CSV
raw_df = pd.read_csv("qa.csv")
# 2. Membagi dataset menjadi train (80%), validation (10%), test (10%)
train_df, temp_df = train_test_split(raw_df, test_size=0.2, random_state=42)
val_df, test_df = train_test_split(temp_df, test_size=0.5, random_state=42)
# 3. Konversi dataframe ke format HuggingFace Dataset
dataset = DatasetDict({
    "train": Dataset.from_pandas(train_df),
    "validation": Dataset.from_pandas(val_df),
    "test": Dataset.from_pandas(test_df)
})
# 4. Menghapus kolom bawaan pandas yang tidak relevan
for split in dataset:
    dataset[split] = dataset[split].remove_columns(['__index_level_0__'])
# 5. Menyimpan dataset ke disk untuk digunakan dalam pelatihan
dataset.save_to_disk("qa_dataset_cleaned")
```

Kode program tersebut digunakan untuk menyusun dataset terstruktur yang siap digunakan dalam proses pelatihan model. Langkah pertama dilakukan dengan membaca file

CSV yang berisi data hasil ekstraksi dan pembersihan menggunakan pustaka pandas. Setelah itu, data dibagi ke dalam tiga subset utama, yaitu 80% untuk data pelatihan (*training*), dan masing-masing 10% untuk data validasi (*validation*) serta pengujian (*test*). Pembagian ini dilakukan dengan fungsi train_test_split dari pustaka scikit-learn dengan parameter random_state agar hasilnya tetap konsisten setiap kali dijalankan.

Setelah terbagi, masing-masing subset dikonversi menjadi objek Dataset dari pustaka datasets milik HuggingFace. Selanjutnya, seluruh bagian tersebut dikumpulkan dalam struktur DatasetDict, yang memungkinkan model mengakses seluruh bagian dataset dengan mudah berdasarkan split-nya. Sebelum disimpan, setiap subset dibersihkan dari kolom teknis bawaan pandas seperti __index_level_0__ yang tidak diperlukan dalam proses pelatihan. Terakhir, dataset disimpan ke disk dalam format yang mendukung pemuatan ulang di tahap selanjutnya menggunakan fungsi save_to_disk. Dengan alur ini, dataset menjadi terstruktur, efisien, dan langsung kompatibel untuk digunakan dalam pipeline pelatihan model berbasis Transformers.

3.4.4 Tokenisasi Dataset

Setelah dataset dibersihkan dan disusun, langkah berikutnya adalah mengubah data teks menjadi representasi numerik agar dapat diproses oleh model bahasa. Tahap ini disebut tokenisasi. Tokenisasi dilakukan dengan menggabungkan nilai dari kolom context dan question menjadi sebuah prompt terstruktur, lalu menerapkannya ke tokenizer model T5.

Format prompt yang digunakan adalah sebagai berikut:

```
Tables:
<context>
Question:
<question>
Answer:
```

Struktur ini dirancang untuk memberi petunjuk eksplisit kepada model mengenai isi tabel dan jenis pertanyaan yang diajukan. Setelah prompt terbentuk, dilakukan proses tokenisasi terhadap prompt dan jawaban (answer) secara terpisah, menghasilkan dua komponen penting: input_ids (input ke model) dan labels (target output).

Berikut pseudocode proses tokenisasi:

ALGORITMA 3.4.4 Tokenisasi Dataset

INPUT: Dataset (question, context, answer)
OUTPUT: Tokenized dataset (input_ids, labels)

1. START
2. FOR setiap data dalam dataset DO
3. Buat prompt ← "Tables:\n" + context + "\nQuestion:\n" + question + "\nAnswer:\n"
4. Tokenisasi prompt menjadi input_ids
5. Tokenisasi jawaban (answer) menjadi labels
6. END FOR

7. SIMPAN hasil tokenisasi
8. END

Pseudocode tersebut menggambarkan proses sistematis dalam mengonversi data berbasis teks ke dalam representasi numerik melalui tokenisasi. Tahap pertama adalah membaca setiap entri dalam dataset. Kemudian, sistem menyusun prompt dengan menggabungkan tiga bagian: konteks dalam bentuk fakta logika Prolog (context), pertanyaan bahasa alami (question), dan penanda awal jawaban. Format prompt ini dirancang untuk membantu model memahami struktur dan maksud pertanyaan secara lebih eksplisit. Setelah prompt disusun, dilakukan tokenisasi terhadap prompt dan jawaban (answer) secara terpisah. Tokenisasi ini akan menghasilkan input_ids sebagai representasi input numerik yang masuk ke model dan labels sebagai target output. Proses ini diulang untuk setiap entri dalam dataset. Terakhir, hasil tokenisasi disimpan untuk digunakan dalam proses pelatihan model.

Implementasi kode Python:

```
from datasets import load_from_disk

def tokenize_function(example):
    start_prompt = "Tables:\n"
    middle_prompt = "\nQuestion:\n"
    end_prompt = "\nAnswer:\n"
    data_zip = zip(example['context'], example['question'])
    prompt = [start_prompt + context + middle_prompt + question + end_prompt for context, question in data_zip]
    example['input_ids'] = tokenizer(prompt, padding="max_length", truncation=True, return_tensors="pt").input_ids
    example['labels'] = tokenizer(example['answer'], padding="max_length", truncation=True, return_tensors="pt").input_ids
    return example

try:
    tokenized_datasets = load_from_disk("tokenized_datasets")
    print("Loaded Tokenized Dataset")
except:
    tokenized_datasets = dataset.map(tokenize_function, batched=True)
    tokenized_datasets = tokenized_datasets.remove_columns(['question', 'context', 'answer'])
    tokenized_datasets.save_to_disk("tokenized_datasets")
    print("Tokenized and Saved Dataset")
```

Potongan kode di atas digunakan untuk mengubah dataset dalam bentuk teks menjadi representasi numerik melalui proses tokenisasi menggunakan tokenizer model T5. Fungsi

tokenize_function() bertugas menyusun sebuah prompt dari kombinasi teks pada kolom context dan question, dengan format khusus:

```
Tables:\n<context>\nQuestion:\n<question>\nAnswer:
```

Prompt ini dirancang untuk memberikan struktur eksplisit kepada model agar dapat memahami konteks dan pertanyaan secara lebih akurat. Fungsi ini kemudian menggunakan tokenizer dari Hugging Face untuk mengonversi prompt menjadi input_ids dan kolom answer menjadi labels, yang keduanya merupakan format input dan output numerik bagi model. Selanjutnya, program mencoba memuat dataset tokenisasi yang telah tersimpan sebelumnya menggunakan load_from_disk. Jika belum tersedia, maka dataset akan diproses melalui fungsi map() yang menerapkan tokenisasi secara batch, lalu menghapus kolom asli (question, context, dan answer) karena sudah direpresentasikan dalam bentuk token. Dataset hasil tokenisasi ini kemudian disimpan ke disk dengan nama tokenized_datasets. Proses ini memastikan bahwa seluruh data siap digunakan oleh model T5 tanpa perlu mengulangi tokenisasi di tahap berikutnya.

Tokenisasi dilakukan terhadap tiga subset data: training, validation, dan test. Setiap proses tokenisasi menampilkan progress bar, jumlah data, kecepatan pemrosesan, dan waktu eksekusi. Setelah selesai, dataset hasil tokenisasi disimpan ke dalam disk, sehingga proses ini tidak perlu diulang saat pelatihan model berikutnya.Dengan tokenisasi ini, data teks mentah berhasil diubah menjadi format numerik yang siap diproses oleh model T5 untuk pelatihan dan evaluasi selanjutnya.

3.5 Training LLM

3.5.1 Pemilihan Model

Dalam penelitian ini, model yang digunakan adalah T5 (*Text-to-Text Transfer Transformer)* yang dikembangkan oleh Google. T5 merupakan arsitektur *sequence-to-sequence* berbasis Transformer yang dirancang untuk menangani berbagai tugas NLP dengan cara mengubah semuanya menjadi format teks ke teks (text-to-text). Kemampuan T5 untuk menangani input dan output dalam bentuk string membuatnya sangat fleksibel, termasuk untuk tugas penerjemahan dari bahasa alami ke logika formal seperti Prolog.

Model yang dipilih secara spesifik adalah t5-small, yaitu varian ringan dari T5 dengan jumlah parameter yang lebih kecil (sekitar 60 juta parameter), namun tetap mampu memberikan hasil yang kompetitif dalam tugas-tugas NLP. Pemilihan varian ini mempertimbangkan keterbatasan sumber daya komputasi yang tersedia, serta kebutuhan akan efisiensi dalam proses pelatihan dan inferensi.

Selain efisiensi, keunggulan T5-small terletak pada kemampuannya untuk memahami konteks secara mendalam karena arsitekturnya yang mendukung mekanisme self-attention dua arah. Ini sangat penting dalam konteks penelitian ini, di mana model harus memahami hubungan antara fakta-fakta logika Prolog (context) dan pertanyaan hukum (question) untuk menghasilkan jawaban yang valid dalam bentuk klausa logika (answer).

Model T5 juga sudah tersedia dalam repositori HuggingFace Transformers, sehingga proses pemanggilan, fine-tuning, dan penyimpanan model dapat dilakukan secara terintegrasi dan efisien menggunakan API Trainer. Hal ini sangat mempercepat proses eksperimen dan replikasi, serta memungkinkan integrasi langsung dengan pipeline pelatihan dan evaluasi.

3.5.2 Konfigurasi Hyperparameter

Pada tahap pelatihan model, pemilihan *hyperparameter* memainkan peranan penting dalam menentukan performa akhir model. *Hyperparameter* adalah parameter yang ditentukan sebelum pelatihan dimulai dan tidak dipelajari langsung oleh model. Pemilihan konfigurasi yang tepat dapat mempercepat konvergensi model, meningkatkan akurasi, serta mencegah masalah *overfitting* atau *underfitting*. Berikut adalah hyperparameter yang digunakan dalam pelatihan model T5 pada penelitian ini:

- **Learning Rate**: 5e-5
