BAB 4
HASIL PENELITIAN

4.1 RINGKASAN HASIL PENELITIAN

Penelitian ini melakukan *fine-tuning* model Mistral-7B-Instruct menggunakan teknik QLoRA untuk mengadaptasi model terhadap domain pedoman akademik Universitas Jenderal Achmad Yani Yogyakarta. Dataset yang digunakan terdiri dari 1.626 entri (413 original + 1.213 variasi augmentasi), dibagi menjadi 1.504 entri training dan 122 entri evaluasi. Fine-tuning dilakukan dengan kuantisasi 4-bit selama 1 *epoch* (47 langkah, 37 menit), melatih 92 juta dari 7,34 miliar parameter. *Training loss* turun dari 2,87 menjadi 0,99, *validation loss* 0,94. Evaluasi BERTScore pada 15 sampel menghasilkan F1 Score 0,6421, *Precision* 0,6323, dan *Recall* 0,6525. Model menunjukkan kemampuan adaptasi pada domain pedoman akademik dengan tingkat kesamaan semantik yang cukup memadai, meskipun masih memerlukan perbaikan untuk mencapai performa optimal.

4.2 DATASET DAN PERSIAPAN DATA

Bagian ini menjelaskan proses persiapan data yang dilakukan sebelum pelatihan model, dimulai dari deskripsi dataset, teknik augmentasi variasi gaya pertanyaan, pembagian data latih dan evaluasi, hingga konversi ke format yang sesuai untuk model LLM. Tahapan ini merujuk pada prinsip CRISP-ML, khususnya pada fase *data understanding* dan *data preparation*.

Dataset yang digunakan bersumber dari dokumen resmi pedoman akademik Universitas Jenderal Achmad Yani Yogyakarta, yang telah disusun dalam format instruksional dan tersedia secara publik di Kaggle melalui tautan berikut: kaggle.com/datasets/riakristi/dataset-pedoman-akademik-unjaya.

4.2.1 Deskripsi Dataset

Dataset yang digunakan dalam penelitian ini bersumber dari dokumen Pedoman Akademik Unjaya Tahun 2024. Dokumen tersebut memuat ketentuan,

peraturan, serta informasi penting terkait sistem akademik yang berlaku di lingkungan Unjaya.

Dataset disusun dalam format pasangan *instruction-response*, di mana setiap entri terdiri dari pertanyaan (*instruction*) yang mencerminkan bentuk tanya dari pengguna, dan jawaban (*response*) yang berasal dari kutipan langsung atau interpretasi isi dokumen. Setiap respons dilengkapi dengan keterangan sumber pasal sebagai referensi. Dataset disusun dalam format CSV dengan struktur kolom sebagai berikut:

a. ID: Id unik setiap entri
b. *Instruction*: Pertanyaan yang diajukan
c. *Response*: Jawaban berdasarkan isi pedoman
d. Sumber: Referensi ayat dalam dokumen
e. ID_Original: Penanda entri original sebelum dilakukan augmentasi
f. Tipe_Variasi: Jenis variasi, seperti original atau gaya_pertanyaan

Jumlah total data original adalah sebanyak 413 entri**,** yang mencakup berbagai aspek dalam pedoman akademik, mulai dari pendaftaran, kurikulum, evaluasi pembelajaran, hingga yudisium dan wisuda.

4.2.2 Augmentasi Variasi Gaya Pertanyaan

Untuk meningkatkan kemampuan generalisasi model dalam memahami berbagai bentuk tanya dari pengguna, dilakukan proses augmentasi dengan cara membuat variasi gaya bertanya dari setiap entri original. Teknik ini bertujuan untuk melatih model mengenali maksud serupa meskipun disampaikan dalam formulasi kalimat yang berbeda.

Setiap entri original dikembangkan menjadi kurang lebih tiga variasi pertanyaan berbeda, dengan tetap mempertahankan respons dan referensi sumber yang sama. Hasil dari proses ini menghasilkan total 1.213 entri variasi, sehingga keseluruhan dataset menjadi 1.626 entri. Contoh hasil augmentasi ditampilkan pada Tabel 4.1.

