

# Technical REPORT

---

Prepared by  
Merta Bhuana  
[105222008]

Prepared by  
Fairus Rajendra  
[101121050]

# Technical Report

Judul Project	Zero-Shot Klasifikasi Tweet
Penulis 1	Fairus Rajendra Wiranata (101121050)
Penulis 2	Ni Putu Merta Bhuana N (105222008)
Institusi	Universitas Pertamina
Tanggal	22 April 2025

## PENDAHULUAN

### Latar Belakang

Analisis teks dalam bahasa Indonesia, khususnya dari media sosial seperti *Twitter*, menghadirkan tantangan tersendiri karena keterbatasan dataset terlabel yang memadai dan kompleksitas bahasa yang digunakan. Klasifikasi topik *tweet* secara otomatis dapat memberikan wawasan penting mengenai opini publik dan tren sosial. Namun, pendekatan tradisional yang mengandalkan pelatihan model pada dataset terlabel seringkali memerlukan sumber daya yang besar dan waktu yang lama. Solusi atas masalah ini dapat dengan, pendekatan Zero-Shot Learning, yang dapat membuat model untuk melakukan klasifikasi pada data yang belum pernah dilihat sebelumnya tanpa perlu pelatihan eksplisit pada setiap label.

Penelitian ini bertujuan untuk menerapkan metode Zero-Shot Learning dengan menggunakan model *transformer* mDeBERTa-v3 untuk mengklasifikasikan *tweet* berbahasa Indonesia ke dalam berbagai kategori topik, seperti Politik, Sumber Daya Alam, Demografi, dan lain-lain. Dengan menggunakan dataset *tweet* yang telah dilabeli, serta data yang belum dilabeli untuk pengujian, proyek ini bertujuan untuk menunjukkan bagaimana model mDeBERTa yang dilatih dengan pendekatan *multi-label classification* dapat mengklasifikasikan *tweet* ke dalam kategori topik yang relevan tanpa perlu pelatihan lebih lanjut. Hasil dari penelitian ini diharapkan dapat memperkaya aplikasi analisis teks dalam bahasa Indonesia, serta memberikan kontribusi dalam pengembangan teknik klasifikasi berbasis Zero-Shot Learning di dunia pemrosesan bahasa alami.

## PEMBAHASAN

### Tinjauan Pustaka

1. Zero-shot learning (ZSL) adalah skenario machine learning di mana model AI dilatih untuk mengenali dan mengkategorikan objek atau konsep tanpa harus melihat contoh kategori atau konsep tersebut sebelumnya [1].
2. Model transformer adalah Model transformer adalah jenis model pembelajaran mendalam yang diperkenalkan pada tahun 2017. Model ini dapat menerjemahkan teks dan ucapan hampir secara real-time. Misalnya, ada aplikasi yang sekarang memungkinkan wisatawan untuk berkomunikasi dengan penduduk setempat di jalan dalam bahasa utama mereka. Mereka membantu peneliti lebih memahami DNA dan mempercepat desain obat. Mereka dapat mendeteksi anomali dan mencegah penipuan di bidang keuangan dan keamanan. Vision transformer juga digunakan untuk tugas-tugas visi komputer [2].
3. Klasifikasi adalah proses memprediksi kelas atau kategori dari nilai yang diamati atau titik data yang diberikan. Output yang dikategorikan dapat memiliki bentuk seperti "Black" atau "White" atau "spam" atau "no spam". Secara matematis, classification adalah tugas mendekati fungsi pemetaan ( $f$ ) dari variabel input ( $X$ ) ke variabel output ( $Y$ ).

