UNIVERSITAS INDONESIA

KEMAMPUAN OPEN-WEIGHT LARGE LANGUAGE MODEL DALAM PENALARAN LOGIKA BERBASIS FRAMEWORK TRANSLATION-DECOMPOSITION-SEARCHRESOLVE PADA DATASET BERBAHASA INDONESIA

SKRIPSI

MIKHAEL DEO BARLI
1906350572

FAKULTAS ILMU KOMPUTER
PROGRAM STUDI ILMU KOMPUTER
DEPOK
BULAN TAHUN

UNIVERSITAS INDONESIA

KEMAMPUAN OPEN-WEIGHT LARGE LANGUAGE MODEL DALAM PENALARAN LOGIKA BERBASIS FRAMEWORK TRANSLATION-DECOMPOSITION-SEARCHRESOLVE 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 Open-Weight Large Language Model dalam Penalaran Logika Berbasis Framework Translation-Decomposition-SearchResolve 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 LaTeX untuk membuat dokumen tugas akhir.

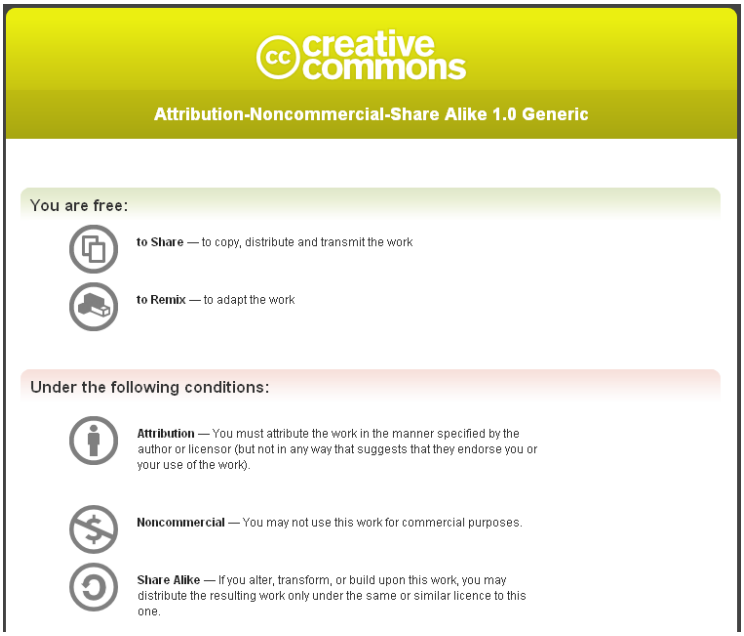
@todo

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 mengentahui 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

demi 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 Open-Weight Large Language Model dalam Penalaran Logika Berbasis Framework Translation-Decomposition-SearchResolve pada Dataset Berbahasa Indonesia

beserta 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 Open-Weight Large Language Model dalam Penalaran Logika Berbasis Framework Translation-Decomposition-SearchResolve 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 Open-Weight Large Language Model in Logical Inference Framework Translation-Decomposition-SearchResolve on Indonesian Language Dataset
Counselor	:	Pembimbing Pertama Anda

Abstract content.

Key words:
Keyword one, keyword two

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR KODE PROGRAM

DAFTAR LAMPIRAN

BAB 1
PENDAHULUAN

Bab ini memaparkan latar belakang, permasalahan, tujuan, batasan, manfaat, ringkasan metodologi, serta sistematika penulisan penelitian ini. Penelitian ini berfokus pada evaluasi kemampuan penalaran logis oleh *Large Language Models* (LLM) yang bersifat *open-weight* ketika bekerja pada dataset berbahasa Indonesia dengan *framework translation-decomposition-searchresolve* juga perbandingannya dengan *naive prompting*

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 melakukan inferensi yang benar dari himpunan premis dan aturan formal, masih menghadapi kendala pada akurasi dari hasil inferensi, terutama di kasus yang memerlukan normalisasi, dekomposisi, pencarian bukti, dan resolusi logika.

Penelitian sebelumnya sudah dilakukan pada berbagai dataset menggunakan framework yang sama. Dataset yang digunakan sebagian besar dalam bahasa Inggris, sehingga studi terhadap kemampuan penalaran LLM pada bahasa lain, termasuk Bahasa Indonesia, masih terbatas. Perbedaan struktur linguistik, idiom, dan masalah tokenisasi serta kualitas terjemahan dapat memengaruhi performa model setelah adaptasi lintas bahasa. Penelitian sebelumnya juga menggunakan *proprietary* LLM yang bukan *open-weight*, sehingga untuk mencari perbedaan dan perbandingan performa *apple-to-apple* tidak memungkinkan.

Dari *gap* penelitian yang sudah disebutkan, belum ada kajian untuk meneliti efektivitas *framework* pada dataset berbahasa Indonesia. Penelitian ini menjadi penting untuk mengevaluasi kemampuan LLM dalam inferensi pada dataset berbahasa Indonesia dan mengeksplorasi sebuah *framework* yang mengintegrasikan modul terjemahan dan normalisasi, dekomposisi logis, mekanisme pencarian bukti, serta resolusi logika. Diharapkan bahwa pendekatan terintegrasi ini dapat secara signifikan meningkatkan kemampuan penalaran LLM pada bahasa Indonesia.

