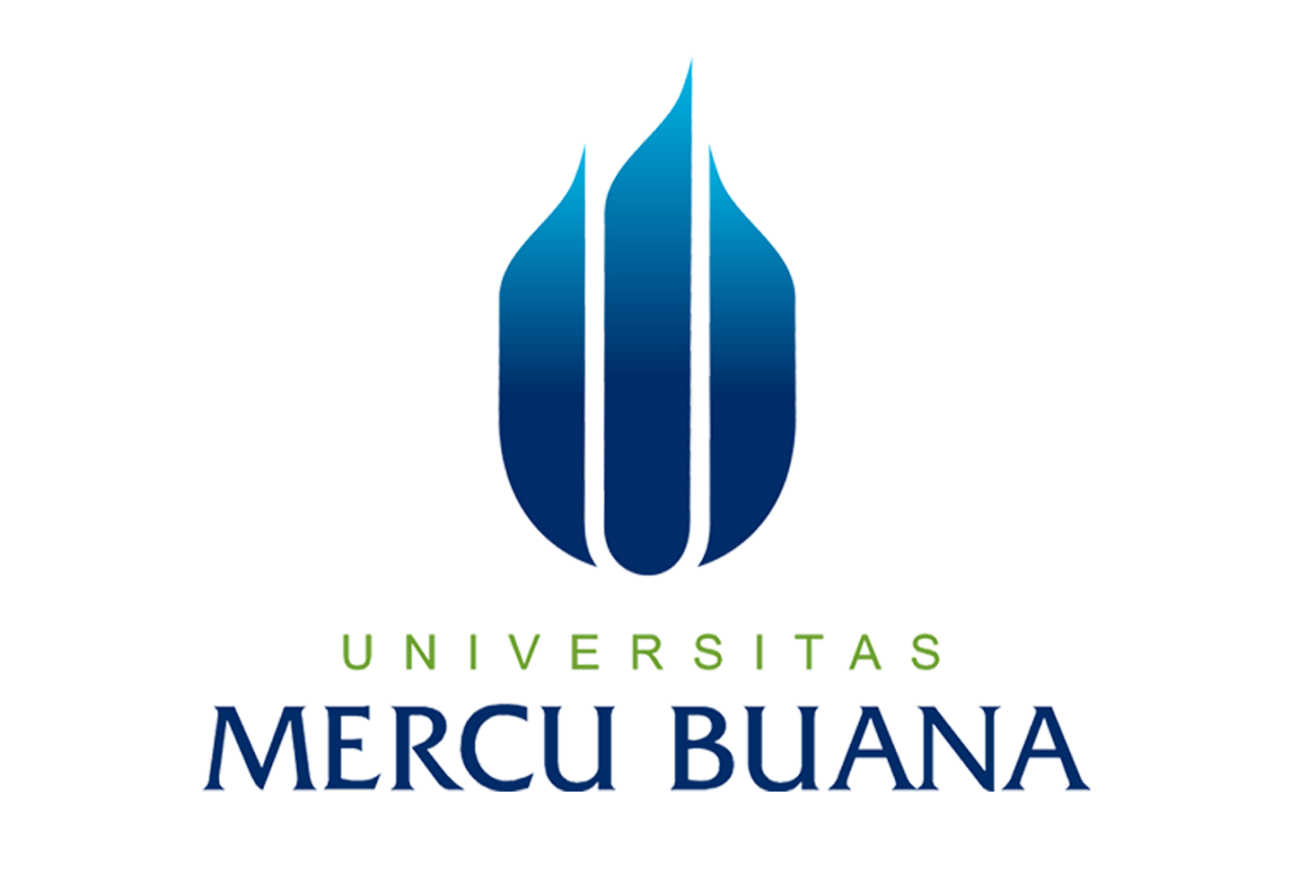
**KLASIFIKASI TEKS MENGGUNAKAN ALGORITMA LSTM**

Mata Kuliah: Natural Language Processing



Dosen Pengampu : Inna Sabily Karima S.kom., M.Kom.

|  |  |
| --- | --- |
| NAMA | : Muhammad Rizky |
| NIM | : 41522010097 |

**PROGRAM STUDI TEKNIK INFORMATIKA**

**FAKULTAS ILMU KOMPUTER**

**UNIVERSITAS MERCU BUANA**

**2024**

**DAFTAR ISI**

[BAB I PENDAHULUAN 3](#_Toc185067185)