Tabel 4.1 Contoh Augmentasi Variasi Pertanyaan

ID	Instruction	Respone	Sumber	ID_Original	Tipe_Variasi
204	Apa saja jenis pendidikan yang diselenggarakan oleh Unjaya?	Jenis pendidikan yang diselenggarakan oleh Unjaya adalah pendidikan akademik, vokasi, dan profesi.	Bab VI Jenis dan Program Pendidikan, Pasal 22 Jenis Pendidikan Ayat (1), Hal. 24 (Pedoman Akademik Unjaya 2024)	204	Original
993	Apa saja jenis pendidikan yang tersedia di Unjaya?	Jenis pendidikan yang diselenggarakan oleh Unjaya adalah pendidikan akademik, vokasi, dan profesi.	Bab VI Jenis dan Program Pendidikan, Pasal 22 Jenis Pendidikan Ayat (1), Hal. 24 (Pedoman Akademik Unjaya 2024)	204	gaya_pertanyaan
994	Jenis pendidikan apa yang ditawarkan oleh Unjaya?	Jenis pendidikan yang diselenggarakan oleh Unjaya adalah pendidikan akademik, vokasi, dan profesi.	Bab VI Jenis dan Program Pendidikan, Pasal 22 Jenis Pendidikan Ayat (1), Hal. 24 (Pedoman Akademik Unjaya 2024)	204	gaya_pertanyaan
995	Di Unjaya ada pendidikan apa saja?	Jenis pendidikan yang diselenggarakan oleh Unjaya adalah pendidikan akademik, vokasi, dan profesi.	Bab VI Jenis dan Program Pendidikan, Pasal 22 Jenis Pendidikan Ayat (1), Hal. 24 (Pedoman Akademik Unjaya 2024)	204	gaya_pertanyaan

4.2.3 Stratifikasi dan Pembagian Data

Dataset dibagi menjadi dua bagian ke dalam *train set* dan *evaluation set*. Pembagian ini dirancang secara adil dan representatif untuk menjaga kualitas evaluasi model, sekaligus menghindari kebocoran data. Dataset yang digunakan terdiri dari dua kelompok utama data original sebanyak 413 entri, yaitu data pertanyaan dan jawaban dan data variasi sebanyak 1.239 entri, yang merupakan hasil augmentasi dari data original dengan masing-masing 3 variasi gaya pertanyaan per entri.

Untuk menjaga cakupan domain secara utuh dalam proses pelatihan, seluruh data original dimasukkan ke dalam *train set.* Sementara itu, data variasi hasil augmentasi dibagi menjadi *train set* dan *evaluation set* menggunakan teknik *Stratified Shuffle Split* dari *library scikit-learn.*

Proses ini dilakukan dengan rasio pembagian 90:10, yaitu 90% data variasi digunakan untuk pelatihan dan 10% untuk evaluasi. Panjang karakter instruksi dijadikan sebagai basis stratifikasi, dengan tujuan agar distribusi kompleksitas pertanyaan tetap seimbang antara data *train* dan data *evaluation*. Strategi ini penting agar performa model yang dihasilkan mampu mencerminkan kemampuan generalisasi terhadap pertanyaan dengan berbagai tingkat kesulitan.

Kode program 4.1 berikut menunjukkan proses pembagian data variasi menggunakan *stratified split*:

Kode Program 4.1 Pembagian Data

```
# Pisahkan data original (ID 1–413) dan data variasi (ID 414 ke
atas)
df_original = df[df['ID'] <= 413].reset_index(drop=True)
df_variasi = df[df['ID'] > 413].reset_index(drop=True)

# Binning untuk stratifikasi hanya pada data variasi (ID 414 ke
atas)
df_variasi['length_bin'] = pd.qcut(
    df_variasi['Instruction'].str.len(),
    q=5,
    labels=False,
    duplicates='drop'
)

# Stratified split hanya untuk data variasi
splitter = StratifiedShuffleSplit(n_splits=1, test_size=0.1,
random_state=42)
```

```
for train_idx, eval_idx in splitter.split(df_variasi,
df_variasi['length_bin']):
    df_variasi_train =
df_variasi.iloc[train_idx].reset_index(drop=True)
    df_variasi_eval =
df_variasi.iloc[eval_idx].reset_index(drop=True)

# Gabungkan seluruh data original + df_variasi_train ke train
df_train = pd.concat([df_original, df_variasi_train],
ignore_index=True)
df_eval = df_variasi_eval.copy()
```

Ringkasan pembagian data ditampilkan pada Tabel 4.2.

Tabel 4.2 Ringkasan Pembagian Dataset

Jenis Data	Jumlah	Keterangan
Original (train)	413	Seluruh data original dimasukkan ke *train set*
Variasi (train)	1.091	Hasil stratifikasi dari data variasi
Variasi (eval)	122	*Stratified* berdasarkan panjang instruksi
Total Train	1.504	Data original + data variasi (*train*)
Total Eval	122	

4.2.4 Konversi ke Format ChatML

Agar kompatibel dengan arsitektur Mistral Instruct, dataset dikonversi ke format ChatML. Dalam format ini, setiap entri terdiri dari dua peran:

a. *role*: *user* untuk kolom *Instruction*

b. *role*: *assistant* untuk kolom *Response* dan Sumber

Potongan kode untuk proses konversi format ChatML pada Kode Program 4.2:

Kode Program 4.2 Konversi Format ChatML

```
def to_chatml_format(df, filename):
    chat_data = []

    for _, row in df.iterrows():
        messages = [
            {"role": "user", "content": row['Instruction']},
            {"role":           "assistant",           "content":
f"{row['Response']}\n\n(Sumber: {row['Sumber']})"}
        ]
        chat_data.append({"messages": messages})

    with open(filename, 'w', encoding='utf-8') as f:
        for item in chat_data:
            json.dump(item, f, ensure_ascii=False)
```

```
            f.write('\n')

    return chat_data

# Konversi dan simpan data
train_jsonl_path = '/kaggle/working/train_chatml.jsonl'
eval_jsonl_path = '/kaggle/working/eval_chatml.jsonl'

train_chatml = to_chatml_format(df_train, train_jsonl_path)
eval_chatml = to_chatml_format(df_eval, eval_jsonl_path)
```

Setelah konversi, data disimpan dalam *file train_chatml.jsonl* dan *eval_chatml.jsonl*, lalu dimuat menggunakan Hugging Face *load_dataset*. Contoh entri format ChatML dapat dilihat pada Kode Program 4.3:

Kode Program 4.3 Contoh Format ChatML

```
{
  "messages": [
    {
      "role": "user",
      "content": "Apa itu pedoman akademik di Unjaya?"
    },
    {
      "role": "assistant",
      "content": "Pedoman akademik adalah jabaran dari kebijakan
akademik Universitas Jenderal Achmad Yani Yogyakarta yang menjadi
pedoman penyelenggaraan program akademik.\n\n(Sumber: Kata
Pengantar, Hal. 3 (Pedoman Akademik Unjaya 2024))"
    }
  ]
}
```


4.3 PROSES FINE-TUNING

Subjudul ini menjelaskan proses implementasi *fine-tuning* terhadap model Mistral-7B-Instruct menggunakan pendekatan. Fokus diarahkan pada efisiensi penggunaan sumber daya GPU serta fleksibilitas dalam pelatihan model berskala besar pada perangkat terbatas.

4.3.1 Teknik QLoRA

QLoRA memungkinkan pelatihan model LLM yang besar dengan sumber daya terbatas melalui kuantisasi 4-bit. Dengan teknik ini, kebutuhan memori dapat dikurangi secara signifikan dibandingkan pelatihan dengan presisi penuh (tanpa kuantisasi). Salah satu format yang digunakan dalam kuantisasi ini adalah *NormalFloat 4-bit (nf4)*, yang dirancang untuk menangani data dengan distribusi

normal secara lebih presisi dan efisien dibandingkan format 4-bit lainnya. Implementasi dilakukan dengan konfigurasi pada Kode Program 4.4:

Kode Program 4.4 Konfigurasi Kuantisasi 4-bit

```
bnb_config = BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_quant_type="nf4",
    bnb_4bit_compute_dtype=torch.bfloat16,
    bnb_4bit_use_double_quant=False
)
```

Model dasar yang digunakan adalah *mistralai/Mistral-7B-Instruct-v0.3*, dimuat dengan konfigurasi kuantisasi menggunakan *AutoModelForCausalLM*. *Tokenizer* juga dimuat dari model dasar dan disesuaikan agar *padding* dilakukan di sisi kanan seperti pada Kode Program 4.5.

Kode Program 4.5 *Base Model* dan *Tokenizer*

```
# Load model dengan quantization
model = AutoModelForCausalLM.from_pretrained(
    base_model,
    quantization_config=bnb_config,
    torch_dtype=torch.bfloat16,
    device_map="auto",
    trust_remote_code=True)
# Load tokenizer
tokenizer      =      AutoTokenizer.from_pretrained(base_model,
trust_remote_code=True)
# Setup tokenizer untuk chat format
if tokenizer.pad_token is None:
    tokenizer.pad_token = tokenizer.eos_token
tokenizer.padding_side = "right"
```

Agar hanya sebagian kecil parameter yang dilatih, digunakan teknik PEFT dengan konfigurasi LoRA pada Kode Program 4.6:

Kode Program 4.6 Konfigurasi LoRA

```
peft_config = LoraConfig(
    lora_alpha=16,
    lora_dropout=0.1,
    r=64,
    bias="none",
    task_type="CAUSAL_LM",
    target_modules=["q_proj",   "k_proj",   "v_proj",   "o_proj",
"gate_proj"])
```

Visualisasi jumlah parameter yang akan dilatih setelah penerapan konfigurasi LoRA ditunjukkan pada Gambar 4.1.

```
trainable params: 92,274,688 || all params: 7,340,298,240 || trainable%: 1.2571
```

Gambar 4.1 Jumlah parameter terlatih setelah LoRA

Dengan konfigurasi ini, hanya sebagian kecil dari total parameter yang benar-benar dilatih, yakni sebanyak 92 juta dari total 7,34 miliar parameter. Jumlah ini berasal dari jumlah parameter adapter LoRA yang ditambahkan pada modul *q_proj, k_proj, v_proj, o_proj, dan gate_proj*.