1.2 Rumusan Masalah

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

1. Bagaimana perbandingan performa antara model open-weight berparameter rendah dalam inferensi dataset berbahasa Indonesia?
2. bagaimana efektivitas *framework translate* $\rightarrow$ *decompose* $\rightarrow$ *search* $\rightarrow$ *resolve* dalam meningkatkan akurasi inferensi pada data Bahasa Indonesia dibanding dengan penalaran secara langsung secara naive?

1.3 Tujuan Penelitian

Berikut adalah tujuan dari penelitian sesuai dengan latar belakang dan rumusan masalah

Tujuan:

1. Mengukur performa beberapa model pada tugas penalaran menggunakan metrik akurasi.
2. Mengevaluasi kemampuan penalaran logika LLM pada dataset berbahasa Indonesia menggunakan *framework* dan tanpa *framework*, seperti *naive prompting*
3. Menyediakan dataset terjemahan, skrip eksperimen, dan laporan yang dapat digunakan penelitian selanjutnya.

1.4 Batasan Penelitian

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

- **Dataset:** Fokus pada dataset diterjemahkan ke Bahasa Indonesia, yaitu ProntoQA saja
- **Model:** Eksperimen menggunakan model open-weight yang dapat dijalankan lokal maupun server, khususnya dengan kuantisasi. Model tersebut antara lain: Qwen2.5-7B-IT-GGUF, SEALIONv3-LLama-8B-IT-GGUF, dan SahabatAIv1-Llama-8B-IT-GGUF
- **Evaluasi:** Metrik utama adalah akurasi jawaban akhir terhadap ground truth.

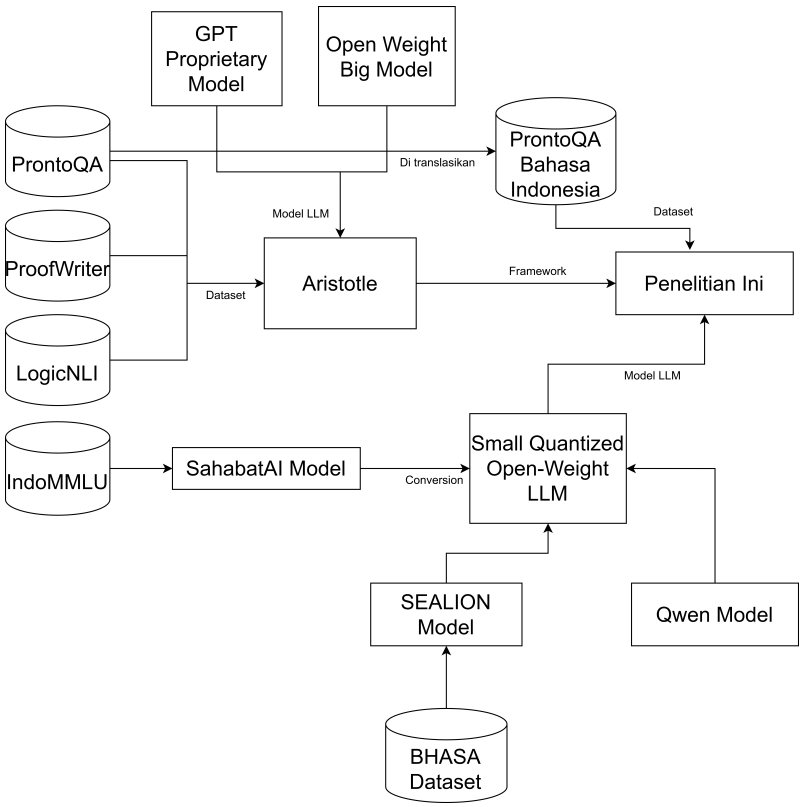
Referensi dari framework dan metode, termasuk skrip seperti `translate_decompose.py`, `negate.py`, dan `search_resolve.py`, serta utilitas evaluasi tersedia pada repositori eksperimen yang menjadi inspirasi implementasi ini, yaitu pada repositori Aristotle.

1.5 Manfaat Penelitian

Penelitian ini diharapkan memberikan kontribusi yang bermakna bagi berbagai pihak:

- **Bagi pengembangan ilmu pengetahuan:** Penelitian ini dapat menambah pemahaman tentang kemampuan penalaran logis LLM pada bahasa Indonesia, sebuah aspek yang masih jarang dikaji. Temuan ini dapat menjadi fondasi bagi penelitian lanjutan dalam evaluasi model bahasa pada tugas-tugas inferensi kompleks dalam bahasa lokal.
- **Bagi praktisi dan pengembang:** Hasil analisis perbandingan model dan efektivitas framework dapat menjadi panduan dalam memilih model open-weight yang tepat dan merancang pipeline pemrosesan bahasa untuk tugas inferensi logis berbahasa Indonesia, terutama dengan sumber daya komputasi terbatas.
- **Bagi komunitas penelitian terbuka:** Dataset ProntoQA yang diterjemahkan ke bahasa Indonesia, skrip eksperimen, serta laporan hasil penelitian akan dibagikan kepada publik. Kontribusi ini memungkinkan peneliti lain untuk mereplikasi, memvalidasi, dan melanjutkan penelitian dalam domain yang sama tanpa perlu melakukan terjemahan dan persiapan data dari awal.

1.6 Posisi Penelitian



Gambar 1.1: Diagram posisi penelitian yang dilakukan

Penelitian ini dilakukan menggunakan *framework* yang sudah diusulkan pada penelitian sebelumnya, tetapi penelitian ini menggunakan dataset berbahasa Indonesia dengan open-weight LLM berparameter rendah dengan kuantisasi.