    - **Penjelasan**: Learning rate menentukan seberapa besar langkah pembaruan bobot pada setiap iterasi. Nilai yang terlalu tinggi bisa membuat model gagal konvergen, sementara nilai yang terlalu rendah memperlambat pelatihan. Nilai 5e-5 dipilih karena memberikan keseimbangan yang baik antara kecepatan dan stabilitas pembaruan bobot.
- **Batch Size**: 8
    - **Penjelasan**: Batch size adalah jumlah data yang diproses dalam satu iterasi pelatihan. Ukuran batch kecil dipilih untuk menyesuaikan dengan kapasitas GPU, tetapi tetap cukup besar untuk menjaga stabilitas pembaruan bobot.
- **Epochs**: 10
    - **Penjelasan**: Epochs menunjukkan jumlah kali dataset diproses oleh model selama pelatihan. Nilai 10 epoch dipilih karena memberikan performa stabil tanpa risiko overfitting atau underfitting.
- **Weight Decay**: 0.01

- **Penjelasan**: Weight decay digunakan untuk **regularisasi**, yang bertujuan mencegah overfitting dengan mengurangi kompleksitas model. Nilai 0.01 membantu menjaga generalisasi model.

- **Evaluation Strategy**: epoch
  - **Penjelasan**: Evaluasi dilakukan pada akhir setiap epoch untuk memantau performa model. Evaluasi yang berkala membantu menghentikan pelatihan lebih awal jika model telah mencapai performa optimal.
- **Save Strategy**: epoch
  - **Penjelasan**: Model disimpan setiap akhir epoch. Hal ini memastikan versi terbaik model disimpan jika pelatihan terhenti tiba-tiba.
- **Logging Steps**: 10
  - **Penjelasan**: Logging dilakukan setiap 10 langkah pelatihan untuk memantau kemajuan dan menganalisis metrik performa model secara real-time

3.5.3 Fine-Tuning Model

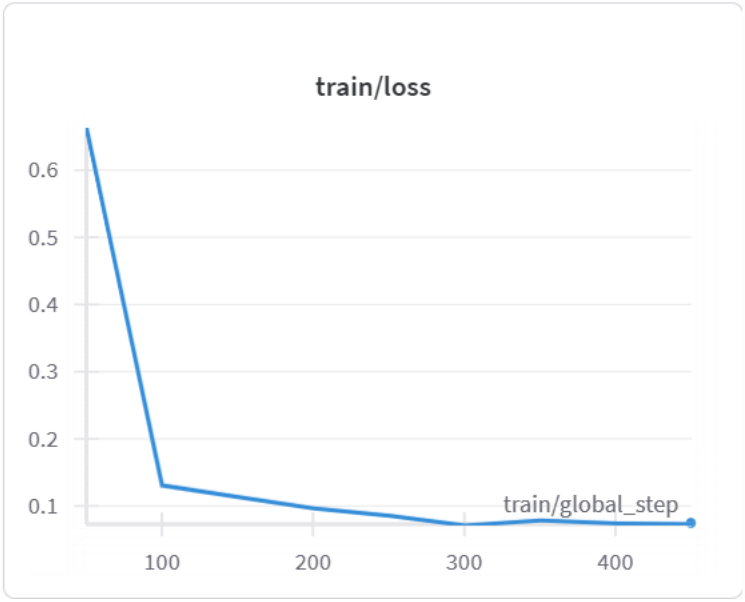
Pada tahap *fine-tuning*, model yang telah dilatih dengan dataset umum diadaptasi lebih lanjut menggunakan dataset yang lebih spesifik untuk meningkatkan akurasi dan kinerja pada tugas tertentu. Fine-tuning adalah langkah penting setelah model pra-pelatihan dilakukan, terutama dalam konteks *transfer learning,* di mana model yang sudah dilatih sebelumnya disesuaikan dengan tugas yang lebih khusus.

Fine-tuning bertujuan untuk menyesuaikan bobot model dengan dataset spesifik yang lebih relevan dengan tugas yang akan dilakukan, dalam hal ini penerjemahan bahasa alami ke dalam klausa logika formal (seperti Prolog). Model yang sudah dipra-latih pada data besar dan umum (seperti T5) kemudian dipertajam untuk tugas yang lebih spesifik.

Evaluasi Metrik Selama Fine-Tuning

Selama **fine-tuning**, grafik-grafik berikut digunakan untuk memantau performa model dan mengevaluasi apakah model melakukan penyesuaian dengan benar:

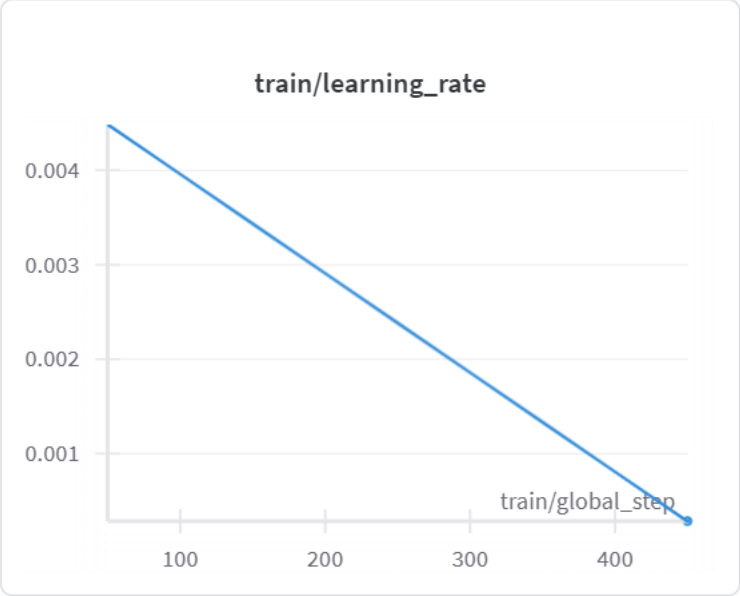
- **Grafik Train Loss**



Gambar 3.2 train/loss

Grafik ini menunjukkan *train loss* selama pelatihan, di mana *train loss* berkurang tajam di awal, menandakan model belajar cepat. Penurunan tajam ini terjadi karena model mulai dengan prediksi yang jauh dari target. Seiring waktu, *train loss* melambat menuju nilai stabil, menunjukkan model mendekati konvergensi dan menemukan solusi optimal. Penurunan stabil ini mencerminkan bahwa model belajar dengan baik, dengan pengaturan *learning rate* dan *epoch* yang membantu mengoptimalkan pelatihan

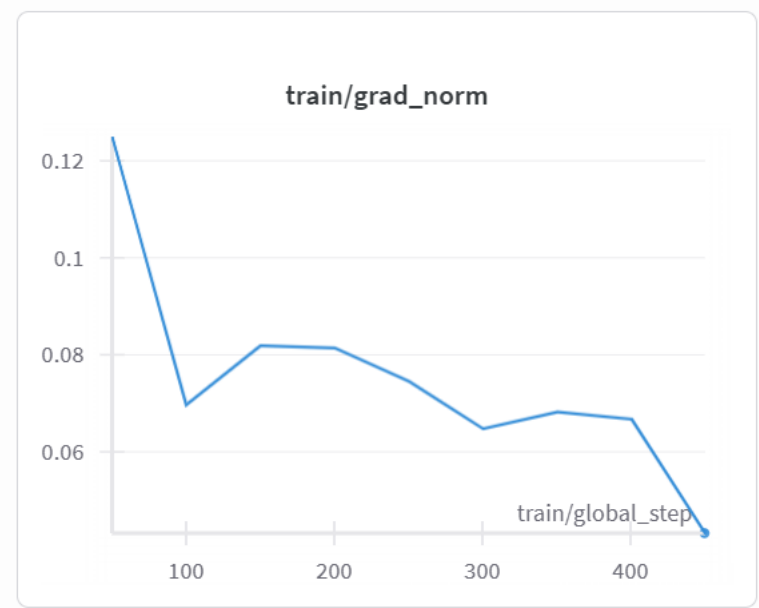
- **Grafik Learning Rate**



Gambar 3.3 learning_rate

Grafik ini menunjukkan bahwa *learning rate* dimulai lebih tinggi di awal pelatihan dan menurun secara linier seiring waktu. Penurunan ini, yang dikenal sebagai *learning rate decay*, memungkinkan model untuk membuat pembaruan lebih besar di awal pelatihan dan lebih halus mendekati konvergensi, mencegah model "melompati" solusi optimal.

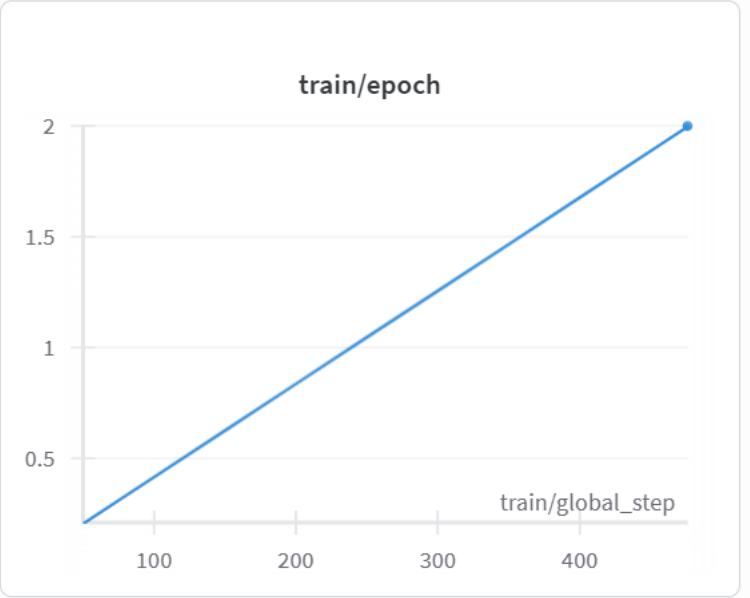
- **Grafik Gradient Norm**



Gambar 3.4 grad_norm

