Topic Level Sentiment Analysis pada Ulasan Pengguna Aplikasi Tix ID

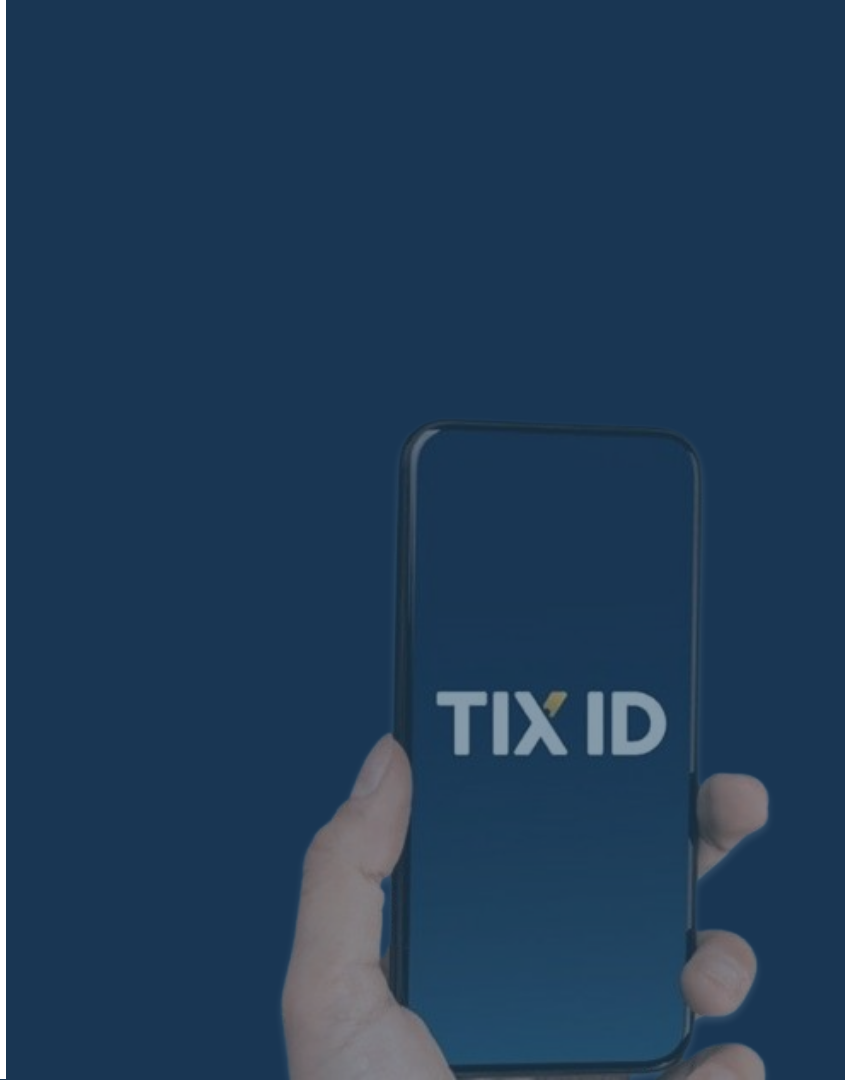
Topik Khusus 2: Web Mining(A)
Dosen Pengampu: Prof. Dr. rer. nat. Hendri Murfi, S.Si., M.Kom.
SCMA604902

Kelompok 2
Athalla Karenza Zimraan (2206826450)
Kirono Dwi Saputro (2106656365)
Hilmy Rahmadani (2206810490)
Geraldus Harry Pascal (2206048663)

Table of Contents

01

Pendahuluan



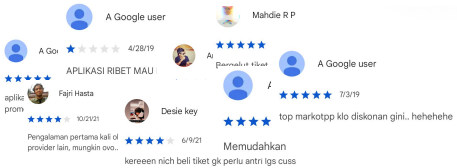
Pendahuluan

Latar Belakang

Profil TIX ID

TIX ID adalah platform digital asal Indonesia yang menyediakan layanan pembelian tiket bioskop secara online. Diluncurkan pada 2017, TIX ID awalnya bekerja sama eksklusif dengan jaringan bioskop seperti CGV dan Cinema XXI. Selain tiket bioskop, TIX ID kini juga menyediakan tiket event dan layanan hiburan lainnya. Aplikasi ini telah berkembang menjadi salah satu layanan terpopuler dalam industri hiburan digital di Indonesia.

Business Services	Business Goals
Penjualan Tiket Digital	**Meningkatkan Transaksi dan Pengguna Aktif**
Pembayaran & Integrasi Dompet Digital	**Meningkatkan Loyalitas dan Kepuasan Pelanggan**
Notifikasi & Konten Hiburan	**Memperkuat Ekosistem Digital & Kolaborasi Mitra**



Topic Level Sentiment Analysis to TIX ID

Untuk memahami opini pelanggan secara mendalam pada tiap aspek layanan seperti pembayaran, notifikasi, dan pemesanan tiket. TIX ID membutuhkan Topic-Level Sentiment Analysis guna meningkatkan pengalaman, loyalitas, dan pertumbuhan bisnis.

Pendahuluan

Rumusan Masalah, dan Tujuan

Rumusan Masalah

Bagaimana mengklasifikasikan sentimen ulasan pengguna aplikasi Tix ID, mengidentifikasi topik utama yang sering muncul, serta menyajikan visualisasi dan insight yang dapat mendukung pengambilan keputusan dalam pengembangan aplikasi?

Tujuan

Mengklasifikasikan sentimen (positif dan non-positif) pada ulasan pengguna aplikasi Tix ID.

Mengidentifikasi topik-topik utama yang sering muncul dalam ulasan pengguna.

Menyediakan visualisasi dan insight dari hasil analisis sentimen dan topik yang dapat digunakan untuk mendukung pengambilan keputusan dalam pengembangan aplikasi Tix ID.

Batasan Masalah

Batasan Masalah

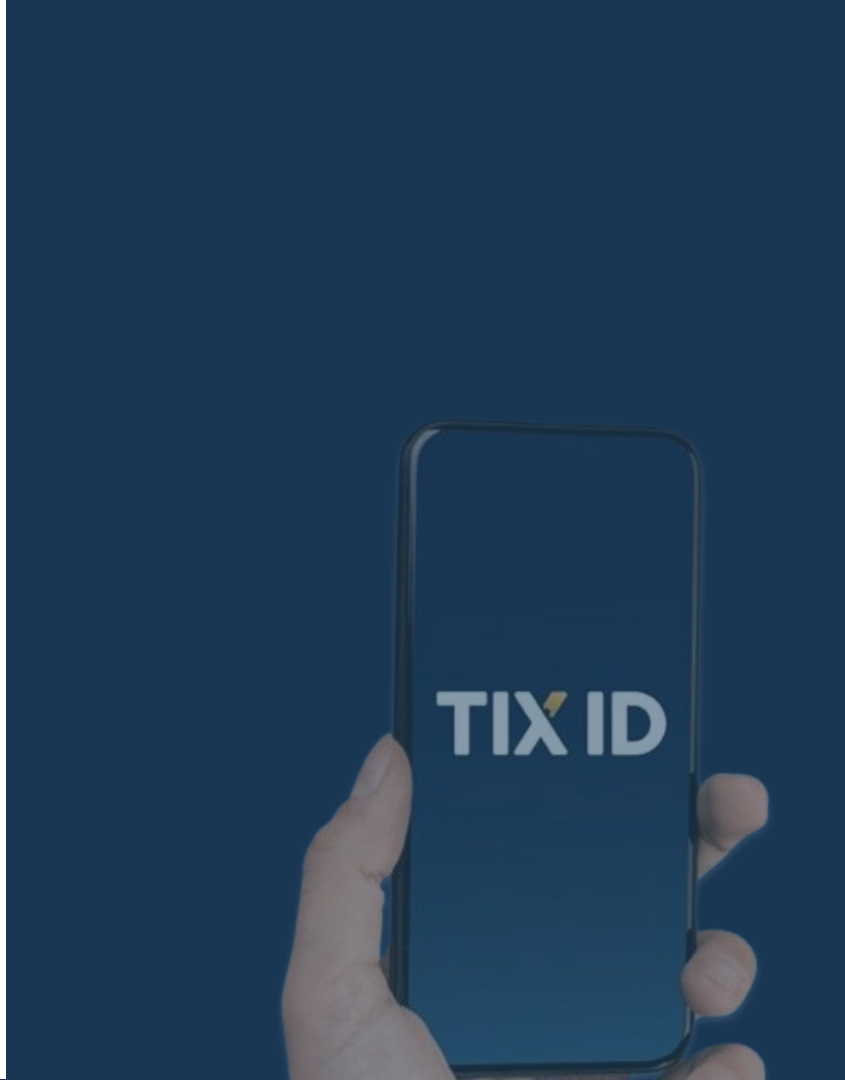
Data ulasan yang digunakan terbatas pada platform Google Play Store, dan tidak mencakup ulasan dari media sosial atau platform lain seperti App Store atau Twitter.

Analisis hanya dilakukan terhadap ulasan berbahasa Indonesia, bahasa inggris, dan campuran, sehingga ulasan dalam bahasa asing lainnya tidak disertakan dalam proses klasifikasi dan deteksi topik.

Model yang digunakan adalah BERT, tanpa melakukan pengujian atau perbandingan dengan model NLP lain atau pendekatan non-transformer.

02

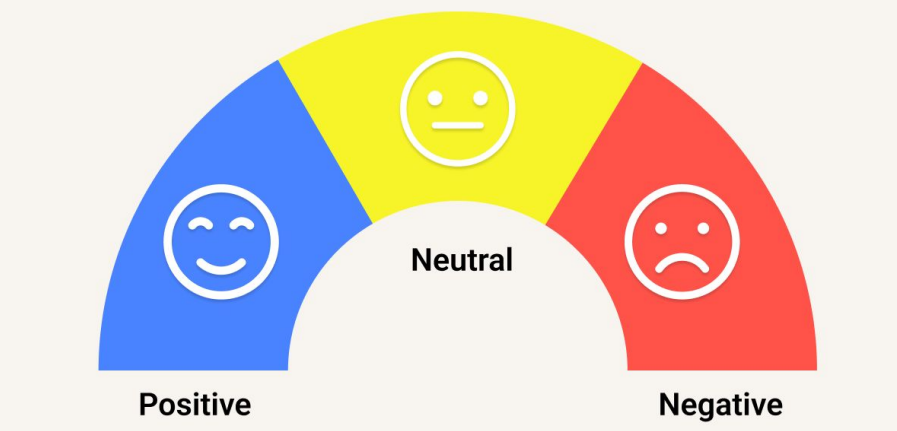
Metode



Metode

Sentiment Analysis dan Topic Detection

Sentiment Analysis



Sentiment analysis atau analisis sentimen adalah proses yang digunakan untuk mengidentifikasi dan menilai emosi atau opini dalam suatu teks, apakah bersifat positif, negatif, atau netral. Teknik ini banyak digunakan dalam bidang bisnis dan media sosial untuk memahami persepsi publik terhadap produk, layanan, atau isu tertentu. Dengan memanfaatkan metode pemrosesan bahasa alami (Natural Language Processing/NLP), analisis sentimen memungkinkan komputer untuk secara otomatis mengkategorikan ekspresi subjektif dalam data teks.

Topic Detection

Topic detection merupakan teknik yang digunakan untuk menemukan dan mengelompokkan tema atau topik utama yang muncul dalam kumpulan dokumen atau teks dalam skala besar. Proses ini membantu dalam memahami konten dan tren informasi, serta mempermudah pengorganisasian data secara otomatis. Salah satu metode populer dalam deteksi topik adalah BERTopic, yang memanfaatkan representasi vektor dari model bahasa seperti BERT serta teknik clustering untuk mengelompokkan dokumen ke dalam topik-topik yang bermakna secara semantik.

