



UNIVERSITAS INDONESIA

**KEMAMPUAN LLM DALAM LOGICAL INFERENCE PADA
DATASET BERBAHASA INDONESIA**

SKRIPSI

**MIKHAEL DEO BARLI
1906350572**

**FAKULTAS ILMU KOMPUTER
PROGRAM STUDI ILMU KOMPUTER
DEPOK
BULAN TAHUN**



UNIVERSITAS INDONESIA

**KEMAMPUAN LLM DALAM LOGICAL INFERENCE PADA
DATASET BERBAHASA INDONESIA**

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar
Gelar Jurusan Anda

**MIKHAEL DEO BARLI
1906350572**

**FAKULTAS ILMU KOMPUTER
PROGRAM STUDI ILMU KOMPUTER
DEPOK
BULAN TAHUN**

HALAMAN PERNYATAAN ORISINALITAS

**Skripsi ini adalah hasil karya saya sendiri,
dan semua sumber baik yang dikutip maupun dirujuk
telah saya nyatakan dengan benar.**

Nama : Mikhael Deo Barli

NPM : 1906350572

Tanda Tangan :

Tanggal : Tanggal Bulan Tahun

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh :

Nama : Mikhael Deo Barli

NPM : 1906350572

Program Studi : Ilmu Komputer

Judul Skripsi : Kemampuan LLM dalam Logical Inference pada
Dataset Berbahasa Indonesia

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana pada Program Studi Ilmu Komputer, Fakultas Ilmu Komputer, Universitas Indonesia.

DEWAN PENGUJI

Pembimbing 1 : Pembimbing Pertama Anda ()

Penguji 1 : Penguji Pertama Anda ()

Penguji 2 : Penguji Kedua Anda ()

Ditetapkan di : Depok

Tanggal : Tanggal Bulan Tahun

KATA PENGANTAR

Template ini disediakan untuk orang-orang yang berencana menggunakan L^AT_EX untuk membuat dokumen tugas akhir.

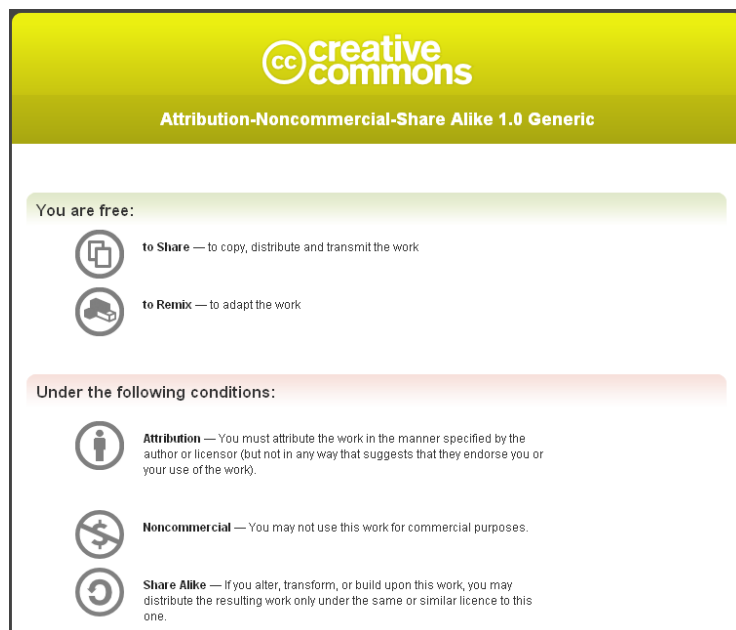
@todo

Silakan ganti pesan ini dengan pendahuluan kata pengantar Anda.

Ucapan Terima Kasih:

1. Pembimbing.
2. Dosen.
3. Instansi.
4. Orang tua.
5. Sahabat.
6. Teman.

Penulis menyadari bahwa laporan Skripsi ini masih jauh dari sempurna. Oleh karena itu, apabila terdapat kesalahan atau kekurangan dalam laporan ini, Penulis memohon agar kritik dan saran bisa disampaikan langsung melalui *e-mail* emailanda@mail.id.



Creative Common License 1.0 Generic

Terkait template ini, gambar lisensi di atas diambil dari http://creativecommons.org/licenses/by-nc-sa/1.0/deed.en_CA. Jika ingin mengetahui lebih lengkap mengenai *Creative Common License 1.0 Generic*, silahkan buka <http://creativecommons.org/licenses/by-nc-sa/1.0/legalcode>. Seluruh dokumen yang dibuat dengan menggunakan template ini sepenuhnya menjadi hak milik pembuat dokumen dan bebas didistribusikan sesuai dengan keperluan masing-masing. Lisensi hanya berlaku jika ada orang yang membuat template baru dengan menggunakan template ini sebagai dasarnya.

Penyusun template ingin berterima kasih kepada Andreas Febrian, Lia Sadita, Fahrurrozi Rahman, Andre Tampubolon, dan Erik Dominikus atas kontribusinya dalam template yang menjadi pendahulu template ini. Penyusun template juga ingin mengucapkan terima kasih kepada Azhar Kurnia atas kontribusinya dalam template yang menjadi pendahulu template ini.

Semoga template ini dapat membantu orang-orang yang ingin mencoba menggunakan \LaTeX . Semoga template ini juga tidak berhenti disini dengan ada kontribusi dari para penggunanya. Jika Anda memiliki perubahan yang dirasa penting untuk disertakan dalam template, silakan lakukan *fork* repositori Git template ini di <https://gitlab.com/ichlaffterlalu/latex-skripsi-ui-2017>, lalu lakukan *merge request* perubahan Anda terhadap *branch* master. Kami berharap agar *template* ini dapat terus diperbarui mengikuti perubahan ketentuan dari pihak Rektorat Universitas Indonesia, dan hal itu tidak mungkin terjadi tanpa kontribusi dari teman-teman sekalian.

Depok, Tanggal Bulan Tahun

Mikhael Deo Barli

**HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI
TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS**

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

Nama : Mikhael Deo Barli
NPM : 1906350572
Program Studi : Ilmu Komputer
Jenis Karya : Skripsi

demikian pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif** (*Non-exclusive Royalty Free Right*) atas karya ilmiah saya yang berjudul:

Kemampuan LLM dalam Logical Inference pada Dataset Berbahasa Indonesia

berserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan memublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok
Pada tanggal : Tanggal Bulan Tahun
Yang menyatakan

(Mikhael Deo Barli)

ABSTRAK

Nama : Mikhael Deo Barli
Program Studi : Ilmu Komputer
Judul : Kemampuan LLM dalam Logical Inference pada Dataset
Berbahasa Indonesia
Pembimbing : Pembimbing Pertama Anda

Isi abstrak.

Kata kunci:

Keyword satu, kata kunci dua

ABSTRACT

Name : Mikhael Deo Barli
Study Program : Computer Science
Title : The Capability of LLM in Logical Inference on Indonesian Language Dataset
Counselor : Pembimbing Pertama Anda

Abstract content.

Key words:

Keyword one, keyword two

DAFTAR ISI

HALAMAN JUDUL	i
HALAMAN ORISINALITAS	ii
LEMBAR PENGESAHAN	iii
KATA PENGANTAR	iv
LEMBAR PERSETUJUAN KARYA ILMIAH	vi
ABSTRAK	vii
DAFTAR ISI	ix
DAFTAR GAMBAR	xi
DAFTAR TABEL	xii
DAFTAR KODE PROGRAM	xii
DAFTAR LAMPIRAN	xiii
1. Pendahuluan	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	1
1.3 Tujuan Penelitian	2
1.4 Batasan Penelitian	2
1.5 Manfaat Penelitian	3
1.6 Posisi Penelitian	3
1.7 Metodologi Singkat	3
1.8 Sistematika Penulisan	4
2. Landasan Teori	5
2.1 Inferensi Logika pada Model Bahasa Besar	5
2.1.1 Pergeseran Paradigma: Dari Pengenalan Pola ke Penalaran	5
2.1.2 Hipotesis <i>Greedy Reasoner</i>	5
2.2 Arsitektur <i>Neuro-Symbolic</i>	6
2.2.1 Kerangka Kerja <i>Translation - Decomposition - Search Resolver - Logic Resolution</i>	6
2.3 Tinjauan Model Terkini	7
2.3.1 Qwen2.5: Generalist Multilingual	7
2.3.2 SEA-LION: Spesialis Regional	7
2.4 Evaluasi dan Benchmark	7
2.4.1 IndoMMLU dan Keterbatasannya	7
2.5 Dampak Kuantisasi pada Inferensi	8
2.6 Rangkuman	8
3. METODOLOGI	9
3.1 Desain Penelitian	9
3.2 Instrumen Data: ProntoQA	10
3.2.1 Adaptasi Linguistik dan Penerjemahan	10
3.3 Konfigurasi Model dan Kuantisasi	10
3.3.1 Model Eksperimen	10
3.3.2 Skema Kuantisasi (Unquantized vs Q4_K_M)	10
3.4 Prosedur Eksperimen	11

3.4.1	Teknik Prompting	11
3.4.2	Parameter Inferensi	11
3.5	Teknik Analisis Data: Regex Adaptif	12
3.5.1	Algoritma <i>Load Template</i>	12
3.5.2	Algoritma Pengiriman Jawaban (<i>Question Sending</i>)	13
3.5.3	Algoritma Penangkapan Jawaban (<i>Result Grabbing</i>)	19
3.5.4	Metrik Evaluasi	23
4.	EKSPERIMEN	24
4.1	thesis.tex	24
4.2	Direktori config	24
4.2.1	settings.tex	24
4.2.2	istilah.tex	24
4.2.3	references.bib	25
4.3	Direktori internals	25
4.3.1	hype.indonesia.tex	25
4.3.2	uithesis.sty	25
4.4	Direktori src/00-frontMatter	26
4.5	Direktori src/01-body	27
5.	ANALISIS	28
5.1	Tugas Akhir Individu S1, Proposal Tesis, dan Tesis S2	28
5.2	Tugas Akhir Kelompok S1	30
5.3	Laporan Ilmiah dan Disertasi S3	31
5.4	Laporan Kerja Praktik	32
5.5	Laporan Kegiatan Merdeka Belajar Kampus Merdeka	33
6.	PENUTUP	35
6.1	Kesimpulan	35
6.2	Saran	35
	DAFTAR REFERENSI	36
	DAFTAR ISTILAH	37

DAFTAR GAMBAR

Gambar 1.1. Penjelasan singkat terkait gambar.	3
--	---

DAFTAR TABEL

DAFTAR KODE PROGRAM

Kode 3.1.	Kode untuk mendefinisikan kelas dan fungsi generate pada LLM	11
Kode 3.2.	Kode untuk memparse prompt template dengan data point	12
Kode 3.3.	Kode untuk memberikan pertanyaan ke LLM	13
Kode 3.4.	Kode untuk mem-parse hasil dari LLM untuk translasi ke FOL	19
Kode 3.5.	Kode untuk mem-parse hasil dari LLM untuk translasi ke FOL	20
Kode 3.6.	Kode untuk mem-parse hasil dari LLM untuk translasi ke FOL	22

DAFTAR LAMPIRAN

Lampiran 1. CHANGELOG	38
Lampiran 2. Judul Lampiran 2	41

BAB 1

PENDAHULUAN

Bab ini memaparkan latar belakang, permasalahan, tujuan, batasan, manfaat, ringkasan metodologi, serta sistematika penulisan penelitian ini. Penelitian berfokus pada evaluasi kemampuan penalaran logis oleh Model Bahasa Besar (Large Language Models, LLM) ketika bekerja pada dataset berbahasa Indonesia.

