Text Summarization

Portfolio Project 2

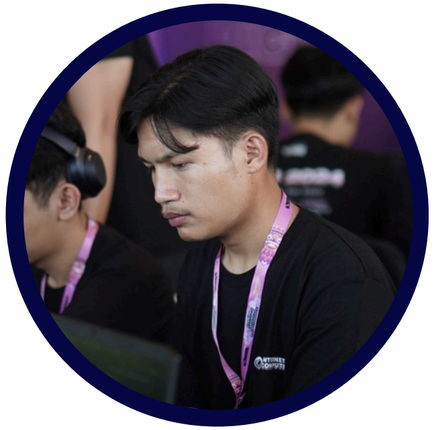
GENESIS

Muhammad Zuama

Data Preparation, Analisis And Developer

Juan Hendy Irmanto

Data Preparation, Analisis And Developer



Rizky Ahsan Syarief

Data Preparation, Analisis And Developer

Samuel David Sutanto

Data Preparation, Analisis And Developer

Jihan Salma Ramadhanti Widodo

Data Preparation, Analisis And Developer

Mohammad Arief Rajendra

Data Preparation, Analisis And Developer

LATAR BELAKANG

Koto et al. (2020) menemukan bahwa dataset untuk melatih *Text Summarizer* dalam Bahasa Indonesia terlalu kecil.

Oleh karena itu, mereka membuat dataset Liputan 6, berdasarkan data dari website berita tersebut, dan mengkoleksi 215,827 *article-summary pairs*.

Tujuan

- Membangun algoritma *Machine Learning* yang dapat melakukan *Text Summarization* terhadap artikel dari kanal berita Liputan 6 dengan akurat dan konsisten

Susunan Project

- Pengumpulan Data
- Data Understanding
- Data Modeling
- Evaluasi dan Pengujian

Business & Data Understanding

Business & Data Understanding

Data yang digunakan adalah dataset Liputan 6, yang dibagi menjadi *Canonical* dan *Xtreme*. *Canonical* digunakan untuk metode *extractive*, sementara *Xtreme* digunakan untuk metode *abstractive* (Koto et al., 2020).

Jumlah Observasi: **215.827 observasi** (*Canonical*)

Jumlah Fitur: **4 variabel**

Business & Data Understanding

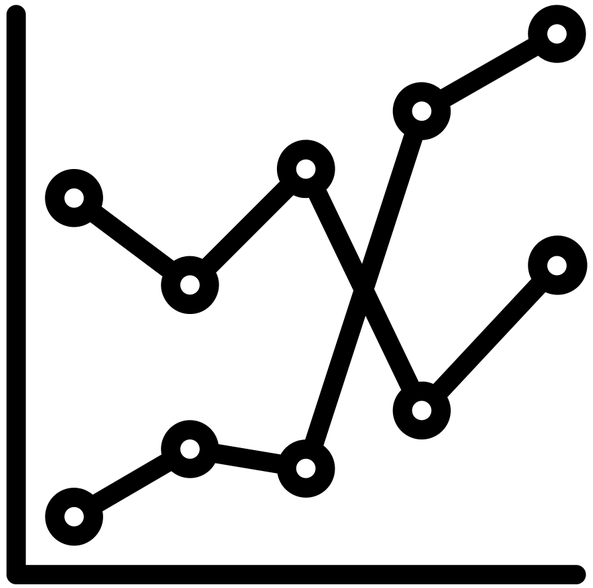
Fitur-fitur yang dapat ditemukan dalam dataset meliputi:

- id: Unique article identifier
- url: Link ke artikel di kanal berita Liputan6
- *clean_article*: Isi dari artikel
- *clean_summary*: Contoh summary

Karena data akan digunakan untuk process *deep learning*, preprocessing yang dilakukan akan cukup minimal.

