1. Mengimport library yang dibutuhkan

In [1]:
```
# Import library yang dibutuhkan
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import random
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.layers import Input, Embedding, Dense, SimpleRNN, Dropout
from tensorflow.keras.models import Sequential
from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlyStopping
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
from nltk.corpus import wordnet
from googletrans import Translator
from wordcloud import WordCloud
```

2. Load Data

Memuat dataset dari file CSV dan menampilkan informasi dasar tentang data seperti tipe data dan jumlah nilai yang hilang.

In [2]:
```
# Load dataset
df = pd.read_csv("liputan6.csv", index_col=0)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 10972 entries, 0 to 10971
Data columns (total 2 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   clean_article  10972 non-null  object
 1   clean_summary  10972 non-null  object
dtypes: object(2)
memory usage: 257.2+ KB
```

Menampilkan statistik deskriptif untuk kolom bertipe objek.

In [3]:
```
df.describe(include='O')
```

Out[3]:

	clean_article	clean_summary
count	10972	10972
unique	10961	10958
top	Liputan6. com, Jakarta : Di masyarakat, televi...	Sebuah tenda didirikan dekat pintu keluar jala...
freq	4	2

Menampilkan 5 baris pertama dari dataset untuk melihat data.

In [4]:
```
df.head()
```

Out[4]:

	clean_article	clean_summary
0	Liputan6. com, Surabaya : Radiogram Direktorat...	Gubernur Jatim Imam Utomo tak mau melantik Bup...
1	Liputan6. com, Jakarta : Berbeda dengan aliran...	Pelukis RM Koestarto memamerkan hasil karyanya...
2	Liputan6. com, Jambi : Ratusan orang dari Kesa...	Dua kelompok pengunjuk rasa di Jambi, menuntut...
3	Liputan6. com, Jakarta : Badan Penyehatan Perb...	BPPN masih mengkaji bank rekap yang dianggap p...
4	Liputan6. com, Jakarta : Ketua Komisi I DPR Ya...	Kendati Dewan Papua membatalkan deklarasi [illegible]

2. Selecting Necessary Columns and Dropping NaN Values

Memilih kolom yang relevan ('clean_article' dan 'clean_summary') dan menghapus data yang hilang (NaN).

In [5]:
```
# Selecting necessary columns and dropping NaN values
data = df[['clean_article', 'clean_summary']].dropna()
```

In [6]:
```
# Menampilkan jumlah total data dalam dataset
print("Total data dalam dataset:", len(df))

# Menampilkan jumlah data yang ada di setiap kolom 'clean_article' dan 'clean_summa
print("Total data di kolom 'clean_article':", df['clean_article'].notnull().sum())
print("Total data di kolom 'clean_summary':", df['clean_summary'].notnull().sum())

# Menjumlahkan total data non-null dari kedua kolom
total_non_null = df['clean_article'].notnull().sum() + df['clean_summary'].notnull(
print("Total dari kedua kolom (clean_article dan clean_summary):", total_non_null)
```

```
Total data dalam dataset: 10972
Total data di kolom 'clean_article': 10972
Total data di kolom 'clean_summary': 10972
Total dari kedua kolom (clean_article dan clean_summary): 21944
```

3. Selecting Subset Data and View It

Mengambil 10% data dari data asli untuk digunakan

```
# Mengambil subset data
df = df.sample(frac=0.10, random_state=50)  # Menggunakan 10% dari dataset
```

Menampilkan jumlah data terkini

In [7]:
```
# Menampilkan jumlah total data dalam dataset
print("Total data dalam dataset:", len(df))

# Menampilkan jumlah data yang ada di setiap kolom 'clean_article' dan 'clean_summa
print("Total data di kolom 'clean_article':", df['clean_article'].notnull().sum())
print("Total data di kolom 'clean_summary':", df['clean_summary'].notnull().sum())

# Menjumlahkan total data non-null dari kedua kolom
total_non_null = df['clean_article'].notnull().sum() + df['clean_summary'].notnull(
print("Total dari kedua kolom (clean_article dan clean_summary):", total_non_null)
```

```
Total data dalam dataset: 1097
Total data di kolom 'clean_article': 1097
Total data di kolom 'clean_summary': 1097
Total dari kedua kolom (clean_article dan clean_summary): 2194
```

3. Visualizing Sentiment Distribution and Wordcloud

Menggambarkan distribusi sentimen dalam data menggunakan grafik batang.