1.1 Latar Belakang

Perkembangan *Large Language Model* (LLM) telah mendorong kemajuan signifikan pada berbagai tugas pemrosesan bahasa natural seperti penerjemahan, ringkasan, dan tanya-jawab. Namun, kemampuan LLM untuk melakukan penalaran logis, yaitu menarik inferensi yang benar dari himpunan premis dan aturan formal, masih menghadapi kendala pada akurasi dan kebenaran dari hasil inferensi, terutama di kasus yang memerlukan normalisasi, dekomposisi, pencarian bukti, dan resolusi logika.

Sebagian besar dataset penalaran dibuat dalam bahasa Inggris, sehingga studi terhadap kemampuan penalaran LLM pada bahasa lain, termasuk Bahasa Indonesia, relatif terbatas. Perbedaan struktur linguistik, idiom, dan masalah tokenisasi serta kualitas terjemahan dapat memengaruhi performa model setelah adaptasi lintas bahasa. Oleh karena itu, diperlukan adaptasi dan evaluasi sistematis pada dataset berbahasa Indonesia serta investigasi *textit*pipeline yang menggabungkan modul terjemahan/normalisasi, dekomposisi aturan, mekanisme pencarian bukti, dan resolusi logika.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, rumusan masalah penelitian ini adalah:

1. Sejauh mana LLM mampu melakukan penalaran logis pada dataset berbahasa Indonesia?
2. Bagaimana perbandingan performa antara model open-source berparameter rendah dalam inferensi dataset?
3. Seberapa efektif pipeline *translate* \rightarrow *decompose* \rightarrow *search* \rightarrow *resolve* dalam meningkatkan akurasi inferensi pada data Bahasa Indonesia dibanding dengan penalaran

secara langsung secara naive?

4. Apa saja sumber utama kegagalan (kesalahan translasi, tokenisasi, ambiguitas budaya/linguistik, format data) ketika memindahkan benchmark penalaran dari Bahasa Inggris ke Bahasa Indonesia?

1.3 Tujuan Penelitian

Tujuan umum: Mengevaluasi dan memperbaiki kemampuan penalaran logis LLM pada dataset berbahasa Indonesia menggunakan pipeline.

Tujuan khusus:

1. Mengukur performa beberapa model pada tugas penalaran menggunakan metrik akurasi dan analisis kesalahan.
2. Mengidentifikasi dan mengkategorikan sumber kesalahan serta memberikan rekomendasi praktis untuk pengolahan data dan desain eksperimen berbahasa Indonesia.
3. Menyediakan dataset terjemahan, skrip eksperimen, dan laporan replikasi yang dapat digunakan peneliti lain.

1.4 Batasan Penelitian

Agar fokus dan ruang lingkup terukur, penelitian ini dibatasi sebagai berikut:

- **Dataset:** Fokus pada dataset *proof-style* yang telah diterjemahkan ke Bahasa Indonesia, yaitu ProntoQA saja
- **Model:** Eksperimen menggunakan model open-source yang dapat dijalankan lokal maupun server, khususnya dengan kuantisasi.
- **Evaluasi:** Metrik utama adalah akurasi jawaban akhir terhadap ground truth.
- **Sumber daya:** Eksperimen disesuaikan dengan kapasitas komputasi, sampling dev/test split dimulai pada 10% hingga 100% tergantung ketersediaan.

Referensi dari pipeline dan metoode, termasuk skrip seperti `translate_decompose.py`, `negate.py`, dan `search_resolve.py`, serta utilitas evaluasi tersedia pada repositori eksperimen yang menjadi inspirasi implementasi ini, yaitu pada repositori Aristotle LaTeX.

1.5 Manfaat Penelitian

Hasil penelitian ini diharapkan memberi manfaat:

- Bagi akademik: memperkaya literatur penalaran LLM pada bahasa Indonesia.
- Bagi praktisi: panduan pemilihan model dan desain pipeline untuk tugas inferensi logis berbahasa lokal.
- Bagi komunitas open-source: dataset dan skrip replikasi yang dapat mempercepat penelitian lanjutan.

1.6 Posisi Penelitian

@todo

Sebutkan posisi penelitian Anda. Ada baiknya jika Anda menggunakan gambar atau diagram. Template ini telah menyediakan contoh cara memasukkan gambar.



Gambar 1.1: Penjelasan singkat terkait gambar.

@todo

Jelaskan Gambar 1.1 di sini. Setiap gambar yang dimasukkan ke tugas akhir **WAJIB** untuk dijelaskan oleh minimal satu paragraf.

1.7 Metodologi Singkat

Pendekatan penelitian dirancang sebagai pipeline yang memadukan teknik pemrosesan teks dan mekanisme reasoning simbolik:

1. **Persiapan Data:** Dataset yang ada di translasikan ke Bahasa Indonesia menggunakan model open-source
2. **Dekomposisi Logis:** Mengurai aturan kompleks menjadi bentuk yang lebih sederhana ke dalam First Order Logic (FOL) dan menormalisasi ke Prenex Normal Form (PNF) atau Conjunctive Normal Form (CNF).
3. **Inisialisasi Dua Jalur Pencarian:** Menegasikan konjektur atau pertanyaan untuk membentuk klausa komplemen yang akan dicari.
4. **Search & Resolve:** Melakukan pencarian klausa komplemen dan resolusi logika untuk menyimpulkan kebenaran atau identifikasi kontradiksi.
5. **Evaluasi:** Mengevaluasi hasil akhir dengan menggabungkan jawaban dari dua jalur tersebut dan membandingkannya dengan ground truth dan di agregasi menggunakan metrik akurasi.

1.8 Sistematika Penulisan

Sistematika penulisan laporan adalah sebagai berikut:

- Bab 1 PENDAHULUAN
Bab ini mencakup latar belakang, cakupan penelitian, dan pendefinisian masalah.
- Bab 2 LANDASAN TEORI
Bab ini mencakup pemaparan terminologi dan teori yang terkait dengan penelitian berdasarkan hasil tinjauan pustaka yang telah digunakan, sekaligus memperlihatkan kaitan teori dengan penelitian.
- Bab 3 METODOLOGI
Bab ini mencakup metodologi atau langkah-langkah yang digunakan untuk melakukan penelitian ini.
- Bab 4 EKSPERIMEN
Bab ini mencakup eksperimen dan hasil eksperimen penelitian
- Bab 5 ANALISIS
Bab ini mencakup analisis dari hasil eksperimen yang sudah dilakukan
- Bab 6 PENUTUP
Bab ini mencakup kesimpulan akhir penelitian dan saran untuk pengembangan berikutnya.

BAB 2

LANDASAN TEORI

Bab ini merupakan kerangka teoretis yang mendasari analisis kemampuan LLM dalam melakukan inferensi logika, khususnya dalam konteks Bahasa Indonesia. Pembahasan mencakup arsitektur kognitif model bahasa, batasan deterministik dari model probabilistik, pendekatan *neuro-symbolic*, serta kendala linguistik dan komputasi yang memengaruhi performa model pada sumber daya rendah (*low-resource setting*).

2.1 Inferensi Logika pada Model Bahasa Besar

2.1.1 Pergeseran Paradigma: Dari Pengenalan Pola ke Penalaran

Perkembangan LLM telah menggeser fokus penelitian *Natural Language Processing* (NLP) dari sekadar pengenalan pola statistik menjadi proses kognitif yang kompleks, termasuk penalaran (*reasoning*). Dalam konteks ini, inferensi logika didefinisikan sebagai kemampuan model untuk menarik kesimpulan yang valid dari serangkaian premis yang diberikan, mengikuti aturan logika formal seperti deduksi, induksi, atau abduksi Saparov and He (2023).

Namun, berbeda dengan sistem logika simbolik tradisional yang bersifat deterministik, LLM beroperasi berdasarkan probabilitas statistik. Hal ini menimbulkan pertanyaan mendasar mengenai apakah LLM benar-benar "bernalar" atau hanya melakukan peniruan pola argumen yang meyakinkan secara statistik.

2.1.2 Hipotesis *Greedy Reasoner*

Salah satu kritik utama terhadap kemampuan penalaran LLM diajukan oleh Saparov dan He (2023) melalui hipotesis "Greedy Reasoner". Riset mereka menunjukkan bahwa model berbasis arsitektur Transformer cenderung menggunakan strategi pencarian (*greedy search*) saat menyelesaikan rantai logika Saparov and He (2023).

- **Sifat Probabilistik Lokal:** Pada setiap langkah inferensi, model memprediksi token selanjutnya $P_{w_t|w_{1:t-1}}$ yang memiliki probabilitas tertinggi berdasarkan konteks lokal, tanpa mempertimbangkan validitas logika global dari keseluruhan argumen.

- **Ketidakmampuan *Backtracking*:** Jika model mengambil langkah logika yang salah di awal, model cenderung "berhalusinasi" untuk membenarkan kesimpulan tersebut daripada kembali dan memperbaiki premis sebelumnya.

Keterbatasan ini sangat penting dalam konteks Bahasa Indonesia, di mana data pelatihan yang lebih sedikit dapat menyebabkan distribusi probabilitas token logika (seperti "jika", "maka", "kecuali") menjadi kurang *robust* dibandingkan Bahasa Inggris.

2.2 Arsitektur *Neuro-Symbolic*

Untuk mengatasi kelemahan penalaran probabilistik murni, pendekatan terkini mulai mengadopsi arsitektur *Neuro-Symbolic* yang menggabungkan fleksibilitas semantik LLM dengan ketepatan logika simbolik. Salah satu kerangka kerja yang relevan adalah **Aristotle** yang diperkenalkan oleh Xu et al. (2025).

2.2.1 Kerangka Kerja *Translation - Decomposition - Search Resolver - Logic Resolution*

Berdasarkan Xu et al. (2025), kerangka kerja Aristotle memisahkan proses pemahaman bahasa dari proses pembuktian logika melalui tiga modul utama:

1. **Translation:** Menggunakan LLM untuk menerjemahkan premis bahasa natural menjadi bentuk logika standar atau *First Order Logic* (FOL). Dalam konteks penelitian ini, modul ini berfungsi mengubah / mentranslasikan premis-premis yang ada menjadi representasi formal terstruktur, termasuk predikat / argumen, operator logika, dan kuantor, serta mengubah subjek yang dalam bentuk *plural* menjadi bentuk *singular*-nya.
2. **Decomposition:** Dari hasil translasi pada tahap sebelumnya, premis-premis tersebut di dekomposisi ke dalam *Prenex Normal Form* (PNF) dan *Conjunctive Normal Form* (CNF). Lalu jika ada kuantor eksistensial, maka hasil dari dekomposisi CNF akan di dekomposisi dengan Skolemisasi untuk menghilangkan kuantor eksistensial tersebut.
3. **Search Resolver:** Melakukan pencarian jalur pembuktian secara sistematis. Berbeda dengan model generatif biasa yang hanya memprediksi kata berikutnya, modul ini mencari klausa pelengkap untuk membuktikan kontradiksi (*Proof by Contradiction*).

4. **Logical Resolution:** Menyelesaikan konflik logika secara deterministik untuk menghasilkan keluaran biner (Benar/Salah), status "Tidak Diketahui" / *Unknown*, dan status *Self-Contradictory*, yang memberikan kepastian lebih tinggi dibandingkan teks generatif biasa.