Data Modeling

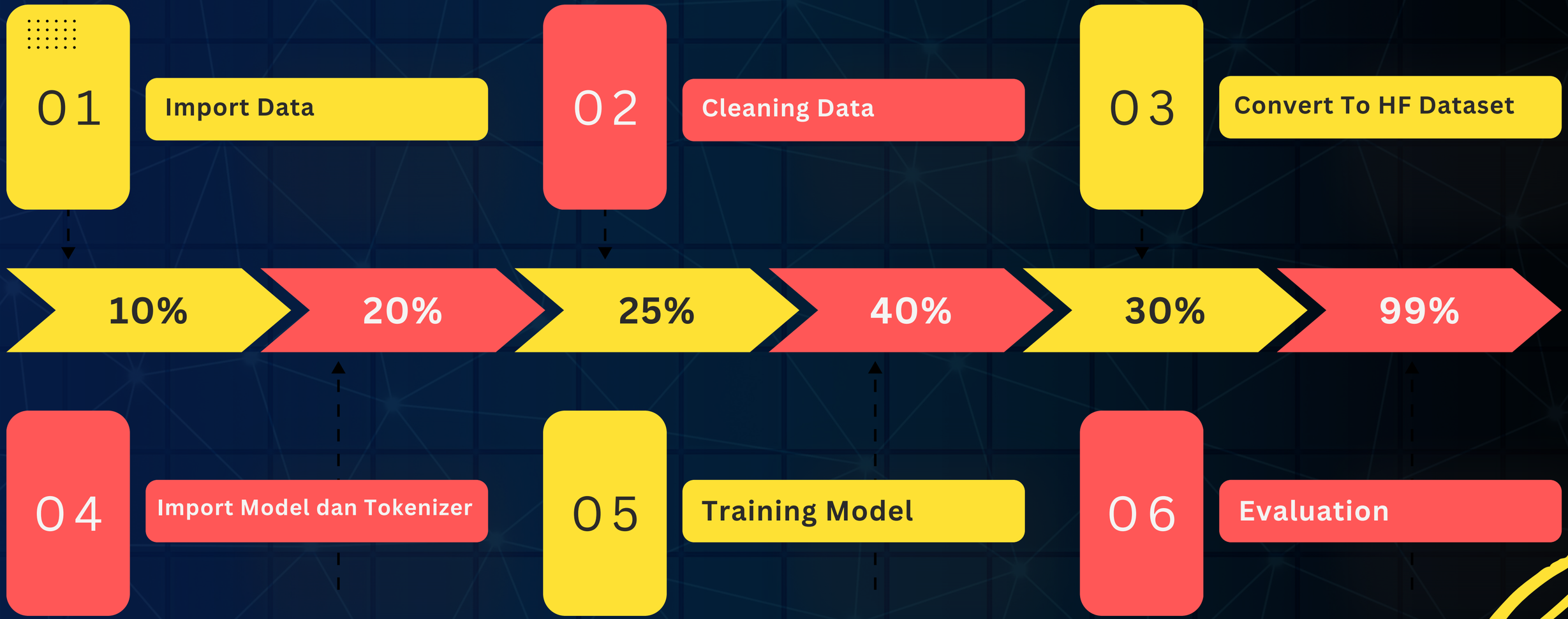
Data Modeling



Deep Learning Algorithms:

- BERT2BERT Indonesian Summarization
- BART Large CNN

ALUR PENGEMBANGAN MODEL



IMPORT DATA

1. **Penggunaan Dataset**

Dataset yang digunakan adalah dataset dari summarisation Liputan 6 dan sudah dibersihkan yang terdiri atas kolom clean_summary dan clean_article

2. **Penggunaan**

Dalam penggunaannya hanya menggunakan 5000 baris untuk training dan 1000 untuk data test

```
Informasi data_train_subset (5000 baris):
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 6 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   id                  5000 non-null   int64 
 1   url                 5000 non-null   object
 2   clean_article       5000 non-null   object
 3   clean_summary       5000 non-null   object
 4   extractive_summary  5000 non-null   object
 5   split               5000 non-null   object
dtypes: int64(1), object(5)
memory usage: 234.5+ KB

Informasi data_test_subset (1000 baris):
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 6 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   id                  1000 non-null   int64 
 1   url                 1000 non-null   object
 2   clean_article       1000 non-null   object
 3   clean_summary       1000 non-null   object
 4   extractive_summary  1000 non-null   object
 5   split               1000 non-null   object
dtypes: int64(1), object(5)
memory usage: 47.0+ KB

Informasi data_dev_subset (1000 baris):
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 6 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   id                  1000 non-null   int64 
 1   url                 1000 non-null   object
 2   clean_article       1000 non-null   object
 3   clean_summary       1000 non-null   object
 4   extractive_summary  1000 non-null   object
 5   split               1000 non-null   object
dtypes: int64(1), object(5)
memory usage: 47.0+ KB
```