[1. Latar Belakang 3](#_Toc185067186)

[2. Rumusan Masalah 4](#_Toc185067187)

[3. Tujuan Penelitian 4](#_Toc185067188)

[4. Manfaat Penelitian 4](#_Toc185067189)

[BAB II PEMBAHASAN 5](#_Toc185067190)

[1. Metodologi 5](#_Toc185067191)

[BAB III HASIL & ANALISIS 7](#_Toc185067192)

[1. Code & Hasil 7](#_Toc185067193)

[BAB IV PENUTUP 19](#_Toc185067194)

[Kesimpulan 19](#_Toc185067195)

# BAB I PENDAHULUAN

## Latar Belakang

Dalam era digital ini, media sosial telah menjadi sumber data yang sangat besar dan beragam. Twitter, sebagai salah satu platform media sosial terkemuka, menghasilkan jutaan tweet setiap hari yang mencakup opini, berita, dan diskusi tentang berbagai topik. Namun, volume data yang besar ini menimbulkan tantangan untuk memahami dan menganalisis informasi secara efisien.

Klasifikasi teks merupakan pendekatan yang digunakan untuk mengelompokkan teks, seperti tweet, ke dalam kategori tertentu berdasarkan isi atau konteksnya. Salah satu aplikasi utama dari klasifikasi teks adalah analisis sentimen, yang dapat memisahkan tweet ke dalam kategori positif, negatif, atau netral. Hal ini penting untuk mendukung berbagai aplikasi, termasuk:

* **Analisis Sentimen**: Membantu perusahaan memahami opini pelanggan terhadap produk atau layanan mereka.
* **Pemantauan Media Sosial**: Mengidentifikasi tren dan isu-isu penting secara real-time.
* **Moderasi Konten**: Membantu mengelola konten yang tidak pantas di platform.

Dengan mengembangkan model klasifikasi teks yang andal, organisasi dapat memanfaatkan data dari Twitter untuk mendukung pengambilan keputusan yang berbasis data. Misalnya, perusahaan dapat mempelajari opini konsumen untuk meningkatkan kualitas layanan, atau institusi pemerintah dapat memantau respons publik terhadap kebijakan baru.

Tujuan dari penelitian ini adalah untuk membangun sistem klasifikasi teks berbasis pembelajaran mesin yang mampu memprediksi sentimen secara akurat dari data Twitter. Dengan sistem ini, diharapkan dapat diperoleh solusi yang efektif untuk memanfaatkan data besar yang tidak terstruktur secara sistematis.

## Rumusan Masalah

Berdasarkan latar belakang tersebut, beberapa masalah yang diidentifikasi adalah:

* Bagaimana memproses data teks dari Twitter agar dapat digunakan untuk membangun model klasifikasi?
* Algoritma atau metode apa yang paling efektif dalam klasifikasi sentimen untuk data teks?
* Bagaimana mengevaluasi kinerja model klasifikasi teks pada data validasi?

## Tujuan Penelitian

Penelitian ini bertujuan untuk:

* Menganalisis dan memproses data teks dari Twitter menggunakan teknik pra-pemrosesan yang sesuai.
* Mengembangkan model klasifikasi teks untuk memprediksi sentimen tweet.
* Mengevaluasi model yang dikembangkan menggunakan metrik evaluasi yang relevan.

## Manfaat Penelitian

Hasil dari penelitian ini diharapkan memberikan manfaat sebagai berikut:

* **Bagi Akademisi**: Menambah referensi dan wawasan dalam pengembangan model klasifikasi teks.
* **Bagi Praktisi**: Memberikan solusi praktis untuk analisis sentimen dan pengelolaan data media sosial.
* **Bagi Masyarakat**: Mendukung pengambilan keputusan berbasis data untuk meningkatkan pengalaman pengguna pada platform digital.

# BAB II PEMBAHASAN

## Metodologi

Penelitian ini dilakukan melalui beberapa tahapan yang terstruktur untuk membangun dan mengevaluasi model klasifikasi teks. Berikut adalah tahapan metodologi yang diterapkan:

1. **Eksplorasi Data**

Dataset yang digunakan terdiri dari dua file utama, Dataset ini saya temukan di kaggle, dengan link: <https://www.kaggle.com/datasets/jp797498e/twitter-entity-sentiment-analysis> :

* twitter\_training.csv: Dataset untuk melatih model klasifikasi.
* twitter\_validation.csv: Dataset untuk mengevaluasi kinerja model.