1.7 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 CARA KERJA FRAMEWORK

  Bab ini mencakup metodologi atau langkah-langkah yang digunakan untuk melakukan

penelitian ini.

- Bab 4 HASIL EKSPERIMEN DAN ANALISA

  Bab ini mencakup eksperimen dan hasil eksperimen penelitian serta analisis dari hasil eksperimen tersebut

- PENUTUP

  Bab ini mencakup kesimpulan akhir penelitian dan saran untuk pengembangan berikutnya.

BAB 2
LANDASAN TEORI

Bab ini membangun kerangka teoretis yang mendasari analisis kemampuan *Large Language Models* (LLM) dalam melakukan inferensi logika, khususnya dalam konteks Bahasa Indonesia. Pembahasan mencakup prinsip formal dari (*First-Order Logic*), transformasi bentuk normal (PNF, Skolemisasi, CNF), serta arsitektur *Neuro-Symbolic* "Aristotle".

2.1 Aturan Inferensi dan Resolusi

2.1.1 Aturan Inferensi Klasik

Sistem ini secara implisit mencakup aturan-aturan logika klasik:

- **Modus Ponens:** Jika $P \rightarrow Q$ dan P, maka Q.
- **Modus Tollens:** Jika $P \rightarrow Q$ dan $\neg Q$, maka $\neg P$.
- **Silogisme Hipotetis:** Jika $P \rightarrow Q$ dan $Q \rightarrow R$, maka $P \rightarrow R$.
- **Silogisme Disjungtif:** Jika $P \vee Q$ dan $\neg P$, maka Q.

2.1.2 Prinsip Resolusi dan Unifikasi

Dalam pembuktian teorema aturan inferensi, aturan-aturan di atas dapat digeneralisasi menjadi satu aturan tunggal yang disebut **Resolusi**.

Untuk resolusi memerlukan proses **Unifikasi**. Unifikasi adalah proses mencari substitusi θ (pemetaan variabel ke term) sehingga dua literal menjadi identik.

$$\frac{L_1 \vee A, \quad \neg L_2 \vee B}{A \vee B\theta} \tag{2.1}$$

Dimana θ adalah *Most General Unifier* (MGU) dari L_1 dan L_2.

Contoh implementasi:

- Premis: "Semua orang menyukai Budi" ($\forall x Loves(x, Budi)$).
- Query: "Apakah Ali menyukai Budi?" ($\neg Loves(Ali, Budi)$).
- Unifikasi: Substitusi $\{x \,/\, Ali\}$ membuat $Loves(x, Budi)$ berlawanan langsung dengan $\neg Loves(Ali, Budi)$.

- Hasil: Klausa kosong ($\perp$), yang berarti terbukti terjadi kontradiksi, sehingga Ali menyukai Budi.

2.2 (*First-Order Logic*)

FOL atau sering disebut Kalkulus Predikat, berfungsi sebagai fondasi untuk merepresentasikan pengetahuan secara tidak ambigu. Berbeda dengan Logika Proposisi yang memperlakukan pernyataan sebagai objek yang tak terbagi, FOL memodelkan dunia sebagai domain objek, properti, dan hubungan antar objek tersebut.

2.2.1 Definisi dan Sintaks

Secara formal, FOL adalah sistem logika deduktif yang memperluas logika proposisi dengan memperkenalkan:

- **Variabel (x, y, z):** Simbol yang mewakili objek dalam suatu domain.
- **Predikat ($P(x, y)$):** Fungsi yang memetakan objek ke nilai kebenaran (*True/False*).
- **Kuantor ($\forall, \exists$):** Operator yang menentukan *range* variabel terhadap domain.

Alfabet dari FOL (Σ) terdiri dari:

1. **Simbol Logika:**
   - Konektif: $\neg$ (Negasi), $\wedge$ (Konjungsi), $\vee$ (Disjungsi), $\rightarrow$ (Implikasi).
   - Kuantor Universal ($\forall$): Dibaca "Untuk semua".
   - Kuantor Eksistensial ($\exists$): Dibaca "Terdapat".
2. **Simbol Non-Logika:** Konstanta (objek spesifik seperti "Wumpus", "Merah") dan Predikat

2.2.2 Semantik dan Interpretasi

Validitas dalam FOL ditentukan oleh sebuah **Model** $\mathcal{M} = \langle \mathcal{D}, I \rangle$, di mana:

- $\mathcal{D}$ adalah **Domain** (misalnya, himpunan semua entitas dalam basis data kependudukan).
- I adalah **Interpretasi** yang memetakan simbol konstanta ke elemen di $\mathcal{D}$ dan simbol predikat ke relasi di $\mathcal{D}$.

Dalam konteks Bahasa Indonesia, struktur kalimat *Subjek-Predikat-Objek* dapat dipetakan ke dalam FOL, namun tantangan muncul pada ambiguitas konteks. Misalnya, kalimat "Ga-

jah itu besar" bisa berarti konstanta ($Big\text{gajah}_1$) atau aturan universal ($\forall x Elephant x \rightarrow Big x$), juga misalnya subjek plural dan singular yang perlu di normalisasi. Modul dekomposisi dalam penelitian ini bertugas menyelesaikan ambiguitas ini menjadi representasi FOL yang benar.

2.3 Bentuk Normal: Standardisasi untuk Algoritma

Algoritma penalaran, khususnya resolusi, tidak dapat bekerja secara efisien pada formula FOL yang tidak terstruktur atau tidak *strict*. Formula harus dikonversi ke dalam bentuk standar atau *Normal Forms*.