Data Modeling

```
Informasi data_train_subset (5000 baris):
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 6 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   id                  5000 non-null   int64
 1   url                 5000 non-null   object
 2   clean_article       5000 non-null   object
 3   clean_summary       5000 non-null   object
 4   extractive_summary  5000 non-null   object
 5   split               5000 non-null   object
dtypes: int64(1), object(5)
memory usage: 234.5+ KB

Informasi data_test_subset (1000 baris):
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 6 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   id                  1000 non-null   int64
 1   url                 1000 non-null   object
 2   clean_article       1000 non-null   object
 3   clean_summary       1000 non-null   object
 4   extractive_summary  1000 non-null   object
 5   split               1000 non-null   object
dtypes: int64(1), object(5)
memory usage: 47.0+ KB

Informasi data_dev_subset (1000 baris):
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 6 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   id                  1000 non-null   int64
 1   url                 1000 non-null   object
 2   clean_article       1000 non-null   object
 3   clean_summary       1000 non-null   object
 4   extractive_summary  1000 non-null   object
 5   split               1000 non-null   object
dtypes: int64(1), object(5)
memory usage: 47.0+ KB
```

```
Informasi data_train_subset (5000 baris):
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 6 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   id                  10000 non-null  int64
 1   url                 10000 non-null  object
 2   clean_article       10000 non-null  object
 3   clean_summary       10000 non-null  object
 4   extractive_summary  10000 non-null  object
 5   split               10000 non-null  object
dtypes: int64(1), object(5)
memory usage: 468.9+ KB

Informasi data_test_subset (1000 baris):
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 6 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   id                  2000 non-null   int64
 1   url                 2000 non-null   object
 2   clean_article       2000 non-null   object
 3   clean_summary       2000 non-null   object
 4   extractive_summary  2000 non-null   object
 5   split               2000 non-null   object
dtypes: int64(1), object(5)
memory usage: 93.9+ KB

Informasi data_dev_subset (1000 baris):
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 6 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   id                  2000 non-null   int64
 1   url                 2000 non-null   object
 2   clean_article       2000 non-null   object
 3   clean_summary       2000 non-null   object
 4   extractive_summary  2000 non-null   object
 5   split               2000 non-null   object
dtypes: int64(1), object(5)
memory usage: 93.9+ KB
```

Dataset yang digunakan adalah dataset dari summarisation Liputan 6 dan sudah dibersihkan yang terdiri atas kolom *clean_summary* dan *clean_article*

Importing Models

```
[ ] #Load BART Model & Tokenization
    from transformers import BartTokenizer, BartForConditionalGeneration, Trainer, TrainingArguments

    model_name = "facebook/bart-large-cnn"
    tokenizer = BartTokenizer.from_pretrained(model_name)
    model = BartForConditionalGeneration.from_pretrained(model_name)
```

```
from transformers import BertTokenizer, EncoderDecoderModel
"""
Loading Tokenizer.
"""
tokenizer = BertTokenizer.from_pretrained("cahya/bert2bert-indonesian-summarization")
tokenizer.bos_token = tokenizer.cls_token
tokenizer.eos_token = tokenizer.sep_token

"""
Loading Summarization Model.
"""
model = EncoderDecoderModel.from_pretrained("cahya/bert2bert-indonesian-summarization")
model = model.to(device) # Moving into GPU if available.
```

```
def tokenize_function(example):
    inputs = tokenizer(example["article"], padding="max_length", truncation=True, max_length=512)
    targets = tokenizer(example["summary"], padding="max_length", truncation=True, max_length=128)

    inputs["labels"] = targets["input_ids"]
    return inputs

train_dataset = train_dataset.map(tokenize_function, batched=True)
val_dataset = val_dataset.map(tokenize_function, batched=True)
```

```
Map: 100% ████████████████████ 5000/5000 [00:07<00:00, 757.34 examples/s]
Map: 100% ████████████████████ 1000/1000 [00:01<00:00, 857.99 examples/s]
```

```
[ ] def tokenize_function(example):
        inputs = tokenizer(example["article"], padding="max_length", truncation=True, max_length=512)
        targets = tokenizer(example["summary"], padding="max_length", truncation=True, max_length=128)

        inputs["labels"] = targets["input_ids"]
        return inputs

    train_dataset = train_dataset.map(tokenize_function, batched=True)
    val_dataset = val_dataset.map(tokenize_function, batched=True)
```

```
Map: 100% ████████████████████ 10000/10000 [00:11<00:00, 778.22 examples/s]
Map: 100% ████████████████████ 2000/2000 [00:02<00:00, 721.71 examples/s]
```