Tahapan eksplorasi data mencakup:

* Memahami struktur dataset (jumlah kolom, baris, dan jenis data).
* Menganalisis distribusi label sentimen (positif, negatif, netral).
* Menangani data yang hilang atau tidak valid.

1. **Pra-pemrosesan Data**

Pra-pemrosesan dilakukan untuk meningkatkan kualitas data teks sehingga lebih cocok digunakan dalam pemodelan. Langkah-langkahnya meliputi:

1. **Membersihkan Data:**

Menghapus URL, tag pengguna (@), simbol, dan karakter khusus. Mengonversi teks menjadi huruf kecil.

* **Tokenisasi:** Memecah teks menjadi token (kata-kata individual).
* **Normalisasi:** Menghapus stopwords (kata umum yang tidak memiliki makna penting). Melakukan stemming atau lemmatization untuk mengembalikan kata ke bentuk dasarnya.
* **Representasi Teks:** Menggunakan teknik seperti TF-IDF (Term Frequency-Inverse Document Frequency) atau word embeddings (contoh: Word2Vec, GloVe) untuk mengonversi teks menjadi vektor numerik.

1. **Pemodelan**

**Model Berbasis Neural Network:**

Menerapkan Long Short-Term Memory (LSTM) atau Bidirectional Encoder Representations from Transformers (BERT) untuk menangkap konteks yang lebih dalam dalam teks.

1. **Evaluasi Model**

Kinerja model dievaluasi menggunakan dataset validasi dengan metrik berikut:

* **Akurasi**: Mengukur proporsi prediksi yang benar dari total prediksi.
* **Precision, Recall, dan F1-Score**: Mengevaluasi performa model pada setiap kategori sentimen.
* **Confusion Matrix**: Memberikan visualisasi pola kesalahan klasifikasi.

1. **Analisis Hasil**

Hasil evaluasi digunakan untuk:

* Mengidentifikasi kekuatan dan kelemahan model.
* Mengusulkan langkah-langkah perbaikan untuk pengembangan di masa depan.

# BAB III HASIL & ANALISIS

## Code & Hasil

*import* pandas *as* pd

*import* numpy *as* np

*import* tensorflow *as* tf

*from* sklearn.model\_selection *import* train\_test\_split

*from* tensorflow.keras.preprocessing.text *import* Tokenizer

*from* tensorflow.keras.preprocessing.sequence *import* pad\_sequences

*from* tensorflow.keras.models *import* Sequential

*from* tensorflow.keras.layers *import* Embedding, GlobalAveragePooling1D, Dense

*from* sklearn.preprocessing *import* LabelEncoder

Kode ini mengimpor beberapa pustaka penting yang digunakan untuk pemrosesan data, pembelajaran mesin, dan deep learning:

* **pandas:** Untuk manipulasi dan analisis data, sering digunakan untuk bekerja dengan data dalam bentuk tabel.
* **numpy:** Untuk operasi numerik, seperti manipulasi array.
* **tensorflow:** Library deep learning untuk membangun dan melatih model jaringan saraf.
* **sklearn.model\_selection.train\_test\_split:** Untuk membagi dataset menjadi set pelatihan dan pengujian.
* **tensorflow.keras.preprocessing.text.Tokenizer:** Untuk mengubah teks menjadi token (angka) agar dapat diproses oleh model.
* **tensorflow.keras.preprocessing.sequence.pad\_sequences:** Untuk memastikan semua input teks memiliki panjang yang sama dengan menambahkan padding.
* **tensorflow.keras.models.Sequential:** Untuk membangun model jaringan saraf secara bertahap.
* **tensorflow.keras.layers:** Menyediakan berbagai jenis lapisan untuk digunakan dalam jaringan saraf, seperti Embedding, GlobalAveragePooling1D, dan Dense.
* **sklearn.preprocessing.LabelEncoder**: Untuk mengonversi label kategorikal menjadi angka.

print("GPU is available:", tf.config.list\_physical\_devices('GPU'))

Kode ini Berfungsi untuk mengecek apakah ada GPU yang bisa digunakan untuk memproses data.