2.3 Tinjauan Model Terkini

2.3.1 Qwen2.5: Generalist Multilingual

Qwen2.5 merupakan model mutakhir yang dilatih pada korpus masif (hingga 18 triliun token). Keunggulan utamanya terletak pada kemampuan generalisasi logika matematika dan pemrograman yang kuat Team (2024); Yang et al. (2024). Meskipun bukan model khusus Bahasa Indonesia, skala pelatihannya yang masif memungkinkan munculnya kemampuan penalaran (*emergent reasoning capabilities*) yang sering kali melampaui model yang lebih kecil namun spesifik.

2.3.2 SEA-LION: Spesialis Regional

SEA-LION (*Southeast Asian Languages in One Network*) adalah inisiatif untuk membangun model yang selaras secara budaya dan linguistik untuk Asia Tenggara. Model ini dilatih ulang (*continued pre-training*) dengan data bahasa yang digunakan pada Asia Tenggara, termasuk bahasa Indonesia, yang signifikan Singapore (2024). Secara teoretis, SEA-LION diharapkan memiliki pemahaman nuansa linguistik yang lebih baik, yang penting untuk tahap *Logical Decomposer* dalam mendeteksi premis implisit dalam teks Bahasa Indonesia.

2.4 Evaluasi dan Benchmark

2.4.1 IndoMMLU dan Keterbatasannya

Benchmark standar untuk evaluasi LLM dalam Bahasa Indonesia saat ini adalah **IndoMMLU** Koto et al. (2023). Dataset ini terdiri dari soal-soal ujian dari tingkat SD hingga Universitas.

Meskipun komprehensif, IndoMMLU memiliki keterbatasan fundamental untuk penelitian inferensi logika murni:

- **Absennya Matematika Simbolik:** Koto et al. (2023) secara eksplisit mengecualikan soal matematika simbolik karena sudah terdapat dataset untuk soal tersebut, seperti GSM-8K Cobbe et al. (2021) dan NumGLUE Mishra et al. (2022)
- **Focus on Knowledge Crystallization:** Sebagian besar tugas menguji ingatan akan fakta (*crystallized knowledge*) seperti Sejarah atau Geografi, bukan kemampuan memanipulasi aturan logika (*fluid reasoning*).

Oleh karena itu, penelitian ini perlu mengadaptasi benchmark logika formal (seperti ProntoQA) ke dalam Bahasa Indonesia untuk pengukuran yang lebih akurat.

2.5 Dampak Kuantisasi pada Inferensi

Dalam skenario penggunaan praktis dengan sumber daya komputasi terbatas, sering kali penelitian menggunakan model yang sudah dikuantisasi (misalnya menggunakan format GGUF via `llama.cpp`) untuk mengurangi kebutuhan memori Gerganov (2023) ataupun mengurangi kebutuhan *processing power*, khususnya untuk dataset yang besar.

Secara teoretis, kuantisasi (misalnya dari 16-bit ke 4-bit) berdampak non-linear terhadap kemampuan model:

1. **Ketahanan Bahasa:** Kemampuan menghasilkan teks yang lancar (*fluency*) relatif baik terhadap kuantisasi.
2. **Logical Fragility:** Inferensi logika sangat sensitif terhadap presisi numerik. Kesalahan pembulatan kecil pada bobot *attention* dapat mengubah probabilitas operator negasi ("tidak") atau implikasi, yang berakibat fatal pada validitas rantai logika.

2.6 Rangkuman

Berdasarkan tinjauan pustaka di atas, dapat disimpulkan bahwa kemampuan inferensi logika LLM pada Bahasa Indonesia bukan hanya fungsi dari ukuran model, tetapi merupakan interaksi kompleks antara arsitektur penalaran (Neuro-Symbolic vs Greedy). Kerangka pemikiran ini menjadi landasan bagi eksperimen yang akan dilakukan dalam penelitian ini.

BAB 3

METODOLOGI

Bab ini menjelaskan desain eksperimen, persiapan dataset ProntoQA, konfigurasi model, serta teknik evaluasi yang diadaptasi untuk Bahasa Indonesia.

Pendekatan penelitian ini bersifat kuantitatif-eksperimental. Evaluasi dilakukan dengan membandingkan dua skenario *prompting*: (1) **Naive Prompting**, prompt langsung tanpa *scaffolding* atau petunjuk langkah demi langkah, tetapi diberikan beberapa contoh pengerjaan. (2) **Prompting Terstruktur dengan Kerangka Aristotle**, *prompt* yang dilakukan dengan *prompt template* beserta contoh keluaran pada setiap tahapannya.

Hasil setiap percobaan diukur menggunakan metrik akurasi yang didefinisikan pada Bagian 3.1 dan untuk melihat perbedaan signifikan pada perbedaan performa antar skenario *prompting*. Semua eksperimen dijalankan deterministik (parameter inferensi seperti $\text{temperature}=0.0$ dan *fixed seed*) agar perbedaan kinerja mencerminkan efek *prompting*, bukan variansi sampling.

3.1 Desain Penelitian

Penelitian ini bertujuan untuk mengisolasi kemampuan *reasoning* model dari pengetahuan umumnya. Oleh karena itu, desain penelitian difokuskan pada parameter-parameter berikut:

1. Variabel Bebas (*Independent Variables*):

- **Presisi Model:** Tingkat kuantisasi bobot model, dibagi menjadi dua level: *Unquantized* (FP16/BF16) dan *4-bit K-Quant Medium* (Q4_K_M).

2. Variabel Terikat (*Dependent Variable*):

- **Akurasi Logika (*Strict Accuracy*):** Persentase jawaban yang benar secara biner (True/False) yang diekstraksi dari luaran *framework*.

3. Variabel Kontrol:

- Parameter model yang sama untuk memastikan kesetaraan antar model.
- *Prompt* dan *data point* yang diberikan ke LLM.

3.2 Instrumen Data: ProntoQA

Penelitian ini menggunakan dataset **ProntoQA** (*Prompting with Ontologies for QA*). Pemilihan dataset ini didasarkan pada karakteristiknya yang sintetik dan tidak terlalu kompleks untuk dilakukan inferensi. Penggunaan entitas fiktif (seperti "Wumpus", "Jompus") mencegah model menggunakan "hafalan" pengetahuan dunia, sehingga memaksa model untuk benar-benar melakukan deduksi berdasarkan premis yang diberikan.

3.2.1 Adaptasi Linguistik dan Penerjemahan

Dataset asli ProntoQA yang berbahasa Inggris diterjemahkan ke dalam Bahasa Indonesia. Adapun kriteria dalam translasi dataset ke dalam Bahasa Indonesia: Translasi dilakukan secara otomatis menggunakan LLM *open-source* dengan *prompting* menggunakan *template* Contoh: "Every X is Y" → "Setiap X adalah Y".

3.3 Konfigurasi Model dan Kuantisasi

Kuantisasi model dilakukan menggunakan `llama.cpp` Gerganov (2023), jika model tersebut belum memiliki versi kuantisasi yang ada pada repositori developer model tersebut

3.3.1 Model Eksperimen

Tiga arsitektur model digunakan untuk mewakili kategori global dan regional:

- **Qwen2.5-7B-Instruct:** Mewakili model *state-of-the-art* global dengan kemampuan matematika dan logika yang kuat.
- **SEA-LION-v3-Llama-8B-Instruct:** Mewakili model regional yang dilatih khusus dengan data Asia Tenggara (termasuk Bahasa Indonesia), untuk melihat apakah spesialisasi bahasa memberikan keuntungan pada penalaran.
- **SahabatAI-v1-Llama-8B-Instruct:** Mewakili model nasional yang dilatih khusus dengan data Bahasa Indonesia sesuai dengan dataset yang sudah ditranslasikan

3.3.2 Skema Kuantisasi (Unquantized vs Q4_K_M)

Penelitian ini juga membandingkan dua kondisi presisi bobot hanya sebagai benchmark pada naive prompting untuk melihat seberapa tingkat error dan kecepatan inferensi jika menggunakan metode kuantisasi dibandingkan dengan *full precision*

1. **Unquantized (FP16/BF16):** Model dijalankan pada presisi aslinya (16-bit). Ini berfungsi sebagai *baseline* performa ideal (maksimal).
2. **Q4_K_M (4-bit K-Quant Medium):** Model dikompresi menggunakan metode *k-quantization* tipe Medium. Metode ini dipilih karena menyeimbangkan ukuran dan akurasi dengan cara:
 - Bobot *Attention.v* (output value) dan sebagian *Feed-Forward* disimpan dengan presisi lebih tinggi (6-bit) karena krusial untuk akurasi.
 - Bobot lainnya dikuantisasi ke 4-bit.

3.4 Prosedur Eksperimen

3.4.1 Teknik Prompting

Setiap input ke model diformat menggunakan prompt template dan menambahkan sebuah data point dan diteruskan ke LLM untuk diproses. Struktur *prompt* terdiri dari:

1. **Instruksi Sistem:** Menetapkan persona (misal: "Anda adalah ahli logika..") dan juga perintah dari user (misal: "Deskripsi tugas: Anda diberikan..")
2. **Contoh:** Contoh soal logika beserta langkah penyelesaiannya (*reasoning steps*).
3. **Soal Target:** Premis dan pertanyaan dari dataset PrOntoQA yang harus dijawab sesuai dengan fungsi dari tahapan yang sedang dijalankan

3.4.2 Parameter Inferensi

Untuk memastikan reproduisibilitas, model dan parameter diatur di dalam *class* berikut:

```

1 class LlamaCPPBackend:
2     def __init__(self, local_model_path: str):
3         """
4         local_model_path: Must point to a specific .gguf FILE, not just a directory.
5         """
6         if not LLAMACPP_AVAILABLE: raise ImportError("llama-cpp-python not installed.
          Run 'pip install llama-cpp-python'")
7
8         # n_gpu_layers=-1 means offload ALL layers to GPU
9         self.llm = Llama(
10             model_path=local_model_path,
11             n_ctx=0,
12             n_gpu_layers=-1,
13             verbose=False
14         )
15

```

```

16     def generate(self, prompt: str, max_new_tokens: int = 512, temperature: float =
17         0.0, **kwargs) -> str:
18         output = self.llm(
19             prompt,
20             max_tokens=max_new_tokens,
21             stop=[],
22             echo=True, # Return prompt + completion to match others
23             temperature=temperature
24         )
25     return output['choices'][0]['text']

```

Kode 3.1: Kode untuk mendefinisikan kelas dan fungsi generate pada LLM

3.5 Teknik Analisis Data: Regex Adaptif

Tantangan utama dalam mengevaluasi respons dalam Bahasa Indonesia adalah variasi linguistik dalam menyatakan kebenaran dan kesalahan. Model tidak selalu menjawab sesuai dengan *template* atau dalam formula yang diinginkan. Oleh karena itu, modul evaluasi standar ProntoQA dimodifikasi pada bagian load prompt template, pengiriman jawaban (*question sending*), dan penangkapan hasil pengiriman (*result grabbing*).