In [9]:
```
# Menghitung jumlah kata rata-rata untuk clean_article dan clean_summary
df['article_word_count'] = df['clean_article'].apply(lambda x: len(x.split()))
df['summary_word_count'] = df['clean_summary'].apply(lambda x: len(x.split()))

# Menghitung rata-rata jumlah kata
avg_word_count = {
    'clean_article': df['article_word_count'].mean(),
    'clean_summary': df['summary_word_count'].mean()
}

# # Membuat bar plot
# plt.figure(figsize=(10, 5))
# plt.bar(avg_word_count.keys(), avg_word_count.values(), color=['blue', 'orange'])
# plt.title('Rata-Rata Jumlah Kata per Kolom', fontsize=16)
# plt.ylabel('Rata-Rata Jumlah Kata', fontsize=12)
# plt.xlabel('Kolom', fontsize=12)
# plt.show()
```

```
# Membuat word cloud untuk clean_article
wordcloud_article = WordCloud(width=800, height=400, background_color='white').gene

# Membuat word cloud untuk clean_summary
wordcloud_summary = WordCloud(width=800, height=400, background_color='white').gene

# Menampilkan word cloud
plt.figure(figsize=(15, 7))
plt.subplot(1, 2, 1)
plt.imshow(wordcloud_article, interpolation='bilinear')
plt.title('Word Cloud untuk clean_article', fontsize=16)
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(wordcloud_summary, interpolation='bilinear')
plt.title('Word Cloud untuk clean_summary', fontsize=16)
plt.axis('off')

plt.tight_layout()
plt.show()
```

Tokenisasi teks untuk mengonversinya menjadi urutan angka berdasarkan frekuensi kata dalam data pelatihan.

In [10]:
```
# Menggunakan Tokenizer untuk teks
tokenizer = Tokenizer()
tokenizer.fit_on_texts(df['clean_article'].tolist() + df['clean_summary'].tolist())

# Konversi teks menjadi sequences
input_sequences = tokenizer.texts_to_sequences(df['clean_article'].tolist())
target_sequences = tokenizer.texts_to_sequences(df['clean_summary'].tolist())

# Padding sequences untuk memastikan panjangnya seragam
max_input_len = 500  # Batasi panjang maksimum input sequence
max_target_len = 50  # Batasi panjang maksimum target sequence

input_sequences = pad_sequences(input_sequences, maxlen=max_input_len, padding='pos
target_sequences = pad_sequences(target_sequences, maxlen=max_target_len, padding='
```

```
# Mendapatkan ukuran vocabulary
vocab_size = len(tokenizer.word_index) + 1
```

5. Augmentasi Data

Fungsi untuk augmentasi teks dengan mengganti beberapa kata dengan berbagai metode berikut

In [11]:
```
# Fungsi untuk augmentasi teks dengan mengganti kata dengan sinonim
def synonym_replacement(text):
    words = text.split()    # Memisahkan teks menjadi kata-kata
    new_words = words.copy()    # Menyalin kata-kata
    word_idx = random.randint(0, len(words)-1)  # Pilih indeks acak
    word = words[word_idx]

    # Mendapatkan sinonim
    synonyms = wordnet.synsets(word)
    if synonyms:
        synonym = random.choice(synonyms).lemmas()[0].name()
        new_words[word_idx] = synonym if synonym != word else word  # Pastikan tida

    return ' '.join(new_words)

# Fungsi back translation (terjemahkan ke bahasa Inggris, kemudian kembali ke bahas
def back_translation(text, src='id', dest='en'):
    translator = Translator()
    translated = translator.translate(text, src=src, dest=dest).text
    back_translated = translator.translate(translated, src=dest, dest=src).text
    return back_translated

# Fungsi untuk augmentasi dataset
def augment_dataset(df):
    augmented_articles = []
    augmented_summaries = []

    for i in range(len(df)):
        article = df.iloc[i]['clean_article']
        summary = df.iloc[i]['clean_summary']

        # Menerapkan augmentasi
        augmented_articles.append(synonym_replacement(article))
        augmented_summaries.append(synonym_replacement(summary))

    augmented_df = pd.DataFrame({
        'clean_article': augmented_articles,
        'clean_summary': augmented_summaries
    })

    return augmented_df
```