Grafik *train/grad_norm* menunjukkan bagaimana stabilitas gradien berubah selama pelatihan. Fluktuasi awal diikuti oleh penurunan stabil mengindikasikan bahwa model berhasil belajar secara efektif tanpa mengalami masalah *exploding gradients*. Penurunan gradien norm ini sangat penting untuk menjaga stabilitas pelatihan dan memastikan model dapat konvergen dengan baik.

- **Grafik Epochs**



Gambar 3.5 train/epoch

Grafik *train/epoch* memberikan gambaran jelas tentang kemajuan pelatihan. Setiap epoch mewakili satu iterasi penuh di mana model memproses seluruh dataset, dan grafik ini menunjukkan bahwa proses pelatihan berlangsung stabil, dengan model terus memproses data hingga mencapai epoch yang ditargetkan.

Kesimpulan Fine-Tuning

Fine-tuning memungkinkan model untuk menyesuaikan diri dengan data yang lebih spesifik dan menghasilkan solusi yang lebih akurat untuk tugas tertentu. Dengan strategi *learning rate decay* dan pembekuan beberapa lapisan, fine-tuning dapat dilakukan dengan efisien, memungkinkan model untuk mendapatkan hasil yang optimal tanpa memerlukan pelatihan ulang dari awal. Proses ini memastikan bahwa model tidak hanya belajar dari dataset umum, tetapi juga mengadaptasi dirinya dengan baik terhadap konteks spesifik yang lebih relevan.

3.6 Evaluasi

Evaluasi merupakan langkah penting dalam proses pengembangan model untuk menilai kinerja model setelah melalui pelatihan dan fine-tuning. Tujuan dari evaluasi adalah untuk memastikan bahwa model yang telah dilatih dapat bekerja dengan baik pada data baru, mengukur akurasi hasil yang diperoleh, serta memastikan bahwa model dapat menggeneralisasi dengan benar pada data yang belum pernah dilihat sebelumnya.

Evaluasi juga bertujuan untuk memberikan wawasan mengenai apakah model mengalami overfitting (terlalu menyesuaikan dengan data pelatihan) atau underfitting (tidak cukup belajar dari data pelatihan). Selain itu, evaluasi membantu untuk memverifikasi apakah model mampu mencapai tujuan yang diinginkan dengan menggunakan metrik yang telah ditentukan sebelumnya.

Langkah-Langkah Evaluasi:

1. Evaluasi Selama Pelatihan:
    - Memantau train loss, gradient norm, dan learning rate untuk memastikan bahwa model belajar secara efektif dan stabil selama pelatihan.
    - Menilai apakah model mengalami overfitting atau underfitting berdasarkan perubahan metrik yang diamati.
2. Evaluasi Setelah Fine-Tuning:
    - Setelah fine-tuning dilakukan, evaluasi dilakukan untuk memverifikasi apakah model yang telah disesuaikan dengan dataset yang lebih spesifik dapat memberikan hasil yang lebih baik dan lebih sesuai dengan tujuan penelitian.
    - Metrik seperti Exact Match dan Execution Match digunakan untuk menilai kualitas output yang dihasilkan oleh model, baik secara sintaksis maupun semantik.

3. Evaluasi pada Data Pengujian:
    - Setelah pelatihan selesai, model diuji pada data pengujian (test set) untuk memastikan bahwa model tidak hanya belajar dari data pelatihan, tetapi juga dapat menggeneralisasi dengan baik pada data yang belum pernah dilihat sebelumnya.

3.6.1 Evaluasi Training Model

Evaluasi Training Model dilakukan selama proses pelatihan untuk memastikan bahwa model belajar dengan baik dan melakukan pembaruan bobot secara stabil. Evaluasi ini dilakukan secara berkala untuk memantau kinerja model dan mengidentifikasi masalah seperti *overfitting* atau *underfitting* sejak dini.

Metrik yang digunakan untuk evaluasi selama pelatihan:

- Train Loss: Mengukur bagaimana kesalahan model berkurang selama pelatihan. Penurunan train loss yang stabil mengindikasikan bahwa model belajar secara efektif.
- Gradient Norm: Digunakan untuk memantau apakah model mengalami masalah exploding gradients, yang dapat merusak pembaruan bobot dan menyebabkan ketidakstabilan dalam pelatihan.