Metode

Metode Penelitian

Workflow

Data

Data Tabular Review Aplikasi TIX.ID

reviewId	userName
ade46460-a93d-4440-9448-2f5039ba9275	Asep Surya
9804a951-cf21-4c52-aaf0-683607a5bf13	Prasetyo P
1b6b070d-01fd-4767-9ad0-2323dbedf847	Riyanti Mulyasari
90aef585-d1af-49b6-b8c6-4827ca6a1a56	Rehan BJbjBJ
92d1c59a-f2af-4d00-97f1-f6eb7e8a0e3e	OndolSatrio

content	score	t
Ini aplikasi di Indonesia tetapi sinopsis film...	3	
Lengkap dan user friendly	5	
Baguss	5	
Aplikasi paling keren baru login mau pesan eh ...	5	
Kepada yang terhormat PT elang jaya ini tolong...	1	

replyContent	
Hi TIX Hunter, terima kasih atas review & sara...	2024-12
Hi TIX Hunter, terima kasih atas review yang d...	2024-12
Hi TIX Hunter, terima kasih atas review yang d...	2024-12
Hi TIX Hunter, terima kasih atas review yang d...	2024-12
Hi TIX Hunter, mohon maaf atas ketidaknyamanan...	2024-12

Pre-Processing

- **Data Cleaning**
- **Data Annotation**
- **Text Cleaning**
- **Text Normalization**
- **Tokenizing (BERT Tokenizer)**
- **Embedding**
- **Label Encoding**

Modelling

Parameter Tuning

- Learning Rate
- Batch Size
- Max Len
- Epochs

BERT (Klasifikasi Sentimen)

BERTopic (Deteksi Topik)

Output

Performance Metrics

- Confusion Matrix
- Precision
- Recall
- F1 Score
- Accuracy

Analysis & Visualization

- Topic Distribution Chart
- Sentiment-Topic Cross Analysis
  a. Top Positive/Negative Topics
- Word Clouds

Metode

Data

Sumber Data

	reviewId	userName	userImage	content	score	thumbsUpCount	reviewCreatedVersion	at	replyContent	repliedAt	appVersion
0	ade46460-a93d-4440-9448-2f5039ba9275	Asep Surya	https://play-lh.googleusercontent.com/a-/ALV-U...	Ini aplikasi di Indonesia tetapi sinopsis film...	3	0	3.12.0	2024-12-12 08:01:53	Hi TIX Hunter, terima kasih atas review & sara...	2024-12-12 08:10:24	3.12.0
1	9804a951-cf21-4c52-aaf0-683607a5bf13	Prasetyo P	https://play-lh.googleusercontent.com/a-/ALV-U...	Lengkap dan user friendly	5	0	3.12.0	2024-12-11 22:24:55	Hi TIX Hunter, terima kasih atas review yang d...	2024-12-12 02:04:21	3.12.0
2	1b6b070d-01fd-4767-9ad0-2323dbedf847	Riyanti Mulyasari	https://play-lh.googleusercontent.com/a-/ALV-U...	Baguss	5	0	3.12.0	2024-12-11 13:29:28	Hi TIX Hunter, terima kasih atas review yang d...	2024-12-11 13:43:47	3.12.0
3	90aef585-d1af-49b6-b8c6-4827ca6a1a56	Rehan BJbjBJ	https://play-lh.googleusercontent.com/a/ACg8oc...	Aplikasi paling keren baru login mau pesan eh ...	5	0	3.12.0	2024-12-11 10:37:52	Hi TIX Hunter, terima kasih atas review yang d...	2024-12-11 10:40:40	3.12.0
4	92d1c59a-f2af-4d00-97f1-f6eb7e8a0e3e	OndolSatrio	https://play-lh.googleusercontent.com/a-/ALV-U...	Kepada yang terhormat PT elang jaya ini tolong...	1	1	3.12.0	2024-12-11 04:36:12	Hi TIX Hunter, mohon maaf atas ketidaknyamanan...	2024-12-11 04:47:21	3.12.0

Dataset yang digunakan dalam penelitian ini berbentuk tabular dengan tipe data teks, berisi total 77487 entri (baris) dan 11 fitur (kolom). Data ini diambil dari *Google Play Store* daerah Indonesia melalui situs web Kaggle.

reviewId	ID unik untuk setiap ulasan yang diberikan pengguna.	**Object**
userName	Nama pengguna yang memberikan ulasan.	**Object**
userImage	URL gambar profil pengguna yang memberikan ulasan.	**Object**
content	Isi teks ulasan yang diberikan oleh pengguna.	**Object**
score	Skor ulasan yang diberikan pengguna, biasanya dalam skala 1-5.	**Int64**

Metode

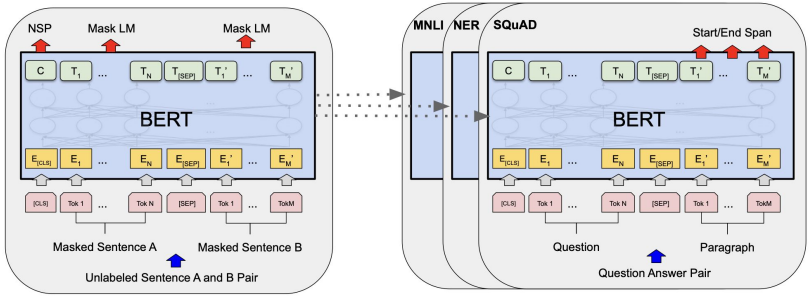
Data

Sumber Data		
thumbsUpCount	Jumlah suka (thumbs up) yang diterima oleh ulasan tersebut.	**Int64**
reviewCreatedVersion	Versi aplikasi yang digunakan pengguna saat membuat ulasan (tidak selalu tersedia).	**Object**
at	Tanggal dan waktu saat ulasan dibuat.	**Object**
replyContent	Isi balasan dari pengembang aplikasi terhadap ulasan (tidak ada data dalam kolom ini).	**Object**
repliedAt	Tanggal dan waktu saat balasan dari pengembang diberikan (tidak ada data dalam kolom ini).	**Object**
appVersion	Versi aplikasi yang digunakan pengguna saat memberikan ulasan (tidak selalu tersedia).	**Object**

Metode

Bidirectional Encoder Representations from Transformers (BERT)

BERT (Devlin dkk, 2018)



Pre-training Fine-Tuning

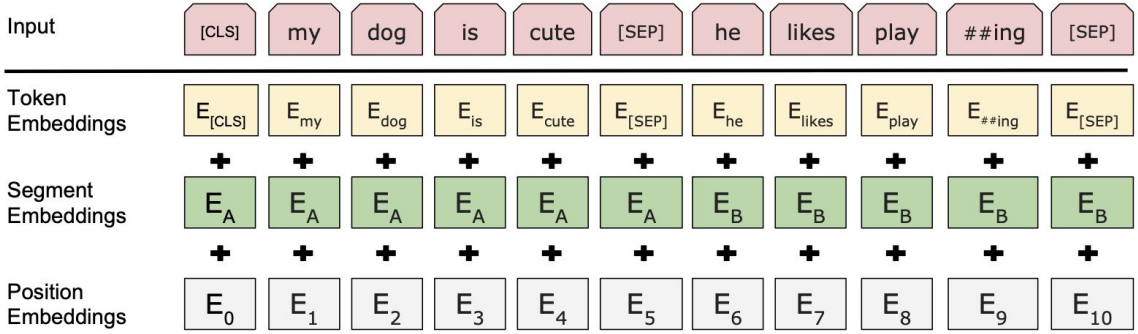
Model ***Bidirectional Encoder Representations from Transformers*** (BERT), yang diperkenalkan oleh peneliti google (Devlin dkk, 2018), adalah model representasi bahasa *pre-trained,* yang dilatih pada korpus teks raksasa (seperti wikipedia), dan digunakan untuk tugas NLP seperti analisis sentimen. Model ini unggul dari model pendahulunya, karena menggunakan sistem *bidirectional*, artinya model ini membaca teks secara menyeluruh dari kiri ke kanan dan dari kanan ke kiri.

Lapisan Utama Arsitektur BERT

Embeddings	Mengonversi urutan token menjadi array vektor bernilai riil yang merepresentasikan token-token tersebut.
Transformer Layers	Menangkap beragam hubungan antar token dalam representasi vektor
Task Head	Lapisan terakhir untuk melakukan tugas downstream spesifik seperti, klasifikasi, penjawaban pertanyaan, dll.

Representasi input BERT

Input dalam BERT



INPUT

Token Embedding

- Diambil dari *WordPiece vocabulary* (jumlahnya sekitar 30.000 token).
- Contoh: kalimat **"playing football"** bisa di-tokenisasi menjadi ["play", "##ing", "football"].

Segment Embedding

Digunakan untuk membedakan dua kalimat:

- Kalimat A → Segment A (semua token bertipe A)
- Kalimat B → Segment B (semua token bertipe B)

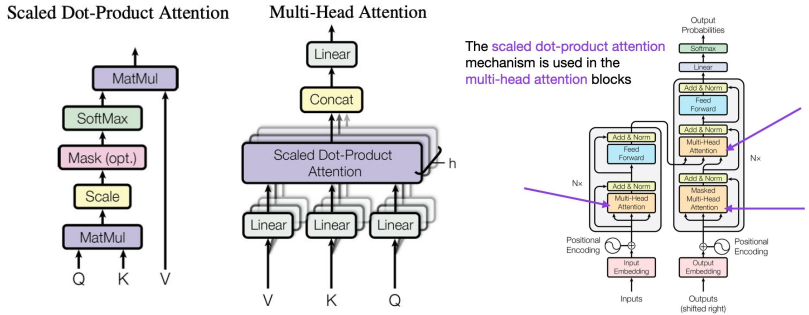
Token Embedding

- Karena BERT tidak menggunakan RNN/CNN, posisi kata dalam kalimat diberikan melalui positional encoding.
- Ini memungkinkan model untuk mengetahui urutan kata.

Metode

Transformer Layer dan Task Head BERT

Transformasi Layer



Attention head memungkinkan model untuk memfokuskan perhatian pada bagian tertentu dari input saat memproses token tertentu. Pada arsitektur BERT digunakan Multi-Head Attention, untuk menangkap beragam hubungan antar token dalam representasi vektor.

Komponen Layer Encoder

Multi-head self-attention

$$\text{Attention}(Q,K,V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \qquad \text{MultiHead}(Q,K,V) = \text{Concat}(\text{head}_1,...,\text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Add & LayerNorm

Feed Forward Neural Network (FFN)

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

BERT menggunakan fungsi aktivasi GELU

$$\text{FFN}(x) = \text{GELU}(xW_1 + b_1)W_2 + b_2 \qquad \text{GELU}(x) = 0.5x(1 + \tanh[\sqrt{2/\pi}(x + 0.044715x^3)])$$