6. Membagi Dataset

Membagi data menjadi fitur (X) dan label (y), kemudian membagi data menjadi data pelatihan, data pengujian, dan data testing.

In [12]:
```
# Model dengan validasi
X_train, X_temp, y_train, y_temp = train_test_split(input_sequences, target_sequenc
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, rand

# Padding target sequences untuk validasi dan testing
y_train = pad_sequences(y_train, maxlen=max_input_len, padding='post')
y_val = pad_sequences(y_val, maxlen=max_input_len, padding='post')
y_test = pad_sequences(y_test, maxlen=max_input_len, padding='post')

# Menampilkan ukuran dataset
print(f"Training data: {len(X_train)}")
print(f"Validation data: {len(X_val)}")
print(f"Testing data: {len(X_test)}")
```

```
Training data: 767
Validation data: 165
Testing data: 165
```

7. Model Simple RNN

Menggunakan simple RNN untuk memprediksi model beserta beberapa layer lainnya.

In [13]:
```
# Model dengan SimpleRNN
model = Sequential()
model.add(Embedding(input_dim=vocab_size, output_dim=256, input_length=max_input_le
model.add(SimpleRNN(256, return_sequences=True))  # SimpleRNN pertama
model.add(Dropout(0.4))  # Dropout pertama
# model.add(SimpleRNN(256, return_sequences=True))  # SimpleRNN kedua
# model.add(Dropout(0.4))  # Dropout kedua
model.add(Dense(vocab_size, activation='softmax'))  # Output layer
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['a

# Menampilkan summary model
model.build(input_shape=(None, max_input_len))
model.summary()
```

```
e:\Software\Python\lib\site-packages\keras\src\layers\core\embedding.py:90: UserWarn
ing: Argument `input_length` is deprecated. Just remove it.
  warnings.warn(
```

Model: "sequential"

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 500, 256)	3,945,216
simple_rnn (SimpleRNN)	(None, 500, 256)	131,328
dropout (Dropout)	(None, 500, 256)	0
dense (Dense)	(None, 500, 15411)	3,960,627

Total params: 8,037,171 (30.66 MB)

Trainable params: 8,037,171 (30.66 MB)

Non-trainable params: 0 (0.00 B)