4.3.2 Konfigurasi Pelatihan

Pelatihan model dilakukan menggunakan *SFTTrainer* dari *library trl*, yang dirancang khusus untuk *fine-tuning* LLM dengan pendekatan supervisi instruksi. Berikut adalah konfigurasi utama yang digunakan dalam pelatihan:

a. Jumlah epoch (*num_train_epochs=1*)
   Model dilatih selama 1 epoch mengingat jumlah data pelatihan relatif kecil. Tujuannya untuk menghindari overfitting dan menjaga efisiensi waktu komputasi.
b. Ukuran batch (*per_device_train_batch_size=2*)
   Ukuran batch per perangkat adalah 2, namun karena *gradient_accumulation_steps=16*, ukuran batch efektif menjadi 32 (2x16).
c. Ukuran *batch* evaluasi (*per_device_eval_batch_size=2*)
   Untuk evaluasi, *batch* per perangkat disamakan agar konsisten dan tidak membebani memori.
d. *Learning rate* (*learning_rate=2e-4*)
   Digunakan nilai *learning rate* sebesar 0.0002, nilai umum untuk *fine-tuning* LLM secara stabil.
e. *Weight decay* (*weight_decay=0.001*)
   Diterapkan regularisasi *weight decay* untuk mencegah *overfitting*.

f. Langkah akumulasi (*gradient_accumulation_steps=16*)

Akumulasi gradien digunakan untuk menyimulasikan *batch size* yang lebih besar tanpa melebihi batas memori GPU.

g. Frekuensi evaluasi (*eval_steps=10*)

Evaluasi dilakukan setiap 10 langkah pelatihan untuk memantau performa model secara berkala.

h. Frekuensi *checkpoint* (*save_steps=10*)

Model disimpan setiap 10 langkah, sehingga hasil pelatihan dapat dipulihkan jika terjadi interupsi.

i. Jumlah maksimum *checkpoint* (*save_total_limit=1*)

Hanya 1 *checkpoint* terbaru yang disimpan untuk menghemat ruang penyimpanan.

j. *Logging* (*logging_steps=10*)

Informasi log dicatat setiap 10 langkah, mendukung pelacakan melalui *Weights & Biases (wandb).*

k. Strategi evaluasi (*eval_strategy="steps"*)

Evaluasi ditentukan berdasarkan jumlah langkah (bukan setiap epoch), agar lebih sering memantau hasil selama proses pelatihan pendek.

l. Seed (seed=42)

Penetapan *seed* untuk memastikan hasil yang *reproducible* dan konsisten antar percobaan.

m. Precision (fp16=True, bf16=False)

Precision ditetapkan ke *FP16* untuk mengurangi konsumsi memori dan mempercepat proses pelatihan. GPU T4 tidak mendukung *BF16*, sehingga opsi ini dimatikan.

Konfigurasi lengkap pelatihan menggunakan *SFTTrainer* ditampilkan pada Kode Program 4.7.

Kode Program 4.7 Konfigurasi Pelatihan

```
sft_config = SFTConfig(
    output_dir = "/kaggle/working/results-pedoman-akademik",
    num_train_epochs=1,
    per_device_train_batch_size=2,
```

```
    per_device_eval_batch_size=2,
    gradient_accumulation_steps=16,
    eval_accumulation_steps=16,
    learning_rate=2e-4,
    weight_decay=0.001,

    # Logging dan checkpointing
    eval_steps=10,
    save_steps=10,
    save_total_limit=1,
    logging_steps=10,
    disable_tqdm=False,

    # Evaluasi dan reproducibility
    eval_strategy="steps",
    seed=42,

    # Precision
    bf16=False,
    fp16=True,

    # Hugging Face Hub
    push_to_hub=True,
    hub_model_id=new_model_name,

    # Wandb monitoring
    report_to="wandb" if wandb.run else None,
    run_name=f"mistral-pedoman-{wandb.run.id}" if wandb.run else
None,

    # SFT-specific params
    max_seq_length=512,
    packing=False,
    neftune_noise_alpha=5)
```


Beberapa pengaturan tambahan seperti *push_to_hub* dan pelaporan ke *Weights & Biases* (*report_to="wandb")* juga disertakan untuk keperluan kolaborasi dan monitoring, namun tidak mempengaruhi proses pelatihan inti.