3.5.1 Algoritma Load Template

Sistem menggunakan logika *Regular Expression* (Regex) untuk mem-*parse template* dan me-*replace* label sesuai dengan formatnya

1. Load file prompt template sesuai dengan fungsi yang ingin dijalankan
2. Replace marker, seperti [[PREMISIS]] sesuai dengan data point

Implementasi *load template* untuk *translation to first order logic*:

```

1 def construct_prompt_a(self, record, in_context_examples_trans):
2     full_prompt = in_context_examples_trans
3     if self.dataset_name == "LogicNLI":
4         context = "\n".join(record['facts'] + record['rules'])
5         question = record['conjecture']
6     else:
7         context = record['context']
8         question = re.search(r'\?(.*)', record['question'].strip()).group(1).strip()
9     full_prompt = full_prompt.replace('[[PREMISES]]', context)
10    full_prompt = full_prompt.replace('[[CONJECTURE]]', question)
11    return full_prompt

```

Kode 3.2: Kode untuk memparse prompt template dengan data point

Kode tersebut kurang lebih sama untuk setiap tahapan fungsi dari translasi, dekomposisi, dan *search_resolve*

3.5.2 Algoritma Pengiriman Jawaban (*Question Sending*)

Sistem menggunakan modul *generate* dari model untuk menghasilkan jawaban dari pertanyaan yang diberikan ke LLM

1. Prompt template + data point yang sudah dimasukkan ke dalam prompt dikirim ke LLM

Implementasi logika pengiriman jawaban untuk *search_resolve*:

```

1 def process_example(example, counter):
2     try:
3         print(f"Running example: {example['id']}")
4
5         flag = 'true'
6         reasoning_step = []
7         search_round = 0
8
9         normalized_context = example['normalized_context']
10        cleaned_normalized_context = self.clean_conjecture(normalized_context)
11        normalized_context_list = cleaned_normalized_context.splitlines()
12        normalized_context_list = [
13            '\n'.join(self.remove_negations(line) for line in
14            item.replace('\textnormal', '').replace('\textrm', '').replace('\\\\text', '\\text')
15                .replace('\\text', '').replace('-', ''))
16                .replace('{', '').replace('}', '').replace('\\right', '')
17                .replace('\\left', '').replace('\\newline', '\\n').replace('$',
18            '').split('\\n'))
19            for item in normalized_context_list if "(" in item and ")" in item
20        ]
21
22        normalized_conjecture = self.clean_conjecture(example['normalized_conjecture'])
23        negated_label = example['negated_label']
24        sos_list = self.clean_conjecture(example['sos_list'])
25        if '(' not in sos_list and ')' not in sos_list:
26            sos_list = normalized_conjecture
27            sos_list = self.remove_negations(sos_list)
28            if example['negated_label'] == 'True':
29                sos_list = self.negate_boolean(sos_list)
30
31        modified_context_list = []
32        for item in normalized_context_list:
33            item = next((p.strip() for p in item.split(":::", 1) if "(" in p), "")
34            if '\\wedge' in item:
35                split_items = item.split('\\wedge')

```

```

34         modified_context_list.extend([sub_item.strip() for sub_item in
split_items])
35         elif '\\land' in item:
36             split_items = item.split('\\land')
37             modified_context_list.extend([sub_item.strip() for sub_item in
split_items])
38         elif '\\u2227' in item:
39             split_items = item.split('\\u2227')
40             modified_context_list.extend([sub_item.strip() for sub_item in
split_items])
41         else:
42             modified_context_list.append(item)
43         normalized_context_list = [item for item in modified_context_list if "(" in
item and ")" in item]
44         print("Normalized Context List: ", normalized_context_list)
45         print("Normalized Conjecture: ", normalized_conjecture)
46         print("sos_list: ", sos_list)
47
48         normalized_context_list.append(sos_list)
49
50         list_of_sos = []
51         list_of_compelment = []
52
53         selected_clause = None
54
55         while flag == 'true':
56             if search_round >= self.search_round:
57                 final_answer = "No final answer found in the text."
58                 break
59
60             if selected_clause == None:
61                 print("Search Router Operating...")
62                 print("Running: ", example['id'])
63                 print("Ground truth: ", example['ground_truth'])
64
65                 complement_indices =
self.filter_complementary_context(normalized_context_list, sos_list)
66                 print("Complement Indices: ", complement_indices)
67
68                 if complement_indices:
69                     potential_clauses = [normalized_context_list[index] for index in
complement_indices]
70
71                 # Normalize sos_list and reasoning_step entries for robust
duplicate checking
72                 norm_sos = normalize_clause_for_compare(sos_list)
73                 used_pairs = set()
74                 for step in reasoning_step:
75                     # step is [sos, selected_clause, new_clause]
76                     used_sos = normalize_clause_for_compare(step[0])

```

```

77         used_selected = normalize_clause_for_compare(step[1])
78         used_pairs.add((used_sos, used_selected))
79
80         # Keep clauses that are not present in used_pairs for (sos, clause)
81         valid_clauses = sorted(
82             [
83                 clause for clause in potential_clauses
84                 if (norm_sos, normalize_clause_for_compare(clause)) not in
used_pairs
85             ],
86             key=len
87         )
88         print("Potential Clauses: ", potential_clauses)
89         print("Valid Clauses: ", valid_clauses)
90
91         if valid_clauses:
92             if len(valid_clauses) > 1:
93                 list_of_sos.append(sos_list)
94                 list_of_complement.append(valid_clauses)
95                 print("Current SOS: ", sos_list, "Current Complement: ",
valid_clauses)
96
97                 selected_clause = list_of_complement[-1].pop(0)
98             else:
99                 selected_clause = valid_clauses[0]
100         else:
101             print("All potential clauses have been used before with this
SOS list.")
102
103             print("Checking cached SOS and complement pairs.")
104             print(f"List of Complements in Cache: {list_of_complement} with
length: {len(list_of_complement)}" )
105             found_new_pair = False
106             if len(list_of_complement) > 0:
107                 for i, complement_clauses in enumerate(list_of_complement):
108                     if complement_clauses:
109                         sos_list = list_of_sos[i]
110                         selected_clause = complement_clauses.pop(0)
111                         found_new_pair = True
112                         break
113                     if not found_new_pair:
114                         print("No more sos and complement pairs found in
cache.")
115
116                         final_answer = "cannot find sos with complement"
117                         break
118
119             if not complement_indices:
120                 if len(list_of_complement) > 0:
121                     all_empty = True
122                     original_sos = sos_list
123                     original_selected_clause = selected_clause
124                     for i, clauses in enumerate(list_of_complement):

```

```

122         if len(clauses) > 0:
123             new_selected_clause = list_of_compelment[i][0]
124             new_sos_list = list_of_sos[i]
125             if new_sos_list != original_sos or new_selected_clause
!= original_selected_clause:
126                 selected_clause = list_of_compelment[i].pop(0)
127                 sos_list = new_sos_list
128                 all_empty = False
129                 print("Check cache: ", sos_list, "Current
Complement: ", selected_clause)
130                 break
131             if all_empty:
132                 final_answer = "No complement found in both context and
cache."
133                 break
134             else:
135                 final_answer = "No complement found in the context."
136                 break
137
138         if any(
139             normalize_clause_for_compare(step[0]) ==
normalize_clause_for_compare(sos_list) and
140             normalize_clause_for_compare(step[1]) ==
normalize_clause_for_compare(selected_clause)
141         ):
142             for step in reasoning_step:
143                 print("Skipping this search round as it has appeared before.")
144                 found_new_pair = False
145                 for i, complement_clauses in enumerate(list_of_compelment):
146                     if complement_clauses:
147                         new_sos_list = list_of_sos[i]
148                         new_selected_clause = complement_clauses[0]
149                         if new_sos_list != sos_list or new_selected_clause !=
selected_clause:
150                             sos_list = new_sos_list
151                             selected_clause = complement_clauses.pop(0)
152                             found_new_pair = True
153                             break
154
155                 if not found_new_pair:
156                     print("No more sos and complement pairs found in cache.")
157                     final_answer = "No sos and complement found"
158                     break
159             else:
160                 print("SOS: ", sos_list)
161                 print("Selected Clause: ", selected_clause)
162                 print("Search round: ", search_round)
163
164         # If after all selection attempts we still have no selected_clause, abort
gracefully
165         if selected_clause is None:

```

```

165         print("No selected_clause available after checking context and cache.
Aborting example as Unknown.")
166         final_answer = "Unknown"
167         flag = 'false'
168         break
169     prompts_e = self.construct_prompt_e(negated_label, normalized_conjecture,
sos_list, selected_clause, in_context_examples_logic_resolver)
170     print(f"\n\nPrompt to Logic Solver: {prompts_e}\n\n", )
171     responses_e, _ = self.model_api.generate(prompts_e)
172     print(f"\n\nResponse from Logic Solver: {responses_e}\n\n", )
173
174     logic_solver_result = self.post_process_logic_solver(responses_e)
175     new_clause = logic_solver_result['new_clause']
176     sufficiency_label = logic_solver_result['sufficiency_label']
177
178     solve_step = [sos_list, selected_clause, new_clause]
179
180     reasoning_step.append(solve_step)
181     print('Reasoning Steps:')
182     for step in reasoning_step:
183         sos_list, selected_clause, new_clause = step
184         solving_step = f"SOS clause: {sos_list}. Selected Clause:
{selected_clause}. New Clause: {new_clause}"
185         print(solving_step)
186
187     if not new_clause.strip():
188         print("No new clause found. Searching from the cache.")
189         all_empty = "True"
190         for i, clause in enumerate(list_of_compelment):
191             if len(clause) > 0:
192                 selected_clause = list_of_compelment[i].pop(0)
193                 sos_list = list_of_sos[i]
194                 all_empty = "False"
195                 print("Searching from cache: Current SOS: ", sos_list, "Current
Complement: ", selected_clause)
196                 break
197
198         if len(list_of_compelment) > 0 and all_empty == "True":
199             final_answer = "Unknown"
200             flag = 'false'
201         elif len(list_of_compelment) == 0:
202             final_answer = "Unknown"
203             flag = 'false'
204     else:
205         sos_list = new_clause
206         normalized_context_list.append(new_clause)
207         selected_clause = None
208
209     if sufficiency_label == "True":
210         if new_clause and (new_clause.lower() == "kontradiksi" or

```

```

new_clause.lower() == "false"):
211         if negated_label.lower() == "true":
212             final_answer = "True"
213         elif negated_label.lower() == "false":
214             final_answer = "False"
215
216         flag = 'false'
217
218     else:
219         all_empty = "True"
220         for i, clause in enumerate(list_of_complement):
221             if len(clause) > 0:
222                 selected_clause = list_of_complement[i].pop(0)
223                 sos_list = list_of_sos[i]
224                 all_empty = "False"
225                 print("Check Cache: ", sos_list, "Current Complement: ",
selected_clause)
226                 break
227
228             if len(list_of_complement) > 0 and all_empty == "True":
229                 final_answer = "Unknown"
230                 flag = 'false'
231             elif len(list_of_complement) == 0:
232                 final_answer = "Unknown"
233                 flag = 'false'
234
235         search_round += 1
236
237     final_choice = self.final_process(final_answer)
238
239     output = {'id': example['id'],
240              'original_context': example['original_context'],
241              'question': example['question'],
242              'translated_context': example['translated_context'],
243              'normalized_context': example['normalized_context'],
244              'normalized_conjecture': example['normalized_conjecture'],
245              'negated_label': negated_label,
246              'reasoning_step': self.list_to_indexed_string(reasoning_step),
247              'ground_truth': example['ground_truth'],
248              'final_answer': final_answer,
249              'final_choice': final_choice,
250              'search_round': search_round}
251
252     print(output)
253     return output