*# # CPU Optimizations*

*# tf.config.experimental.set\_virtual\_device\_configuration(*

*#     tf.config.experimental.list\_physical\_devices("CPU")[0],*

*#     [tf.config.experimental.VirtualDeviceConfiguration()],*

*# )*

*# GPU Memory Limit Optimization*

tf.config.set\_logical\_device\_configuration(

     tf.config.list\_physical\_devices("GPU")[0],

     [tf.config.LogicalDeviceConfiguration(memory\_limit=5899)],

)

Kode ini berfungsi untuk mengkonfigurasikan Pemrosesan data yang tadinya menggunakan prosesor laptop dengan sebuah GPU.

train\_df = pd.read\_csv('twitter\_training.csv')

validation\_df = pd.read\_csv('twitter\_validation.csv')

Kode ini Berfungsi untuk memasukkan dataset ke workspace Jupyter agar bisa di proses

train\_df.head()

Kode ini berfungsi untuk mencetak 4 row teratas dalam dataset train

validation\_df.head()

Kode ini berfungsi untuk mencetak 4 row teratas dalam dataset validation.

*# Assign column names*

train\_df.columns = ['id', 'platform', 'label', 'text']  *# Update names based on actual data*

validation\_df.columns = ['id', 'platform', 'label', 'text']

*# Display the updated dataframes*

print(train\_df.head())

print(validation\_df.head())

Kode ini berfungsi untuk menambahkan nama kolom untuk masing-masing kolom pada dataframe train dan validation karena sebelumnya tidak mempunyai nama kolom.

*# Extract relevant columns*

train\_texts = train\_df['text']

train\_labels = train\_df['label']

validation\_texts = validation\_df['text']

validation\_labels = validation\_df['label']

Kode ini Berfungsi untuk hanya memanggil 2 kolom yang akan di gunakan pada masing-masing dataframe yaitu kolom text, dan label.

*import* re

def *clean\_text*(text):

*if* not isinstance(text, str):

*return* ""

*# Remove URLs*

    text = re.sub(r'http\S+|www\S+|https\S+', '', text, flags=re.MULTILINE)

*# Remove user mentions (@username)*

    text = re.sub(r'@\w+', '', text)

*# Remove hashtags*

    text = re.sub(r'#\w+', '', text)

*# Remove special characters and numbers*

    text = re.sub(r'[^A-Za-z\s]', '', text)

*# Convert to lowercase*

    text = text.lower()

*return* text.strip()

Kode ini berfungsi untuk membersihkan teks, mengubah-nya menjadi lower case, dan menghilangkan tanda-tanda baca seperti @, #, dan lain-lain.

train\_texts = train\_texts.fillna("").apply(clean\_text)

validation\_texts = validation\_texts.fillna("").apply(clean\_text)

kode ini akan mengaplikasikan function pembersihan tersebut ke text train, dan text validasi.

*# Encode labels as integers (label encoding)*

label\_map = {label: idx *for* idx, label *in* enumerate(train\_df['label'].unique())}

train\_labels = train\_labels.map(label\_map)

validation\_labels = validation\_labels.map(label\_map)

Kode ini Berfungsi untuk melakukan encoding pada label, agar label yang tadinya teks bisa berubah menjadi integer.

*# Step 3: Split Data (if necessary)*

X\_train, X\_test, y\_train, y\_test = train\_test\_split(train\_texts, train\_labels, test\_size=0.2, random\_state=42)

Kode ini berfungs untuk men-split data. Menjadi data train, dan data test. Dataframe yang digunakan di sini adalah dataframe train.

*# Parameters*

max\_vocab\_size = 10000  *# Maximum size of the vocabulary*

max\_sequence\_length = 100  *# Maximum length of each sequence*

padding\_type = 'post'  *# Pad at the end of sequences*

truncating\_type = 'post'  *# Truncate sequences at the end*

oov\_token = "<OOV>"  *# Token for out-of-vocabulary words*

*# Initialize the tokenizer*

tokenizer = Tokenizer(num\_words=max\_vocab\_size, oov\_token=oov\_token)

*# Fit the tokenizer on the training data*

tokenizer.fit\_on\_texts(X\_train)

*# Convert texts to sequences*

X\_train\_sequences = tokenizer.texts\_to\_sequences(X\_train)

X\_test\_sequences = tokenizer.texts\_to\_sequences(X\_test)

validation\_sequences = tokenizer.texts\_to\_sequences(validation\_texts)

*# Pad sequences*

X\_train\_padded = pad\_sequences(X\_train\_sequences, maxlen=max\_sequence\_length, padding=padding\_type, truncating=truncating\_type)