Setelah konfigurasi ditentukan, proses pelatihan dimulai menggunakan *SFTTrainer* seperti pda Kode Program 4.8:

Kode Program 4.8 Inisialisasi dan Pelatihan SFTTrainer

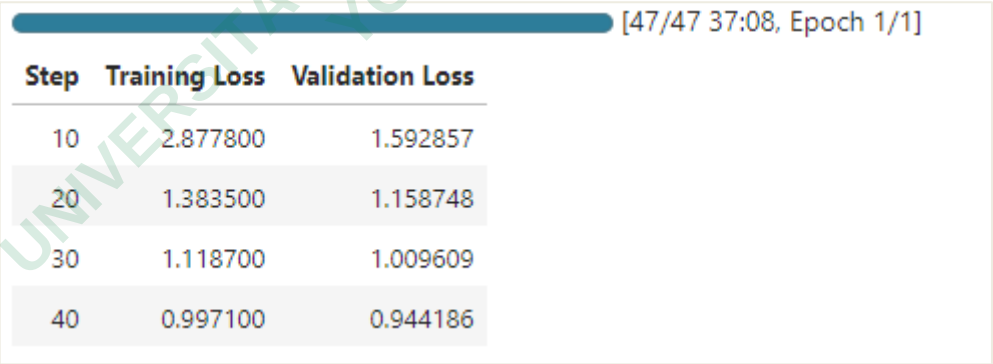
```
trainer = SFTTrainer(
    model=model,
    args=sft_config,
    train_dataset=train_dataset,
    eval_dataset=eval_dataset,
    peft_config=peft_config,
)

trainer.tokenizer = tokenizer
```

```
# Training
trainer.train()
```

4.3.3 Hasil Pelatihan

Pelatihan model dilakukan selama 1 *epoch* dengan total 47 langkah optimisasi (*training steps*). Jumlah langkah ini dihitung berdasarkan jumlah total data latih yang tersedia, yaitu sebanyak 1.504 sampel. Dengan konfigurasi *per_device_train_batch_size*=2 dan *gradient_accumulation_steps*=16, maka *batch size* efektif yang digunakan adalah 32. Dengan demikian, jumlah langkah pelatihan dihitung dengan membagi total sampel pelatihan dengan *batch size* efektif, yaitu 1.504 dibagi 32, sehingga menghasilkan 47 langkah.

Proses pelatihan memakan waktu sekitar 37 menit, dan model disimpan secara berkala dalam beberapa *checkpoint* sesuai dengan konfigurasi *save_steps*=10. Selama pelatihan, nilai *training loss* menunjukkan penurunan yang konsisten, dari 2,87 pada awal pelatihan menjadi 0,99 pada langkah ke-40. Sementara itu, nilai *validation loss* terakhir tercatat sebesar 0,94, yang menunjukkan bahwa model mampu belajar dari data tanpa mengalami *overfitting* secara signifikan. Rangkuman hasil pelatihan pada beberapa langkah evaluasi dapat dilihat pada Gambar 4.2:



Gambar 4.2 Ringkasan *Training* dan *Validation Loss* selama Pelatihan

Seluruh proses pelatihan dapat diakses melalui notebook publik yang tersedia di Kaggle: kaggle.com/code/riakristi/mistral-7b-instruct-fine-tuning-akademik-unjaya

4.4 EVALUASI

Evaluasi dilakukan untuk menilai sejauh mana model yang telah dilatih mampu menghasilkan respons yang relevan dan sesuai secara semantik terhadap referensi dalam domain pedoman akademik. Metrik yang digunakan adalah BERTScore, sebagaimana telah dijelaskan pada Bab II. Dengan data uji yang representatif, evaluasi ini memberikan gambaran kuantitatif terhadap kualitas keluaran model, baik secara keseluruhan maupun per sampel.

BERTScore menghitung kesamaan semantik antar token dengan membandingkan representasi vektor dari teks prediksi dan referensi. Dengan pendekatan ini, BERTScore menilai kualitas jawaban berdasarkan makna, bukan sekadar kesamaan kata. Tiga metrik utama yang digunakan adalah:

a. *Precision*: Mengukur seberapa banyak token prediksi yang relevan dengan referensi.
b. *Recall*: Mengukur seberapa banyak token referensi yang berhasil ditangkap oleh prediksi.
c. F1 *Score*: Rata-rata harmonis dari *precision* dan *recall*, memberikan gambaran umum terhadap performa model.

Tahapan evaluasi diawali dengan memuat model hasil *fine-tuning* dan mengatur mode evaluasi menggunakan *model.eval()* untuk memastikan bahwa tidak terjadi pembaruan parameter selama proses inferensi. Selanjutnya, *pipeline* untuk *text generation* disiapkan menggunakan *transformers.pipeline,* dengan penyesuaian tipe data (*torch_dtype*) berdasarkan ketersediaan GPU. *Pipeline* ini memungkinkan inferensi dilakukan secara efisien dan fleksibel terhadap input teks , sebagaimana ditunjukkan pada Kode Program 4.9.