2.3.1 Bentuk Normal Prenex (PNF)

Langkah pertama standardisasi adalah *Prenex Normal Form* (PNF). Sebuah formula dikatakan dalam bentuk PNF jika seluruh kuantor ditarik ke depan (prefix), meninggalkan bagian bebas-kuantor (matriks) di belakangnya.

Bentuk umum:

$$Q_1 x_1 Q_2 x_2 \ldots Q_n x_n \; \mathcal{M} \tag{2.2}$$

Dimana $Q_i \in \{\forall, \exists\}$ dan $\mathcal{M}$ adalah matriks tanpa kuantor. Contoh transformasi dalam Bahasa Indonesia:

- *Asli:* "Tidak benar bahwa semua orang menyukai durian." ($\neg \forall x Likes x, Durian$).
- *PNF:* "Terdapat seseorang yang tidak menyukai durian." ($\exists x \neg Likes x, Durian$).

2.3.2 Skolemisasi (Skolemization)

Skolemisasi adalah proses menghilangkan **Kuantor Eksistensial** ($\exists$) untuk menghasilkan formula yang *equisatisfiable* (memiliki keterpenuhan yang setara). Kuantor eksistensial digantikan oleh **Konstanta Skolem** atau **Fungsi Skolem**.

Algoritma:

1. Jika $\exists y$ tidak berada dalam lingkup $\forall$, ganti y dengan konstanta baru c.
2. Jika $\exists y$ berada dalam lingkup $\forall x_1, \ldots, \forall x_k$, ganti y dengan fungsi $f x_1, \ldots, x_k$.

Contoh kasus:

”Setiap warga negara memiliki sebuah NIK.”
FOL: $\forall x\,(Citizen(x) \rightarrow \exists y\,(NIK(y) \wedge Has(x,y)))$
Skolemisasi: $\forall x\,(Citizen(x) \rightarrow (NIK(\text{nikOf}(x)) \wedge Has(x, \text{nikOf}(x))))$

Di sini, nikOf(x) adalah fungsi Skolem yang memetakan setiap warga ke NIK spesifik mereka.

2.3.3 Bentuk Normal Konjungtif (CNF)

Conjunctive Normal Form (CNF) adalah representasi akhir yang dibutuhkan oleh mesin inferensi. CNF adalah konjungsi dari klausa-klausa, di mana setiap klausa adalah disjungsi dari literal.

$$L_{1,1} \vee \cdots \vee L_{1,n} \wedge L_{2,1} \vee \cdots \vee L_{2,m} \tag{2.3}$$

2.4 Paradigma Neuro-Symbolic

Integrasi LLM dengan penalaran logika formal, sering disebut juga sebagai *Neuro-Symbolic AI*, merupakan pergeseran penting dalam riset *artificial intelligence*. Komponen ”neuro” (model *deep learning* seperti Qwen atau Llama) unggul dalam pengenalan pola, namun memiliki keterbatasan dalam melakukan inferensi multi-langkah yang *strict* dan deterministik (Saparov and He (2023)). Keterbatasan ini menjadi poin penting pada bahasa dengan sumber daya rendah hingga menengah seperti Bahasa Indonesia, di mana korpus pelatihan untuk penalaran kompleks jauh lebih sedikit dibandingkan Bahasa Inggris.

Penelitian ini mengadopsi kerangka kerja yang tidak hanya mengandalkan prediksi token probabilitas, melainkan memisahkan proses menjadi translasi simbolik, pencarian terstruktur, dan resolusi formal, sebagaimana diusulkan dalam arsitektur **Aristotle** (Xu et al. (2025)).

2.5 Kerangka Kerja Aristotle

Penelitian ini mengimplementasikan *framework* **Aristotle** (Xu et al. (2025)) yang membagi proses inferensi menjadi empat modul utama:

2.5.1 Logika Translasi (*Logical Translation*)

Modul ini menggunakan LLM *open-weight* untuk menerjemahkan premis bahasa natural menjadi FOL

- **Input:** Data poin dalam bahasa Indonesia
- **Proses:** Menerjemahkan data poin menjadi 3 bagian, sesuai dengan formula, yaitu premis dari konteks $\rightarrow$ fakta dan aturan, pertanyaan dari konteks $\rightarrow$ konjektur

2.5.2 Dekomposer (*Decomposer*)

Modul ini menggunakan LLM *open-weight* untuk dekomposisi aturan yang sudha berbentuk FOL ke dalam PNF, CNF, dan Skolemisasi

- **Input:** data poin yang sudah melewati Logical Translation dalam bentuk FOL
- **Proses:** Dekomposisi data poin yang sudha menjadi FOL ke dalam bentuk NF, lalu Skolemisasi, dan terakhir CNF.

2.5.3 Penyelesai Pencarian (*Search Router*)

Mesin inti inferensi terdiri dari dua sub-komponen:

1. ***Logical Search Router*:** Memilih klausa mana yang relevan untuk diproses. Hal ini mencegah terjadinya ledakan kombinatorial yang mengakibatkan proses komputasi yang lama dan pencarian bukti yang sesuai dengan pembuatan set untuk menyimpan premis yang sudah ditelusuri dan memberi batasan banyaknya ronde dalam pencarian bukti
2. ***Logical Resolver*:** Menerapkan aturan resolusi secara iteratif hingga ditemukan kontradiksi (untuk pembuktian salah) atau hingga ruang pencarian habis.