Add & LayerNorm

Task Head

Fine Tuning

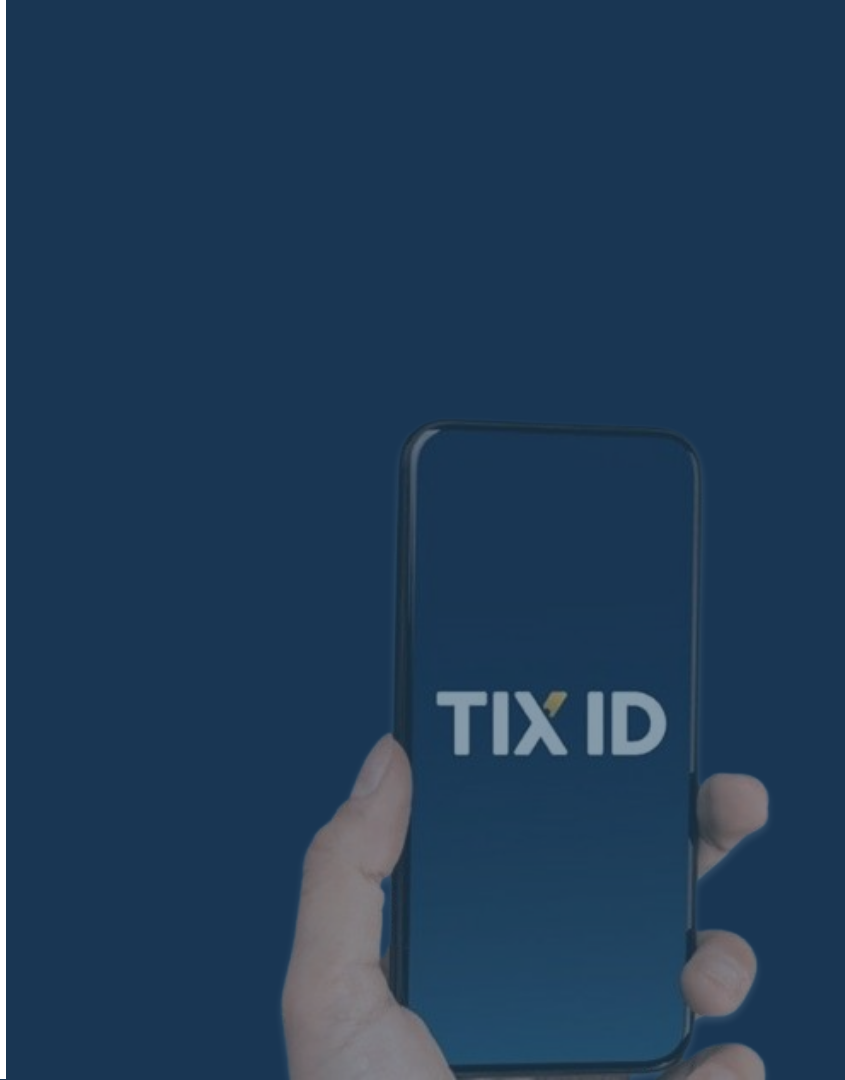
Fine tuning: Sequence Classification (e.g., Sentiment, Natural Language Inference), Token Classification (e.g., Named Entity Recognition), Question Answering (e.g., SQuAD)

Pre-training task

Pre-training task: Next Sentence Prediction (NSP), Masked Language Modeling (MLM)

03

Simulasi Analisis



Metode

BERTopic

Komponen Utama BERTopic (Grootendorst, 2022)

Document Embedding Layer

Mengubah dokumen teks menjadi representasi vektor yang kaya konteks semantik.

- BERTopic menggunakan model pre-trained berbasis Transformer seperti BERT atau Sentence-BERT (SBERT).
- Setiap dokumen (kalimat, paragraf, tweet, dll.) diubah menjadi vektor berdimensi tinggi (misalnya panjang 384 atau 768).
- Vektor ini disebut embedding, dan dokumen yang mirip secara makna akan memiliki embedding yang letaknya dekat satu sama lain di ruang vektor

Clustering Layers

Dimensi Reduksi (UMAP)

Mengecilkan ukuran vektor (embedding) supaya lebih mudah dikelompokkan (cluster).

- BERTopic memakai teknik UMAP, metode untuk mereduksi dimensi tanpa menghilangkan banyak makna semantik.
- Setelah direduksi, vektor jadi berdimensi rendah (misalnya 5–50 dimensi) namun tetap mewakili isi dokumen.

Clustering (HDBSCAN)

- Setelah vektor direduksi, dokumen bisa dikelompokkan menggunakan algoritma clustering bernama HDBSCAN.
- HDBSCAN tidak perlu diberi tahu jumlah klaster terlebih dahulu → cocok untuk topik yang jumlahnya tidak diketahui.
- Dokumen yang tidak cocok masuk ke klaster mana pun dianggap sebagai outlier atau “tidak punya topik”.

Topic Representation Layer (c-TF-IDF)

Menentukan kata kunci/topik utama dari setiap cluster dokumen.

- Semua dokumen dalam satu cluster dianggap satu dokumen besar. Lalu dihitung kata-kata apa yang paling khas dan membedakan topik tersebut dari topik lain.
- Teknik yang digunakan adalah modifikasi dari TF-IDF (Term Frequency-Inverse Document Frequency), yang sering dipakai dalam information retrieval. Hasil akhirnya adalah daftar kata-kata yang paling mewakili topik tersebut

Simulasi Analisis

Pre-Processing

Import Dataset dan EDA

```
#Pengunduhan dan Pembacaan Dataset
!gdown --id 1s4TFNtN4QcJs6YQiwO7WVgGO7KGqw2FG --output dataset.csv

import pandas as pd

df = pd.read_csv("dataset.csv")
df.info()

#Pengecekan nilai yang hilang dari setiap kolom pada dataset
df.isnull().sum().sort_values(ascending=False)

#visualisasi distribusi skor ulasan
import seaborn as sns
import matplotlib.pyplot as plt

sns.countplot(data=df, x='score')
plt.title('Distribusi Skor Ulasan')
plt.show()
```

1 Import Dataset

Langkah pertama adalah pengambilan dataset dari sumber eksternal (Google Drive) menggunakan perintah gdown, yang kemudian dibaca ke dalam program menggunakan pustaka pandas. Data disimpan dalam bentuk file CSV dan dimuat ke dalam DataFrame untuk memudahkan manipulasi dan analisis.

2 Eksplorasi Data Awal

Selanjutnya, dilakukan eksplorasi terhadap struktur data menggunakan **df.info()** dan **df.isnull().sum()**, yang bertujuan untuk mengetahui tipe data, jumlah entri, serta apakah terdapat nilai yang hilang pada kolom-kolom tertentu. Selain itu, dilakukan visualisasi distribusi skor ulasan dengan **seaborn.countplot**, untuk mengetahui proporsi skor ulasan.

Simulasi Analisis

Pre-Processing

Import dataset dan EDA

```
# Menampilkan 10 versi aplikasi yang paling banyak menerima ulasan
df['appVersion'].value_counts().head(10).plot(kind='barh')
plt.title('10 Versi Aplikasi dengan Jumlah Ulasan Terbanyak')
plt.xlabel('Jumlah Ulasan')
plt.show()

# Menghitung panjang masing-masing ulasan dan memvisualisasikannya dalam histogram
df['review_length'] = df['content'].astype(str).apply(len)
sns.histplot(df['review_length'], bins=50, kde=True)
plt.title('Distribusi Panjang Ulasan')
plt.xlabel('Jumlah Karakter')
plt.show()

# Menampilkan scatterplot untuk melihat hubungan antara skor dan jumlah like (thumbs up)
sns.scatterplot(data=df, x='score', y='thumbsUpCount')
plt.title('Hubungan antara Skor dan Jumlah Like')
plt.show()

# Membuat Word Cloud dari kata-kata yang paling sering muncul dalam ulasan
from collections import Counter
from wordcloud import WordCloud
import re

all_words = ' '.join(df['content'].dropna().astype(str).tolist())
all_words = re.sub(r'[^a-zA-Z\s]', '', all_words.lower())
word_freq = Counter(all_words.split())

wc = WordCloud(width=800, height=400).generate_from_frequencies(word_freq)
plt.figure(figsize=(10, 5))
plt.imshow(wc, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud dari Ulasan')
plt.show()
```

3 Analisis Fitur

Pada tahap ini, dilakukan analisis lebih lanjut terhadap kolom **appVersion** dan **content**. Versi aplikasi yang paling banyak diulas divisualisasikan dengan grafik batang horizontal. Selain itu, dihitung panjang setiap ulasan dalam satuan karakter dan ditampilkan distribusinya menggunakan histogram. Analisis ini berguna untuk memahami sejauh mana pengguna menjelaskan pengalaman mereka.

4 Visualisasi Word Cloud

Kata-kata dalam ulasan digabung menjadi satu teks panjang, kemudian dibersihkan dari karakter non-alfabet dan dikonversi menjadi huruf kecil. Dengan bantuan pustaka **WordCloud**, dibuat representasi visual kata-kata yang paling sering muncul. Ini memberikan gambaran cepat tentang kata kunci atau topik umum yang sering dibahas oleh pengguna.

Data Cleaning, Data Annotation, dan Text Cleaning

```python
import pandas as pd
from transformers import BertTokenizer
import torch
from torch.utils.data import Dataset, DataLoader
from tqdm import tqdm
import re
import string

# 1. Load data
# Data Cleaning: memilih hanya kolom yang diperlukan
df = df[['content', 'score']].copy()

# 2. Label mapping: 1-2 = negatif, 3 = netral, 4-5 = positif
# Data Annotation: mengubah skor menjadi label biner (non-positive vs positive)
def map_sentiment(score):
    if score <= 3:
        return 0  # non-positive
    else:
        return 1  # positive

df['label'] = df['score'].apply(map_sentiment)

# 3. Text preprocessing function
# Text Cleaning dan Text Normalization:
# - lowercase (normalization)
# - hapus URL, angka, tanda baca (cleaning)
# - hapus spasi berlebih (normalization)
def preprocess_text(text):
    text = text.lower()  # Text Normalization
    text = re.sub(r'http\S+|www\S+|https\S+', '', text)  # Text Cleaning
    text = re.sub(r'\d+', '', text)  # Text Cleaning
    text = text.translate(str.maketrans('', '', string.punctuation))  # Text Cleaning
    text = re.sub(r'\s+', ' ', text).strip()  # Text Normalization
    return text

# Apply text preprocessing
df['cleaned_content'] = df['content'].apply(preprocess_text)
```

5 Pemilihan Kolom dan Label Mapping

Data difokuskan hanya pada dua kolom utama: **content** dan **score**. Untuk kebutuhan klasifikasi biner, dilakukan pemetaan nilai skor menjadi dua kategori: skor 1 hingga 3 dianggap sebagai "non-positif" dan diberi label 0, sedangkan skor 4 hingga 5 dianggap sebagai "positif" dan diberi label 1. Proses ini penting dalam membangun model klasifikasi sentimen.

6 Text Cleaning dan Text Normalization

Teks ulasan dibersihkan dengan berbagai teknik: mengubah huruf menjadi kecil (lowercasing), menghapus URL, angka, tanda baca, dan spasi berlebih. Fungsi ini diterapkan ke setiap entri pada kolom **content**. Hasil dari pembersihan ini disimpan dalam kolom baru bernama **cleaned_content**.

Simulasi Analisis

Pre-Processing

Tokenizing (BERT Tokenizer)

```python
# 4. Tokenisasi Menggunakan BERT Tokenizer
# Menggunakan tokenizer dari model multilingual BERT
tokenizer = BertTokenizer.from_pretrained('bert-base-multilingual-cased')

# 5. Pembentukan Dataset dan DataLoader
# Membuat class dataset kustom yang akan mengatur input dan label dalam format tensor
class ReviewDataset(Dataset):
    def __init__(self, texts, labels, tokenizer, max_len=128):
        self.texts = texts
        self.labels = labels
        self.tokenizer = tokenizer
        self.max_len = max_len

    def __len__(self):
        return len(self.texts)

    def __getitem__(self, idx):
        text = str(self.texts[idx])
        label = self.labels[idx]

        # Proses tokenisasi dan pembentukan tensor input
        encoding = self.tokenizer.encode_plus(
            text,
            add_special_tokens=True,
            max_length=self.max_len,
            return_token_type_ids=False,
            padding='max_length',
            truncation=True,
            return_attention_mask=True,
            return_tensors='pt',
        )

        return {
            'input_ids': encoding['input_ids'].squeeze(0),
            'attention_mask': encoding['attention_mask'].squeeze(0),
            'label': torch.tensor(label, dtype=torch.long)
        }
```

7 **Tokenisasi Dengan BERT Tokenizer**

Untuk mempersiapkan data masuk ke dalam model BERT, teks perlu ditokenisasi. Digunakan **BertTokenizer** dari pustaka Transformers dengan model **bert-base-multilingual-cased**, yang mendukung berbagai bahasa termasuk Bahasa Indonesia. Tokenizer ini mengubah teks menjadi ID token yang dimengerti oleh model BERT.