BERT 2 BERT

```
from transformers import AutoTokenizer

model_name = "facebook/bart-large-cnn"
tokenizer = AutoTokenizer.from_pretrained(model_name)

max_input_length = 256
max_target_length = 256

def preprocess_function(example):
    input_enc = tokenizer(
        example['article'],
        max_length=max_input_length,
        truncation=True,
        padding="max_length"
    )
    target_enc = tokenizer(
        example['summary'],
        max_length=max_target_length,
        truncation=True,
        padding="max_length"
    )
    return {
        "input_ids": input_enc["input_ids"],
        "attention_mask": input_enc["attention_mask"],
        "labels": target_enc["input_ids"]
    }
```

```
from datasets import Dataset

train_dataset = Dataset.from_pandas(train_df)
train_dataset = train_dataset.map(preprocess_function)
train_dataset.set_format(type='torch', columns=['input_ids', 'attention_mask', 'labels'])
```

```
Map: 100%|██████████| 193883/193883 [04:15<00:00, 758.39 examples/s]
```

```
test_dataset = Dataset.from_pandas(test_df)
test_dataset = test_dataset.map(preprocess_function)
test_dataset.set_format(type='torch', columns=['input_ids', 'attention_mask', 'labels'])
```

```
Map: 100%|██████████| 10972/10972 [00:12<00:00, 860.33 examples/s]
```

BART Large CNN

Model Training and Evaluation

Proses fine-tuning dilakukan pada model BERT2BERT dan BART CNN menggunakan dataset ringkasan berita berbahasa Indonesia. Model dilatih ulang selama 3 epoch dengan batch size 8 menggunakan Trainer, yang juga mengatur evaluasi dengan metrik ROUGE. Fine-tuning difokuskan pada penyesuaian bobot encoder dan decoder agar mampu menghasilkan ringkasan yang sesuai dengan struktur dan gaya bahasa dataset target.

```python
from transformers import TrainingArguments

training_args = TrainingArguments(
    output_dir="/content/results",
    num_train_epochs=3,
    per_device_train_batch_size=4,
    per_device_eval_batch_size=4,
    # evaluation_strategy="no",
    save_strategy="no",
    logging_dir="/content/logs",
    logging_steps=1000,
    fp16=True,  # Mixed precision (hanya jika pakai GPU T4/V100/A100)
)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=val_dataset,
    tokenizer=tokenizer
)


trainer.train()
```

```
Using the `WANDB_DISABLED` environment variable is deprecated and will be
/tmp/ipython-input-30-753933343.py:15: FutureWarning: `tokenizer` is depre
  trainer = Trainer(
/usr/local/lib/python3.11/dist-packages/transformers/models/encoder_decode
  decoder_attention_mask = decoder_input_ids.new_tensor(decoder_input_ids
/usr/local/lib/python3.11/dist-packages/transformers/models/encoder_decode
  warnings.warn(DEPRECATION_WARNING, FutureWarning)
```

[4546/7500 20:30 < 13:19, 3.69 it/s, Epoch 1.82/3]

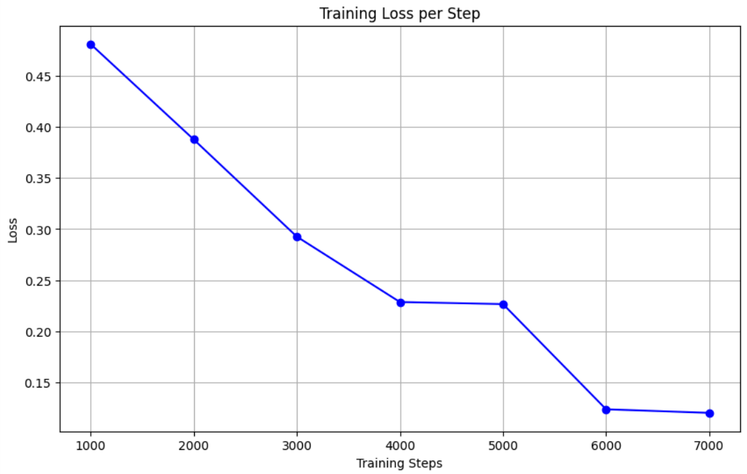
Step	Training Loss
1000	0.481300
2000	0.387800
3000	0.292600
4000	0.228500