2.5.4 Resolusi (*Resolver*)

Sistem memverifikasi sebuah hipotesis S melalui dua jalur paralel dengan menerapkan pembuktian dengan kontradiksi (*Proof By Contradiction*)

- **Jalur 1 :** Asumsikan S. Jika ditemukan kontradiksi, maka S adalah **Benar**.
- **Jalur 2 :** Asumsikan $\neg S$ (Negasi S). Jika ditemukan kontradiksi, maka S adalah **Benar**.

BAB 3
CARA KERJA FRAMEWORK

Bab ini menguraikan rancangan operasional yang digunakan untuk menjawab rumusan masalah penelitian. Metodologi ini disusun untuk menguji secara empiris apakah pendekatan *Neuro-Symbolic*, khususnya kerangka kerja Aristotle yang telah dibahas pada Bab 2, dapat mengatasi kelemahan penalaran probabilistik pada *open-weight Large Language Models* (LLM) dalam konteks Bahasa Indonesia.

Pendekatan penelitian ini bersifat kuantitatif-eksperimental. Fokus utamanya adalah mengukur akurasi penalaran logis yang diperoleh melalui manipulasi struktur *prompting* (dari Naive ke Neuro-Symbolic) dan dampaknya ketika model dijalankan dengan sumber daya terbatas (melalui kuantisasi).

3.1 Desain Eksperimen

Eksperimen dirancang dengan membandingkan performa model dalam dua skenario inferensi:

1. **Skenario Pertama (*Naive Prompting*):** Pada skenario ini, model diuji kemampuan "intuisi"-nya. Masalah logika diberikan dalam bentuk narasi langsung dengan instruksi standar dan beberapa contoh pengerjaan (*few-shot*). Skenario ini merepresentasikan cara penggunaan LLM pada umumnya, di mana model dipaksa melakukan logika secara implisit di dalam *hidden states*-nya.
2. **Skenario Kedua (*Framework Aristotle*):** Skenario ini menerapkan (*pipeline*) yang memaksa model untuk mengeksternalisasi proses berpikirnya ke dalam simbol-simbol logika formal. Sesuai landasan teori, proses ini dipecah menjadi tahapan sekuensial: Translasi $\rightarrow$ Dekomposisi $\rightarrow$ Pencarian Bukti $\rightarrow$ Resolusi.

Struktur variabel penelitian didefinisikan sebagai berikut:

- **Variabel Bebas (*Independent Variables*):**
  - **Model:** Tiga kategori model yang mewakili spektrum pemahaman bahasa: Global (Qwen), Regional (SEA-LION), dan Nasional (SahabatAI).
  - **Metode Prompting:** *Naive* vs *Aristotle Framework*.

  - **Tingkat Presisi:** *Unquantized* (FP16/BF16) vs Terkuantisasi (4-bit GGUF)
- **Variabel Terikat (*Dependent Variables*):**
  - **Akurasi (*Accuracy*):** Ketepatan hasil akhir (Benar/Salah) yang diverifikasi terhadap *ground truth*.
  - **Kepatuhan Format (*Parsability*):** Kemampuan model menghasilkan sintaks logika (FOL/CNF) yang valid secara komputasi. Kegagalan menghasilkan format yang benar dianggap sebagai kegagalan penalaran.
- **Variabel Kontrol:** Untuk memastikan bahwa variasi hasil murni disebabkan oleh variabel bebas, seluruh eksperimen dijalankan secara deterministik dengan parameter yang tetap. Hal ini meniadakan faktor (*randomness*) dalam resolusi inferensi.

3.2 Instrumen Data dan Adaptasi Linguistik

Penelitian ini menggunakan dataset **ProntoQA** (*Prompting with Ontologies for QA*). Pemilihan dataset ini didasarkan pada kebutuhan untuk menguji kemampuan LLM dalam deduksi sintetis, terlepas dari pengetahuan dunia nyata dari LLM tersebut. ProntoQA menggunakan ontologi fiktif (misalnya: *"Setiap Wumpus adalah Jompus"*), sehingga model tidak dapat menjawab dengan mengandalkan hafalan dalam *pre-training*.

3.2.1 Proses Adaptasi ke Bahasa Indonesia

Proses penerjemahan dilakukan secara otomatis dengan *prompting* dan memeberikan beberapa contoh dalam proses tranlasinya. Penerjemahan dilakukan dengan *open-weight* LLM yaitu **Gemma-SEA-LION-v3-9B-IT**. Model dipilih karena kemampuan yang baik dalam translasi dari bahasa Inggris ke bahasa Indonesia sesuai dengan SEA-HELM *leaderboard*

- **Konsistensi Kuantor:** Frasa "Every X is Y" diterjemahkan menjadi "Setiap X adalah Y" atau "Semua X adalah Y" untuk menjaga pola *Universal Quantifier* ($\forall$).
- **Konsistensi Negasi:** Struktur "X is not Y" dipetakan menjadi "X bukan Y" atau "X tidak Y".
- **Konsistensi Plural:** Subjek seperti "Wumpuses", "Jompuses", dan subjek lainnya, tidak diterjemahkan imbuhannya untuk menjaga struktur kalimat.

3.3 Konfigurasi Model

Mengingat bahwa adanya batasan sumber daya komputasi, maka eksperimen dijalankan menggunakan pustaka `llama.cpp` Gerganov (2023) yang dioptimalkan untuk inferensi model.