- Learning Rate: Mengukur bagaimana learning rate decay diterapkan selama pelatihan. Hal ini membantu model melakukan pembaruan bobot yang lebih besar pada awal pelatihan dan pembaruan yang lebih halus mendekati konvergensi.

Evaluasi selama pelatihan bertujuan untuk memastikan bahwa model berada di jalur yang benar dan tidak mengalami masalah stabilitas dalam pembaruan bobot atau pelatihan yang terlalu cepat atau lambat.

3.6.2 Metrik Evaluasi

Setelah model selesai dilatih dan di-fine-tune, metrik evaluasi digunakan untuk menilai performa akhir model. Metrik evaluasi ini sangat penting untuk menilai seberapa baik model dapat menghasilkan output yang sesuai dengan tujuan yang diinginkan, yaitu menghasilkan klausa logika formal yang valid dan dapat dieksekusi dalam sistem berbasis logika.

Beberapa metrik evaluasi yang digunakan adalah:

- Exact Match (EM):
    - Metrik ini mengukur seberapa tepat jawaban yang dihasilkan model dibandingkan dengan jawaban yang diinginkan. EM akan bernilai 1 jika jawaban model persis sama dengan ground truth, dan 0 jika tidak.
    - Contoh: Jika model menghasilkan klausa logika yang tepat, maka EM akan bernilai 1, menunjukkan bahwa model berhasil menghasilkan output yang akurat dan tepat.
- Execution Match (XM):
    - Metrik ini mengukur kemampuan model untuk menghasilkan klausa logika yang dapat dieksekusi dengan benar. Dalam sistem seperti Prolog, klausa tersebut harus dapat dijalankan dan menghasilkan hasil yang valid.
    - Contoh: Jika model menghasilkan klausa yang secara semantik benar dan dapat dieksekusi dengan benar dalam sistem logika, maka XM akan bernilai 1.

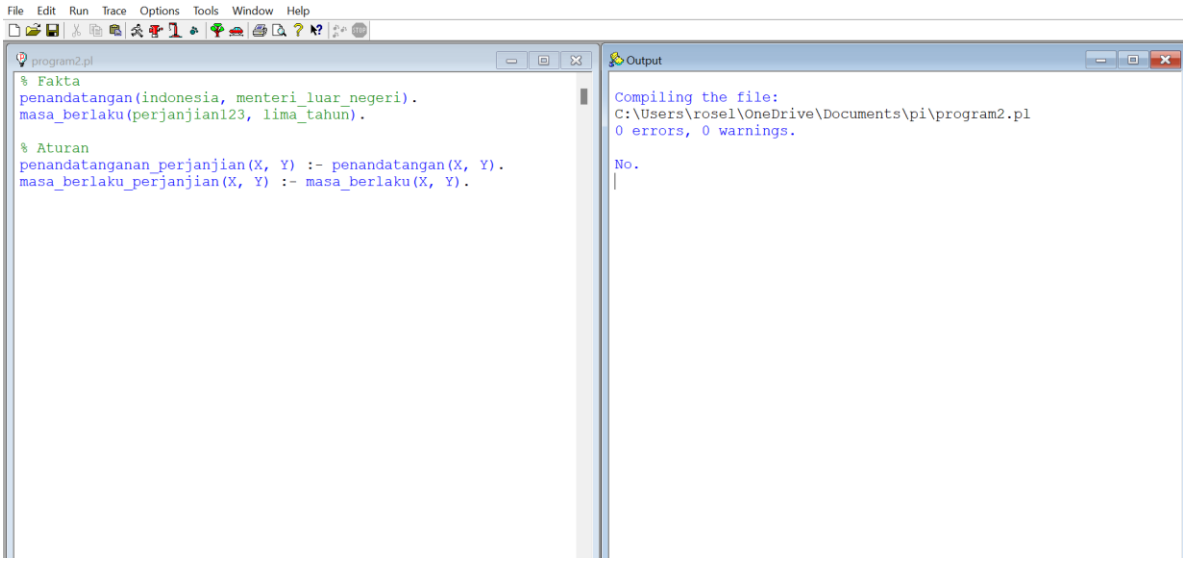
Metrik Exact Match dan Execution Match digunakan untuk mengevaluasi seberapa baik model menghasilkan klausa logika yang benar secara sintaksis (EM) dan semantik (XM), yang merupakan tujuan utama dari penelitian ini.

3.7 Deployment

Setelah evaluasi pada tahap *Deployment* dalam siklus hidup model, model yang telah dilatih dan dievaluasi diterapkan dalam aplikasi dunia nyata agar bisa digunakan untuk memberikan manfaat langsung. *Deployment* mencakup serangkaian langkah yang melibatkan penerapan model ke dalam sistem yang dapat diakses oleh pengguna atau aplikasi lain.

3.7.1 Menggunakan Prolog dalam Aplikasi QA:

- **Prolog** digunakan untuk mendefinisikan pengetahuan berbasis **fakta** dan **aturan** yang berkaitan dengan penandatanganan perjanjian. Misalnya, fakta seperti **penandatangan(indonesia, menteri_luar_negeri)** menunjukkan siapa yang menandatangani perjanjian, dan **masa_berlaku(perjanjian123, lima_tahun)** menunjukkan masa berlaku perjanjian.
- Sebagai contoh, jika pertanyaan yang diajukan adalah **"Siapa yang menandatangani perjanjian?"**, Prolog akan mencari dan mengeksekusi klausa **penandatanganan_perjanjian(X, Y)**, yang menghubungkan fakta-fakta tentang siapa yang menandatangani perjanjian.



Gambar 3.7 Contoh representasi fakta dan aturan dalam bahasa Prolog menggunakan Strawberry Prolog

Pada gambar yang Anda unggah:

1. Fakta :
    - **penandatangan(indonesia, menteri_luar_negeri)**: Menyatakan bahwa Indonesia menandatangani perjanjian dengan Menteri Luar Negeri.
    - **masa_berlaku(perjanjian123, lima_tahun)**: Menyatakan bahwa perjanjian123 berlaku selama 5 tahun.
2. Aturan :
    - **penandatanganan_perjanjian(X, Y)**: Jika X menandatangani perjanjian dengan Y, maka X adalah penandatangan perjanjian.
    - **masa_berlaku_perjanjian(X, Y)**: Jika X adalah perjanjian dan Y adalah durasi, maka X memiliki masa berlaku Y.