X\_test\_padded = pad\_sequences(X\_test\_sequences, maxlen=max\_sequence\_length, padding=padding\_type, truncating=truncating\_type)

validation\_padded = pad\_sequences(validation\_sequences, maxlen=max\_sequence\_length, padding=padding\_type, truncating=truncating\_type)

*# Output a summary*

print("Example Tokenized Training Sequence:", X\_train\_sequences[0])

print("Example Padded Training Sequence:", X\_train\_padded[0])

print("Tokenizer Word Index Size:", len(tokenizer.word\_index))

Kode ini melakukan preprocessing data teks untuk digunakan dalam model pembelajaran mendalam (deep learning). Secara garis besar:

* **Parameter:** Menentukan ukuran kosakata, panjang maksimal urutan teks, serta cara menangani kata baru dan padding.
* **Tokenizer:** Membuat kosakata berdasarkan data pelatihan dan mengonversi teks menjadi urutan angka.
* **Padding:** Menyamakan panjang semua urutan agar sesuai dengan kebutuhan model.
* **Output:** Menampilkan contoh urutan tokenisasi, padding, dan ukuran kosakata.

*from* keras.utils *import* to\_categorical

*# Convert labels to categorical (one-hot encoding)*

y\_train = to\_categorical(y\_train, num\_classes=4)

y\_test = to\_categorical(y\_test, num\_classes=4)

validation\_labels = to\_categorical(validation\_labels, num\_classes=4)

Kode ini mengubah label target menjadi bentuk *one-hot encoding*, format yang sesuai untuk digunakan dalam klasifikasi multi-kelas.

Berikut prosesnya secara umum:

1. **Library**: to\_categorical dari keras.utils digunakan untuk mengubah label numerik menjadi matriks *one-hot encoding*.
2. **Input**:
   * y\_train, y\_test, dan validation\_labels adalah label asli dalam bentuk angka (0, 1, 2, 3) yang mewakili empat kelas.
3. **Output**:
   * Label diubah menjadi matriks biner dengan ukuran [jumlah\_data, num\_classes=4], di mana setiap baris berisi satu elemen "1" untuk menunjukkan kelasnya, dan sisanya "0".

*# Build the model*

model = tf.keras.Sequential([

    tf.keras.layers.Embedding(input\_dim=max\_vocab\_size, output\_dim=128, input\_length=max\_sequence\_length),

    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64, return\_sequences=True)),

    tf.keras.layers.GlobalMaxPooling1D(),

    tf.keras.layers.Dense(64, activation='relu'),

    tf.keras.layers.Dropout(0.5),

    tf.keras.layers.Dense(4, activation='softmax')

])

*# Compile the model with categorical cross-entropy loss*

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

*# Train the model*

history = model.fit(

    X\_train\_padded, y\_train,

    validation\_data=(validation\_padded, validation\_labels),

    epochs=20,

    batch\_size=32,

    verbose=1

)

Kode ini membangun, mengompilasi, dan melatih model pembelajaran mendalam untuk klasifikasi teks multi-kelas. Berikut penjelasan singkatnya:

**1. Membangun Model**

* **Layer Embedding**: Mengonversi indeks kata menjadi vektor numerik berdimensi 128 untuk menangkap representasi kata.
* **Bidirectional LSTM**: Memproses data sekuensial dalam dua arah (maju dan mundur) untuk menangkap konteks lebih baik. Parameter return\_sequences=True memastikan keluaran tiap langkah disimpan.
* **GlobalMaxPooling1D**: Mengambil nilai maksimum dari setiap fitur di sepanjang dimensi waktu, mengurangi dimensi tanpa kehilangan fitur penting.
* **Dense (64 unit)**: Lapisan dengan fungsi aktivasi ReLU untuk menangkap hubungan non-linear.
* **Dropout (0.5)**: Menambahkan regularisasi untuk mengurangi risiko *overfitting*.
* **Output Dense (4 unit)**: Lapisan akhir dengan fungsi aktivasi softmax untuk menghasilkan probabilitas pada 4 kelas (multi-kelas).

**2. Kompilasi Model**

* **Optimizer**: adam, metode adaptif untuk pembaruan bobot yang efisien.
* **Loss Function**: categorical\_crossentropy, digunakan untuk klasifikasi multi-kelas dengan *one-hot encoded labels*.
* **Metrics**: Mengukur *accuracy* selama pelatihan.