3.3.1 Pemilihan Model

Model yang dipilih mewakili tiga tingkatan spesialisasi bahasa untuk menguji hipotesis bahwa pemahaman dalam bahasa lokal membantu proses dekomposisi logika:

1. **Qwen2.5-7B-Instruct (Representasi Global):** Model ini dipilih karena performanya yang unggul dalam benchmark logika matematika global, menjadi titik ukur batas atas kemampuan model *open-weight* saat ini.
2. **SEA-LION-v3-Llama-8B-Instruct (Representasi Regional):** Model yang telah melalui *continued pre-training* dengan data Asia Tenggara, diharapkan memiliki pemahaman semantik yang lebih baik terhadap struktur kalimat regional.
3. **SahabatAI-v1-Llama-8B-Instruct (Representasi Nasional):** Model yang dikhususkan untuk Bahasa Indonesia, juga sudah melalui *continued pre-training*, digunakan untuk melihat apakah spesialisasi bahasa yang mendalam dapat menyelesaikan penalaran lebih baik dalam tugas logika.

3.3.2 Parameterisasi Inferensi

Setiap model dijalankan dengan pengaturan parameter yang sama / konsisten untuk memastikan hasil eksperimen mencerminkan pengaruh variabel bebas, bukan perbedaan konfigurasi teknis. Strategi deterministik diterapkan untuk menghilangkan unsur *randomness* dalam proses inferensi. Parameter utama yang dikontrol adalah:

- ***Temperature*:** Ditetapkan ke 0.0 sehingga model hanya memilih token dengan probabilitas tertinggi tanpa penambahan noise acak (*greedy decoding*). Hal ini memastikan setiap prompt menghasilkan output yang identik.
- ***Max Tokens*:** Dibatasi sesuai dengan panjang output yang wajar untuk setiap tahap dalam pipeline, mencegah generasi yang berlebihan. Adapun token yang dibutuhkan untuk setiap tahap inferensi dalam *framework* antara lain: proses translasi ke FOL memerlukan output setidaknya 400 token, proses dekomposisi setidaknya memerlukan

output 700 token, dan proses pencarian bukti setidaknya memerlukan 1000 token., sehingga ***max tokens*** yang di set untuk semua proses adalah 2500 untuk mengatasi *overhead* atau halusinasi jika model tidak secara *strict* mengikuti format.

- ***n_ctx*:** Ukuran konteks diatur untuk menentukan berapa banyak token sebelumnya yang dapat dipertimbangkan model saat menghasilkan respons. Nilai yang lebih besar memungkinkan model memahami konteks yang lebih panjang, namun meningkatkan penggunaan memori. ***n_ctx*** di set ke 0, sehingga ukuran konteks akan sesuai dengan kemampuan model masing-masing.
- ***n_gpu_layers*:** Menentukan jumlah layer model yang dijalankan di GPU untuk akselerasi. Nilai yang lebih tinggi mempercepat inferensi tetapi memerlukan VRAM yang lebih besar. Nilai -1 berarti semua komputasi dilakukan di CPU.

Konfigurasi lengkap ini disajikan dalam kode berikut:

```
class LlamaCPPBackend:
    def __init__(self, local_model_path: str):
        """
        local_model_path: Must point to a specific .gguf FILE, not just a directory.
        """
        if not LLAMACPP_AVAILABLE: raise ImportError("llama-cpp-python not installed. Run 'pip install llama-cpp-python'")

        # n_gpu_layers=-1 means offload ALL layers to GPU
        self.llm = Llama(
            model_path=local_model_path,
            n_ctx=0,
            n_gpu_layers=-1,
            verbose=False
        )

    def generate(self, prompt: str, max_new_tokens: int = 512, temperature: float = 0.0, **kwargs) -> str:
        output = self.llm(
            prompt,
            max_tokens=max_new_tokens,
            stop=[],
            echo=True, # Return prompt + completion to match others
            temperature=temperature
        )
        return output['choices'][0]['text']
```

Kode 3.1: Konfigurasi parameter deterministik untuk meniadakan halusinasi kreatif

3.4 Prosedur Eksekusi: Pipeline Aristotle

Berbeda dengan *Naive Prompting* yang hanya berupa satu kali pemanggilan (*single-turn*), implementasi kerangka kerja Aristotle menuntut beberapa pemanggilan model (*chain-of-calls*).

3.4.1 Tahap 1: Translasi Logika (Logical Translator)

Langkah pertama bertujuan memetakan kalimat Bahasa Indonesia yang sering kali implisit (tanpa penanda waktu atau subjek yang jelas) menjadi representasi *First Order Logic* (FOL). Model diberikan instruksi untuk mengidentifikasi:

- **Fakta Atomik:** Pernyataan spesifik tentang entitas (misal: $Pahit(Bernard, False)$).
- **Aturan Implikasi:** Hubungan sebab-akibat (misal: $Dermawan(x, False)$ ¿¿¿ $Murni(Axel, False)$).

3.4.2 Tahap 2: Dekomposisi dan Normalisasi

Pada tahap ini, output FOL diproses lebih lanjut untuk memenuhi standar mesin pembukti teorema. Model diinstruksikan untuk melakukan konversi ke *Prenex Normal Form* (PNF) dan kemudian ke *Conjunctive Normal Form* (CNF). Proses penting di sini adalah **Skolemisasi**, yaitu penghapusan kuantor eksistensial ($\exists$) yang sering menjadi sumber ambiguitas bagi model bahasa. Dalam dataset ProntoQA, tidak perlu dilakukan skolemisasi karena premis-premis dari dataset tersebut tidak memiliki kuantor eksistensial.