Dengan menggunakan aturan ini, ketika pertanyaan **"Siapa yang menandatangani perjanjian?"** diajukan, Prolog akan memeriksa **penandatanganan_perjanjian** dan memberikan hasil seperti **"indonesia, menteri_luar_negeri"**.

BAB IV
PENUTUP

4.1 Kesimpulan

Dari hasil penelitian ini, dapat disimpulkan bahwa pendekatan berbasis Transformer, terutama model T5, terbukti sangat efektif dalam mengonversi bahasa alami menjadi klausa logika formal seperti yang digunakan dalam Prolog. Model yang dikembangkan berhasil menghasilkan klausa logika yang akurat secara sintaksis (Exact Match/EM) dan sah secara semantik (Execution Match/XM), yang merupakan tujuan utama dari penelitian ini. Keberhasilan model dalam tugas tersebut menunjukkan potensi besar dalam aplikasi sistem tanya jawab (QA) berbasis logika formal. Model T5 yang telah di-fine-tune menunjukkan kemampuan luar biasa dalam memahami hubungan semantik dan mentransformasikan kalimat bahasa alami ke dalam format logika formal dengan tingkat akurasi yang tinggi.

Prolog, yang digunakan sebagai sistem untuk menguji kevalidan klausa logika, juga terbukti sangat efisien dalam memverifikasi klausa yang dihasilkan oleh model. Meskipun hasilnya menjanjikan, masalah halusinasi tetap menjadi tantangan utama dalam model Large Language Models (LLM), terutama dalam konteks penerjemahan bahasa alami yang sangat bergantung pada data pelatihan dan teknik prompting. Penerapan teknik neuro-symbolic, dengan mengintegrasikan Transformer dan Prolog, berhasil mengurangi kesalahan logika yang disebabkan oleh halusinasi dan meningkatkan akurasi semantik dalam penerjemahan ke klausa logika formal. Secara keseluruhan, penelitian ini membuka peluang besar untuk penggunaan pendekatan neuro-symbolic berbasis Transformer dalam berbagai aplikasi praktis, seperti legaltech, sistem tanya jawab berbasis pengetahuan, dan berbagai aplikasi berbasis AI simbolik.

4.2 Saran

Meskipun penelitian ini berhasil mencapai tujuan yang diinginkan, masih ada beberapa area yang bisa ditingkatkan untuk pengembangan lebih lanjut. Salah satunya adalah masalah halusinasi yang tetap menjadi tantangan dalam model *Large Language Models (LLM)*. Untuk mengatasi hal ini, penerapan pendekatan *Retrieval-Augmented Generation (RAG)* yang menggabungkan model generatif dengan pencarian informasi dari sumber eksternal dapat membantu meningkatkan akurasi dan mengurangi ketergantungan pada data yang terbatas. Selain itu, eksperimen dengan dataset yang lebih besar dan lebih beragam sangat dianjurkan untuk mengukur kemampuan model dalam menangani variasi bahasa alami yang lebih kompleks. Pengembangan model ini juga bisa diperluas untuk mendukung penerjemahan multibahasa, yang akan membuat model ini lebih fleksibel dan dapat diterapkan pada sistem berbasis logika di berbagai negara.

Penerapan model pada platform produksi perlu memperhatikan optimasi model untuk latency rendah dan kecepatan eksekusi yang tinggi, yang bisa dicapai dengan teknik seperti *quantization* atau *pruning*. Selain itu, integrasi model ini dengan sistem berbasis pengetahuan yang lebih besar, seperti manajemen dokumen hukum atau sistem berbasis aturan, juga bisa memperluas aplikasi model. Model ini juga dapat diuji pada tugas yang lebih kompleks dalam pemrosesan bahasa alami, seperti terjemahan otomatis atau analisis sentimen, yang akan menguji kemampuannya dalam mengatasi analisis semantik yang lebih mendalam. Dengan langkah-langkah tersebut, penelitian ini membuka peluang lebih besar dalam pengembangan model AI simbolik yang dapat digunakan di berbagai bidang dan aplikasi praktis di masa depan.

DAFTAR PUSTAKA

Baral, C. (2003). *Knowledge Representation, Reasoning, and Declarative Problem Solving*. Cambridge University Press.