**3. Pelatihan Model**

* **Data Input**:
  + X\_train\_padded: Data teks yang sudah diproses.
  + y\_train: Label dalam format *one-hot encoding*.
  + validation\_padded, validation\_labels: Data validasi untuk mengevaluasi performa selama pelatihan.
* **Parameter**:
  + **Epochs**: 20 iterasi melalui data.
  + **Batch Size**: 32 sampel diproses sekaligus dalam satu langkah.
* **Output**: Riwayat pelatihan (*loss* dan *accuracy* di setiap epoch).

Kode ini bertujuan untuk melatih model yang dapat memprediksi kelas teks dengan memanfaatkan representasi kontekstual dari teks menggunakan LSTM.

Hasil Output Code setelah fitting adalah:

Epoch 1/20

1867/1867 [==============================] - 53s 25ms/step - loss: 0.9411 - accuracy: 0.6229 - val\_loss: 0.4658 - val\_accuracy: 0.8418

Epoch 2/20

1867/1867 [==============================] - 38s 20ms/step - loss: 0.5442 - accuracy: 0.8012 - val\_loss: 0.2845 - val\_accuracy: 0.9099

Epoch 3/20

1867/1867 [==============================] - 39s 21ms/step - loss: 0.3744 - accuracy: 0.8624 - val\_loss: 0.2115 - val\_accuracy: 0.9419

Epoch 4/20

1867/1867 [==============================] - 39s 21ms/step - loss: 0.2811 - accuracy: 0.8949 - val\_loss: 0.1890 - val\_accuracy: 0.9520

Epoch 5/20

1867/1867 [==============================] - 36s 20ms/step - loss: 0.2269 - accuracy: 0.9133 - val\_loss: 0.1836 - val\_accuracy: 0.9580

Epoch 6/20

1867/1867 [==============================] - 37s 20ms/step - loss: 0.1898 - accuracy: 0.9262 - val\_loss: 0.2183 - val\_accuracy: 0.9499

Epoch 7/20

1867/1867 [==============================] - 38s 20ms/step - loss: 0.1657 - accuracy: 0.9343 - val\_loss: 0.1813 - val\_accuracy: 0.9670

Epoch 8/20

1867/1867 [==============================] - 38s 21ms/step - loss: 0.1441 - accuracy: 0.9424 - val\_loss: 0.2134 - val\_accuracy: 0.9590

Epoch 9/20

1867/1867 [==============================] - 36s 19ms/step - loss: 0.1323 - accuracy: 0.9453 - val\_loss: 0.2089 - val\_accuracy: 0.9630

Epoch 10/20

1867/1867 [==============================] - 38s 21ms/step - loss: 0.1245 - accuracy: 0.9473 - val\_loss: 0.2145 - val\_accuracy: 0.9660

Epoch 11/20

1867/1867 [==============================] - 38s 20ms/step - loss: 0.1154 - accuracy: 0.9509 - val\_loss: 0.2545 - val\_accuracy: 0.9580

Epoch 12/20

1867/1867 [==============================] - 38s 20ms/step - loss: 0.1098 - accuracy: 0.9530 - val\_loss: 0.2256 - val\_accuracy: 0.9680

Epoch 13/20

1867/1867 [==============================] - 37s 20ms/step - loss: 0.1050 - accuracy: 0.9546 - val\_loss: 0.2593 - val\_accuracy: 0.9640

Epoch 14/20

1867/1867 [==============================] - 39s 21ms/step - loss: 0.1002 - accuracy: 0.9558 - val\_loss: 0.3106 - val\_accuracy: 0.9610

Epoch 15/20

1867/1867 [==============================] - 38s 20ms/step - loss: 0.0973 - accuracy: 0.9563 - val\_loss: 0.2819 - val\_accuracy: 0.9590

Epoch 16/20

1867/1867 [==============================] - 36s 19ms/step - loss: 0.0933 - accuracy: 0.9587 - val\_loss: 0.2945 - val\_accuracy: 0.9660

Epoch 17/20

1867/1867 [==============================] - 36s 19ms/step - loss: 0.0905 - accuracy: 0.9596 - val\_loss: 0.2915 - val\_accuracy: 0.9670

Epoch 18/20

1867/1867 [==============================] - 38s 20ms/step - loss: 0.0890 - accuracy: 0.9599 - val\_loss: 0.3495 - val\_accuracy: 0.9670

