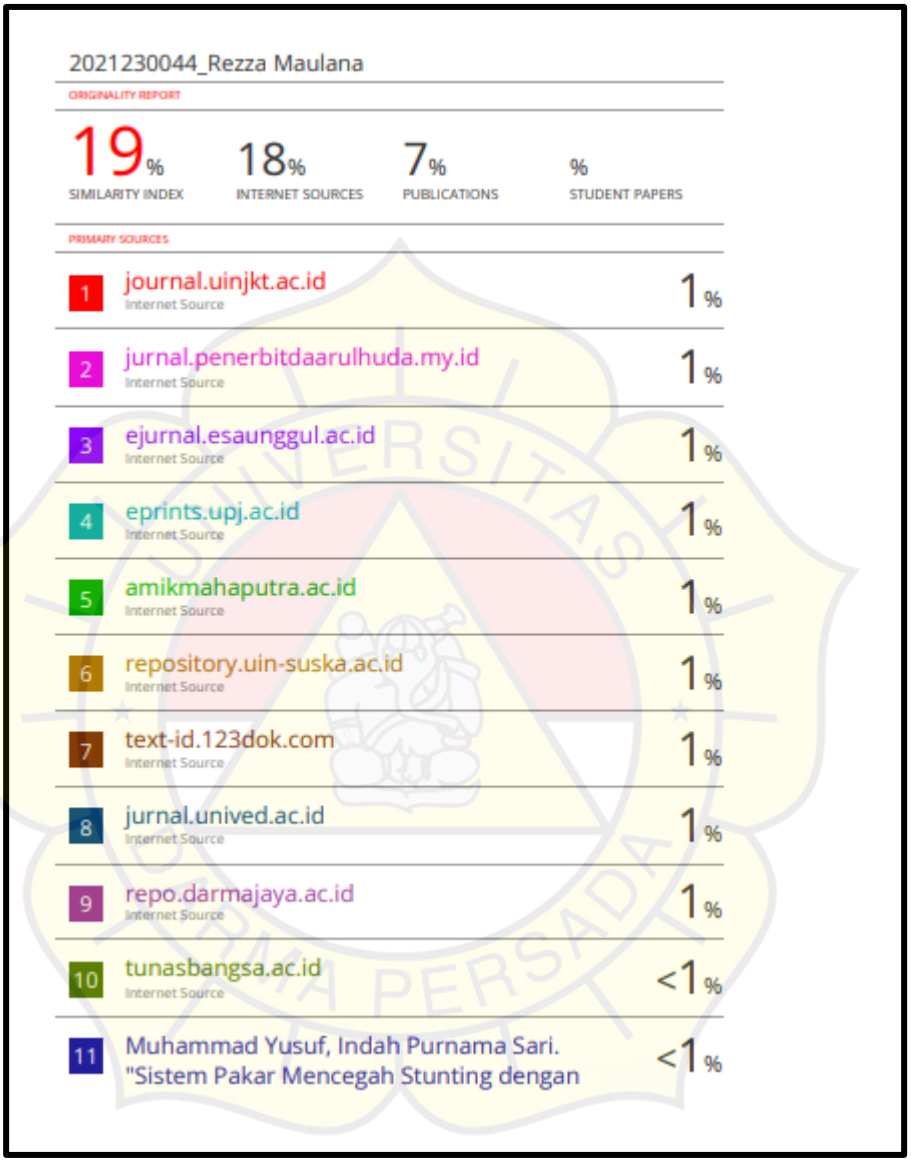
DAFTAR LAMPIRAN



2021230044_Rezza Maulana

ORIGINALITY REPORT

19%	18%	7%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	journal.uinjkt.ac.id Internet Source	1%
2	jurnal.penerbitdaarulhuda.my.id Internet Source	1%
3	ejurnal.esaunggul.ac.id Internet Source	1%
4	eprints.upj.ac.id Internet Source	1%
5	amikmahaputra.ac.id Internet Source	1%
6	repository.uin-suska.ac.id Internet Source	1%
7	text-id.123dok.com Internet Source	1%
8	jurnal.unived.ac.id Internet Source	1%
9	repo.darmajaya.ac.id Internet Source	1%
10	tunasbangsa.ac.id Internet Source	<1%
11	Muhammad Yusuf, Indah Purnama Sari. "Sistem Pakar Mencegah Stunting dengan	<1%