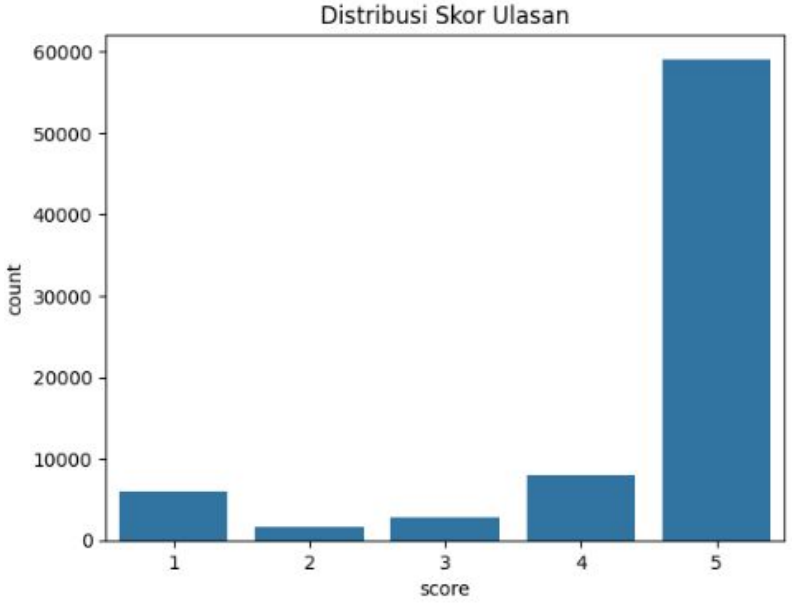
8 **Pembentukan Dataset dan DataLoader**

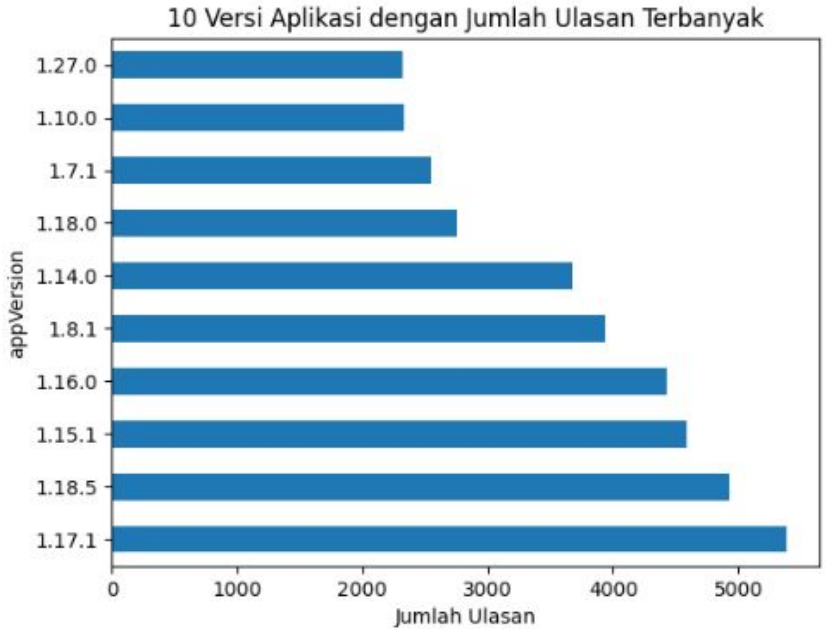
Dibuat kelas kustom **ReviewDataset** yang mewarisi **torch.utils.data.Dataset**. Kelas ini bertugas mengatur bagaimana teks dan label dibaca dan dikonversi menjadi tensor yang siap digunakan dalam pelatihan model. Selanjutnya, **DataLoader** digunakan untuk mengelompokkan data ke dalam batch dan memungkinkan pengacakan data selama pelatihan.

Simulasi Analisis

Pre-Processing

Hasil Visualisasi



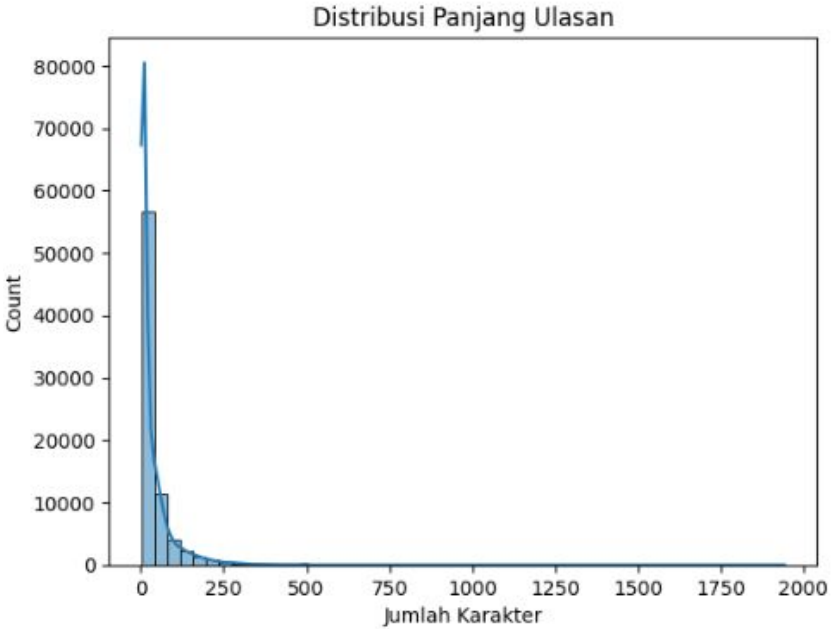


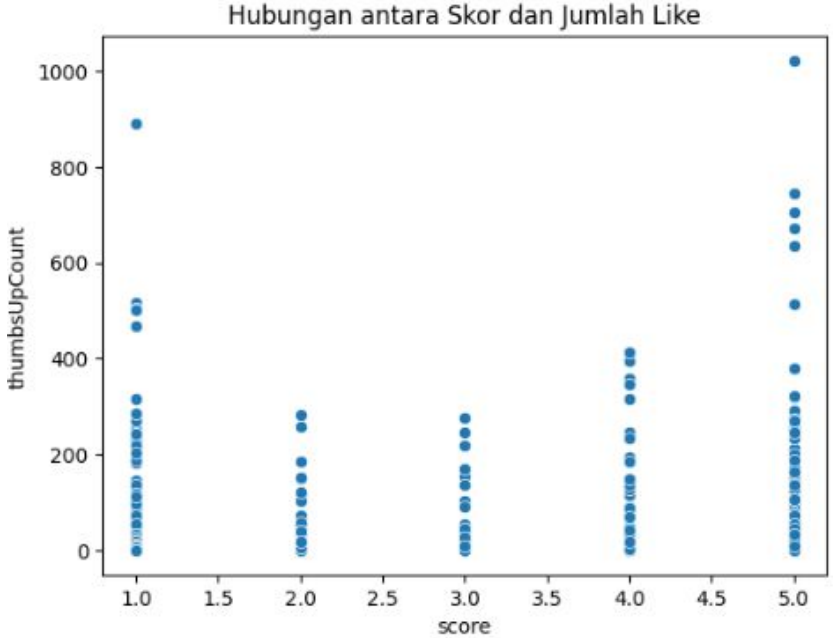
Skor 5 mendominasi dengan hampir 60.000 ulasan, menunjukkan mayoritas pengguna sangat puas terhadap aplikasi. Sebaliknya, skor 1 hingga 4 jauh lebih sedikit, dengan skor 2 paling minim. Versi 1.17.1 mendapat ulasan terbanyak, diikuti oleh 1.18.5 dan 1.15.1, yang mengindikasikan tingginya jumlah pengguna aktif atau adanya pembaruan signifikan yang mendorong lebih banyak ulasan.

Simulasi Analisis

Pre-Processing

Hasil Visualisasi





Dari visualisasi terlihat bahwa mayoritas ulasan memiliki panjang kurang dari 100 karakter, menunjukkan kecenderungan pengguna untuk menulis ulasan singkat. Selain itu, ulasan dengan skor 1 dan 5 cenderung memperoleh lebih banyak like dibandingkan skor lainnya. Hal ini mengindikasikan bahwa opini yang kuat, baik sangat positif maupun sangat negatif, lebih menarik perhatian pengguna lain dan lebih mungkin mendapatkan apresiasi dalam bentuk like.

Simulasi Analisis

Pre-Processing

Word Cloud

Dari word cloud terlihat bahwa kata "tiket", "good", "bisa", "aplikasi", "beli", dan "mudah" paling menonjol, mencerminkan fokus ulasan pada pembelian tiket dan pengalaman pengguna yang umumnya positif. Kata seperti "kode", "voucher", dan "antri" juga muncul, mengindikasikan pengalaman terkait promo atau kemudahan menghindari antrean. Secara keseluruhan, word cloud menunjukkan sentimen positif terhadap kemudahan dan manfaat penggunaan aplikasi.

Simulasi Analisis

Pre-Processing

Hasil Tokenisasi dan Dataset setelah Pre-Pocessing

```
Tokenizing data: 100%|██████████| 4843/4843 [00:26<00:00, 183.30it/s]
0  Ini aplikasi di Indonesia tetapi sinopsis film...
1                          Lengkap dan user friendly
2                                             Baguss
3  Aplikasi paling keren baru login mau pesan eh ...
4  Kepada yang terhormat PT elang jaya ini tolong...

                                    cleaned_content  label  score
0  ini aplikasi di indonesia tetapi sinopsis film...      0      3
1                          lengkap dan user friendly      1      5
2                                             baguss      1      5
3  aplikasi paling keren baru login mau pesan eh ...      1      5
4  kepada yang terhormat pt elang jaya ini tolong...      0      1
```

	content	cleaned_content	label	score
0	Ini aplikasi di Indonesia tetapi sinopsis film...	ini aplikasi di indonesia tetapi sinopsis film...	0	3
1	Lengkap dan user friendly	lengkap dan user friendly	1	5
2	Baguss	baguss	1	5
3	Aplikasi paling keren baru login mau pesan eh ...	aplikasi paling keren baru login mau pesan eh ...	1	5
4	Kepada yang terhormat PT elang jaya ini tolong...	kepada yang terhormat pt elang jaya ini tolong...	0	1

Sentiment Analysis

```
# HYPERPARAMETER SECTION
# ============================
LEARNING_RATE = 2e-5
BATCH_SIZE = 32
MAX_LEN = 128
EPOCHS = 3
PATIENCE = 1

# Dataset kamu
df = clean_data

# Tokenizer
tokenizer = BertTokenizer.from_pretrained('bert-base-multilingual-cased')

# Dataset class
class ReviewDataset(Dataset):
    def __init__(self, texts, labels, tokenizer, max_len=128):
        self.texts = texts
        self.labels = labels
        self.tokenizer = tokenizer
        self.max_len = max_len

    def __len__(self):
        return len(self.texts)

    def __getitem__(self, idx):
        text = str(self.texts[idx])
        label = self.labels[idx]
        encoding = self.tokenizer.encode_plus(
            text,
            add_special_tokens=True,
            max_length=self.max_len,
            return_token_type_ids=False,
            padding='max_length',
            truncation=True,
            return_attention_mask=True,
            return_tensors='pt',
        )
        return {
            'input_ids': encoding['input_ids'].squeeze(0),
            'attention_mask': encoding['attention_mask'].squeeze(0),
            'label': torch.tensor(label, dtype=torch.long)
        }
```

1 Import Library, Data dan Tokenizer

Langkah awal mencakup impor library penting seperti torch untuk deep learning dan transformers untuk model BERT. Dataset hasil pembersihan dimuat ke dalam DataFrame. Tokenizer bert-base-multilingual-cased dipanggil untuk mengubah teks menjadi token ID numerik sesuai dengan vocabulary BERT, yang akan menjadi input ke dalam model.

2 Dataset Class

Kelas ReviewDataset digunakan untuk menyiapkan data agar kompatibel dengan model. Setiap teks diubah menjadi tensor melalui proses tokenisasi, padding, dan attention masking. Label juga dikonversi menjadi tensor. Outputnya berupa dictionary tensor yang siap digunakan dalam proses pelatihan.