```

Kode 3.3: Kode untuk memberikan pertanyaan ke LLM

3.5.3 Algoritma Penangkapan Jawaban (*Result Grabbing*)

Sistem menggunakan logika *Regular Expression* (Regex) bertingkat untuk memarsing jawaban model. Logika ini dirancang untuk kasus-kasus dari hasil generasi oleh LLM

1. **Pendeteksian Bentuk Akhir:** Fokus pencarian dibatasi pada bagian respons pertama pada LLM karena menggunakan LLM berbasis instruksi *"*-Instruct"*, sehingga lebih mudah untuk diinstruksikan dan hasilnya tidak jauh dari ekspektasi

2. **Pola Regex Hirarkis:**

- **Naive Prompting:** Mendeteksi blok "Penjelasan" dan blok "Jawaban" pada hasil generasi LLM
- **Aristotle:** Mendeteksi *boundary* antara jawaban dan template dan mengambil blok sesuai dengan regex
 - **Translation to FOL:** Mendeteksi blok "Fakta", "Konjektur", dan "Aturan" sebagai hasil translasi ke FOL Implementasi kode:

```

1 def extract_facts_rules_conjecture(self, content, context_sentence_count=None):
2     # Clean invisible characters
3     content = (content or "").replace('\u200b', '').replace('\uffff', '')
4
5     prompt_marker = re.compile(
6         r'Di bawah ini(?:\s+adalah(?:\s+yang\s+perlu\s+Anda\s+terjemahkan)?):?',
7         re.IGNORECASE
8     )
9     m_prompt = prompt_marker.search(content)
10    search_start_pos = m_prompt.end() if m_prompt else 0
11
12    block_header = re.compile(r'\s*\{0,3\}Bentuk Akhir\s*\{0,3\}', re.IGNORECASE)
13    m_block = block_header.search(content, pos=search_start_pos)
14
15    if m_block:
16        area = content[m_block.end():]
17    else:
18        area = content[search_start_pos:]
19
20    # Define Patterns for possible headers
21    # Include the "End Markers" as a header type.
22    patterns = {
23        "facts": re.compile(r'(?:(Fakta|Facts)\s*[:\-\]\s*', re.IGNORECASE),
24        "rules": re.compile(r'(?:(Aturan|Rules)\s*[:\-\]\s*', re.IGNORECASE),
25        "conj": re.compile(r'(?:(Konjektur|Conjecture)\s*[:\-\]\s*',
26        re.IGNORECASE),
27        "stop": re.compile(r'(?:(\s*\{0,3\}\s*Akhir Blok\s*\{0,3\}###|``|-{3,})',
28        re.IGNORECASE)
29    }

```

```

28
29 def extract_section(target_key):
30     start_match = patterns[target_key].search(area)
31     if not start_match:
32         return ""
33
34     content_start_idx = start_match.end()
35
36     # Find the next header
37     next_indices = []
38     for key, pat in patterns.items():
39         m = pat.search(area, pos=content_start_idx)
40         if m:
41             next_indices.append(m.start())
42
43     # If found upcoming headers, stop at the nearest one (min index).
44     # If no headers found, go to end of string.
45     if next_indices:
46         cutoff_idx = min(next_indices)
47         raw_text = area[content_start_idx:cutoff_idx]
48     else:
49         raw_text = area[content_start_idx:]
50
51     return raw_text.strip()
52
53 facts = extract_section("facts")
54 rules = extract_section("rules")
55 conjecture = extract_section("conj")
56
57 return facts, rules, conjecture

```

Kode 3.4: Kode untuk mem-parse hasil dari LLM untuk translasi ke FOL

– **Decomposition:** Mendeteksi blok "Aturan dalam CNF" dan "Skolemisasi" jika ada Implementasi kode:

```

1 def post_process_decompose(self, content, rules_count=None):
2
3     content = (content or "").replace('\u200b', '').replace('\uffff', '')
4
5     marker_pattern = r'Di bawah ini adalah yang perlu Anda konversikan
menggunakan normalisasi.'
6     marker_match = re.search(marker_pattern, content, flags=re.IGNORECASE)
7
8     block_header = re.compile(r'\{*0,3\}Bentuk Akhir\{*0,3\}', re.IGNORECASE)
9     m_block = block_header.search(content, pos=marker_match.end())
10
11     area = content[m_block.end():]
12
13     cnf_label_re = re.compile(
14         r'(?:(Aturan dalam CNF|Aturan CNF|Aturan|Rules)\s*[:\-\]?\s*',

```



```

15         flags=re.IGNORECASE
16     )
17     skolem_label_re = re.compile(
18         r'(? :Aturan dalam Skolem|Skolemisasi|Skolem|Bentuk Akhir Setelah
Skolemisasi|Skolemization)\s*[:\~]?s*',
19         flags=re.IGNORECASE
20     )
21
22     # Construct the pattern string explicitly to avoid parentheses nesting
errors.
23     boundary_pattern = (
24         r'(?:'                                     # Start outer
group
25         r'\r?\n\s*'                                 # Newline +
whitespace
26         r'(?:'                                     # Start inner
grouping for headers
27         r'(? :Aturan dalam CNF|Aturan CNF|Aturan \((CNF\)|Aturan|Rules)|'
# CNF headers
28         r'(? :Skolemisasi|Skolem|Bentuk Akhir)|'       # Skolem
headers
29         r'(? :*\{0,3\}\s*Akhir Blok\s*\{0,3\}|Final Form|###)' # End
markers
30         r')'                                         # End inner
grouping
31         r')'                                         # End outer
group
32         r'|$'                                       # OR End of
String
33     )
34
35     boundary_re = re.compile(boundary_pattern, flags=re.IGNORECASE)
36
37     def extract_after_label(label_re):
38         """Finds label, returns text until next boundary."""
39         lab_match = label_re.search(area)
40         if not lab_match:
41             return None
42         start = lab_match.end()
43         bound = boundary_re.search(area, pos=start)
44         end = bound.start() if bound else len(area)
45         return area[start:end].strip()
46
47     cnf_raw = extract_after_label(cnf_label_re)
48     skolem_raw = extract_after_label(skolem_label_re)
49
50     def to_lines(raw):
51         if not raw:
52             return []
53         return [ln.strip() for ln in raw.splitlines() if ln.strip()]

```

```

54
55     cnf_lines = to_lines(cnf_raw)
56     skolem_lines = to_lines(skolem_raw) if skolem_raw else None
57
58     print(f"CNF Raw: {cnf_lines}")
59     print(f"Skolem Raw: {skolem_lines}")
60
61     return cnf_lines, skolem_lines

```

Kode 3.5: Kode untuk mem-parse hasil dari LLM untuk translasi ke FOL

– Search Resolve: Mendeteksi blok "Clause Baru" dan "Label Cukup" Implementasi kode:

```

1 def post_process_logic_solver(self, response_d):
2     content = response_d
3     marker_pattern = r'(.*)Dibawah ini tugas yang perlu Anda lakukan(.*)'
4     marker_match = re.search(marker_pattern, content, flags=re.IGNORECASE)
5     search_area = content[marker_match.end():] if marker_match else content
6
7     final_block_pattern = (
8         r'\s*\{0,3\}(?:Bentuk Akhir)\s*\{0,3\}\s*'
9         r'(.*)'
10        r'(?=\s*\{0,3\}(?:Akhir Blok)\s*\{0,3\})|'
11        r'$)' # or end of string
12    )
13
14    final_block_match = re.search(final_block_pattern, search_area,
15                                  flags=re.DOTALL | re.IGNORECASE)
16
17    if not final_block_match:
18        return [], None
19
20    block = final_block_match.group(1)
21
22    block_clean = block.strip()
23    print(f"\n\nCHOSEN BLOCK:\n\n{block_clean}\n\n")
24    print("END OF CHOSEN BLOCK\n\n")
25
26    clause_pos = re.search(r'Clause\s*Baru', block_clean, flags=re.IGNORECASE)
27    clause_after = block_clean[clause_pos.end():]
28    m_new = re.search(r'\{(.*)\}', clause_after, flags=re.DOTALL)
29
30    if not m_new:
31        raise ValueError(f"'Clause Baru:' with '{...}' not found in expected form.")
32
33    new_clause = m_new.group(1).strip()
34
35    label_pos = re.search(r'Label\s*Cukup', block_clean, flags=re.IGNORECASE)
36    label_after = block_clean[label_pos.end():]

```

```

36     m_label = re.search(r'\[(.*?)\]', label_after, flags=re.DOTALL)
37     if not m_label:
38         raise ValueError(f"'Label Cukup' with ' [...] ' not found in expected form
[True|False].")
39
40     sufficiency_label = m_label.group(1).strip()
41
42     return {
43         "new_clause": new_clause,
44         "sufficiency_label": sufficiency_label,
45     }

```

Kode 3.6: Kode untuk mem-parse hasil dari LLM untuk translasi ke FOL

3.5.4 Metrik Evaluasi

Akurasi dihitung berdasarkan perbandingan antara hasil parsing regex dengan kunci jawaban (*Ground Truth*) dari dataset. Jika model menghasilkan jawaban yang tidak dapat diparsing (misalnya, meracau atau tidak memberikan kesimpulan), maka dianggap sebagai jawaban dengan status "tidak diketahui" / *unknown*. Hal ini memperketat standar evaluasi, karena kemampuan mengikuti instruksi (*instruction following*) dianggap sebagai prasyarat dari penalaran yang valid.

BAB 4

EKSPERIMEN

TTTTTTTTTTTTTTTTT Bab ini menjelaskan tentang struktur dari *template* tugas akhir ini. Dengan memahami struktur *template*, pekerjaan Anda akan menjadi lebih terarah karena Anda tahu di mana Anda harus melakukan sesuatu.

@todo

Sejatinya bab ini digunakan untuk membahas inti dari penelitian Anda. Sesuaikan saja dengan kebutuhan Anda: misalkan bab empat Anda adalah penjelasan terkait implementasi sistem.

4.1 `thesis.tex`

Berkas `thesis.tex` berisi seluruh berkas \LaTeX yang dibaca, jadi bisa dikatakan sebagai berkas utama. Dari berkas ini kita dapat mengatur bab apa saja yang ingin kita tampilkan dalam dokumen.

4.2 Direktori `config`

Direktori `config` berisi berkas-berkas yang menyimpan konfigurasi variabel dan istilah-istilah yang bisa dimodifikasi sesuai dengan kebutuhan tugas akhir.

4.2.1 `settings.tex`

Berkas `settings.tex` berguna untuk mempermudah pembuatan beberapa template standar. Anda diminta untuk menuliskan judul laporan, nama, NPM, dan hal-hal lain yang dibutuhkan untuk pembuatan template.

4.2.2 `istilah.tex`

Berkas `istilah.tex` digunakan untuk mencatat istilah-istilah yang digunakan. Fungsinya hanya untuk memudahkan penulisan. Pada beberapa kasus, ada kata-kata yang harus selalu muncul dengan tercetak miring atau tercetak tebal. Anda juga bisa menggunakan berkas ini untuk mencatat istilah atau akronim khusus yang perlu dimunculkan di Daftar Istilah. Penggunaan lebih lanjut terkait berkas `_internals/istilah.tex` untuk menyim-

pan istilah atau akronim ada di Subbab ???. Dengan menjadikan kata-kata tersebut sebagai sebuah perintah \LaTeX tentu akan mempercepat dan mempermudah pengerjaan laporan.