UNIVERSITAS DARMA PERSADA
UPT PERPUSTAKAAN
Gedung Rektorat Lantai 3,
Jl.Taman Malaka Selatan, Pondok Kelapa – Jakarta Timur 13450

SURAT KETERANGAN
HASIL PENGECEKAN TURNITIN

UPT Perpustakaan Universitas Darma Persada menerangkan telah selesai melakukan pemeriksaan duplikasi/*similarity* menggunakan perangkat lunak Turnitin terhadap hasil karya sebagai berikut:

Judul	: Penerapan Natural Language Processing pada Pengelolaan Berkas Digital menggunakan Chatbot di Pusdatin Dinas PPKUKM
Penulis	: Rezza Maulana
NIM	: **2021230044**
Tgl pemeriksaan	: 25 Juli 2025

Dengan hasil Tingkat Kesamaan (*similarity index*) **19%**

Demikian Surat Keterangan kami buat, untuk dipergunakan sebagaimana mestinya.

Jakarta, 25 Juli 2025

Ka.UPT Perpustakaan Unsada

Yus Rusmiyati, SS., MM

Batas maksimal similarity 30% untuk Fakultas Sastra dan Ekonomi

Batas maksimal similarity 25% untuk Fakultas Teknik, Kelautan dan Pasca Sarjana

Source code

Nlp_api.py

```
from fastapi import FastAPI, Query
from pydantic import BaseModel
import re
import os
import fitz  # PyMuPDF
import torch
from transformers import AutoTokenizer, AutoModelForSequenceClassification, pipeline

app = FastAPI()

    # Load intent classification model (BERT)
    intent_model_path = "bert-intent-model"  # Folder hasil training
    intent_model = AutoModelForSequenceClassification.from_pretrained(
        intent_model_path,
        use_safetensors=True
    )
    intent_tokenizer = AutoTokenizer.from_pretrained(intent_model_path)
    label_map = intent_model.config.id2label

    # Load summarization pipeline
    summarizer = pipeline("summarization", model="facebook/bart-large-cnn")

    # Model input
    class QueryModel(BaseModel):
        question: str

    # Fungsi klasifikasi intent
    def classify_intent(text: str) -> str:
        try:
            inputs = intent_tokenizer(text, return_tensors="pt", truncation=True, padding=True)
            with torch.no_grad():
                outputs = intent_model(**inputs)
            predicted_class_id = torch.argmax(outputs.logits, dim=1).item()
            return label_map[predicted_class_id]
        except Exception as e:
```

```
            print(f"❌ classify_intent error: {e}")
            return "unknown"

    # Daftar sinonim topik
    TOPIC_SYNONYMS = {
        "prestasi": ["prestasi", "promosi", "penghargaan", "kenaikan jabatan"],
        "cuti": ["cuti", "izin tidak masuk", "izin kerja", "libur"],
        "mutasi": ["mutasi", "perpindahan", "rotasi"],
        "absensi": ["absensi", "kehadiran", "presensi"],
        "Permohonan": ["zoom meeting", "jaringan", "desain grafis"],
        # Tambahkan sesuai kebutuhan
    }

    # Endpoint NLP utama
    @app.post("/nlp")
    def get_entities(query: QueryModel):
        text = query.question
        print(f"📥 Pertanyaan user: {text}")
        intent = classify_intent(text)
        print(f"🎯 Intent dikenali: {intent}")

        nama = bulan = tahun = file_name = topic = None
        matching_files = []

        # Deteksi nama file PDF
        if intent in ["read_pdf_content", "summarize_pdf"]:
            match_file = re.search(r"(?:file|berkas|dokumen)(?:bernama|)?\s+(.*?\.pdf)", text.lower())
            if match_file:
                file_name = match_file.group(1).strip().replace(" ", "%20")

        # Deteksi berdasarkan kriteria
        if intent == "search_by_criteria":