Sentiment Analysis

```python
# Split data
train_df, val_df = train_test_split(df, test_size=0.1)
train_dataset = ReviewDataset(train_df['cleaned_content'].tolist(), train_df['label'].tolist(), tokenizer, max_len=MAX_LEN)
val_dataset = ReviewDataset(val_df['cleaned_content'].tolist(), val_df['label'].tolist(), tokenizer, max_len=MAX_LEN)

train_dataloader = DataLoader(train_dataset, batch_size=BATCH_SIZE, shuffle=True)
val_dataloader = DataLoader(val_dataset, batch_size=BATCH_SIZE)

# Model
model = BertForSequenceClassification.from_pretrained('bert-base-multilingual-cased', num_labels=2)

# Optimizer dan loss
optimizer = AdamW(model.parameters(), lr=LEARNING_RATE)
criterion = nn.CrossEntropyLoss()
scaler = GradScaler()
device = torch.device('cuda') if torch.cuda.is_available() else torch.device('cpu')
model.to(device)

# Training function
def train_model(model, dataloader, optimizer, criterion, device, accumulation_steps=4):
    model.train()
    total_loss, total_correct, total_samples = 0, 0, 0
    optimizer.zero_grad()
    for step, batch in enumerate(tqdm(dataloader, desc="Training", ncols=100)):
        input_ids = batch['input_ids'].to(device)
        attention_mask = batch['attention_mask'].to(device)
        labels = batch['label'].to(device)
        with autocast():
            outputs = model(input_ids, attention_mask=attention_mask, labels=labels)
            loss = outputs.loss
            logits = outputs.logits
        scaler.scale(loss).backward()
        if (step + 1) % accumulation_steps == 0 or (step + 1) == len(dataloader):
            scaler.step(optimizer)
            scaler.update()
            optimizer.zero_grad()
        total_loss += loss.item()
        _, preds = torch.max(logits, dim=1)
        total_correct += (preds == labels).sum().item()
        total_samples += labels.size(0)
    return total_loss / len(dataloader), total_correct / total_samples
```

3 Split Data dan DataLoader

Data dibagi menjadi latih dan validasi, masing-masing dimasukkan ke ReviewDataset lalu ke DataLoader. Proses ini memungkinkan batching dan shuffling data agar pelatihan berjalan lebih efisien dan hasil model menjadi lebih general.

4 Inisialisasi Model dan Komponen Pelatihan

Model yang digunakan adalah BertForSequenceClassification, yaitu model BERT yang dilengkapi layer klasifikasi di bagian akhir dan telah di-pretrained. Model ini disiapkan untuk klasifikasi dua label: positif dan non-positif. Optimizer AdamW dipilih karena lebih sesuai untuk fine-tuning model transformer, dan loss function yang digunakan adalah CrossEntropyLoss yang cocok untuk tugas klasifikasi dua kelas. GradScaler diaktifkan untuk memfasilitasi mixed precision training, sehingga proses training berjalan lebih cepat dan hemat memori. Seluruh komponen ini kemudian dipindahkan ke GPU jika tersedia untuk mempercepat komputasi.

Sentiment Analysis

```python
# Evaluation function
def evaluate_model(model, dataloader, criterion, device):
    model.eval()
    total_loss, total_correct, total_samples = 0, 0, 0
    with torch.no_grad():
        for batch in tqdm(dataloader, desc="Evaluating", ncols=100):
            input_ids = batch['input_ids'].to(device)
            attention_mask = batch['attention_mask'].to(device)
            labels = batch['label'].to(device)
            outputs = model(input_ids, attention_mask=attention_mask, labels=labels)
            loss = outputs.loss
            logits = outputs.logits
            total_loss += loss.item()
            _, preds = torch.max(logits, dim=1)
            total_correct += (preds == labels).sum().item()
            total_samples += labels.size(0)
    return total_loss / len(dataloader), total_correct / total_samples

# Training loop dengan early stopping
best_val_loss = float('inf')
epochs_no_improve = 0

for epoch in range(EPOCHS):
    print(f"\nEpoch {epoch+1}/{EPOCHS}")
    train_loss, train_acc = train_model(model, train_dataloader, optimizer, criterion, device)
    val_loss, val_acc = evaluate_model(model, val_dataloader, criterion, device)
    print(f"Train Loss: {train_loss:.4f}, Train Acc: {train_acc:.4f}")
    print(f"Val Loss: {val_loss:.4f}, Val Acc: {val_acc:.4f}")

    # Early stopping check
    if val_loss < best_val_loss:
        best_val_loss = val_loss
        epochs_no_improve = 0
        # Kamu bisa simpan model terbaik di sini, contoh:
        # torch.save(model.state_dict(), 'best_model.pt')
    else:
        epochs_no_improve += 1
        print(f"EarlyStopping counter: {epochs_no_improve}/{PATIENCE}")
        if epochs_no_improve >= PATIENCE:
            print("Early stopping triggered!")
            break
```

5 Fungsi Pelatihan dan Evaluasi

Fungsi train_model() digunakan untuk melatih model dalam satu epoch, dengan setiap batch diproses menggunakan autocast() untuk efisiensi GPU melalui presisi campuran. Gradien dikumpulkan sebelum dilakukan update oleh optimizer (gradient accumulation). Sementara itu, evaluate_model() mengevaluasi performa model pada data validasi tanpa pembaruan parameter, menghitung rata-rata loss dan akurasi, serta dijalankan tanpa menyimpan gradien agar lebih ringan. Evaluasi ini bertujuan untuk menilai kemampuan generalisasi model terhadap data yang belum pernah dilatih.

6 Training Loop dan Early Stopping

Model dilatih dalam beberapa epoch. Setelah setiap epoch, performa pada data validasi diuji. Jika tidak ada peningkatan dalam beberapa epoch, pelatihan dihentikan lebih awal untuk mencegah overfitting.

Sentiment Analysis

```python
# Tuning loop
# ======================================
for lr, batch_size, max_len, epochs in param_combinations:
    print(f"\n=== LR: {lr}, Batch: {batch_size}, MaxLen: {max_len}, Epochs: {epochs} ===")

    # Dataset ulang sesuai max_len
    train_dataset = ReviewDataset(train_df['cleaned_content'].tolist(), train_df['label'].tolist(), tokenizer, max_len=max_len)
    val_dataset = ReviewDataset(val_df['cleaned_content'].tolist(), val_df['label'].tolist(), tokenizer, max_len=max_len)
    train_dataloader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
    val_dataloader = DataLoader(val_dataset, batch_size=batch_size)

    # Model baru
    model = BertForSequenceClassification.from_pretrained('bert-base-multilingual-cased', num_labels=2)
    optimizer = AdamW(model.parameters(), lr=lr)
    criterion = nn.CrossEntropyLoss()
    scaler = GradScaler()
    model.to(device)

    # Early stopping reset
    best_epoch_val_loss = float('inf')
    epochs_no_improve = 0

    for epoch in range(epochs):
        train_loss, train_acc = train_model(model, train_dataloader, optimizer, criterion, device)
        val_loss, val_acc = evaluate_model(model, val_dataloader, criterion, device)
        print(f"Epoch {epoch+1}/{epochs} - Train Acc: {train_acc:.4f}, Val Acc: {val_acc:.4f}")

        if val_loss < best_epoch_val_loss:
            best_epoch_val_loss = val_loss
            epochs_no_improve = 0
            # Simpan model jika ini model terbaik
            if val_acc > best_val_acc:
                best_val_acc = val_acc
                best_params = (lr, batch_size, max_len, epochs)
                torch.save(model.state_dict(), best_model_path)
        else:
            epochs_no_improve += 1
            if epochs_no_improve >= PATIENCE:
                print("Early stopping triggered!")
                break

    results.append({
        'learning_rate': lr,
        'batch_size': batch_size,
        'max_len': max_len,
        'epochs': epochs,
        'val_accuracy': val_acc
    })
```

7 Tuning Hyperparameter

Untuk mendapatkan performa optimal, dilakukan tuning terhadap beberapa kombinasi hyperparameter seperti learning_rate, max_len, dan batch_size. Grid search dilakukan dengan mencoba semua kombinasi yang memungkinkan. Untuk setiap konfigurasi, model dilatih dan diuji, lalu akurasi validasi dicatat. Model dengan akurasi validasi tertinggi disimpan, dan konfigurasinya dianggap sebagai parameter terbaik. Proses ini penting karena model transformer seperti BERT sangat sensitif terhadap pengaturan parameter, sehingga tuning berperan besar dalam mencapai kinerja maksimal.

8 Evaluasi Model Terbaik

Model terbaik dimuat ulang dan diuji kembali. Confusion matrix dan classification report digunakan untuk mengevaluasi performa model terhadap masing-masing kelas, terutama dalam hal precision dan recall.

Sentiment Analysis

```python
# Load best model
best_model = BertForSequenceClassification.from_pretrained('bert-base-multilingual-cased', num_labels=2)
best_model.load_state_dict(torch.load(best_model_path))
best_model.to(device)

# Buat ulang val dataloader dengan best max_len
_, _, best_max_len, _ = best_params
val_dataset = ReviewDataset(val_df['cleaned_content'].tolist(), val_df['label'].tolist(), tokenizer, max_len=best_max_len)
val_dataloader = DataLoader(val_dataset, batch_size=32)

# Evaluasi akhir dengan tqdm progress bar
best_model.eval()
all_labels, all_preds = [], []
with torch.no_grad():
    for batch in tqdm(val_dataloader, desc="Evaluating"):
        input_ids = batch['input_ids'].to(device)
        attention_mask = batch['attention_mask'].to(device)
        labels = batch['label'].to(device)
        outputs = best_model(input_ids, attention_mask=attention_mask)
        logits = outputs.logits
        _, preds = torch.max(logits, dim=1)
        all_labels.extend(labels.cpu().numpy())
        all_preds.extend(preds.cpu().numpy())

# Confusion matrix
cm = confusion_matrix(all_labels, all_preds)
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Non-positive', 'Positive'], yticklabels=['Non-positive', 'Positive'])
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()

# Classification report
report = classification_report(all_labels, all_preds, target_names=['Non-positive', 'Positive'])
print("\nClassification Report:\n", report)

# Insight: Distribusi Prediksi
val_df = val_df.reset_index(drop=True)
val_df['predicted'] = all_preds
sentiment_counts = val_df['predicted'].value_counts()
plt.figure(figsize=(6, 4))
sns.barplot(x=['Non-positive', 'Positive'], y=sentiment_counts.sort_index(), palette='viridis')
plt.title('Sentiment Distribution (Predicted)')
plt.ylabel('Count')
plt.show()

print(f"\nBest model saved to: {best_model_path}")
```

9 Visualisasi Distribusi Prediksi

Untuk mengetahui bagaimana distribusi prediksi model terhadap kedua kelas, dilakukan visualisasi hasil prediksi dalam bentuk grafik batang. Setiap bar menggambarkan jumlah prediksi untuk kelas positif dan non-positif. Visualisasi ini berguna untuk mendeteksi bias model, misalnya jika model terlalu banyak memprediksi positif karena dominasi data di kelas tersebut. Hasil grafik ini juga bisa digunakan sebagai insight tambahan dalam pelaporan analisis sentimen.

10 Penyimpanan Model dan Tokenizer

Setelah pelatihan selesai, model dan tokenizer disimpan dalam format transformers menggunakan fungsi save_pretrained(). Proses ini menyimpan konfigurasi model, bobot (weights), serta tokenizer ke dalam folder terstruktur. Dengan demikian, model dapat dimuat ulang untuk keperluan prediksi atau integrasi ke aplikasi lain tanpa perlu pelatihan ulang, sangat ideal untuk deployment ke sistem real-time.