[7500/7500 33:42, Epoch 3/3]

Step	Training Loss
1000	0.481300
2000	0.387800
3000	0.292600
4000	0.228500
5000	0.226500
6000	0.123400
7000	0.119900

```
TrainOutput(global_step=7500, training_loss=0.25610795237223305, metrics=
```

Model Evaluation

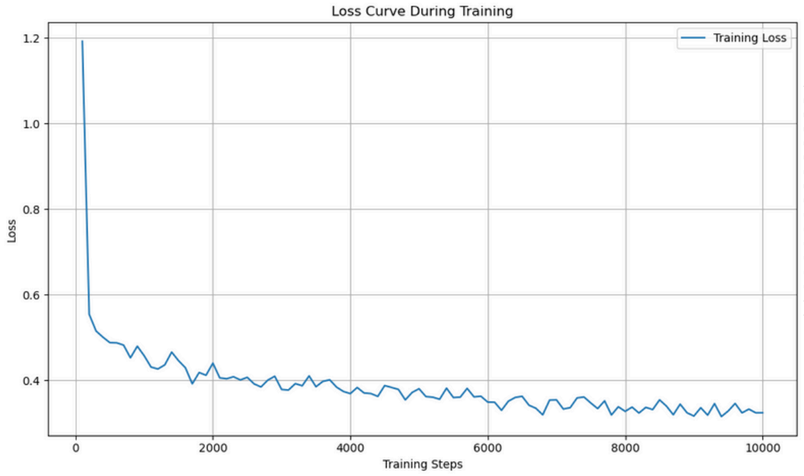


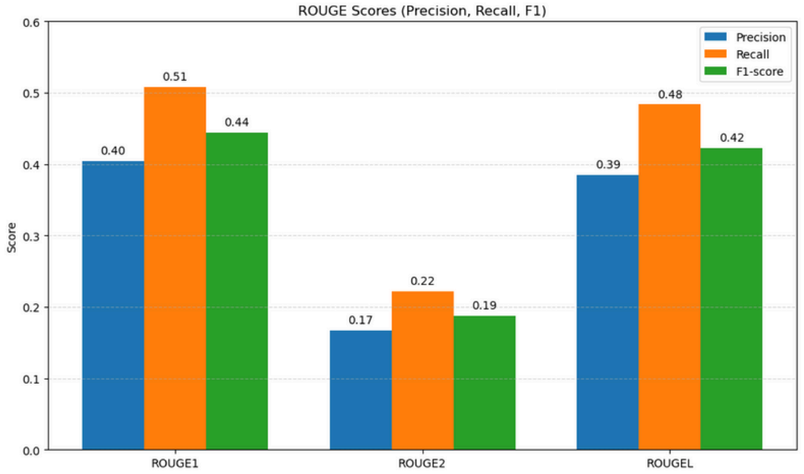

```
    references=reference_formatted,
    use_stemmer=True
)

# 8. Tampilkan hasil dengan pembulatan
print("📊 Evaluasi ROUGE pada 1000 data test sample:")
for key in ["rouge1", "rouge2", "rougeL", "rougeLsum"]:
    score = round(rouge_scores[key] * 100, 2)
    print(f"{key}: {score}")
```

```
Map: 100% 1000/1000 [00:01<00:00, 917.81 examples/s]
Map: 100% 1000/1000 [04:10<00:00,  4.50 examples/s]
📊 Evaluasi ROUGE pada 1000 data test sample:
rouge1: 38.52
rouge2: 22.51
rougeL: 34.4
rougeLsum: 34.49
```

```
def buat_prediksi(teks_input, model, tokenizer, device):
    """
    Membuat prediksi ringkasan dari teks input menggunakan model yang sudah dilat

    Args:
        teks_input (str): Teks artikel yang ingin diringkas.
        model: Model Transformer (EncoderDecoderModel) yang sudah dilatih.
        tokenizer: Tokenizer yang sesuai dengan model.
```