Besold, T. R., Garcez, A. d., Bader, S., Bowman, H., Domingos, P., Hitzler, P., ... & Lamb, L. C. (2017).
Neural-Symbolic Learning and Reasoning: A Survey and Interpretation.

Bommasani, R., et al. (2021). *On the Opportunities and Risks of Foundation Models*. Communications of the ACM, 64(3), 62-71.

Brown, T. B., et al. (2020). *Language models are few-shot learners*. In *Proceedings of NeurIPS 2020* (pp. 1-10). Curran Associates, Inc.

Buchanan, B. G., & Shortliffe, E. H. (1984). *Rule-based expert systems: The MYCIN experiments of the Stanford Heuristic Programming Project*. Addison-Wesley.

Buntine, W. (1994). *Operations for learning with probabilistic graphical models*. *Journal of Artificial Intelligence Research*, 2, 159-225.

Cambria, E., White, B. (2014). Jumping NLP Curves: A Review of Natural Language Processing Research. *IEEE Computational Intelligence Magazine*, 9(2), 48–57.

Chen, X., Liu, C., & Song, D. (2020).
Neural Symbolic Machines: Learning Semantic Parsers on Freebase with Weak Supervision.

Chollet, F. (2018). *Deep Learning with Python*. Manning Publications.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.*

Harris, Z. (1952). *Discourse Analysis*. *Language*, 28(1), 1-30.

Jurafsky, D., & Martin, J. H. (2023). *Speech and Language Processing* (3rd ed.). Pearson Education.

Lewis, M., et al. (2020). *Retrieval-augmented generation for knowledge-intensive NLP tasks*. In *Proceedings of NeurIPS 2020* (pp. 1003-1014). Curran Associates, Inc.

Lindsay, R. K. (1990). *Artificial Intelligence: Foundations of Computational Agents*. Addison-Wesley.

Marcus, G. (2020). *The next decade in AI: Four steps to safer AI*. *Nature Machine Intelligence*, 2(9), 500-507.

Minsky, M. (1961). *Steps Toward Artificial Intelligence*. *Proceedings of the IRE*, 49(1), 8-30.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill.

Ouyang, L., et al. (2022). *Training language models to follow instructions with human feedback*. In *Proceedings of the 2022 Conference on Neural Information Processing Systems (NeurIPS 2022)*.

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). *Improving Language Understanding by Generative Pre-Training (GPT-1).*

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... & Liu, P. J. (2020). *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer.*

Russell, S., & Norvig, P. (2016). *Artificial Intelligence: A Modern Approach* (3rd ed.). Pearson Education.

Shannon, C. E. (1948). *A Mathematical Theory of Communication*. *The Bell System Technical Journal*, 27(3), 379-423.

Shieber, S. M. (1986). *An Introduction to Unification-based Approaches to Grammar*. *CSLI Lecture Notes* (Vol. 4). Center for the Study of Language and Information.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (2nd ed.). MIT Press.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M. A., Lacroix, T., ... & Jegou, H. (2023).
LLaMA: Open and Efficient Foundation Language Models.

Turing, A. M. (1936). *On Computable Numbers, with an Application to the Entscheidungsproblem*. *Proceedings of the London Mathematical Society*, 42(1), 230-265.

Vaswani, A., et al. (2017). *Attention is all you need*. In *Proceedings of NeurIPS 2017* (pp. 5998-6008). Curran Associates, Inc.

Wang, J., et al. (2022). *ReAct: Synergizing Reasoning and Acting for Complex Decision-Making*. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing* (pp. 102-113). Association for Computational Linguistics.

Winograd, T. (1972). *Understanding Natural Language*. *Cognitive Psychology*, 3(1), 1-191.

Xu, Z., Jain, S., & Kankanhalli, M. (2024). *Hallucinations in Large Language Models: A review*. *Journal of AI Research*, 12(1), 1-27.

LAMPIRAN LISTING PROGRAM