4.2.3 `references.bib`

Berkas `references.bib` berisi seluruh daftar referensi yang digunakan dalam laporan. Anda bisa membuat model daftar referensi lain dengan menggunakan BibTeX. Untuk menambahkan referensi dengan format BibTeX, Anda bisa mengisi berkas `references.bib`. Untuk merujuk pada salah satu referensi yang ada, gunakan perintah `\cite`, e.g. `\cite{book:sample}` yang akan akan memunculkan ?. Informasi lebih lanjut mengenai referensi bisa dilihat di Subbab 4.2.3. Untuk mempelajari bibtex lebih lanjut, silahkan buka <http://www.bibtex.org/Format>.

4.3 Direktori `_internals`

Direktori `_internals` berisi halaman-halaman dan *styling* yang tidak perlu diubah untuk penggunaan normal dari template ini. *Styling* bisa diubah jika diperlukan untuk menyesuaikan beberapa fitur template dengan kebutuhan tugas akhir, atau untuk menyesuaikan dengan aturan terbaru yang dirilis oleh Universitas Indonesia.

4.3.1 `hype.indonesia.tex`

Berkas `hype.indonesia.tex` berisi cara pemenggalan beberapa kata dalam bahasa Indonesia. \LaTeX memiliki algoritma untuk memenggal kata-kata sendiri, namun untuk beberapa kasus algoritma ini memenggal dengan cara yang salah. Untuk memperbaiki pemenggalan yang salah inilah cara pemenggalan yang benar ditulis dalam berkas `hype.indonesia.tex`.

4.3.2 `uithesis.sty`

Berkas `uithesis.sty` berisi konfigurasi inti dari *layoutting* untuk *template* ini. Secara umum, Anda tidak perlu mengubah apapun pada berkas ini. Akan tetapi, untuk kasus-kasus lanjutan, seperti menambahkan daftar konten *custom* atau menyalakan dukungan terhadap *multi-language*, Anda bisa mengubahnya secara langsung pada `uithesis.sty`. Salah satu contohnya adalah ketika Anda ingin mendefinisikan daftar suatu jenis objek baru, seperti yang dicontohkan pada Subbab ???. Atau bisa juga ketika Anda ingin

mengganti tipe referensi, seperti yang dicontohkan pada Subbab ???. Jika Anda memiliki feedback maupun ingin berkontribusi terhadap perbaikan *layout*, selama ke arah yang sesuai dengan ketentuan Peraturan Rektor UI terkait format Tugas Akhir, Anda bisa mengubah berkas ini dan berkas lainnya yang terkait lalu membuat Merge Request di repositori. Keterangan lebih lanjut terkait cara kontribusi dapat dilihat di berkas `README.md` dan `CONTRIBUTING`.

4.4 Direktori `src/00-frontMatter`

Direktori `src/00-frontMatter` berisi bagian depan yang memuat halaman-halaman administratif untuk laporan ilmiah Anda. Sedangkan direktori `src/99-backMatter` berisikan berkas-berkas lampiran. Berikut adalah daftar berkas yang tersedia di `src/00-frontMatter`:

1. `pernyataanOrisinalitas.tex` untuk halaman pernyataan orisinalitas. Berlaku untuk semua tipe dokumen kecuali Laporan Kerja Praktik dan Kampus Merdeka.
2. `pengesahanKP.tex` untuk halaman pengesahan spesifik tipe dokumen Laporan Kerja Praktik.
3. `pengesahanMBKM.tex` untuk halaman pengesahan spesifik tipe dokumen Kampus Merdeka.
4. `pengesahanSidang.tex` untuk halaman pengesahan sidang. Berlaku untuk semua tipe dokumen kecuali laporan ilmiah mahasiswa S3 (Disertasi), Laporan Kerja Praktik, dan Kampus Merdeka.
5. `pengesahanSidangS3.tex` untuk halaman pengesahan sidang khusus mahasiswa S3.
6. `kataPengantar.tex` untuk kata pengantar. Berlaku untuk semua tipe dokumen kecuali Laporan Kerja Praktik dan Kampus Merdeka.
7. `persetujuanPublikasi.tex` untuk halaman persetujuan publikasi karya intelektual. Berlaku untuk semua tipe dokumen kecuali Laporan Kerja Praktik dan Kampus Merdeka.
8. `abstrak.tex` untuk halaman abstrak berbahasa Indonesia.
9. `abstract.tex` untuk halaman abstrak berbahasa Inggris.

Umumnya, Anda hanya perlu mengisi bagian-bagian seperti Abstrak dan Kata Pengantar. Berkas sisanya berisi kode yang akan menghasilkan halaman-halaman terkait secara otomatis, sehingga hanya bisa diubah jika diperlukan penyesuaian, misal ukuran *line*

spacing.

4.5 Direktori `src/01-body`

Direktori ini berisi isi laporan yang Anda tulis. Setiap nama berkas e.g. `bab1.tex` merepresentasikan bab dimana tulisan tersebut akan muncul. Sebagai contoh, kode dimana tulisan ini dibuat berada dalam berkas dengan nama `bab4.tex`. Ada enam buah berkas yang telah disiapkan untuk mengakomodir enam bab dari laporan Anda, diluar bab kesimpulan dan saran. Jika Anda tidak membutuhkan sebanyak itu, silahkan hapus kode dalam berkas `thesis.tex` yang memasukan berkas \LaTeX yang tidak dibutuhkan; contohnya perintah `\include{bab6.tex}` merupakan kode untuk memasukan berkas `bab6.tex` kedalam laporan.

BAB 5

ANALISIS

TTTTTTTTTTTTTTTTTT Awalnya, *template* ini hanya digunakan untuk Tugas Akhir (Skripsi) mahasiswa S1 di Fakultas Ilmu Komputer, Universitas Indonesia. Seiring berkembangnya kegiatan pendidikan dan kemahasiswaan di lingkup Fakultas Ilmu Komputer hingga tingkat universitas, penyusun *template* menyadari ada kasus-kasus lain yang bisa menggunakan format Tugas Akhir UI. Beberapa di antaranya adalah tesis S2, disertasi S3, dan laporan kegiatan/kerja praktik. Oleh karena itu, perlu ada penjelasan terkait berbagai kasus penggunaan (*use case*) untuk *template* L^AT_EX ini, dan bagaimana cara pengguna bisa memanfaatkan *template* untuk kasus tersebut.

@todo

Sejatinya bab ini digunakan untuk membahas inti penelitian Anda. Bab lima pada tugas akhir S1 umumnya merupakan pembahasan analisis dari penelitian. Namun, sekali lagi, sesuaikan dengan kebutuhan Anda. Tesis atau disertasi tentunya berbeda dengan skripsi.

5.1 Tugas Akhir Individu S1, Proposal Tesis, dan Tesis S2

Tugas Akhir Individu di Fakultas Ilmu Komputer Universitas Indonesia berlaku sama dengan Tugas Akhir atau Skripsi mahasiswa S1 di fakultas lain di Universitas Indonesia. Proposal Tesis dan Tesis (di beberapa jurusan disebut Karya Akhir) di Fakultas Ilmu Komputer Universitas Indonesia juga berlaku sama dengan Tesis mahasiswa S2 di fakultas lain di Universitas Indonesia. Format yang digunakan untuk semua fakultas juga sama, mengacu ke Keputusan Rektor Universitas Indonesia nomor 2143/SK/R/UI/2017 tentang Pedoman Teknis Penulisan Tugas Akhir Mahasiswa Universitas Indonesia. Sejak versi 2.0.0, *template* ini sudah mengacu ke Keputusan Rektor UI tersebut. Pada versi tersebut juga dukungan untuk cetak skripsi atau tesis bolak-balik sudah tersedia. Tidak ada perubahan khusus yang perlu dilakukan terhadap konfigurasi *template* untuk Tugas Akhir untuk Mahasiswa S1 atau Proposal Tesis dan Tesis untuk Mahasiswa S2. Anda bisa mengikuti tahapan berikut untuk memulai penulisan Anda:

1. Buka `config/settings.tex`. Terdapat lima bagian yang perlu dilengkapi:

- **Judul dokumen:** Anda bisa memasukkan judul dalam bahasa Indonesia dan bahasa Inggris di sini.
 - **Tipe dokumen:** Pada variabel `\type`, cukup tuliskan "Skripsi" atau "Tugas Akhir", sesuaikan dengan aturan dari Fakultas masing-masing. Isi variabel `\jenjang` dengan "Sarjana" atau "Magister". Kosongkan variabel lainnya yang tidak relevan (jangan dihapus).
 - **Informasi penulis:** Karena pada kasus ini, tugas akhir Anda bersifat individu, cukup isi variabel `\penulisSatu` dengan nama Anda, `\npmSatu` dengan NPM Anda, `\programSatu` dengan nama program studi Anda dalam bahasa Indonesia, dan `\studyProgramSatu` dengan nama program studi Anda dalam bahasa Inggris. Untuk variabel lain mohon agar tetap dikosongkan (namun jangan dihapus) sehingga *template* bisa mendeteksi bahwa Anda akan menuliskan skripsi individu.
 - **Informasi dosen pembimbing dan penguji:** Pada umumnya, dosen pembimbing skripsi di UI terdiri dari satu atau dua orang dosen, dan penguji skripsi di UI terdiri dari dua orang dosen. Silakan isi variabel yang relevan dan kosongkan variabel lainnya (namun jangan dihapus).
 - **Informasi lain:** Anda bisa melihat komentar di setiap variabel untuk mengetahui apa yang harus diisi di setiap variabel.
 - **Judul setiap bab:** Silakan isi variabel yang ada untuk judul setiap bab. Jika ada bab yang ingin ditambahkan sebelum bab kesimpulan (misal: bab 6, bab 7), Anda dapat membuat variabel baru, contohnya: `\Var{\bab6}{Analisis Pendapat Pengguna Aplikasi}`.
 - Bagian lainnya seperti "Capitalized Variables" tidak perlu dimodifikasi. Variabel-variabel tersebut menunjang fungsi-fungsi khusus di *template*, salah satunya adalah versi *all caps* dari judul skripsi di halaman judul.
2. Setelah mengisi konfigurasi, Anda bisa periksa halaman-halaman awal dokumen. Jika terdapat ketidaksesuaian pada ukuran atau jarak antar elemen, Anda bisa mengatur melalui berkas-berkas yang ada di `src/00-frontMatter`. Halaman pengesahan sidang yang dipakai di format Tugas Akhir Individu ada di `src/00-frontMatter/pengesahanSidang.tex`. Silakan perbesar atau perkecil ukuran yang ada pada kode `\vspace*{...}`, untuk menyesuaikan *spacing*. Tahapan ini akan berguna terutama jika judul tugas akhir Anda cukup panjang sehingga beberapa teks ada yang terlempar ke halaman berikutnya. Jika ada perubahan kode yang sig-

nifikan, Anda bisa mengusulkan ke penyusun *template*. Keterangan lebih lanjut terkait cara kontribusi dapat dilihat di berkas `README.md` dan `CONTRIBUTING`.