- **PNF:** Konversi FOL ke bentuk PNF, di mana semua kuantor diletakkan di awal formula, diikuti oleh matriks bebas kuantor.
- **Skolemisasi:** Penghapusan kuantor eksistensial ($\exists$) dengan mengganti variabel eksistensial dengan fungsi Skolem. Meskipun dataset *ProntoQA* tidak memerlukan langkah ini, tetpai inklusinya memastikan pipeline dapat menangani formula umum dengan kuantor campuran untuk dataset lain, seperti *ProofWriter* dan *LogicNLI*
- **CNF:** Transformasi akhir ke CNF, di mana formula diekspresikan dalam bentuk konjungsi (AND) dan disjungsi (OR) antar premis / kalimat.

3.4.3 Tahap 3: Pencarian dan Resolusi (Search & Resolve)

Ini adalah inti dari arsitektur neuro-symbolic. Alih-alih membiarkan model menebak jawaban akhir, model diminta bertindak sebagai mesin *Resolution Prover*.

1. **Clause Selection:** Model memilih premis yang relevan dari premis yang sudah didekomposisi dan disimpan ke memori.
2. **Resolution Step:** Model menerapkan aturan inferensi (seperti Modus Ponens atau Silogisme) untuk menurunkan klausa baru.
3. **Contradiction Check:** Proses berhenti ketika ditemukan kontradiksi (misal: P dan $\neg P$ hadir bersamaan), yang membuktikan bahwa asumsi awal salah / terjadinya kontradiksi.

Mekanisme injeksi instruksi ke dalam *prompt template* untuk setiap tahap diimplementasikan menggunakan kode Python berikut:

```
def construct_prompt_a(self, record, in_context_examples_trans):
    full_prompt = in_context_examples_trans
    if self.dataset_name == "LogicNLI":
        context = "\n".join(record['facts'] + record['rules'])
        question = record['conjecture']
    else:
        context = record['context']
        question = re.search(r'\?(.*)', record['question'].strip()).group(1).strip()
    full_prompt = full_prompt.replace('[[PREMISES]]', context)
    full_prompt = full_prompt.replace('[[CONJECTURE]]', question)
    return full_prompt
```

Kode 3.2: Mekanisme injeksi data poin ke dalam template prompt Aristotle

3.5 Teknik Evaluasi dan Analisis Data

Untuk menjamin objektivitas, evaluasi tidak dilakukan secara manual melainkan menggunakan sistem ekstraksi berbasis pola (*Regular Expression* / Regex).

3.5.1 Parsing Bertingkat (Multi-stage Parsing)

Sistem evaluasi dirancang untuk ”menangkap” struktur logika dari output teks model. Kegagalan model dalam mematuhi format yang diminta pada tahap apa pun akan langsung dianggap sebagai kesalahan (*failure*), tanpa upaya perbaikan manual. Ini adalah standar ketat yang diterapkan untuk menguji keandalan model sebagai komponen sistem logika.

- **Ekstraksi FOL:** Mengambil Fakta, Aturan, dan Konjektur sesuai bloknya

```
def extract_facts_rules_conjecture(self, content, context_sentence_count=None):
    # Clean invisible characters
    content = (content or "").replace('\u200b', '').replace('\ufeff', '')

    prompt_marker = re.compile(
        r'Di bawah ini(?:\s+adalah(?:\s+yang\s+perlu\s+Anda\s+terjemahkan)?)?:?',
        re.IGNORECASE
    )
    m_prompt = prompt_marker.search(content)
    search_start_pos = m_prompt.end() if m_prompt else 0

    block_header = re.compile(r'\*{0,3}Bentuk Akhir\*{0,3}', re.IGNORECASE)
    m_block = block_header.search(content, pos=search_start_pos)

    if m_block:
        area = content[m_block.end():]
    else:
        area = content[search_start_pos:]

    # Define Patterns for possible headers
    # Include the "End Markers" as a header type.
    patterns = {
        "facts": re.compile(r'(?:Fakta|Facts)\s*[:\-]?\s*', re.IGNORECASE),
        "rules": re.compile(r'(?:Aturan|Rules)\s*[:\-]?\s*', re.IGNORECASE),
        "conj": re.compile(r'(?:Konjektur|Conjecture)\s*[:\-]?\s*', re.IGNORECASE),
        "stop": re.compile(r'(?:\*{0,3}\s*Akhir Blok\s*\*{0,3}|###|```|-{3,})',
re.IGNORECASE)
    }

    def extract_section(target_key):
        start_match = patterns[target_key].search(area)
        if not start_match:
            return ""

        content_start_idx = start_match.end()

        # Find the next header
        next_indices = []
        for key, pat in patterns.items():
            m = pat.search(area, pos=content_start_idx)
            if m:
                next_indices.append(m.start())

        # If found upcoming headers, stop at the nearest one (min index).
        # If no headers found, go to end of string.
        if next_indices:
            cutoff_idx = min(next_indices)
            raw_text = area[content_start_idx:cutoff_idx]
        else:
            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 3.3: Regex untuk ekstraksi blok Translasi

- **Ekstraksi CNF:** mengambil hasil akhir dari dekomposisi dari Aturan FOL.