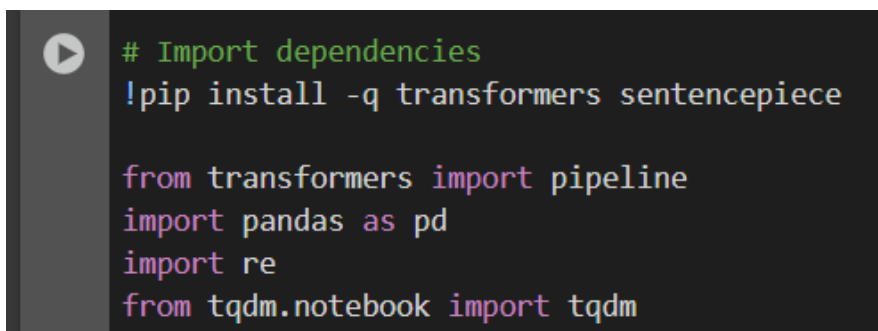
### Metodologi

Dataset terdiri dari *tweet* berbahasa Indonesia yang diberi label dengan topik yang berbeda. Data terbagi menjadi dua bagian, yaitu dataset terlabel dan dataset tidak terlabel. Dataset terlabel digunakan untuk evaluasi model. Pembersihan data dilakukan dengan menghapus URL, *mention* (@), dan *hashtag*, serta melakukan normalisasi teks seperti pengubahan teks ke huruf kecil. Model mDeBERTa digunakan dalam *pipeline Zero-Shot Classification*. Model ini memungkinkan klasifikasi berdasarkan prediksi tanpa pelatihan langsung pada data label tertentu. Evaluasi dilakukan menggunakan metrik *accuracy*, *precision*, *recall*, dan *F1-score* untuk mengukur kinerja model.

### Implementasi

#### 1. Import Dependencies

Bagian pertama dari *project* ini adalah pengimporan pustaka (*libraries*) yang diperlukan untuk menjalankan *script*. Pustaka ini memberikan fungsionalitas untuk operasi yang akan dilakukan di dalam kode.



```
# Import dependencies
!pip install -q transformers sentencepiece

from transformers import pipeline
import pandas as pd
import re
from tqdm.notebook import tqdm
```

Perintah pip instal ini menginstal dua pustaka Python yang penting, yaitu *transformers* dan *sentencepiece*, yang sering digunakan dalam pemrosesan bahasa alami (NLP). *transformers* digunakan untuk bekerja dengan model-model bahasa yang canggih, sedangkan *sentencepiece* adalah pustaka yang digunakan untuk pemrosesan tokenisasi teks. Selanjutnya, ada beberapa *import* dari pustaka yang sudah diinstal, seperti *pipeline* dari *transformers*, yang digunakan untuk mempermudah pemanggilan model pre-trained dalam tugas NLP.

Pustaka pandas (disingkat pd) diimpor untuk manipulasi dan analisis data dalam format tabel, sedangkan re digunakan untuk operasi pencocokan dan manipulasi ekspresi reguler (*regex*) pada teks. Terakhir, tqdm.notebook diimpor untuk menampilkan progress bar yang berguna saat menjalankan proses iterasi dalam notebook Jupyter, memberikan tampilan yang lebih interaktif dan informatif bagi pengguna. Semua *import* ini mendukung tujuan untuk memproses dan menganalisis teks dengan model-model NLP modern serta mempermudah pengolahan data. Tambahan, pustaka torch dari PyTorch diimpor, yang menyediakan fungsi untuk bekerja dengan tensor dan melakukan komputasi berbasis GPU.

## 2. Import Dataset

```
[ ] # Load labeled & unlabeled dataset
    from google.colab import files

    uploaded = files.upload() # Upload `dataset_labeled.csv` and `dataset_unlabeled.csv`
```

Mengunggah dataset yang telah diberi label (labeled) dan yang tidak diberi label (unlabeled) ke Google Colab.

## 3. Membaca Dataset

```
# Read dataset
df_labeled = pd.read_csv('dataset_labeled.csv', sep=';')
df_unlabeled = pd.read_csv('dataset_unlabeled.csv', sep=';')

df_labeled.columns = ['tweet', 'label']
df_unlabeled.columns = ['id', 'tweet']

df_labeled.head()
```

Membaca dataset yang telah diunggah sebelumnya. Selanjutnya, kolom-kolom dataset diberi nama yang sesuai, yaitu 'tweet' dan 'label' untuk dataset yang sudah diberi label, serta 'id' dan 'tweet' untuk dataset yang tidak diberi label. Terakhir, `df_labeled.head()` digunakan untuk menampilkan lima baris pertama dari dataset yang sudah diberi label, untuk memeriksa struktur data yang dimuat.