BERT 2 BERT



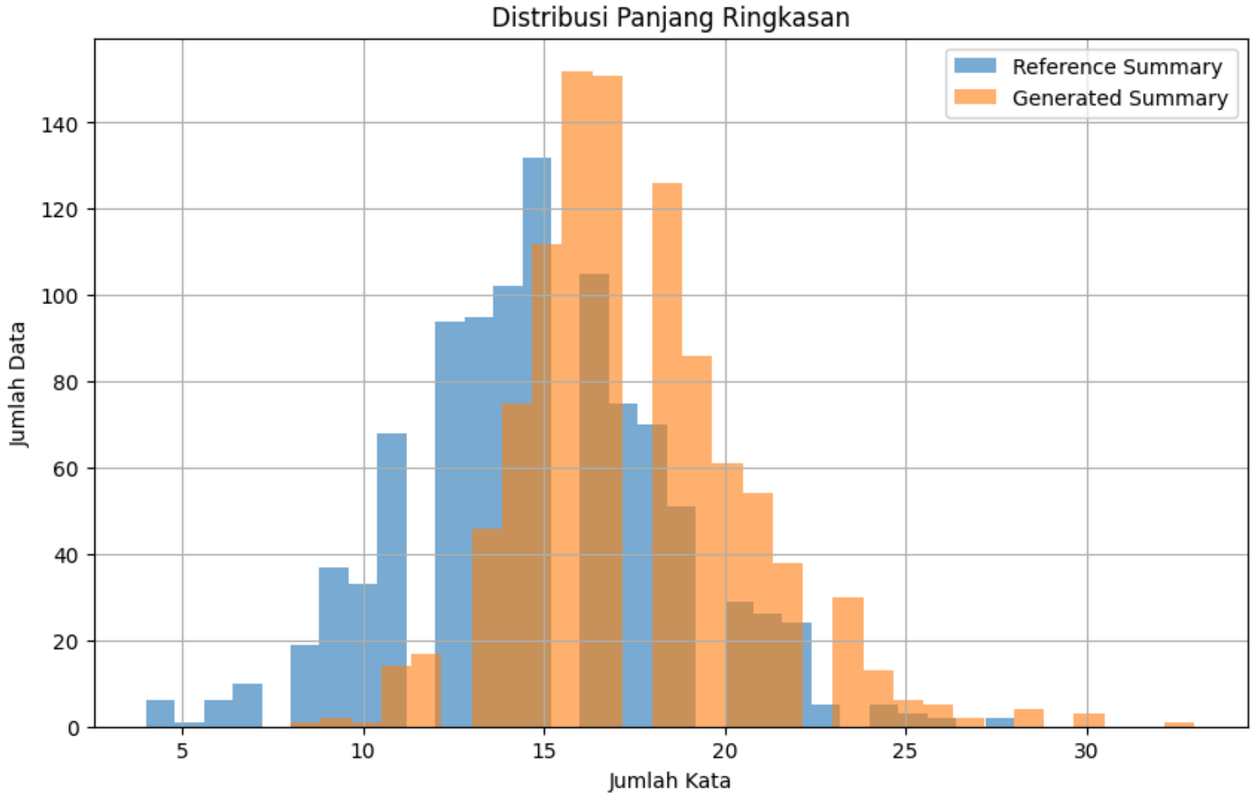


BART Large CNN

Hasil evaluasi menunjukkan kedua model memiliki performa cukup baik untuk membuat ringkasan artikel.

Berdasarkan score ROUGE, model BART Large CNN memiliki performa lebih baik.