Kode Program 4.9 Pipeline Text Generation

```
# Setup pipeline
pipe = pipeline(
    task="text-generation",
    model=model,
    tokenizer=tokenizer,
    torch_dtype=torch.bfloat16 if torch.cuda.is_available() else
torch.float32,
    return_full_text=False)
```

Kemudian sampel data uji sebanyak 15 entri diambil secara acak dari *eval_dataset* menggunakan *seed* tetap (*seed=42*) untuk menjaga reprodusibilitas. Setiap *prompt* diformat ke dalam *template ChatML* sebelum digunakan sebagai *input* model. Setelah diformat, prompt digunakan dalam pipeline untuk menghasilkan prediksi *respone*. Pengaturan inferensi meliputi jumlah maksimum token baru (*max_new_tokens=256*), sampling aktif (*do_sample=True*), suhu sampling (*temperature=0.7*), dan nilai *top_p=0.9*, sebagaimana ditunjukkan pada Kode Program 4.10.

Kode Program 4.10 Inferensi Template ChatML

```
chat = [{"role": "user", "content": prompt}]
input_text = tokenizer.apply_chat_template(chat, tokenize=False,
add_generation_prompt=True)

output = pipe(
    input_text,
    max_new_tokens=256,
    do_sample=True,
    temperature=0.7,
    top_p=0.9,
    pad_token_id=tokenizer.eos_token_id)
```

Setelah semua prediksi dihasilkan, pada Kode Program 4.11 perhitungan BERTScore dilakukan menggunakan *library evaluate*, dengan pengaturan bahasa Indonesia (*lang="id"*).

Kode Program 4.11 Perhitungan BERTScore

```
bertscore = evaluate.load("bertscore")

results = bertscore.compute(
    predictions=generated,
    references=references,
    lang="id"
)
```

4.4.1 Hasil Evaluasi

Rata-rata hasil evaluasi dari 15 sampel adalah sebagai berikut:

a. F1 Score : 0.6421
b. *Precision* : 0.6323
c. *Recall* : 0.6525

Hasil ini menunjukkan bahwa meskipun respons model belum sepenuhnya optimal, model mampu menghasilkan jawaban yang secara semantik cukup mendekati referensi dalam mayoritas kasus.

Contoh hasil evaluasi ditampilkan pada Tabel 4.3:

Tabel 4.3 Contoh Hasil Evaluasi

Komponen	**Teks**
Prompt	Apa definisi pedoman akademik menurut Unjaya?
Prediksi Model	Pedoman akademik merupakan suatu petunjuk umum yang ditetapkan oleh lembaga pendidikan tinggi untuk mengatur, mengatur, dan mengatur proses pembelajaran dan penilaian di institusi tersebut. Pedoman akademik ini biasanya dikembangkan dan diatur oleh komite akademik atau badan pengelola akademik di institusi tersebut. Sesuai dengan Unjaya, pedoman akademik merupakan pedoman yang berfungsi untuk mengatur, mengatur, dan mengatur proses pembelajaran, penilaian, dan pengelolaan kualitas akademik di universitas tersebut. Pedoman akademik ini dikembangkan dan diatur oleh Komite Pedoman Akademik (KPA) Unjaya, yang terdiri dari para pakar akademik yang berwisuda di institusi tersebut.
Referensi	Pedoman akademik adalah jabaran dari kebijakan akademik Universitas Jenderal Achmad Yani Yogyakarta yang menjadi pedoman penyelenggaraan program akademik. (Sumber: Kata Pengantar, Hal. 3 (Pedoman Akademik Unjaya 2024))
Precision	0.655
Recall	0.693
F1 Score	0.674

Berdasarkan hasil evaluasi menggunakan BERTScore, model yang telah dilatih mampu menghasilkan jawaban yang cukup relevan secara semantik terhadap referensi. Namun, skor F1 yang belum mencapai angka tinggi (di bawah 0,70) serta kecenderungan model menghasilkan respons yang bersifat umum mengindikasikan adanya potensi *hallucination*. Rata-rata skor sebesar 0,64 menunjukkan bahwa performa model masih dapat ditingkatkan, misalnya melalui penambahan jumlah data pelatihan, perpanjangan durasi *fine-tuning*, atau penerapan teknik augmentasi data.

4.5 INFERENCE MODEL

Melakukan inference atau pengujian terhadap model untuk memastikan bahwa model mampu memberikan respons sesuai konteks data pelatihan, yakni Pedoman Akademik Unjaya. Model hasil *fine-tuning* yang telah digabung (*merged*) dengan *adapter* dan *base model* dipublikasikan melalui *platform* Hugging Face Hub dengan nama repositori: huggingface.co/riakrst/mistral-7b-pedoman-akademik-unjaya-merged.

Untuk membuktikan bahwa model dapat digunakan untuk menjawab pertanyaan, dilakukan proses *inference* lokal menggunakan pendekatan *pipeline*() dari pustaka *transformers* seperti pada Kode Program 4.12.