```
1  def post_process_decompose(self, content, rules_count=None):
2
3      content = (content or "").replace('\u200b', '').replace('\ufeff', '')
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
```

```
    )

    boundary_re = re.compile(boundary_pattern, flags=re.IGNORECASE)

    def extract_after_label(label_re):
        """Finds label, returns text until next boundary."""
        lab_match = label_re.search(area)
        if not lab_match:
            return None
        start = lab_match.end()
        bound = boundary_re.search(area, pos=start)
        end = bound.start() if bound else len(area)
        return area[start:end].strip()

    cnf_raw = extract_after_label(cnf_label_re)
    skolem_raw = extract_after_label(skolem_label_re)

    def to_lines(raw):
        if not raw:
            return []
        return [ln.strip() for ln in raw.splitlines() if ln.strip()]

    cnf_lines = to_lines(cnf_raw)
    skolem_lines = to_lines(skolem_raw) if skolem_raw else None

    print(f"CNF Raw: {cnf_lines}")
    print(f"Skolem Raw: {skolem_lines}")

    return cnf_lines, skolem_lines
```

Kode 3.4: Regex untuk ekstraksi blok Dekomposisi

- **Ekstraksi Resolusi:** Mendeteksi klausa baru dan kesimpulan akhir dalam label kecukupan.

```
def post_process_logic_solver(self, response_d):
    content = response_d
    marker_pattern = r'(.*?)Dibawah ini tugas yang perlu Anda lakukan(.*?)'
    marker_match = re.search(marker_pattern, content, flags=re.IGNORECASE)
    search_area = content[marker_match.end():] if marker_match else content

    final_block_pattern = (
        r'\*{0,3}(?:Bentuk Akhir)\*{0,3}\s*'
        r'(.*?)'
        r'(?=(\*{0,3}(?:Akhir Blok)\*{0,3})|'
        r'$)' # or end of string
    )

    final_block_match = re.search(final_block_pattern, search_area, flags=re.DOTALL |
re.IGNORECASE)

```

```
    if not final_block_match:
        return [], None

    block = final_block_match.group(1)

    block_clean = block.strip()
    print(f"\n\nCHOSEN BLOCK:\n\n{block_clean}\n")
    print("END OF CHOSEN BLOCK\n\n")

    clause_pos = re.search(r'Clause\s*Baru', block_clean, flags=re.IGNORECASE)
    clause_after = block_clean[clause_pos.end():]
    m_new = re.search(r'\{(.*?)\}', clause_after, flags=re.DOTALL)

    if not m_new:
        raise ValueError(f"'Clause Baru:' with '{{...}}' not found in expected form.")

    new_clause = m_new.group(1).strip()

    label_pos = re.search(r'Label\s*Cukup', block_clean, flags=re.IGNORECASE)
    label_after = block_clean[label_pos.end():]
    m_label = re.search(r'\[(.*?)\]', label_after, flags=re.DOTALL)
    if not m_label:
        raise ValueError(f"'Label Cukup' with '[...]' not found in expected form
[True|False].")

    sufficiency_label = m_label.group(1).strip()

    return {
        "new_clause": new_clause,
        "sufficiency_label": sufficiency_label,
    }
```

Kode 3.5: Regex untuk memvalidasi langkah Resolusi

3.5.2 Matrix Keberhasilan

Kinerja dari sebuah LLM diukur menggunakan ***Accuracy***. Berikut adalah matrix kebenaran dari sebuah jawaban:

$$A = \begin{cases} \text{True}, & P_n \vdash S_n \wedge P_n \nvdash \neg S_n \\ \text{False}, & P_n \nvdash S_n \wedge P_n \vdash \neg S_n \\ \text{Unknown}, & P_n \nvdash S_n \wedge P_n \nvdash \neg S_n \\ \text{Self-Contradictory}, & P_n \vdash S_n \wedge P_n \vdash \neg S_n \end{cases}$$

A merupakan output dari matrix kebenaran tersebut dan merupakan jawaban akhir dari sebuah data poin. P_n merupakan premis, S_n merupakan konjektur dan $\neg S_n$ merupakan konjektur yang dinegasi yang diinisialisasi dalam dua jalur pencarian bukti.

BAB 4
HASIL EKSPERIMEN DAN ANALISA

Bab ini menjelaskan tentang eksperimen dan hasil eksperimen dari penelitian

4.1 Naive Prompting

Tabel 4.1: Hasil Eksperimen dengan Naive Prompting

	Qwen2.5 7B-Instruct-GGUF	**SEA-LION v3-Llama-8B-GGUF**	**SahabatAI v1-Llama-8B-GGUF**
Naive Prompting			
After Answer	51.40%	56.20%	61.40%
Before Answer	81.00%	76.80%	67.80%

4.2 Aristotle

Tabel 4.2: Hasil Eksperimen dengan Aristotle Framework

	Qwen2.5 7B-Instruct-GGUF	**SEA-LION v3-Llama-8B-GGUF**	**SahabatAI v1-Llama-8B-GGUF**
Aristotle	14.00%	81.60%	61.20%

BAB 5
PENUTUP

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

5.1 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.

5.2 Saran

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

1. Saran 1.
2. Saran 2.

DAFTAR REFERENSI

Gerganov, G. (2023). llama.cpp. https://github.com/ggml-org/llama.cpp. Accessed: November 30, 2025.

Saparov, A. and He, H. (2023). Language models are greedy reasoners: A systematic formal analysis of chain-of-thought.

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*.

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.
  - Perbaikan 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.
  - Perbaikan tutorial berdasarkan perubahan terbaru pada versi 2.1.0 dan 2.1.1.
- versi 2.1.2 (13 Agustus 2022):
  - Modifikasi penamaan beberapa berkas.
  - Perbaikan 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, perbaikan format sitasi, dan *quality-of-life*.
  - Perbaikan 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* `pdflscape`.
  - 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.