            match_nama = re.search(r"(?:atas nama|milik|punya|untuk|dari|berkas(?:dari)?)(?:\s+(?:pak|ibu))?\s+([a-zA-Z]+(?:\s[a-zA-Z]+)*)", text.lower())
            if match_nama:
                nama = match_nama.group(1).strip().lower()

            month_map = {
```

```
                "januari": "01", "februari": "02", "maret": "03", "april": "04",
                "mei": "05", "juni": "06", "juli": "07", "agustus": "08",
                "september": "09", "oktober": "10", "november": "11", "desember": "12"
            }
            for month_name, month_code in month_map.items():
                if month_name in text.lower():
                    bulan = month_code
                    break

            for y in range(2020, 2031):
                if str(y) in text:
                    tahun = str(y)
                    break

        # Deteksi berdasarkan topik
        if intent == "search_by_topic":
            topic = None
            if isinstance(text, str):
                text = text.lower()

            match_topic = re.search(
                r"(?:tentang|topik|dengan topik|berkas(?:\s+terkait)?|dokumen|file|mengenai|seputar)(?:\s+yang)?(?:\s+membahas)?\s+([a-zA-Z\s]+)",
                text
            )

            if match_topic:
                topic = match_topic.group(1).strip().lower()
            else:
                match_simple = re.search(r"tentang\s+([a-zA-Z\s]+)", text)
                if match_simple:
                    topic = match_simple.group(1).strip().lower()

            if topic:
                # Ekspansi keyword berdasarkan sinonim
                expanded_keywords = []
                for key, synonyms in TOPIC_SYNONYMS.items():
                    if topic in synonyms:
                        expanded_keywords = synonyms
                        break
```

```
                if not expanded_keywords:
                    expanded_keywords = [topic]

                folder_path = os.path.join("storage", "app", "public")
                for file in os.listdir(folder_path):
                    if file.endswith(".pdf"):
                        normalized_filename = file.lower().replace("_", " ").replace("-", " ")
                        for keyword in expanded_keywords:
                            if keyword in normalized_filename:
                                matching_files.append(file)
                                break

        return {
            "intent": intent,
            "nama": nama,
            "bulan": bulan,
            "tahun": tahun,
            "file_name": file_name,
            "topic": topic,
            "matching_files": matching_files
        }

    # Endpoint baca isi PDF
    @app.get("/read-pdf")
    def read_pdf(file_name: str = Query(..., alias="file_name")):
        try:
            decoded_name = file_name.replace('%20', ' ')
            file_path = os.path.join("storage", "app", "public", decoded_name)

            if not os.path.exists(file_path):
                return {"error": "File tidak ditemukan."}

            with fitz.open(file_path) as doc:
                text = "\n".join([page.get_text() for page in doc])
                return {"content": text if text.strip() else "Isi dokumen kosong atau tidak dapat dibaca."}
        except Exception as e:
            return {"error": f"Gagal membaca isi dokumen: {str(e)}"}