Epoch 19/20

1867/1867 [==============================] - 37s 20ms/step - loss: 0.0871 - accuracy: 0.9605 - val\_loss: 0.3568 - val\_accuracy: 0.9600

Epoch 20/20

1867/1867 [==============================] - 35s 19ms/step - loss: 0.0867 - accuracy: 0.9609 - val\_loss: 0.3047 - val\_accuracy: 0.9620

*# Evaluate the model*

loss, accuracy = model.evaluate(X\_test\_padded, y\_test, verbose=1)

print(f'Test Loss: {loss}')

print(f'Test Accuracy: {accuracy}')

dan kode ini berfungsi untuk melakukan evaluasi dari modelnya yang memiliki output akhir:  
467/467 [==============================] - 5s 10ms/step - loss: 0.8705 - accuracy: 0.8593

Test Loss: 0.8704943656921387

Test Accuracy: 0.8592756390571594

*# Predict on test set*

y\_pred = model.predict(X\_test\_padded)

y\_pred\_classes = np.argmax(y\_pred, axis=1)  *# Convert probabilities to class labels*

*# Convert y\_test to class labels (for evaluation)*

y\_test\_classes = np.argmax(y\_test, axis=1)

*# Classification Report*

*from* sklearn.metrics *import* classification\_report

print(classification\_report(y\_test\_classes, y\_pred\_classes))

Kode ini berfungsi untuk memberikan classification report pada data

467/467 [==============================] - 5s 9ms/step

precision recall f1-score support

0 0.87 0.85 0.86 4254

1 0.82 0.87 0.84 3551

2 0.88 0.89 0.89 4471

3 0.86 0.81 0.83 2661

accuracy 0.86 14937

macro avg 0.86 0.85 0.86 14937

weighted avg 0.86 0.86 0.86 14937

*from* sklearn.metrics *import* confusion\_matrix

*import* seaborn *as* sns

*# Generate the confusion matrix*

cm = confusion\_matrix(y\_test\_classes, y\_pred\_classes)

*# Plot confusion matrix*

plt.figure(figsize=(8, 6))

sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=np.unique(y\_test\_classes), yticklabels=np.unique(y\_test\_classes))

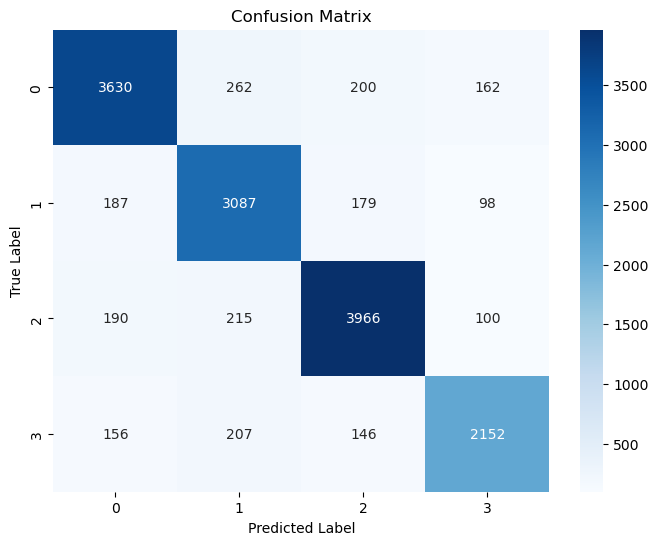
plt.title('Confusion Matrix')

plt.xlabel('Predicted Label')

plt.ylabel('True Label')

plt.show()

Dan ini adalah kode untuk memberikan confusion matrix



Confusion matrix pada gambar ini menunjukkan kinerja model klasifikasi yang memprediksi empat kelas (0, 1, 2, dan 3). Berikut penjelasan dari masing-masing elemen matriks:

1. **Diagonal utama (sel [0,0], [1,1], [2,2], [3,3]):**
   * Menunjukkan jumlah prediksi yang benar untuk masing-masing kelas:
     + **3630** sampel kelas 0 diprediksi dengan benar sebagai kelas 0.
     + **3087** sampel kelas 1 diprediksi dengan benar sebagai kelas 1.
     + **3966** sampel kelas 2 diprediksi dengan benar sebagai kelas 2.
     + **2152** sampel kelas 3 diprediksi dengan benar sebagai kelas 3.
2. **Di luar diagonal utama:**
   * Menunjukkan jumlah prediksi yang salah, di mana model memprediksi kelas yang berbeda dari label sebenarnya:
     + Baris 0:
       - **262** sampel kelas 0 diprediksi sebagai kelas 1.
       - **200** sampel kelas 0 diprediksi sebagai kelas 2.
       - **162** sampel kelas 0 diprediksi sebagai kelas 3.
     + Baris 1:
       - **187** sampel kelas 1 diprediksi sebagai kelas 0.
       - **179** sampel kelas 1 diprediksi sebagai kelas 2.
       - **98** sampel kelas 1 diprediksi sebagai kelas 3.
     + Baris 2:
       - **190** sampel kelas 2 diprediksi sebagai kelas 0.
       - **215** sampel kelas 2 diprediksi sebagai kelas 1.
       - **100** sampel kelas 2 diprediksi sebagai kelas 3.
     + Baris 3:
       - **156** sampel kelas 3 diprediksi sebagai kelas 0.
       - **207** sampel kelas 3 diprediksi sebagai kelas 1.
       - **146** sampel kelas 3 diprediksi sebagai kelas 2.
3. **Maksud dan Evaluasi Keseluruhan:**
   * Model memiliki performa yang cukup baik karena angka pada diagonal utama (prediksi benar) lebih besar dibandingkan angka di luar diagonal (prediksi salah).
   * Namun, terdapat kesalahan prediksi di mana sampel dari satu kelas sering salah diklasifikasikan ke kelas lain. Misalnya:
     + Sebanyak **262** sampel kelas 0 salah diprediksi sebagai kelas 1.
     + Sebanyak **207** sampel kelas 3 salah diprediksi sebagai kelas 1.
4. **Analisis Berdasarkan Kelas:**
   * Kelas 2 memiliki jumlah prediksi benar tertinggi (**3966**), menunjukkan bahwa model bekerja sangat baik untuk kelas ini.
   * Kelas 3 memiliki jumlah kesalahan yang lebih tinggi dibandingkan kelas lainnya.

Kesimpulannya, confusion matrix ini menunjukkan distribusi prediksi benar dan salah pada masing-masing kelas, yang dapat digunakan untuk mengidentifikasi area yang memerlukan perbaikan, seperti kelas yang sering salah prediksi.

# BAB IV PENUTUP

## Kesimpulan

Kesimpulan dari confusion matrix yang ditampilkan adalah sebagai berikut:

1. **Performa Model Secara Keseluruhan:**
   * Model menunjukkan performa yang cukup baik, dengan jumlah prediksi benar (angka pada diagonal utama) yang lebih tinggi dibandingkan dengan jumlah prediksi salah (angka di luar diagonal utama).
   * Model bekerja lebih baik untuk kelas **2** (3966 prediksi benar) dibandingkan kelas lainnya, yang berarti data kelas ini cenderung lebih mudah dikenali oleh model.
2. **Kesalahan Prediksi yang Signifikan:**
   * Ada kesalahan prediksi yang cukup sering terjadi antara kelas tertentu, misalnya:
     + **262 sampel kelas 0** salah diprediksi sebagai kelas 1.
     + **215 sampel kelas 2** salah diprediksi sebagai kelas 1.
     + **207 sampel kelas 3** salah diprediksi sebagai kelas 1.
   * Kelas **1 dan 3** tampaknya memiliki tumpang tindih fitur yang menyebabkan model sering salah prediksi ke kelas-kelas ini.
3. **Area untuk Perbaikan:**
   * Model perlu ditingkatkan dalam membedakan kelas-kelas yang sering terjadi kesalahan, seperti antara kelas **0 dan 1**, atau kelas **3 dan 1**.
   * Kemungkinan perbaikan dapat mencakup:
     + Memperbaiki distribusi data pelatihan (jika tidak seimbang).
     + Menggunakan teknik ekstraksi fitur yang lebih baik.
     + Melakukan tuning parameter model atau mencoba model lain.
4. **Akurasi Keseluruhan:**
   * Meskipun akurasi tidak dihitung langsung dari confusion matrix, total jumlah pada diagonal utama memberikan gambaran bahwa model cukup akurat karena banyak sampel yang diklasifikasikan dengan benar.

Secara ringkas, model memiliki performa yang baik, tetapi masih terdapat kesalahan pada beberapa kelas tertentu yang memerlukan perhatian lebih.