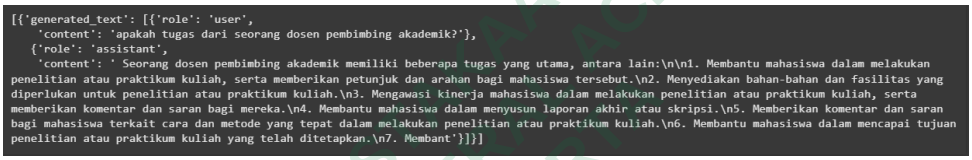
Kode Program 4.13 *Pipeline Inference Local*

```
from transformers import pipeline

# Memuat pipeline text-generation dengan model hasil fine-tuning
pipe = pipeline("text-generation", model="riakrst/mistral-7b-pedoman-akademik-unjaya-merged", device=0)

# Menyusun pesan dari pengguna
messages = [
    {"role": "user", "content": "apakah tugas dari seorang dosen pembimbing akademik?"},
]

# print inference
pipe(
    messages,
    max_new_tokens=300,
    temperature=0.7,
    top_p=0.9,
    repetition_penalty=1.1,
```

```
    do_sample=True
)
```

Pengaturan *parameter* seperti *temperature=0.7* dan *top_p=0.9* digunakan untuk mengontrol keluaran model agar tetap bervariasi namun tetap relevan secara semantik. Nilai *temperature* yang lebih rendah membantu menjaga koherensi dan fokus jawaban, sementara top-p *sampling* memungkinkan model mengeksplorasi beberapa kemungkinan respons yang memiliki probabilitas tinggi secara kumulatif. Pengaturan ini terbukti mampu menghasilkan respons yang informatif dan tidak terlalu acak, sebagaimana terlihat pada jawaban model terhadap pertanyaan seperti tugas pembimbing akademik.

Hasil output dari proses *inference* ditampilkan dalam format percakapan (ChatML) pada Gambar 4.4.



Gambar 4.3 Contoh Hasil *Inference*

Notebook inference secara lengkap dapat diakses melalui Kaggle: kaggle.com/code/riakristi/local-inference-mistral-pedoman-akademik

4.5.1 Implementasi Prototipe Chatbot

Pengembangan antarmuka prototipe chatbot berbasis model hasil fine-tuning telah direncanakan dalam bentuk aplikasi web sederhana menggunakan Streamlit. Namun, pada tahap pelaksanaan, implementasi prototipe belum berhasil dibangun secara utuh karena beberapa kendala teknis dan non-teknis.

Model hasil *fine-tuning* telah berhasil dipublikasikan ke Hugging Face Hub, tetapi layanan *inference endpoint* yang tersedia pada platform tersebut bersifat berbayar. Oleh karena itu, penulis belum dapat menggunakan layanan tersebut untuk integrasi *chatbot* secara penuh. Alternatif yang berhasil diterapkan adalah *deploy* prototipe ke Hugging Face Spaces, namun hanya menggunakan CPU gratis, sehingga performa *inference* model masih terbatas dan belum optimal untuk

interaksi *real-time*. Tautan prototipe chatbot yang telah disiapkan: https://huggingface.co/spaces/riakrst/chatbot-akademik-unjaya.

Prototipe *chatbot* belum berhasil dibangun secara menyeluruh, tetapi model yang tersedia saat ini telah siap untuk diintegrasikan lebih lanjut dalam proses pengembangan sistem *chatbot* akademik yang fungsional.

4.6 PEMBAHASAN

Penelitian ini bertujuan untuk mengadaptasi model LLM ke dalam domain spesifik pedoman akademik Unjaya melalui proses *fine-tuning* yang efisien. Berdasarkan hasil eksperimen yang diperoleh, terdapat sejumlah poin penting yang dapat dianalisis untuk menjawab pertanyaan penelitian mengenai proses *fine-tuning* serta performa model dalam merespons pertanyaan seputar pedoman akademik secara akurat dan kontekstual.

4.6.1 Interpretasi Hasil *Fine-Tuning* dan Evaluasi Model

Proses *fine-tuning* menggunakan model dasar *Mistral-7B-Instruct* dan metode QLoRA menunjukkan efisiensi yang tinggi dalam melatih model LLM dengan keterbatasan sumber daya komputasi. Selama satu *epoch* pelatihan, *training loss* secara konsisten menurun dari 2.87 menjadi 0.99, sedangkan *validation loss* akhir tercatat sebesar 0.94. Penurunan ini menunjukkan bahwa model berhasil mempelajari pola dari data tanpa menunjukkan indikasi *overfitting* yang berarti.