```
!pip install datasets

!pip install evaluate


from sklearn.model_selection import train_test_split

from datasets import Dataset, DatasetDict, load_dataset, interleave_datasets,
load_from_disk

from transformers import AutoModelForSeq2SeqLM, AutoTokenizer, GenerationConfig,
TrainingArguments, Trainer

import torch

import time

import evaluate

import pandas as pd

import numpy as np


import warnings

warnings.filterwarnings("ignore")


torch.cuda.is_available()


model_name='t5-small'


tokenizer = AutoTokenizer.from_pretrained(model_name)
```

```
original_model = AutoModelForSeq2SeqLM.from_pretrained(model_name,
torch_dtype=torch.bfloat16)

original_model = original_model.to('cuda')


from google.colab import files

uploaded = files.upload()


import pandas as pd


df = pd.read_csv("qa.csv")

print(df.head())

print(df.columns)


from datasets import Dataset, DatasetDict

import pandas as pd

from sklearn.model_selection import train_test_split
```

```
# 1. Load CSV
df = pd.read_csv("qa.csv")

# 2. Optional: Strip whitespace dari kolom
df["question"] = df["question"].str.strip()
df["context"] = df["context"].str.strip()
df["answer"] = df["answer"].str.strip()

# 3. Split data: 80% train, 10% validation, 10% test
train_df, temp_df = train_test_split(df, test_size=0.2, random_state=42)
val_df, test_df = train_test_split(temp_df, test_size=0.5, random_state=42)

# 4. Convert ke HuggingFace Dataset
dataset = DatasetDict({
    "train": Dataset.from_pandas(train_df),
    "validation": Dataset.from_pandas(val_df),
    "test": Dataset.from_pandas(test_df)
})

# 5. Save ke disk (opsional, supaya bisa di-load lagi nanti)
dataset.save_to_disk("qa_dataset")
```

```
# 6. Print summary
print(dataset)


from datasets import DatasetDict


# Misalnya dataset kamu bernama `dataset`
dataset = DatasetDict({
    'train': dataset['train'].remove_columns(['__index_level_0__']),
    'validation': dataset['validation'].remove_columns(['__index_level_0__']),
    'test': dataset['test'].remove_columns(['__index_level_0__']),
})
# Tokenization
from transformers import T5Tokenizer


tokenizer = T5Tokenizer.from_pretrained('t5-small')


def tokenize_function(example):
    return tokenizer(example['question'], example['context'], truncation=True,
padding="max_length")


tokenized_datasets = dataset.map(tokenize_function, batched=True)
```

```
# Model Setup
from transformers import T5ForConditionalGeneration


model = T5ForConditionalGeneration.from_pretrained('t5-small')


# Training
training_args = TrainingArguments(
    output_dir="./results",
    evaluation_strategy="epoch",
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    logging_dir="./logs",
    num_train_epochs=3
)
```

```
# Evaluasi Model
from sklearn.metrics import accuracy_score, f1_score, recall_score
from datasets import load_metric

accuracy = accuracy_score(true_labels, predicted_labels)
f1_binary = f1_score(true_labels, predicted_labels)
recall = recall_score(true_labels, predicted_labels)

print("==== EVALUASI MODEL ====")
print(f"Accuracy: {accuracy:.2%}")
print(f"F1 Score: {f1_binary:.2%}")
print(f"Recall: {recall:.2%}")
# Execute Evaluation Metrics
!pip install pyswip

!apt-get install swi-prolog

from pyswip import Prolog

def exact_match(predictions, references):
    em_score = sum([1 if pred == ref else 0 for pred, ref in zip(predictions, references)])
    return em_score / len(references)
```

```
def execution_match(predictions, references):
    prolog = Prolog()
    match_score = 0
    for pred, ref in zip(predictions, references):
        prolog.assertz(f"{pred}")
        result = list(prolog.query(ref))
        if result:
            match_score += 1
        prolog.retractall(f"{pred}")
    return match_score / len(references)


predictions = ["penandatangan(indonesia, menteri_luar_negeri)"]
references = ["penandatangan(indonesia, menteri_luar_negeri)"]


em_score = exact_match(predictions, references)
xm_score = execution_match(predictions, references)


print(f"Exact Match (EM) Score: {em_score}")
print(f"Execution Match (XM) Score: {xm_score}")
```