3. Anda juga bisa mengatur beberapa hal sebagai berikut:
 - Pelajari cara sitasi dengan melihat Subbab ?? dan cara melakukan *cross-reference* dengan melihat Subbab ?. Kedua fitur tersebut merupakan fitur yang sangat penting dalam penulisan skripsi menggunakan L^AT_EX.
 - Jika fakultas Anda memerlukan format sitasi selain APA (yang menjadi *default* di tingkat universitas), silakan baca Subbab ?.
 - Jika Anda membutuhkan support untuk selain tulisan alfabet, silakan baca Subbab ?. Jika Anda membutuhkan penulisan notasi matematis, silakan baca Subbab ?. Jika Anda membutuhkan penulisan kode program, silakan baca Subbab ?.
4. Di akhir penulisan, Anda perlu memeriksa ulang tulisan Anda secara lebih teliti untuk memaksimalkan penggunaan kertas, sebisa mungkin hindari *unused space*. Selain itu, perhatikan juga pemenggalan yang dilakukan L^AT_EX apakah sudah sesuai atau belum. Jika ada pemenggalan yang kurang sesuai, silakan tambahkan di `_internals/hypeindonesia.tex` dan *request* untuk kontribusi. Keterangan lebih lanjut terkait cara kontribusi dapat dilihat di berkas `README.md` dan `CONTRIBUTING`.

5.2 Tugas Akhir Kelompok S1

Beberapa fakultas, salah satunya Fakultas Ilmu Komputer Universitas Indonesia (sejak tahun 2022) mengizinkan pengerjaan skripsi secara berkelompok paling banyak 3 (tiga) orang. Format yang digunakan juga mengacu ke Keputusan Rektor Universitas Indonesia nomor 2143/SK/R/UI/2017 tentang Pedoman Teknis Penulisan Tugas Akhir Mahasiswa Universitas Indonesia, namun ada penyesuaian di beberapa hal. Sejak versi 2.1.3, *template* ini mendukung *format* Tugas Akhir kelompok dengan menyesuaikan bagian depan dari *template*. Untuk memanfaatkan *format* tersebut, silakan ikuti tahapan berikut:

1. Buka `config/settings.tex`. Isi variabel pada bagian "**Informasi Penulis**" untuk penulis pertama, kedua dan ketiga secara berurutan. Misal: `\penulisSatu` untuk nama penulis pertama, `\penulisDua` untuk nama penulis kedua, dan `\penulisTiga` untuk nama penulis ketiga. Pastikan Anda mengisi data secara lengkap pada variabel yang sesuai. Jika kelompok Anda hanya terdiri dari 2 (dua) orang, maka variabel-variabel data penulis ketiga harus dikosongkan (namun jangan dihapus). *Template*

akan menyesuaikan *format* sesuai dengan jumlah anggota kelompok di skripsi Anda.

2. Setelah mengisi konfigurasi, Anda bisa periksa halaman-halaman awal dokumen. Jika terdapat ketidaksesuaian pada ukuran atau jarak antar elemen, Anda bisa mengatur melalui berkas-berkas yang ada di `src/00-frontMatter`. Halaman pengesahan sidang yang dipakai di format Tugas Akhir Kelompok ada di `src/00-frontMatter/pengesahanSidang.tex`. Silakan perbesar atau perkecil ukuran yang ada pada kode `\vspace*{...}`, untuk menyesuaikan *spacing*. Tahapan ini akan berguna terutama jika judul tugas akhir Anda dan data kelompok Anda cukup panjang sehingga beberapa teks ada yang terlempar ke halaman berikutnya. Jika ada perubahan kode yang signifikan, Anda bisa mengusulkan ke penyusun *template*. Keterangan lebih lanjut terkait cara kontribusi dapat dilihat di berkas `README.md` dan `CONTRIBUTING`.

5.3 Laporan Ilmiah dan Disertasi S3

Disertasi S3 dan laporan-laporan lain yang diwajibkan untuk jenjang S3 juga menggunakan format sesuai Keputusan Rektor Universitas Indonesia nomor 2143/SK/R/UI/2017 tentang Pedoman Teknis Penulisan Tugas Akhir Mahasiswa Universitas Indonesia, namun ada penyesuaian di beberapa hal. Salah satu penyesuaian yang perlu dilakukan adalah istilah pembimbing yang berganti menjadi Promotor, Kopromotor. Jumlah penguji juga lebih banyak, bisa mencapai 6 orang dosen penguji. Sejak versi 2.1.2, *template* ini mendukung *format* disertasi dengan menyesuaikan bagian depan dari *template*. Untuk memanfaatkan *format* tersebut, silakan ikuti tahapan berikut:

1. Buka `config/settings.tex`.
 - Pada bagian **”Tipe Dokumen”**, variabel `\type` bisa diisi dengan **”Disertasi”** atau tipe dokumen lainnya. Variabel `\jenjang` wajib diisi dengan **”Doktor”**.
 - Pada bagian **”Informasi Pembimbing dan Penguji”**, isi nama lengkap dan gelar Promotor pada variabel `\pembimbingSatu`, dan Kopromotor pada variabel `\pembimbingDua` (jika kopromotor ada dua orang, variabel `\pembimbingTiga` bisa diisi). Untuk penguji, Anda bisa mengisi secara berurutan dari `\pengujiSatu` hingga `\pengujiEnam`.

Konfigurasi untuk dokumen laporan ilmiah S3 tidak mendukung format Tugas Akhir Kelompok.

2. Setelah mengisi konfigurasi, Anda bisa periksa halaman-halaman awal dokumen. Jika terdapat ketidaksesuaian pada ukuran atau jarak antar elemen, Anda bisa mengatur melalui berkas-berkas yang ada di `src/00-frontMatter`. Halaman pengesahan sidang yang dipakai di format laporan ilmiah S3 ada di `src/00-frontMatter/pengesahanSidangS3.tex`. Jika "Halaman Pengesahan" menjadi dua halaman, hal tersebut adalah lumrah. Jika ada hal yang tidak lumrah, silakan perbesar atau perkecil ukuran yang ada pada kode `\vspace*{...}`, untuk menyesuaikan *spacing*. Jika ada perubahan kode yang signifikan, Anda bisa mengusulkan ke penyusun *template*. Keterangan lebih lanjut terkait cara kontribusi dapat dilihat di berkas `README.md` dan `CONTRIBUTING`.

5.4 Laporan Kerja Praktik

Mata kuliah Kerja Praktik umumnya ditawarkan bagi individu sebagai mata kuliah bernilai SKS untuk mempresentasikan dan mendokumentasikan pekerjaan magang di industri melalui laporan karya ilmiah. Laporan Kerja Praktik di Fakultas Ilmu Komputer UI (dan sebagian fakultas yang menyediakan mata kuliah Kerja Praktik) juga menggunakan format sesuai Keputusan Rektor Universitas Indonesia nomor 2143/SK/R/UI/2017 tentang Pedoman Teknis Penulisan Tugas Akhir Mahasiswa Universitas Indonesia, namun ada penyesuaian di beberapa hal. Salah satu penyesuaian yang perlu dilakukan adalah halaman persetujuan yang berbeda karena Kerja Praktik tidak memerlukan sidang. Selain itu, ada beberapa halaman yang tidak diperlukan seperti Pernyataan Orisinalitas dan Persetujuan Publikasi. Sejak versi 2.1.2, *template* ini mendukung *format* laporan kerja praktik dengan menyesuaikan bagian depan dari *template*. Untuk memanfaatkan *format* tersebut, silakan ikuti tahapan berikut:

1. Buka `config/settings.tex`.
 - Pada bagian "**Tipe Dokumen**", variabel `\type` wajib diisi dengan "Laporan Kerja Praktik". Variabel `\jenjang` wajib diisi dengan "Sarjana".
 - Pada bagian "**Informasi Pembimbing dan Penguji**", isi nama lengkap dan gelar dosen kelas Kerja Praktik pada `\pembimbingSatu`, dan kosongkan semua variabel lain pada bagian tersebut (namun jangan dihapus).
- Konfigurasi untuk Laporan Kerja Praktik tidak mendukung format Tugas Akhir Kelompok.

2. Setelah mengisi konfigurasi, Anda bisa periksa halaman-halaman awal dokumen. Jika terdapat ketidaksesuaian pada ukuran atau jarak antar elemen, Anda bisa mengatur melalui berkas-berkas yang ada di `src/00-frontMatter`. Halaman persetujuan yang dipakai di format Laporan Kerja Praktik ada di `src/00-frontMatter/pengehasilanKP.tex`. Silakan perbesar atau perkecil ukuran yang ada pada kode `\vspace*{...}`, untuk menyesuaikan *spacing*. Jika ada perubahan kode yang signifikan, Anda bisa mengusulkan ke penyusun *template*. Keterangan lebih lanjut terkait cara kontribusi dapat dilihat di berkas `README.md` dan `CONTRIBUTING`.

5.5 Laporan Kegiatan Merdeka Belajar Kampus Merdeka

Program Merdeka Belajar Kampus Merdeka¹ merupakan program *flagship* dari Kementerian Pendidikan, Kebudayaan, Riset, dan Teknologi (Kemendikbud) Republik Indonesia yang bertujuan untuk memberikan peluang mahasiswa mendapatkan pengalaman belajar di luar kampus. Terdapat banyak pilihan program Kampus Merdeka yang tersedia bagi mahasiswa UI, beberapa di antaranya adalah Magang Bersertifikat, Studi Independen Bersertifikat (termasuk Program Bangkit²), dan beberapa program lain di tingkat UI seperti *Build Your Own Course* (BYOC). Pada akhir program, mahasiswa diminta menyusun laporan dengan format yang disediakan untuk Kemendikbud, yang tentunya hanya tersedia untuk program dan jalur yang dikelola Kemendikbud. Beberapa program seperti BYOC dan jalur yang diselenggarakan UI seperti Kampus Merdeka Mandiri tidak memiliki akses ke template Kemendikbud. Di Fakultas Ilmu Komputer, laporan MBKM yang tidak melewati jalur yang dikelola Kemendikbud menggunakan laporan akhir layaknya Laporan Kerja Praktik yang formatnya menggunakan aturan Keputusan Rektor Universitas Indonesia nomor 2143/SK/R/UI/2017 tentang Pedoman Teknis Penulisan Tugas Akhir Mahasiswa Universitas Indonesia, dengan beberapa penyesuaian. Salah satu penyesuaian yang perlu dilakukan adalah halaman persetujuan yang berbeda karena Kampus Merdeka tidak memerlukan sidang, namun berbeda dengan Kerja Praktik, laporan kegiatan Kampus Merdeka membutuhkan persetujuan dari mitra. Sejak versi 2.1.3, *template* ini mendukung *format* laporan kerja praktik dengan menyesuaikan bagian depan dari *template*. Untuk memanfaatkan *format* tersebut, silakan ikuti tahapan berikut:

1. Buka `config/settings.tex`.

¹<https://kampusmerdeka.kemdikbud.go.id/>

²<https://www.dicoding.com/programs/bangkit>

- Pada bagian **”Tipe Dokumen”**, variabel `\type` wajib diisi dengan **”Kampus Merdeka”**. Variabel `\jenjang` wajib diisi dengan **”Sarjana”**. Variabel `\kampusMerdekaType` wajib diisi dengan tipe kegiatan atau jalur yang diambil, misal: Magang, Studi Independen, Bangkit, dsb. Jika program memiliki mitra, variabel `\partnerPosition` wajib diisi dengan jabatan yang dimiliki perwakilan mitra yang akan menandatangani laporan Anda. Jika program memiliki mitra, variabel `\partnerInstance` wajib diisi dengan instansi, perusahaan, atau program yang menjadi tempat kerja perwakilan mitra yang akan menandatangani laporan Anda.
- Pada bagian **”Informasi Pembimbing dan Penguji”**, isi nama lengkap dan gelar dosen penanggungjawab program Kampus Merdeka yang diambil (untuk mahasiswa Fasilkom UI) atau Pembimbing Akademik (untuk fakultas lain) pada `\pembimbingSatu`. Kemudian, isi nama lengkap perwakilan penyelia atau manajer dari mitra tempat kegiatan pada `\pembimbingDua`. Jika program tidak memiliki mitra (misalkan BYOC), kosongkan variabel `\pembimbingDua`. Kosongkan semua variabel lain pada bagian tersebut (namun jangan dihapus).