```

```
150.    # Endpoint ringkasan PDF
151.    @app.get("/summarize-pdf")
152.    def    summarize_pdf(file_name:    str    =    Query(...,
  alias="file_name")):
153.        try:
154.            decoded_name = file_name.replace('%20', ' ')
155.            file_path   =   os.path.join("storage",   "app",
  "public", decoded_name)
156.
157.            if not os.path.exists(file_path):
158.                return {"error": "File tidak ditemukan."}
159.
160.            with fitz.open(file_path) as doc:
161.                text = "\n".join([page.get_text() for page in
  doc])
162.
163.            if not text.strip():
164.                return {"summary": "Isi dokumen kosong atau
  tidak dapat dibaca."}
165.
166.            max_len = 1024
167.            text = text.replace("\n", " ")
168.            chunks = [text[i:i + max_len] for i in range(0,
  len(text), max_len)][:3]
169.
170.            summaries = []
171.            for chunk in chunks:
172.                try:
173.                    result        =        summarizer(chunk,
  max_length=130, min_length=30, do_sample=False)
174.
  summaries.append(result[0]['summary_text'])
175.                except Exception as e:
176.                    summaries.append("[Gagal        meringkas
  sebagian isi dokumen]")
177.
178.            return {"summary": " ".join(summaries)}
179.
180.        except Exception as e:
181.            return {"error": f"Gagal meringkas isi dokumen:
  {str(e)}"}
182.
183.    # Endpoint untuk pencarian berdasarkan topik langsung
184.    @app.get("/search-by-topic")
185.    def    search_by_topic(topic:    str    =    Query(...,
  alias="topic")):
```

```
        try:
            topic = topic.lower()
            matching_files = []

            # Ekspansi keyword berdasar sinonim
            expanded_keywords = []
            for key, synonyms in TOPIC_SYNONYMS.items():
                if topic in synonyms:
                    expanded_keywords = synonyms
                    break
            if not expanded_keywords:
                expanded_keywords = [topic]

            folder_path = os.path.join("storage", "app", "public")
            for file in os.listdir(folder_path):
                if file.endswith(".pdf"):
                    normalized_filename = file.lower().replace("_", " ").replace("-", " ")
                    for keyword in expanded_keywords:
                        if keyword in normalized_filename:
                            matching_files.append(file)
                            break

            if not matching_files:
                return {"message": f"Tidak ditemukan berkas terkait topik '{topic}'."}

            return {
                "topic": topic,
                "matching_files": matching_files
            }

        except Exception as e:
            return {"error": f"Gagal mencari berkas berdasarkan topik: {str(e)}"}
```

Source code
Train_intent_classifier.py

```
from datasets import load_dataset
from transformers import AutoTokenizer, AutoModelForSequenceClassification, Trainer, TrainingArguments
import torch

# Model IndoBERT
model_name = "indobenchmark/indobert-base-p1"

# Load dataset
dataset = load_dataset("json", data_files="intent_dataset.jsonl", split="train")

# Label encoding
unique_labels = sorted(set(example["label"] for example in dataset))
    label2id = {label: idx for idx, label in enumerate(unique_labels)}
    id2label = {idx: label for label, idx in label2id.items()}

    # Map label string ke ID numerik
    dataset = dataset.map(lambda x: {"label": label2id[x["label"]]})

    # Tokenizer
    tokenizer = AutoTokenizer.from_pretrained(model_name)

    # Tokenisasi + padding dan truncation dengan panjang maksimal 128
    tokenized_dataset = dataset.map(
    lambda x: tokenizer(x["text"], truncation=True, padding="max_length", max_length=128),
    batched=True
    )

    # Load model untuk klasifikasi
    model = AutoModelForSequenceClassification.from_pretrained(
    model_name,
    num_labels=len(label2id),
    id2label=id2label,
    label2id=label2id
    )

    # Argumen pelatihan
    training_args = TrainingArguments(
```

```
30.       output_dir="./bert-intent-model",
31.       per_device_train_batch_size=8,
32.       num_train_epochs=5,
33.       logging_dir="./logs",
34.       save_strategy="epoch",  # Simpan model setiap epoch
35.       save_total_limit=2,
36.       load_best_model_at_end=False,  # Tidak perlu karena
   tidak ada eval dataset
37.       logging_steps=50
38.       )

39.       # Trainer
40.       trainer = Trainer(
41.       model=model,
42.       args=training_args,
43.       train_dataset=tokenized_dataset
44.       )

45.       # Jalankan training
46.       trainer.train()

47.       # Simpan model dan tokenizer
48.       model.save_pretrained("bert-intent-model")
49.       tokenizer.save_pretrained("bert-intent-model")
```