## 4. Exploratory Data Analysis

```
# Simple EDA
print("Jumlah data label:", len(df_labeled))
print("Distribusi label:\n", df_labeled['label'].value_counts())
df_labeled['text_length'] = df_labeled['tweet'].apply(lambda x: len(str(x)))
print(df_labeled['text_length'].describe())
```

Eksplorasi data dasar (EDA) pada dataset yang telah diberi label. Pertama, baris pertama mencetak jumlah total data pada dataset berlabel menggunakan `len()`. Selanjutnya, menghitung distribusi jumlah setiap label dalam kolom 'label' dan mencetaknya. Kemudian, sebuah kolom baru bernama `text_length` ditambahkan ke dataset, yang menghitung panjang setiap tweet dengan menggunakan `apply()` dan fungsi `len()` untuk menghitung jumlah karakter dalam kolom 'tweet'. Terakhir, `df_labeled['text_length'].describe()` digunakan untuk menampilkan statistik deskriptif (seperti mean, standar deviasi, nilai minimum, dan maksimum) dari panjang teks dalam dataset.

## 5. Visualisasi EDA

```
# EDA visualization
import matplotlib.pyplot as plt

# Check class distribution
label_counts = df_labeled['label'].value_counts()
label_counts.plot(kind='bar', title='Distribusi Label Tweet', figsize=(10, 5), color='skyblue')
plt.xticks(rotation=45)
plt.ylabel("Jumlah Tweet")
plt.show()

# Tweet length analysis
df_labeled['length'] = df_labeled['tweet'].apply(lambda x: len(str(x).split()))
df_labeled['length'].hist(bins=30, figsize=(10, 5), color='orange')
plt.title('Distribusi Panjang Tweet (dalam kata)')
plt.xlabel('Jumlah Kata')
plt.ylabel('Jumlah Tweet')
plt.show()
```

Visualisasi EDA pada dataset yang telah diberi label. Pertama, distribusi label divisualisasikan dengan membuat grafik batang menggunakan `matplotlib`. Fungsi `value_counts()` digunakan untuk menghitung jumlah tweet per label, dan `plot(kind='bar')` digunakan untuk menggambarkan distribusinya. Grafik tersebut diberi judul "Distribusi Label Tweet", dengan sumbu x yang berisi label dan sumbu y yang menunjukkan jumlah tweet untuk setiap label.

Setelah itu, pada bagian kedua, panjang tweet dihitung berdasarkan jumlah kata dalam setiap tweet (dengan `len(str(x).split())`), dan hasilnya divisualisasikan dalam histogram untuk menunjukkan distribusi panjang tweet dalam jumlah kata. Histogram ini diberi warna oranye dengan 30 bin, serta diberi judul dan label sumbu x dan y yang sesuai. Kedua visualisasi ini membantu dalam memahami distribusi label dan panjang tweet dalam dataset.

## 6. Preprocessing

```
# Preprocessing (sanitize text and regularize to lowercase)
def clean_tweet(text):
    text = re.sub(r"http\S+", "", text)
    text = re.sub(r"@[A-Za-z0-9_]+", "", text)
    text = re.sub(r"#[A-Za-z0-9_]+", "", text)
    text = re.sub(r"^[^a-zA-Z0-9\s]", "", text)
    return text.strip().lower()

df_labeled['cleaned_tweet'] = df_labeled['tweet'].apply(clean_tweet)
df_unlabeled['cleaned_tweet'] = df_unlabeled['tweet'].apply(clean_tweet)

df_labeled.head()
```

Pembersihan pada teks tweet untuk menghilangkan elemen-elemen yang tidak diinginkan dan memastikan teks berada dalam format yang lebih seragam. Setelah teks dibersihkan, fungsi ini juga mengubah seluruh teks menjadi huruf kecil dengan lower() dan menghapus spasi ekstra di awal atau akhir teks menggunakan strip(). Kolom cleaned\_tweet ditambahkan ke kedua dataset (df\_labeled dan df\_unlabeled) dengan hasil pemrosesan teks yang telah dibersihkan. Terakhir, df\_labeled.head() digunakan untuk menampilkan lima baris pertama dari dataset yang telah diproses untuk memverifikasi hasilnya. Tambahan, setelah itu melihat isi data paling atas dari data yang tidak berlabel