Konfigurasi untuk Kampus Merdeka tidak mendukung format Tugas Akhir Kelompok.

2. Setelah mengisi konfigurasi, Anda bisa periksa halaman-halaman awal dokumen. Jika terdapat ketidaksesuaian pada ukuran atau jarak antar elemen, Anda bisa mengatur melalui berkas-berkas yang ada di `src/00-frontMatter`. Halaman persetujuan yang dipakai di format Laporan Kerja Praktik ada di `src/00-frontMatter/pengesahanMBKM.tex`. Silakan perbesar atau perkecil ukuran yang ada pada kode `\vspace*{...}`, untuk menyesuaikan *spacing*. Jika ada perubahan kode yang signifikan, Anda bisa mengusulkan ke penyusun *template*. Keterangan lebih lanjut terkait cara kontribusi dapat dilihat di berkas `README.md` dan `CONTRIBUTING`.

BAB 6

PENUTUP

Pada bab ini, Penulis akan memaparkan kesimpulan penelitian dan saran untuk penelitian berikutnya.

6.1 Kesimpulan

Berikut ini adalah kesimpulan terkait pekerjaan yang dilakukan dalam penelitian ini:

1. Poin pertama

Penjelasan poin pertama.

2. Poin kedua

Penjelasan poin kedua.

Tulis kalimat penutup di sini.

6.2 Saran

Berdasarkan hasil penelitian ini, berikut ini adalah saran untuk pengembangan penelitian berikutnya:

1. Saran 1.

2. Saran 2.

DAFTAR REFERENSI

- Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. (2021). Training verifiers to solve math word problems.
- Gerganov, G. (2023). llama.cpp. <https://github.com/ggml-org/llama.cpp>. Accessed: November 27, 2025.
- Koto, F., Aisyah, N., Li, H., and Baldwin, T. (2023). Large language models only pass primary school exams in Indonesia: A comprehensive test on IndoMMLU. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Singapore. Association for Computational Linguistics.
- Mishra, S., Mitra, A., Varshney, N., Sachdeva, B., Clark, P., Baral, C., and Kalyan, A. (2022). Numglue: A suite of fundamental yet challenging mathematical reasoning tasks.
- Saparov, A. and He, H. (2023). Language models are greedy reasoners: A systematic formal analysis of chain-of-thought.
- Singapore, A. (2024). Sea-lion (southeast asian languages in one network): A family of large language models for southeast asia. <https://github.com/aisingapore/sealion>.
- Team, Q. (2024). Qwen2.5: A party of foundation models.
- Xu, J., Fei, H., Luo, M., Liu, Q., Pan, L., Wang, W. Y., Nakov, P., Lee, M., and Hsu, W. (2025). Aristotle: Mastering logical reasoning with A logic-complete decompose-search-resolve framework. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*.
- Yang, A., Yang, B., Hui, B., Zheng, B., Yu, B., Zhou, C., Li, C., Li, C., Liu, D., Huang, F., Dong, G., Wei, H., Lin, H., Tang, J., Wang, J., Yang, J., Tu, J., Zhang, J., Ma, J., Xu, J., Zhou, J., Bai, J., He, J., Lin, J., Dang, K., Lu, K., Chen, K., Yang, K., Li, M., Xue, M., Ni, N., Zhang, P., Wang, P., Peng, R., Men, R., Gao, R., Lin, R., Wang, S., Bai, S., Tan, S., Zhu, T., Li, T., Liu, T., Ge, W., Deng, X., Zhou, X., Ren, X., Zhang, X., Wei, X., Ren, X., Fan, Y., Yao, Y., Zhang, Y., Wan, Y., Chu, Y., Liu, Y., Cui, Z., Zhang, Z., and Fan, Z. (2024). Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

DAFTAR ISTILAH

L^AT_EX Sebuah *mark up language* yang didesain khusus untuk karya tulis ilmiah. 24, 25, 27, 28, 30

LAMPIRAN

Lampiran 1: CHANGELOG

@todo

Silakan hapus lampiran ini ketika Anda mulai menggunakan *template*.

Template versi terbaru bisa didapatkan di <https://gitlab.com/ichlaffterlalu/latex-skripsi-ui-2017>. Daftar perubahan pada *template* hingga versi ini:

- versi 1.0.3 (3 Desember 2010):
 - *Template* Skripsi/Tesis sesuai ketentuan *formatting* tahun 2008.
 - Bisa diakses di <https://github.com/edom/uistyle>.
- versi 2.0.0 (29 Januari 2020):
 - *Template* Skripsi/Tesis sesuai ketentuan *formatting* tahun 2017.
 - Menggunakan BibTeX untuk sitasi, dengan format *default* sitasi IEEE.
 - *Template* kini bisa ditambahkan kode sumber dengan *code highlighting* untuk bahasa pemrograman populer seperti Java atau Python.
- versi 2.0.1 (8 Mei 2020):
 - Menambahkan dan menyesuaikan tutorial dari versi 1.0.3, beserta cara kontribusi ke *template*.
- versi 2.0.2 (14 September 2020):
 - Versi ini merupakan hasil *feedback* dari peserta skripsi di lab *Reliable Software Engineering* (RSE) Fasilkom UI, semester genap 2019/2020.
 - BibTeX kini menggunakan format sitasi APA secara *default*.
 - Penambahan tutorial untuk `longtable`, agar tabel bisa lebih dari 1 halaman dan header muncul di setiap halaman.
 - Menambahkan tutorial terkait penggunaan BibTeX dan konfigurasi *header/footer* untuk pencetakan bolak-balik.
 - Label "Universitas Indonesia" kini berhasil muncul di halaman pertama tiap bab dan di bagian abstrak - daftar kode program.
 - *Hyphenation* kini menggunakan `babel Bahasa Indonesia`. Aktivasi dilakukan di `hype-indonesia.tex`.
 - Minor adjustment untuk konsistensi *license* dari *template*.
- versi 2.0.3 (15 September 2020):

- Menambahkan kemampuan orientasi *landscape* beserta tutorialnya.
- `\captionsource` telah diperbaiki agar bisa dipakai untuk `longtable`.
- Daftar lampiran kini telah tersedia, lampiran sudah tidak masuk daftar isi lagi.
- Nomor halaman pada lampiran dilanjutkan dari halaman terakhir konten (daftar referensi).
- Kini sudah bisa menambahkan daftar isi baru untuk jenis objek tertentu (custom), seperti: "Daftar Aturan Transformasi". Sudah termasuk mekanisme *captioning* dan tutorialnya.
- Perbaiki minor pada tutorial.
- versi 2.1.0 (8 September 2021):
 - Versi ini merupakan hasil *feedback* dari peserta skripsi dan tesis di lab *Reliable Software Engineering* (RSE) Fasilkom UI, semester genap 2020/2021.
 - Minor edit: "Lembar Pengesahan", dsb. di daftar isi menjadi all caps.
 - Experimental multi-language support (Chinese, Japanese, Korean).
 - *Support* untuk justifikasi dan word-wrapping pada tabel.
 - Penggunaan suffix "(sambungan)" untuk tabel lintas halaman. Tambahan support suffix untuk `\captionsource`.
- versi 2.1.1 (7 Februari 2022):
 - Update struktur mengikuti fork template versi 1.0.3 di <https://github.com/rkkautsar/edom/ui-thesis-template>.
 - *Support* untuk simbol matematis `amsfonts`.
 - Kontribusi komunitas terkait improvement GitLab CI, atribusi, dan format sitasi APA bahasa Indonesia.
 - Perbaiki tutorial berdasarkan perubahan terbaru pada versi 2.1.0 dan 2.1.1.
- versi 2.1.2 (13 Agustus 2022):
 - Modifikasi penamaan beberapa berkas.
 - Perbaiki beberapa halaman depan (halaman persetujuan, halaman orisinalitas, dsb.).
 - *Support* untuk lembar pengesahan yang berbeda dengan format standar, seperti Laporan Kerja Praktik dan Disertasi.
 - Kontribusi komunitas terkait kesesuaian dengan format Tugas Akhir UI, kelengkapan dokumen, perbaiki format sitasi, dan *quality-of-life*.
 - Perbaiki tutorial.
- versi 2.1.3 (22 Februari 2023):

- Dukungan untuk format Tugas Akhir Kelompok di Fasilkom UI.
- Dukungan untuk format laporan Kampus Merdeka Mandiri di Fasilkom UI.
- Minor *bugfix*: Perbaikan kapitalisasi variabel.
- Quality-of-Life: Pengaturan kembali `config/settings.tex`.
- Tutorial untuk beberapa *use case*.
- versi 2.2.0 (28 Agustus 2024):
 - Perbaikan format agar sesuai dengan format Tugas Akhir terbaru. Hal ini mencakup halaman judul, halaman pernyataan orisinalitas, header/footer, dan lampiran.
- versi 2.2.1 (16 Desember 2024):
 - *Bugfix*: isu *header* dan *footer* untuk halaman bolak-balik.
 - *Bugfix*: isu *auto-wrapping* pada kode yang tidak bisa berjalan sejak v2.2.0.
 - *Bugfix*: isu penomoran objek kustom yang tidak sesuai konvensi `[bab].[objek]`.
 - *Bugfix*: penomoran bab di Daftar Isi yang belum sesuai konvensi Tugas Akhir UI.
 - *Bugfix*: hal-hal lain pada *formatting* sesuai dengan permintaan dari Perpustakaan Fasilkom UI.
 - Perbaikan *formatting* untuk *landscape* dengan *library* `pdfscape`.
 - Perbaikan cara memasukkan sebuah persamaan ke dalam daftar persamaan.
 - Perbaikan penggunaan "saya" menjadi "kami" untuk dokumen-dokumen awal pada Tugas Akhir Kelompok.
 - Fitur baru: *Support* untuk *code highlighting* pada berbagai bahasa pemrograman yang tidak di-*support* secara *default* oleh *library listings*.
 - Fitur baru: *Support* untuk *glossary* (daftar istilah).
 - Perbaikan *major* pada tutorial, termasuk menampilkan contoh kode ke dalam PDF tutorial, dan pengaturan ulang subbab.

Lampiran hadir untuk menampung hal-hal yang dapat menunjang pemahaman terkait tugas akhir, namun akan mengganggu *flow* bacaan sekiranya dimasukkan ke dalam bacaan. Lampiran bisa saja berisi data-data tambahan, analisis tambahan, penjelasan istilah, tahapan-tahapan antara yang bukan menjadi fokus utama, atau pranala menuju halaman luar yang penting.

Subbab dari Lampiran 2

@todo

Isi subbab ini sesuai keperluan Anda. Anda bisa membuat lebih dari satu judul lampiran, dan tentunya lebih dari satu subbab.