Sentiment Analysis

```python
def predict_sentiment(text, model, tokenizer, device, max_len=128):
    model.eval()

    # Tokenisasi teks
    encoding = tokenizer.encode_plus(
        text,
        add_special_tokens=True,
        max_length=max_len,
        return_token_type_ids=False,
        padding='max_length',
        truncation=True,
        return_attention_mask=True,
        return_tensors='pt',
    )

    input_ids = encoding['input_ids'].to(device)
    attention_mask = encoding['attention_mask'].to(device)

    with torch.no_grad():
        outputs = model(input_ids=input_ids, attention_mask=attention_mask)
        logits = outputs.logits
        prediction = torch.argmax(logits, dim=1).item()

    label_map = {0: "Non-Positive", 1: "Positive"}
    return label_map[prediction]
```

11 **Fungsi Prediksi untuk Teks Baru**

Untuk memungkinkan penggunaan model dalam konteks dunia nyata, dibuat fungsi predict_sentiment() yang menerima input berupa satu kalimat ulasan. Teks di-tokenisasi dan dikonversi menjadi tensor, lalu dimasukkan ke model untuk menghasilkan prediksi. Hasil berupa logits dikonversi menjadi kelas (0 atau 1) dengan argmax, lalu dipetakan ke label sentimen (“Non-Positive” atau “Positive”). Fungsi ini dapat digunakan untuk mengintegrasikan model ke dalam chatbot, dashboard monitoring, atau sistem klasifikasi otomatis lainnya.

```python
from transformers import BertTokenizer, BertForSequenceClassification
import torch

# Load tokenizer & model
tokenizer = BertTokenizer.from_pretrained("bert-base-multilingual-cased")
model = BertForSequenceClassification.from_pretrained("sentiment_model")
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model.to(device)

# Prediksi kalimat baru
input_text = "kenapa loadingnya lama?"
result = predict_sentiment(input_text, model, tokenizer, device)
print("Prediksi Sentimen:", result)
```

12 **Contoh Inferensi Kalimat**

Sebuah kalimat seperti “kenapa loadingnya lama?” dimasukkan untuk pengujian. Model berhasil mengklasifikasikannya sebagai non-positif, menunjukkan kemampuannya dalam mengenali konteks keluhan dari teks pendek.

Simulasi Analisis

Modelling

Topic Level Sentiment

```
import pandas as pd
import torch
from transformers import BertTokenizer, BertForSequenceClassification
from sentence_transformers import SentenceTransformer
from bertopic import BERTopic
import umap
import hdbscan
from sklearn.feature_extraction.text import CountVectorizer
```

1 Import Library dan Model Pendukung

Tahap awal mencakup impor pustaka yang mendukung proses pemodelan topik berbasis transformer. Selain pandas dan torch, digunakan sentence-transformers untuk menghasilkan representasi vektor teks (embedding), serta BERTopic sebagai model utama. Untuk proses clustering dan reduksi dimensi, digunakan HDBSCAN dan UMAP, sedangkan CountVectorizer dari sklearn digunakan untuk ekstraksi fitur kata berbasis n-gram. Semua komponen ini disiapkan agar dapat diintegrasikan dalam pipeline BERTopic.

```
umap_model = umap.UMAP(n_neighbors=15, n_components=5, min_dist=0.0, metric='cosine')
hdbscan_model = hdbscan.HDBSCAN(min_cluster_size=15, metric='euclidean', cluster_selection_method='eom')
vectorizer = CountVectorizer(ngram_range=(1, 2), stop_words='english')
embedding_model = SentenceTransformer('paraphrase-multilingual-MiniLM-L12-v2')  # Multilingual (termasuk Bahasa Indonesia)
```

2 Inisialisasi

UMAP digunakan untuk mereduksi embedding dokumen menjadi 5 dimensi dengan metrik cosine, agar struktur semantik lokal tetap terjaga. HDBSCAN berperan sebagai algoritma clustering tanpa perlu menentukan jumlah klaster, sehingga cocok untuk data ulasan yang bervariasi. CountVectorizer digunakan untuk mengekstrak unigram dan bigram dengan menghapus stopwords bahasa Inggris, meskipun embedding-nya bersifat multibahasa.

Simulasi Analisis

Modelling

Topic Level Sentiment

```
import numpy as np
# Ambil dokumen
documents = clean_data['cleaned_content'].astype(str).tolist()

# Buat embeddings dengan progress bar
print("Generating embeddings...")
embeddings = []
for doc in tqdm(documents, desc="Embedding documents"):
    emb = embedding_model.encode(doc)
    embeddings.append(emb)

embeddings = np.array(embeddings)
print("\nTraining BERTopic model...")

topic_model = BERTopic(
    embedding_model=None,  # Karena kita sudah pakai manual embeddings
    umap_model=umap_model,
    hdbscan_model=hdbscan_model,
    vectorizer_model=vectorizer,
    nr_topics=10,
    verbose=True
)

# Fit BERTopic dengan dokumen dan embeddings yang sudah dihitung
new_topics, new_probs = topic_model.fit_transform(documents, embeddings)

# Tambahkan topik ke dataframe
clean_data['10_topics'] = new_topics
```

3 **Pembuatan Embedding dan Pelatihan BERTopic**

Setiap dokumen ulasan yang telah dibersihkan dari kolom cleaned_content dikonversi menjadi vektor embedding menggunakan model paraphrase-multilingual-MiniLM-L12-v2, yang mendukung Bahasa Indonesia. Embedding dihitung satu per satu dengan progress bar untuk memudahkan pemantauan. Setelah semua vektor dokumen tersedia, dilakukan pelatihan model BERTopic dengan menyuplai embedding secara langsung, sambil menonaktifkan embedding bawaan. Model dilatih untuk menghasilkan 10 topik menggunakan kombinasi UMAP, HDBSCAN, dan vectorizer yang telah dikonfigurasi sebelumnya.

Topic Level Sentiment

```
topic_info = topic_model.get_topic_info()
print("\nTop topics:")
print(topic_info)

# Visualisasi (ditampilkan di notebook)
topic_model.visualize_topics(custom_labels=True)
```

4 **Hasil Pelatihan dan Visualisasi Topik**

Setelah pelatihan selesai, setiap dokumen dikaitkan dengan salah satu dari 10 topik yang dihasilkan, dan label topik ini ditambahkan ke dataframe clean_data. Informasi tentang topik-topik yang ditemukan ditampilkan melalui get_topic_info(), yang memberikan ringkasan topik beserta jumlah dokumen yang terkait. Kemudian, dilakukan visualisasi relasi antar topik menggunakan visualize_topics()

```
# Hitung jumlah sentimen per label topik
sentiment_distribution = clean_data.groupby(['topic_label', 'sentiment']).size().reset_index(name='count')

# Hitung total per topik
total_per_topic = sentiment_distribution.groupby('topic_label')['count'].transform('sum')

# Tambahkan kolom persentase
sentiment_distribution['percentage'] = (sentiment_distribution['count'] / total_per_topic) * 100
```

5 **Distribusi Sentimen per Topik**

Dihiitung jumlah ulasan berdasarkan kombinasi topik dan label sentimen, lalu menghitung persentasenya dalam tiap topik. Data dikelompokkan berdasarkan topic_label dan sentiment, kemudian dihitung jumlahnya (count). Selanjutnya, total ulasan per topik dihitung dan digunakan untuk menambahkan kolom percentage, yang menunjukkan proporsi tiap sentimen dalam satu topik. Hasil akhir memberikan gambaran distribusi sentimen di setiap topik yang terbentuk dari proses topic modeling.

Topic Level Sentiment

```
# Set gaya visual
sns.set(style="whitegrid")

# Tetapkan urutan topik agar sesuai label yang kamu buat
sentiment_distribution['topic_label'] = pd.Categorical(
    sentiment_distribution['topic_label'],
    categories=[topic_labels[t] for t in sorted(topic_labels.keys())],
    ordered=True
)

# Buat palet warna custom
custom_palette = {
    'Positive': '#2ecc71',       # Hijau terang
    'Non-positive': '#e74c3c'    # Merah terang
}

# Buat plot
plt.figure(figsize=(10, 6))
barplot = sns.barplot(
    data=sentiment_distribution,
    y='topic_label',
    x='percentage',
    hue='sentiment',
    palette=custom_palette
)

# Tambahkan label persentase di atas bar
for p in barplot.patches:
    width = p.get_width()
    if width > 1:
        plt.text(width + 0.5, p.get_y() + p.get_height() / 2,
                 f'{width:.1f}%', va='center')

# Finalisasi tampilan
plt.title('Distribusi Sentimen per Topik (%)', fontsize=14, weight='bold')
plt.xlabel('Persentase (%)')
plt.ylabel('Topik')
plt.legend(title='Sentimen', loc='lower right')
plt.tight_layout()
plt.show()
```

6 **Visualisasi Distribusi Sentimen per Topik**

Terakhir, dilakukan visualisasi berbentuk barplot horizontal yang menunjukkan proporsi sentimen positif dan non-positif untuk setiap topik. Topik diurutkan sesuai label yang telah ditentukan, dan digunakan palet warna khusus untuk membedakan jenis sentimen (hijau untuk positif, merah untuk non-positif). Setiap batang bar diberi label persentase untuk memperjelas perbandingan antar sentimen di tiap topik. Visualisasi ini membantu memahami topik mana yang didominasi sentimen positif atau sebaliknya, serta memudahkan analisis tematik dalam konteks sentimen pengguna.

Simulasi Analisis

Output

Sentiment Analysis				
Learning Rate	**Batch Size**	**Max Len**	**Epochs**	**Accuracy**
0.00003	32	256	3	0.941412
0.00002	32	256	3	0.941154
0.00002	32	256	3	0.940379
0.00003	32	256	3	0.939089

Tabel ini menampilkan hasil eksperimen pelatihan model dengan parameter tetap untuk batch size (32), max length (256), dan epochs (3), namun dengan variasi learning rate. Hasil menunjukkan bahwa perubahan learning rate berdampak langsung terhadap akurasi model. Nilai learning rate 0.00003 menghasilkan akurasi tertinggi yaitu 0.941412, mengungguli nilai 0.00002 yang menghasilkan akurasi antara 0.939089 hingga 0.941154. Meskipun perbedaannya relatif kecil, hal ini menunjukkan bahwa pemilihan learning rate yang tepat sangat penting untuk mengoptimalkan performa model. Nilai 0.00003 tampaknya merupakan titik optimal pada konfigurasi ini.

Simulasi Analisis

Output

Sentiment Analysis

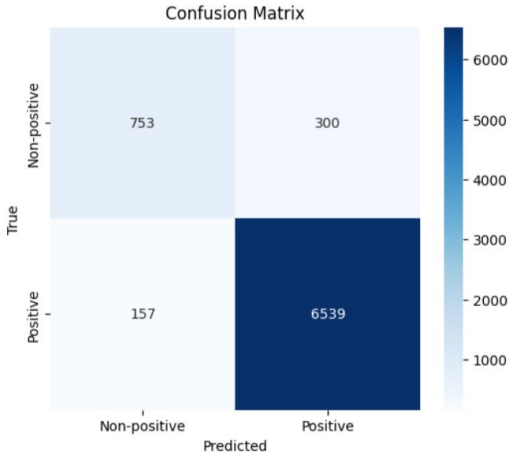
Contoh Output

Contoh prediksi klasifikasi sentimen, model mengklasifikasikan kalimat “kenapa loadingnya lama?” ke dalam kelas ***negative* (*non-positive*)**

```
Classification Report:
              precision    recall  f1-score   support

Non-positive       0.83      0.72      0.77      1053
    Positive       0.96      0.98      0.97      6696

    accuracy                           0.94      7749
   macro avg       0.89      0.85      0.87      7749
weighted avg       0.94      0.94      0.94      7749
```



Metrik Evaluasi

Model yang telah dilatih mencapai akurasi keseluruhan sebesar 94%, dengan kinerja yang sangat baik pada kelas positive, ditunjukkan oleh precision sebesar 96% dan recall mencapai 98%. Sementara itu, performa model pada kelas non-positive masih tergolong lebih rendah, dengan precision sebesar 83% dan recall hanya 72%, yang menunjukkan bahwa sebagian ulasan non-positive masih sering salah diklasifikasikan sebagai positive. Ketimpangan distribusi data—di mana jumlah data positive (6696) jauh lebih besar dibanding non-positive (1053)—kemungkinan berkontribusi terhadap bias model yang cenderung memprioritaskan kelas positive. Hal ini menyebabkan ketidakseimbangan performa antar kelas dan mengurangi kemampuan model dalam mendeteksi sentimen negatif secara konsisten.