Dari sisi evaluasi kinerja, penggunaan metrik BERTScore menghasilkan rata-rata nilai F1 sebesar 0.6421, dengan nilai *Precision* dan *Recall* masing-masing sebesar 0.6323 dan 0.6525. Meskipun nilai ini belum mencapai ambang batas ideal untuk sistem produksi (misalnya F1 > 0.70), performa tersebut menunjukkan bahwa model telah mampu memahami konteks dan memberikan respons yang secara semantik mendekati referensi. Namun, masih terdapat indikasi *hallucination*, terutama pada respons yang cenderung bersifat umum dan tidak merujuk secara spesifik pada isi pedoman akademik. Sebagai contoh, pada pertanyaan mengenai tugas pembimbing akademik, model mampu menghasilkan jawaban yang mengandung makna esensial meskipun tidak identik secara leksikal dengan

referensi. Hal ini menandakan bahwa model memiliki kemampuan generalisasi semantik, namun akurasinya masih dapat ditingkatkan.

4.6.2 Analisis Faktor yang Mempengaruhi Performa Model

Terdapat beberapa faktor utama yang memengaruhi performa model hasil *fine-tuning* dalam penelitian ini:

1. Kuantitas dan Kualitas Dataset

   Dataset berjumlah 1.626 entri, meskipun telah diperluas melalui augmentasi, masih relatif kecil untuk ukuran model Mistral-7B. Keterbatasan dalam variasi gaya pertanyaan juga dapat membatasi kemampuan model dalam melakukan generalisasi terhadap pertanyaan baru yang belum pernah dilihat sebelumnya.

2. Durasi Pelatihan (*Epoch*):

   Pelatihan dilakukan hanya selama satu *epoch* sebagai bentuk mitigasi terhadap risiko *overfitting*. Walaupun strategi ini konservatif dan aman, potensi peningkatan performa model dapat dieksplorasi lebih lanjut dengan menambahkan jumlah *epoch* disertai penggunaan *early stopping* atau teknik *regularisasi* tambahan.

3. Ukuran dan Karakteristik *Base* Model

   Mistral-7B-Instruct adalah LLM yang dirancang untuk penggunaan umum (*general-purpose*). Untuk mencapai performa optimal di domain pedoman akademik yang bersifat formal dan spesifik, dibutuhkan domain *adaptation* yang lebih intensif. Karakteristik bahasa pada dokumen pedoman akademik yang cenderung formal dan repetitif juga menjadi tantangan tersendiri bagi proses pembelajaran model.

4. Pengaturan *Hyperparameter*

   Penggunaan *learning rate*, *batch size*, dan konfigurasi inferensi seperti *max_new_tokens* dan *temperature* telah disesuaikan dengan praktik terbaik. Namun, eksplorasi lebih lanjut terhadap kombinasi parameter lain seperti *lora_r, lora_alpha, gradient_accumulation_steps* berpotensi menghasilkan performa yang lebih baik.

4.6.3 Keterbatasan dan Potensi Pengembangan

Meskipun model menunjukkan performa yang menjanjikan, terdapat beberapa keterbatasan yang perlu dicatat:

1. Cakupan dan Variasi Dataset

   Dataset yang digunakan dalam proses fine-tuning masih terbatas dari segi variasi gaya pertanyaan dan format respons. Untuk meningkatkan generalisasi model, diperlukan ekspansi dataset yang mencakup:

   a. Variasi kompleksitas jawaban. Penambahan data dengan jenis pertanyaan yang menuntut jawaban lebih analitis atau inferensial, bukan sekadar kutipan eksplisit dari dokumen pedoman.
   b. Beragam format respons. Pelatihan model agar mampu menghasilkan jawaban dalam format berbeda, seperti jawaban singkat, poin-poin, atau paragraf naratif, sesuai konteks pertanyaan.

2. Optimasi *Hyperparameter* Komprehensif

   Melakukan eksperimen sistematis dengan berbagai kombinasi hyperparameter dan strategi *learning rate scheduler* untuk mengidentifikasi konfigurasi yang paling optimal. Penelitian lanjutan dapat melakukan eksperimen sistematis terhadap berbagai kombinasi *learning rate*, *batch size*, dan strategi *scheduler* untuk memperoleh konfigurasi yang optimal terhadap dataset spesifik ini.

3. Implementasi Aplikasi Fungsional

   Model yang dihasilkan saat ini masih berada dalam bentuk file model (checkpoint) yang belum sepenuhnya diintegrasikan ke dalam aplikasi chatbot siap pakai. Pengembangan lebih lanjut dapat difokuskan pada:

   a. Pembangunan antarmuka pengguna (*frontend*) untuk penggunaan interaktif.
   b. *Deploy inference endpoint* agar model dapat diakses melalui API.
   c. Integrasi ke sistem informasi akademik Unjaya.

Dengan mempertimbangkan aspek-aspek tersebut, hasil penelitian ini dapat dikembangkan lebih lanjut untuk menghasilkan sistem chatbot akademik yang lebih tangguh, adaptif, dan siap digunakan di lingkungan pendidikan tinggi.