## 7. Klasifikasi Zero-Shot

```
# Zero-shot classifier
# English labels to fit the model's language
candidate_labels = [
    "Ideology", "Politics", "Economy", "Social Culture",
    "Defense and Security", "Natural Resources", "Geography", "Demographics"
]

# Label map to match English and Indonesian prediction
label_map = {
    "Politik": "Politics",
    "Ekonomi": "Economy",
    "Sosial Budaya": "Social Culture",
    "Pertahanan dan Keamanan": "Defense and Security",
    "Sumber Daya Alam": "Natural Resources",
    "Geografi": "Geography",
    "Demografi": "Demographics",
    "Ideologi": "Ideology"
}

df_labeled['mapped_label'] = df_labeled['label'].map(label_map)
df_labeled = df_labeled[df_labeled['mapped_label'].notnull()]

# 0 = GPU, -1 = CPU
device = 0 if useGPU else -1
classifier = pipeline("zero-shot-classification", model="joeddav/xlm-roberta-large-xnli", device=device)
```

Baris kode ini untuk mengklasifikasikan tweet ke dalam berbagai topik yang telah ditentukan. Label-label ini adalah kategori yang akan diprediksi oleh model.

Kemudian, terdapat `label_map` yang digunakan untuk mencocokkan label dalam bahasa Indonesia yang ada dalam dataset (seperti "Politik", "Ekonomi", dll.) dengan label yang sesuai dalam bahasa Inggris. Hal ini memastikan bahwa label dalam bahasa Indonesia dapat dipetakan dengan benar ke dalam format yang digunakan oleh model. Setelah itu, kolom baru `mapped_label` ditambahkan ke dataset `df_labeled`, yang berisi label yang telah dipetakan ke dalam bahasa Inggris. Baris-baris yang tidak dapat dipetakan ke label bahasa Inggris akan dihapus.

*Pipeline zero-shot-classification* diinisialisasi menggunakan model pre-trained `xlm-roberta-large-xnli`, yang mampu melakukan klasifikasi teks dengan pendekatan zero-shot. Model ini akan digunakan untuk memprediksi topik tweet tanpa memerlukan pelatihan lebih lanjut.

```
# Batch of 16 tweets per GPU batch
if useGPU:
    BATCH_SIZE = 16

    def batch_classify(texts, labels, batch_size=16):
        predictions = []
        for i in tqdm(range(0, len(texts), batch_size)):
            batch = texts[i:i+batch_size]
            results = classifier(batch, labels)
            for result in results:
                predictions.append(result['labels'][0])
        return predictions

# Classify labeled dataset
tqdm.pandas()
if useGPU:
    df_labeled['predicted'] = batch_classify(df_labeled['cleaned_tweet'].tolist(), candidate_labels)
else:
    df_labeled['predicted'] = df_labeled['cleaned_tweet'].progress_apply(
        lambda x: classifier(x, candidate_labels)['labels'][0]
    )
```

Mengklasifikasikan tweet dalam dataset yang telah diberi label dengan model zero-shot classification. Jika GPU tersedia, ukuran batch untuk klasifikasi ditetapkan menjadi 16. Fungsi `batch_classify()` digunakan untuk memproses tweet dalam batch, mengklasifikasikan teks secara bertahap dan mengumpulkan hasil prediksi. Hasil prediksi disimpan dalam kolom baru `predicted` di dataset `df_labeled`. Jika GPU tidak tersedia, klasifikasi dilakukan per tweet menggunakan `progress_apply()` dengan menampilkan progres menggunakan `tqdm`. Dengan cara ini, proses klasifikasi dapat dilakukan secara efisien, baik dengan CPU maupun GPU.



## 8. Evaluasi

```
# Model evaluation in the same language
from sklearn.metrics import classification_report
print(classification_report(df_labeled['mapped_label'], df_labeled['predicted']))
```

Mengevaluasi kinerja model klasifikasi dengan membandingkan hasil prediksi yang dihasilkan oleh model dengan label yang sebenarnya. Hasil evaluasi ini memberikan gambaran seberapa baik model dalam mengklasifikasikan tweet sesuai dengan label yang benar.

## 9. Confusion Matrix

```
# Confusion matrix
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

cm = confusion_matrix(df_labeled['mapped_label'], df_labeled['predicted'], labels=candidate_labels)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=candidate_labels)
disp.plot(xticks_rotation=45, cmap='Blues', values_format='d')
```

Menampilkan matriks (confusion matrix) untuk mengevaluasi hasil klasifikasi model. Fungsi `confusion_matrix()` dari pustaka `sklearn.metrics` digunakan untuk menghitung matriks dengan membandingkan label asli (`mapped_label`) dan label yang diprediksi (`predicted`). Matriks ini menunjukkan jumlah prediksi yang benar dan salah untuk setiap kategori dan memberikan gambaran yang jelas tentang seberapa baik model mengklasifikasikan setiap kategori.