Simulasi Analisis

Output

Sentiment Analysis

Intepretasi WordCloud Sentimen Non-Positive

WordCloud sentimen **non-positif** menunjukkan bahwa pengguna banyak mengeluhkan masalah teknis seperti aplikasi yang *error*, transaksi *gagal*, serta kesulitan dalam proses *refund* dan penggunaan *saldo dana*. Kata-kata seperti “tidak bisa”, “kenapa”, dan “tolong” mencerminkan frustrasi dan permintaan bantuan.

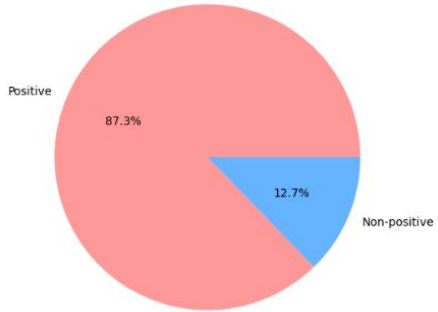
Intepretasi WordCloud Sentimen Positive

WordCloud sentimen **positif** didominasi oleh ungkapan kepuasan seperti “sangat membantu”, “mudah”, dan “mantap”. Pengguna mengapresiasi kemudahan beli tiket, fitur tanpa antre, serta promo dan kode diskon yang memudahkan dan memberi nilai tambah. Ini menunjukkan perbedaan mencolok antara keluhan teknis dan pujian terhadap fitur praktis.

Simulasi Analisis

Output

Sentiment Analysis		
Aspek	**Non-Positive**	**Positive**
Nada	Keluhan, frustrasi, dan kekecewaan	Apresiasi, bantuan, dan kepuasan
Fokus Utama	Masalah teknis, refund, kegagalan penggunaan	Kemudahan akses, efisiensi, pengalaman menyenangkan
Kata Kunci	"tidak bisa", "gagal", "refund", "kenapa", "tolong", "susah"	"sangat membantu", "mantap", "tanpa antre", "bagus", "oke"
Fitur yang Dibahas	Dana, refund, login, server, kendala input kode	Pembelian tiket, input kode, promo, fitur antre, aplikasi



Proporsi Sentimen

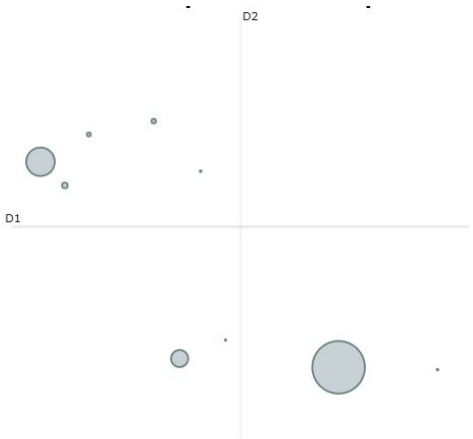
Diagram menunjukkan bahwa **mayoritas besar sentimen dalam data yang dianalisis adalah positif**, dengan proporsi sebesar **87,3%**, sedangkan **hanya 12,7%** yang tergolong sentimen **non-positif**. Hal ini mengindikasikan bahwa, mayoritas pengguna memiliki persepsi atau pengalaman yang baik terhadap layanan atau produk yang dikaji.

Simulasi Analisis

Output

Topic Modelling

Intertopic Distance Map



Peta Intertropic Distance Map menunjukkan bahwa 10 topik tersebar merata, mencerminkan perbedaan tematik yang jelas. Topik besar seperti “Feedback Positif dan Kepuasan” dan “Pembelian Tiket & Promo” dominan, sedangkan topik kecil seperti “Film & Bioskop di Indonesia” memiliki cakupan terbatas.

Topic Labels

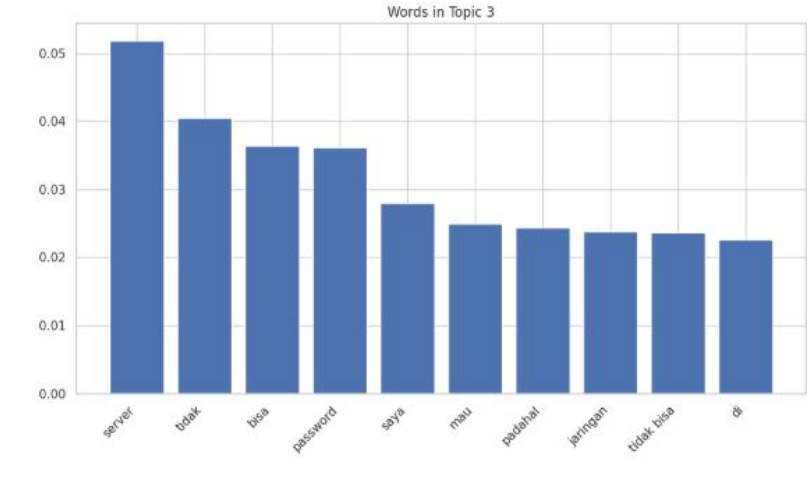
1	Feedback Positif dan Kepuasan
2	Pembelian Tiket & Promo
3	Aplikasi TIX dan Pengalaman Pengguna
4	Masalah Teknis & Akses Login
5	Efisiensi Waktu & Penjadwalan
6	Film & Bioskop di Indonesia
7	Ucapan Harapan & Doa Positif
8	Update Aplikasi yang Terlalu Sering
9	Kekecewaan pada Promo (PHP = harapan palsu)

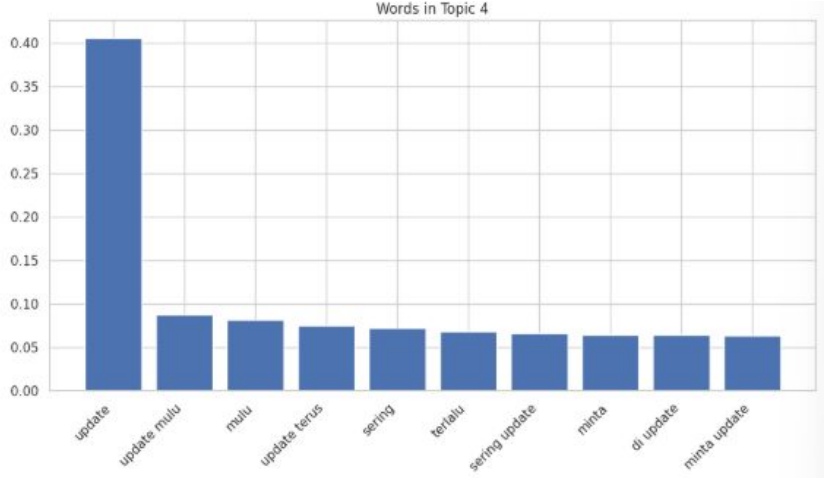
Pelabelan topik dilakukan berdasarkan dua komponen utama: kata kunci topik (top words) yang diperoleh dari **topic_model.get_topic(topic_id)[:3]**, serta representative documents (ulasan-ulasan dengan skor tertinggi dan terendah) yang diambil dengan **nlargest(3, 'score')** dan **nsmallest(3, 'score')** dari subset **topic_docs.** Kode ini menampilkan tiga kata teratas yang memiliki bobot tertinggi untuk masing-masing topik, yang secara statistik menggambarkan kata-kata paling representatif dalam klaster tersebut. Kemudian, untuk menambah konteks, kode menampilkan contoh ulasan dari topik tersebut yang memiliki skor tertinggi (positif) dan terendah (negatif), sehingga memungkinkan pengguna untuk memahami nuansa sentimen dan isi dominan dari topik tersebut.

Simulasi Analisis

Output

Top Words by Topic

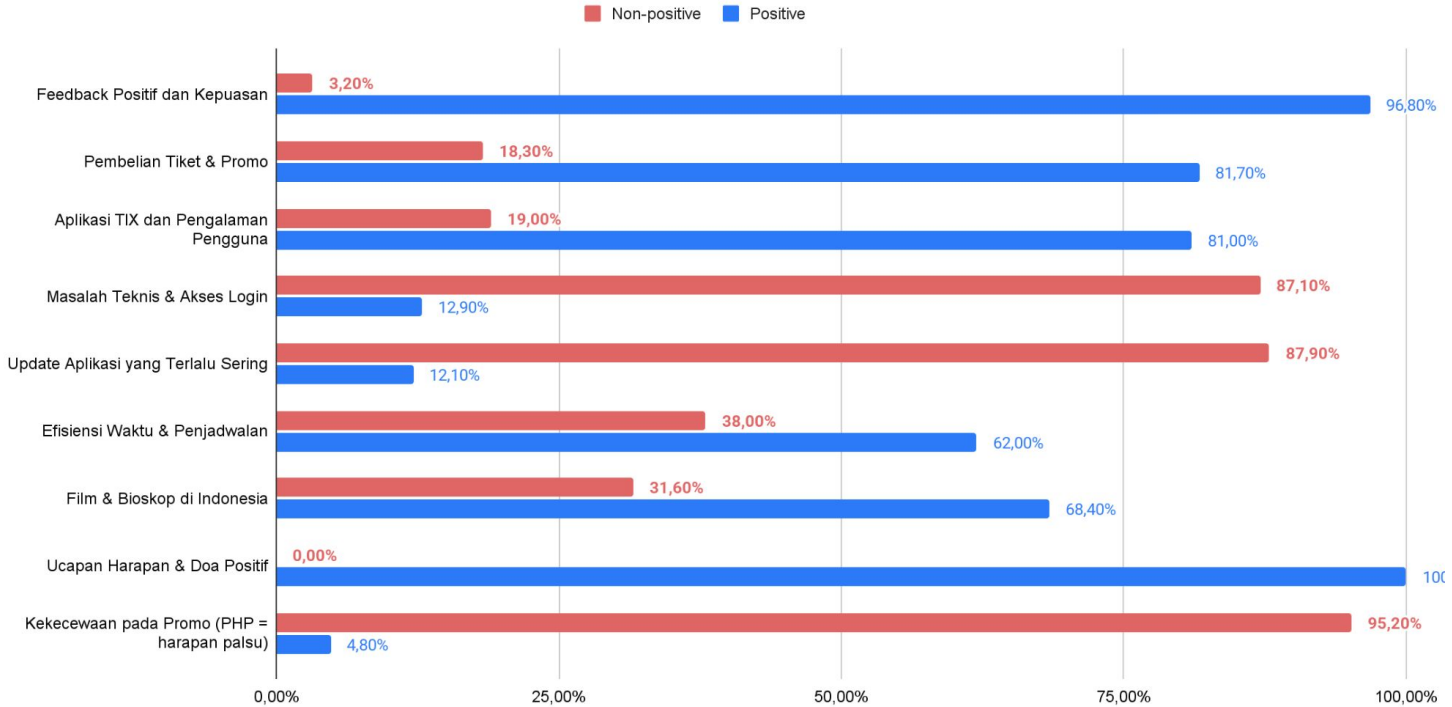




Berdasarkan hasil pemodelan topik, ulasan pengguna terhadap aplikasi menunjukkan variasi tema yang cukup beragam. Banyak pengguna memberikan tanggapan positif terkait kualitas aplikasi secara umum, menyebutkan bahwa aplikasi ini membantu, bagus, dan mudah digunakan. Di sisi lain, terdapat juga apresiasi terhadap manfaat praktis seperti kemudahan mengatur waktu dan jadwal, adanya promo tiket bioskop, serta dukungan terhadap film lokal di Indonesia. Namun demikian, tiga topik utama muncul sebagai sorotan kritis yang menjadi titik fokus dalam analisis ini. **Topik 3 ('Masalah Teknik & Akses Login')** mencerminkan keluhan mengenai kendala teknis dalam mengakses aplikasi, termasuk masalah server, jaringan, dan kegagalan login meskipun kata sandi sudah benar. Sementara itu, **Topik 4 ('Update Aplikasi yang Terlalu Sering')** menunjukkan keberatan pengguna terhadap frekuensi pembaruan aplikasi yang dianggap berlebihan, yang menimbulkan kesan bahwa aplikasi belum stabil. Selain itu, **Topik 8 ('Isu Promo PHP')** mengindikasikan ketidakpuasan pengguna terhadap sistem promo dalam aplikasi yang berkaitan dengan istilah "PHP" — yang dalam konteks lokal sering diartikan sebagai harapan palsu — sehingga muncul persepsi bahwa promo yang ditawarkan tidak sesuai dengan ekspektasi atau tidak dapat digunakan sebagaimana mestinya. Ketiga isu ini menandakan adanya hambatan signifikan, baik secara teknis maupun dalam hal kepercayaan pengguna, yang perlu segera ditangani oleh pengembang guna meningkatkan pengalaman dan kepuasan pengguna secara menyeluruh.