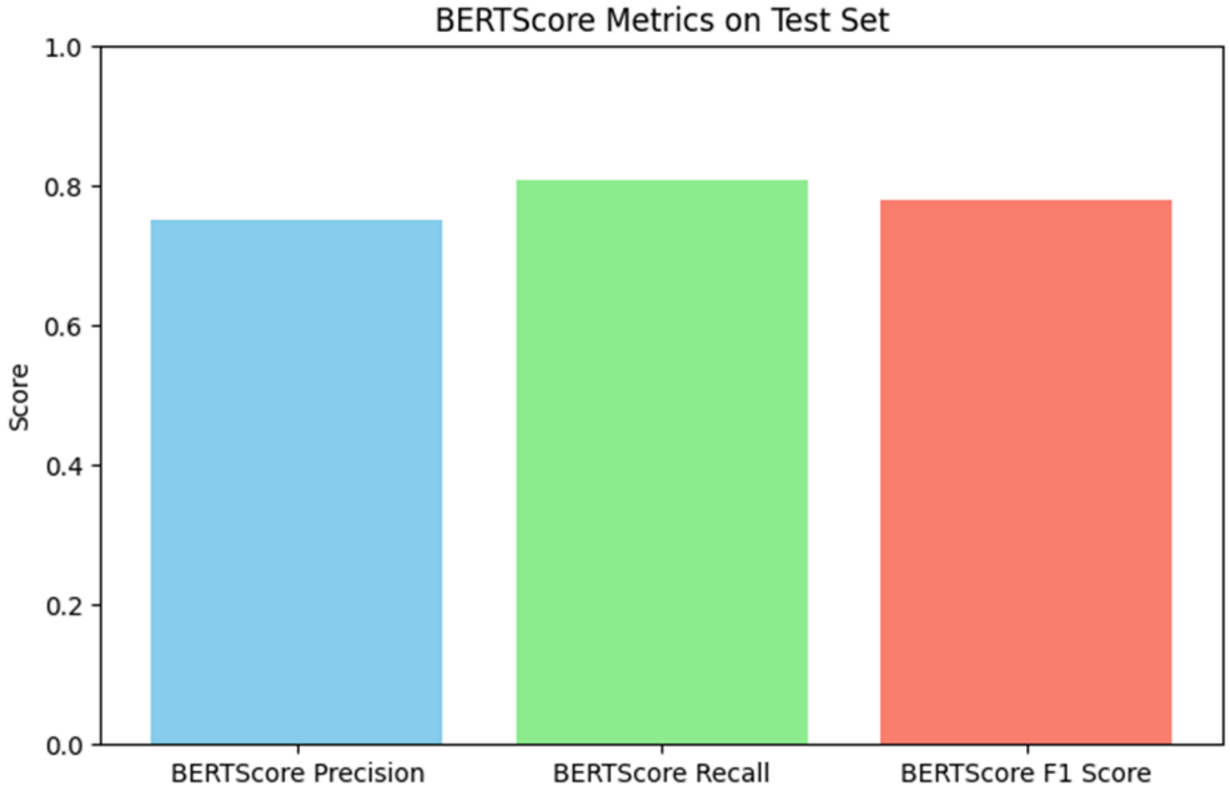
Model Evaluation

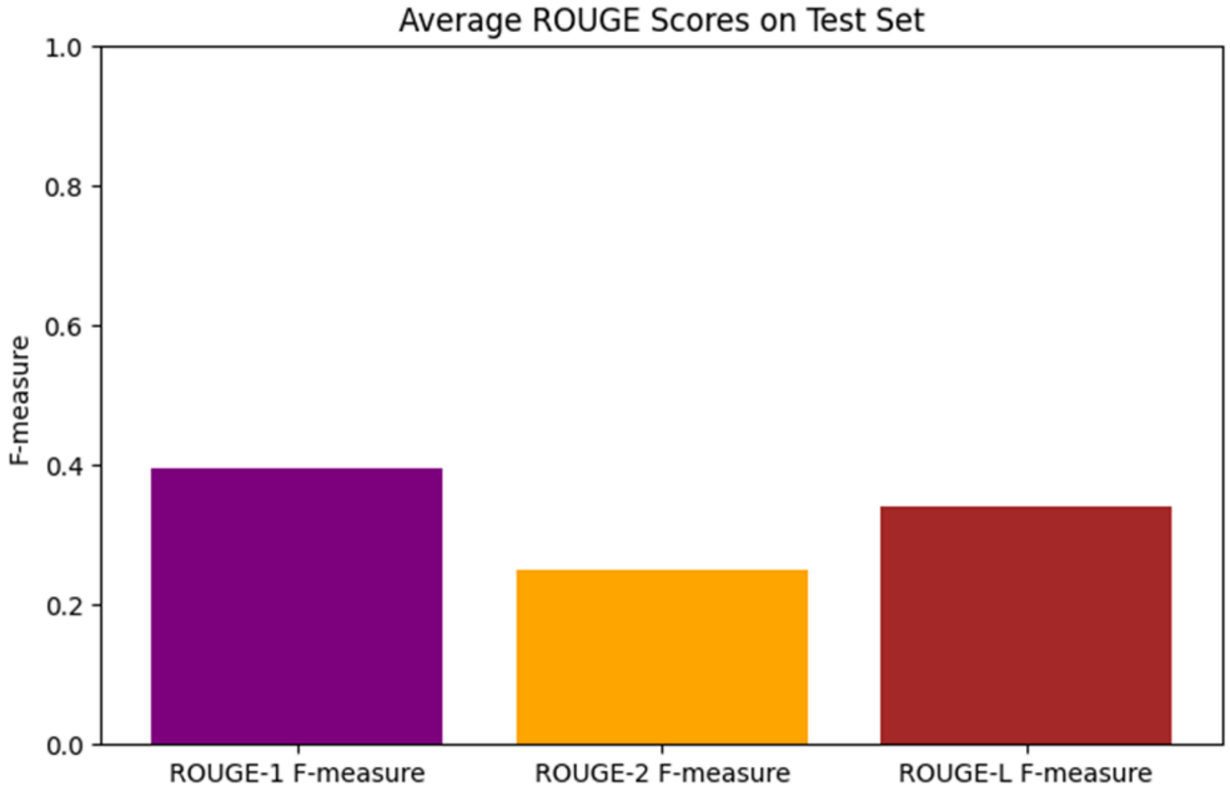


Evaluasi menggunakan 10.000 data Train dan 2000 data test menghasilkna evaluasi model yang lebih baik di banding yang menggunakna 5000 data train dan 1000 data test

```
Map: 100% 1000/1000 [00:01<00:00, 917.81 examples/s]
Map: 100% 1000/1000 [04:10<00:00,  4.50 examples/s]
📊 Evaluasi ROUGE pada 1000 data test sample:
rouge1: 38.52
rouge2: 22.51
rougeL: 34.4
rougeLsum: 34.49
```

Model Evaluation





BERTScore Precision: 0.7520
BERTScore Recall: 0.8092
BERTScore F1 Score: 0.7791

--- ROUGE Scores ---
ROUGE-1 F-measure: 0.3954
ROUGE-2 F-measure: 0.2495
ROUGE-L F-measure: 0.3421

Result Example

```python
import random

random_idx = random.randint(0, len(dev_df) - 1)
article_to_summarize = dev_df.loc[random_idx, "article"]

print(f"Selected article index: {random_idx}")
print(f"Article:\n{article_to_summarize[:500]}...")

input_ids = tokenizer.encode(article_to_summarize, return_tensors='pt')
summary_ids_1 = model.generate(input_ids,
            min_length=30,
            max_length=80,
            num_beams=4,
            repetition_penalty=2.0,
            length_penalty=1.0,
            early_stopping=True,