## 10. Penerapan Model

```
# Apply prediction to unlabeled dataset
df_unlabeled['predicted'] = batch_classify(df_unlabeled['cleaned_tweet'].tolist(), candidate_labels)
df_unlabeled.head()
```

Menerapkan model zero-shot classification pada dataset yang belum diberi label (`df_unlabeled`). Pertama, fungsi `batch_classify()` digunakan untuk mengklasifikasikan tweet dalam dataset tersebut, dan hasil prediksi disimpan dalam kolom baru `predicted`. Setelah klasifikasi selesai, lima baris pertama dari dataset yang telah diprediksi ditampilkan menggunakan `df_unlabeled.head()`.

```
# Save prediction results
predicted_result = 'predicted_unlabeled.csv'
df_unlabeled.to_csv(predicted_result, index=False)
files.download(predicted_result)
```

Selanjutnya, hasil prediksi disimpan dalam file CSV dengan nama `predicted_unlabeled.csv` menggunakan `to_csv()`.



## Hasil

Hasil yang diperoleh menunjukkan bahwa pendekatan Zero-Shot Learning dengan model transformer dapat bekerja dengan baik dalam mengklasifikasikan *tweet* dalam berbagai kategori meskipun model tidak dilatih secara eksplisit pada setiap kategori. Namun, ketidakseimbangan dalam distribusi label dan panjang tweet yang sangat bervariasi dapat mempengaruhi akurasi model. Model ini sangat efektif untuk penggunaan yang tidak memerlukan data label yang sangat banyak, tetapi untuk meningkatkan kinerja pada kategori dengan jumlah data yang sedikit, fine-tuning atau pendekatan lain seperti oversampling atau pengoptimalan algoritma bisa dilakukan.

Model mDeBERTa-v3 berhasil menangani klasifikasi Zero-Shot dengan baik, dan meskipun akurasi untuk kategori tertentu dapat ditingkatkan, hasil yang diperoleh menunjukkan bahwa model ini dapat diandalkan untuk aplikasi klasifikasi teks di bahasa Indonesia, terutama dalam konteks analisis sentimen atau pengelompokan topik di media sosial.

## KESIMPULAN

Penerapan Zero-Shot Learning menggunakan model transformer mDeBERTa-v3 dalam mengklasifikasikan tweet berbahasa Indonesia berhasil menunjukkan potensi yang menjanjikan, meskipun model ini tidak dilatih secara eksplisit pada label yang ada. Hasil yang diperoleh dari eksperimen menunjukkan bahwa model mampu mengklasifikasikan tweet dengan akurasi yang cukup baik, terutama pada kategori dengan jumlah data yang lebih banyak seperti Politik. Namun, ketidakseimbangan jumlah data antar kategori dan panjang teks tweet yang bervariasi tetap memberikan tantangan bagi model, terutama dalam mengklasifikasikan kategori dengan data terbatas.

Dengan demikian, meskipun model ini tidak memerlukan pelatihan khusus untuk setiap label, hasilnya masih sangat bergantung pada distribusi data yang digunakan. Untuk meningkatkan kinerja, terutama pada kategori dengan sedikit data, penerapan teknik tambahan seperti fine-tuning, pengoptimalan distribusi data, atau penggunaan model lain mungkin diperlukan. Terlepas dari itu, pendekatan Zero-Shot Learning terbukti efektif dan efisien, memberikan solusi bagi analisis teks dalam bahasa Indonesia tanpa memerlukan dataset yang sangat besar atau pelatihan yang memakan banyak waktu. Hasil ini membuka peluang besar untuk penggunaan teknologi serupa dalam analisis media sosial atau aplikasi lain yang memerlukan klasifikasi teks dalam berbagai kategori.

## DAFTAR PUSTAKA

- [1] <https://www.ibm.com/id-id/think/topics/zero-shot-learning>
- [2] <https://www.ibm.com/id-id/think/topics/transformer-model>
- [3] <https://medium.com/@bondansatrio99/apa-itu-classification-dalam-machine-learning-bcdf4fcdc614>