Simulasi Analisis

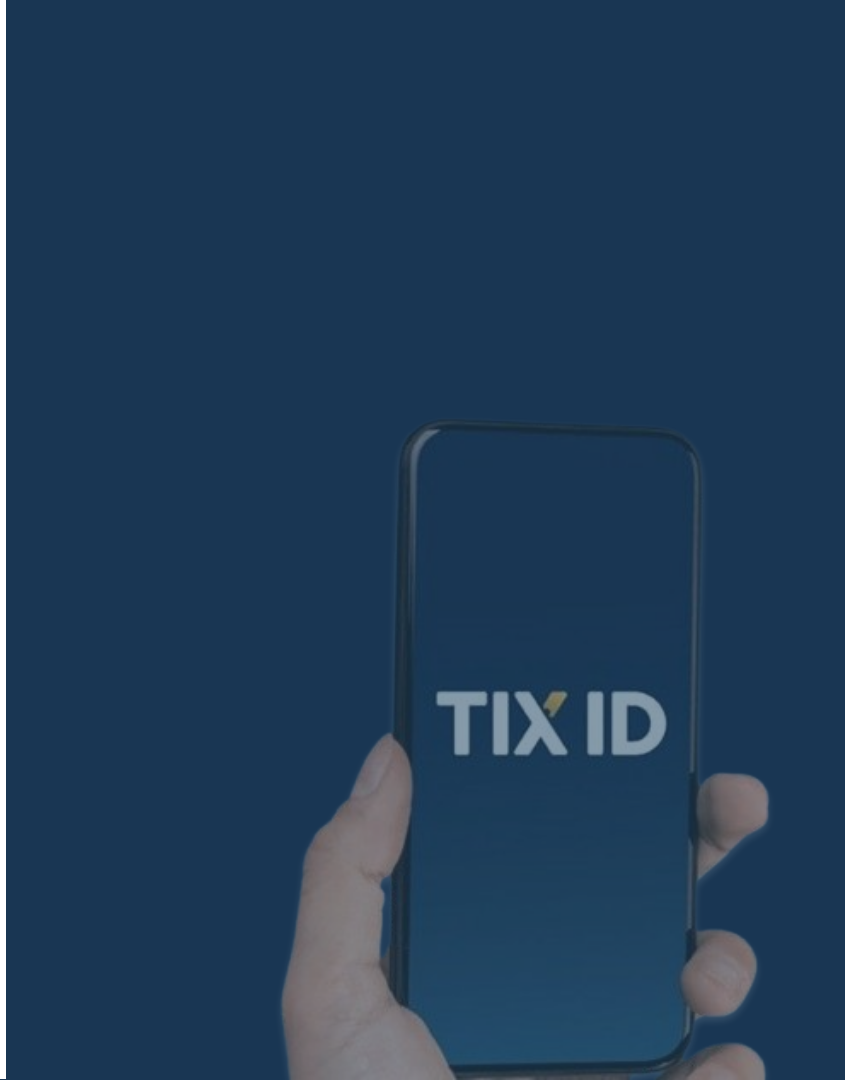
Sentiment Per Topic



Analisis sentimen per topik menunjukkan bahwa sebagian besar kategori didominasi oleh sentimen positif. Topik dengan proporsi positif tertinggi adalah "Jadwal Film & Pemilihan Tiket" dan "Pembelian Tiket & Masalah Transaksi", menunjukkan pengalaman pengguna yang memuaskan. Namun, topik seperti "Perbandingan dengan M-Tix & Efisiensi" dan "Ketersediaan Lokasi & Kota" memiliki proporsi non-positif yang relatif tinggi, mengindikasikan adanya ketidakpuasan pengguna terkait efisiensi layanan dan cakupan lokasi.

04

Kesimpulan



Kesimpulan

Output

Kesimpulan

1

Model ***BERT*** mencapai akurasi 94.3% dalam prediksi klasifikasi sentimen ulasan pengguna aplikasi Tix ID. Model sangat baik dalam mengenali kelas *positive*. Namun, masih perlu ditingkatkan kemampuannya dalam mengenali *Non-Positive*, terutama untuk menurunkan *false positive*

2

Sentimen positif terhadap Tix ID didominasi oleh kepuasan pengguna yang tinggi terhadap berbagai aspek layanan. Topik **"Feedback Positif dan Kepuasan"** mencapai proporsi sentimen positif sebesar **96,8%**, sedangkan **"Ucapan Harapan & Doa Positif"** bahkan mencapai 100%. Pengguna mengapresiasi kemudahan dalam pembelian tiket serta promo menarik yang ditawarkan, dengan sentimen positif pada topik **"Pembelian Tiket & Promo"** sebesar **81,7%**. Selain itu, pengalaman penggunaan aplikasi TIX juga mendapat tanggapan positif sebesar **81,0%, menegaskan kenyamanan dan kepraktisan layanan**. Aspek efisiensi waktu dan penjadwalan yang mencapai 62,0% serta perhatian terhadap film dan bioskop di Indonesia sebesar 68,4% turut memperkuat kesan positif pengguna. Secara keseluruhan, sentimen ini menunjukkan bahwa Tix ID berhasil memenuhi harapan pelanggan dalam hal kemudahan akses, penawaran menarik, dan pengalaman menonton yang memuaskan.

3

Keluhan utama meliputi **"Masalah Teknis & Akses Login" (87,1%), "Update Aplikasi yang Terlalu Sering" (87,9%), dan "Kekecewaan pada Promo (PHP)" (95,2%)**. Masalah teknis dan pembaruan yang berlebihan menimbulkan ketidaknyamanan dan menurunkan kepercayaan pengguna.

Kesimpulan

Strategi Mengatasi Sentimen Negatif Demi Meningkatkan Pengalaman Pengguna

Kesimpulan

Business Services

- Penjualan Tiket Digital
- Pembayaran & Integrasi Dompet Digital
- Notifikasi & Konten Hiburan

Business Goals

- Meningkatkan Transaksi dan Pengguna Aktif
- Meningkatkan Loyalitas dan Kepuasan Pelanggan
- Memperkuat Ekosistem Digital & Kolaborasi Mitra

Complication

Berdasarkan hasil Level Topic Sentiment Analysis, ditemukan beberapa masalah dengan proporsi sentimen non-positif yang tinggi, yaitu pada topik...

Masalah Teknis & Akses Login

Update Aplikasi yang Terlalu Sering

Kekecewaan pada Promo (PHP)

Solution

Perencanaan dan Pengelolaan Update Terstruktur

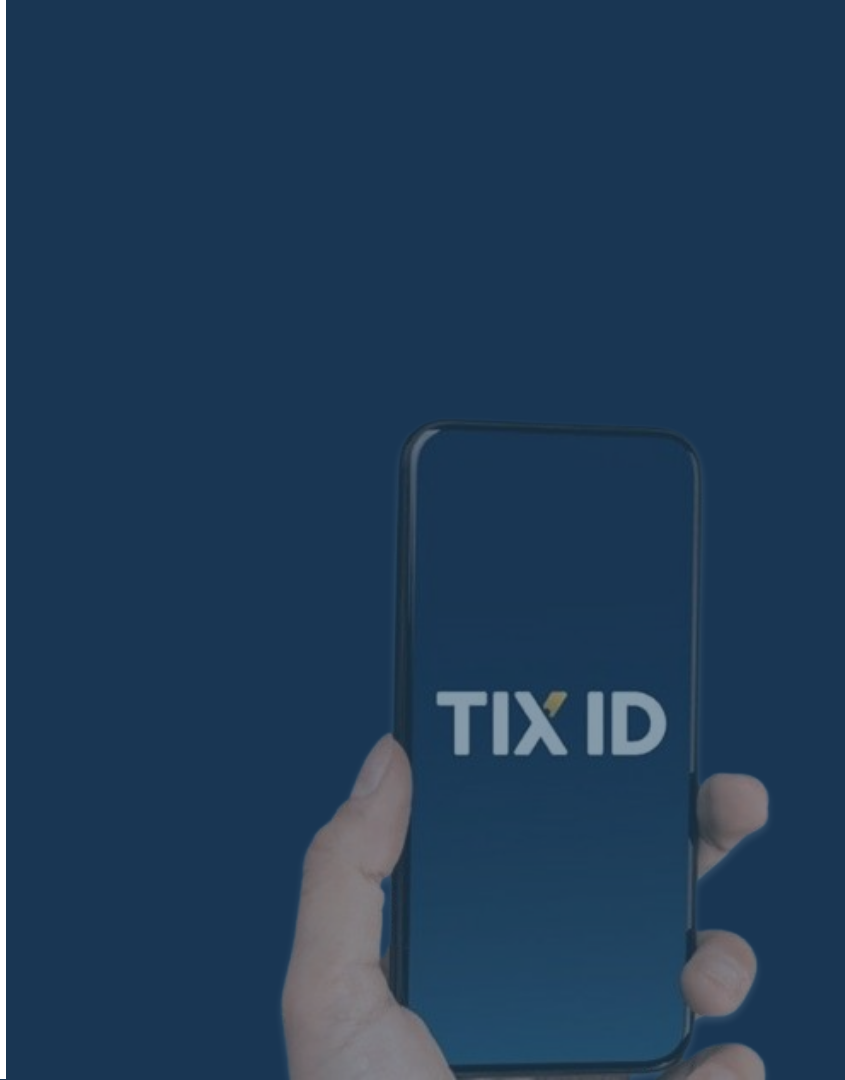
Susun jadwal update aplikasi yang terencana dengan baik, termasuk tahap persiapan, pengujian, dan peluncuran. Pastikan setiap update membawa perbaikan nyata tanpa mengganggu kenyamanan pengguna, serta komunikasikan perubahan secara transparan.

Marketing Promo yang Lebih Jelas dan Terbuka

Buat materi promo yang mudah dipahami dengan syarat dan ketentuan yang jelas untuk menghindari harapan palsu (PHP). Tingkatkan komunikasi dan edukasi kepada pengguna tentang promo yang berlaku agar kepercayaan tetap terjaga.

05

Daftar Pustaka



Daftar Pustaka

1. Permana, M. E., Ramadhan, H., Budi, I., Santoso, A. B., & Putra, P. K. (2020). *Sentiment analysis and topic detection of mobile banking application review*. In **2020 Fifth International Conference on Informatics and Computing (ICIC)** (pp. 1–6). IEEE. https://doi.org/10.1109/ICIC50835.2020.9288616
2. Liu, B. (2012). *Sentiment Analysis and Opinion Mining*. Morgan & Claypool Publishers.
3. Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). *Latent Dirichlet Allocation*. Journal of Machine Learning Research.
4. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). *Attention Is All You Need*. http://arxiv.org/abs/1706.03762
5. Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. http://arxiv.org/abs/1810.04805
6. Grootendorst, M. (2022). *BERTopic: Neural topic modeling with a class-based TF-IDF procedure*. http://arxiv.org/abs/2203.05794

Lampiran

1. Google Colab: https://colab.research.google.com/drive/1bMAUAKpooL3UKCkJ5eenwEHOThrA3i21?usp=sharing

Terima Kasih