            no_repeat_ngram_size=2,
            use_cache=True,
            do_sample=True,
            temperature=1.2,
            top_k=50,
            top_p=0.95)

# Decode the summary
summary_text = tokenizer.decode(summary_ids_1[0], skip_special_tokens=True)
print("Summary 1: ", summary_text)
```

Python

```
[INFO|configuration_utils.py:1135] 2025-06-19 23:42:47,251 >> Generate config GenerationConfig {
  "bos_token_id": 0,
  "decoder_start_token_id": 2,
  "eos_token_id": 2,
  "forced_bos_token_id": 0,
  "forced_eos_token_id": 2,
  "pad_token_id": 1
}

Selected article index: 409
Article:
Jumat kemarin , Kejaksaan Agung mengumumkan tujuh tersangka baru dalam kasus penyalahgunaan dana Bantuan Likuiditas Bank Indonesia ( BLBI ) . Besar dugaan , dari ketujuh tersangka itu , negara telah dirugikan senilai Rp 12 triliun . Menurut Kepala Pus
Summary 1:  Tujuh tersangka baru dalam kasus penyalahgunaan dana BLBI ditetapkan Kejagung . Negara telah dirugikan senilai Rp 12 triliun . Mereka terbukti melanggar UU Nomor 3/1971 tentang Pemberantasan Korupsi
```

Hasil testing dari model BART Large CNN menunjukan ringkasan yang cukup baik dari artikel Liputan6

Genesis

Terimakasih Atas Perhatiannya