Melatih Model dengan Callbacks untuk Penurunan Learning Rate dan Early Stopping

In [14]:
```
# Callbacks untuk training
lr_scheduler = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=3, min_lr
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights

# Melatih model
history = model.fit(
    X_train,
    y_train,
    epochs=30,
    batch_size=160,
    validation_data=(X_val, y_val),
    callbacks=[lr_scheduler, early_stopping]
)
```

```
Epoch 1/30
5/5 ━━━━━━━━━━━━━━━━━━━━ 150s 29s/step - accuracy: 0.3826 - loss: 9.1968 - val_accur
acy: 0.9555 - val_loss: 7.3784 - learning_rate: 0.0010
Epoch 2/30
5/5 ━━━━━━━━━━━━━━━━━━━━ 131s 26s/step - accuracy: 0.9556 - loss: 6.8112 - val_accur
acy: 0.9555 - val_loss: 4.8293 - learning_rate: 0.0010
Epoch 3/30
5/5 ━━━━━━━━━━━━━━━━━━━━ 108s 22s/step - accuracy: 0.9557 - loss: 4.2762 - val_accur
acy: 0.9555 - val_loss: 2.4179 - learning_rate: 0.0010
Epoch 4/30
5/5 ━━━━━━━━━━━━━━━━━━━━ 109s 22s/step - accuracy: 0.9554 - loss: 2.0011 - val_accur
acy: 0.9555 - val_loss: 0.8531 - learning_rate: 0.0010
Epoch 5/30
5/5 ━━━━━━━━━━━━━━━━━━━━ 120s 24s/step - accuracy: 0.9557 - loss: 0.7728 - val_accur
acy: 0.9555 - val_loss: 0.5625 - learning_rate: 0.0010
Epoch 6/30
5/5 ━━━━━━━━━━━━━━━━━━━━ 131s 22s/step - accuracy: 0.9555 - loss: 0.5657 - val_accur
acy: 0.9555 - val_loss: 0.5543 - learning_rate: 0.0010
Epoch 7/30
5/5 ━━━━━━━━━━━━━━━━━━━━ 109s 22s/step - accuracy: 0.9558 - loss: 0.5521 - val_accur
acy: 0.9555 - val_loss: 0.5552 - learning_rate: 0.0010
Epoch 8/30
5/5 ━━━━━━━━━━━━━━━━━━━━ 109s 21s/step - accuracy: 0.8821 - loss: 2.3388 - val_accur
acy: 0.6457 - val_loss: 8.1381 - learning_rate: 0.0010
Epoch 9/30
5/5 ━━━━━━━━━━━━━━━━━━━━ 118s 24s/step - accuracy: 0.7420 - loss: 5.7756 - val_accur
acy: 0.9556 - val_loss: 0.5442 - learning_rate: 0.0010
Epoch 10/30
5/5 ━━━━━━━━━━━━━━━━━━━━ 144s 29s/step - accuracy: 0.9559 - loss: 0.5354 - val_accur
acy: 0.9556 - val_loss: 0.5360 - learning_rate: 0.0010
Epoch 11/30
5/5 ━━━━━━━━━━━━━━━━━━━━ 141s 29s/step - accuracy: 0.9559 - loss: 0.5267 - val_accur
acy: 0.9556 - val_loss: 0.5284 - learning_rate: 0.0010
Epoch 12/30
5/5 ━━━━━━━━━━━━━━━━━━━━ 141s 29s/step - accuracy: 0.9556 - loss: 0.5243 - val_accur
acy: 0.9556 - val_loss: 0.5242 - learning_rate: 0.0010
Epoch 13/30
5/5 ━━━━━━━━━━━━━━━━━━━━ 218s 46s/step - accuracy: 0.9506 - loss: 0.6379 - val_accur
acy: 0.8605 - val_loss: 2.7796 - learning_rate: 0.0010
Epoch 14/30
5/5 ━━━━━━━━━━━━━━━━━━━━ 224s 46s/step - accuracy: 0.8776 - loss: 2.3733 - val_accur
acy: 0.9556 - val_loss: 0.5199 - learning_rate: 0.0010
Epoch 15/30
5/5 ━━━━━━━━━━━━━━━━━━━━ 216s 44s/step - accuracy: 0.9555 - loss: 0.5168 - val_accur
acy: 0.9556 - val_loss: 0.5199 - learning_rate: 0.0010
Epoch 16/30
5/5 ━━━━━━━━━━━━━━━━━━━━ 225s 45s/step - accuracy: 0.9530 - loss: 0.5739 - val_accur
acy: 0.9555 - val_loss: 0.5200 - learning_rate: 0.0010
Epoch 17/30
5/5 ━━━━━━━━━━━━━━━━━━━━ 121s 21s/step - accuracy: 0.9548 - loss: 0.5342 - val_accur
acy: 0.9555 - val_loss: 0.5185 - learning_rate: 0.0010
Epoch 18/30
5/5 ━━━━━━━━━━━━━━━━━━━━ 88s 18s/step - accuracy: 0.9549 - loss: 0.5271 - val_accura
cy: 0.9555 - val_loss: 0.5174 - learning_rate: 0.0010
Epoch 19/30
5/5 ━━━━━━━━━━━━━━━━━━━━ 93s 19s/step - accuracy: 0.9554 - loss: 0.5177 - val_accura
```

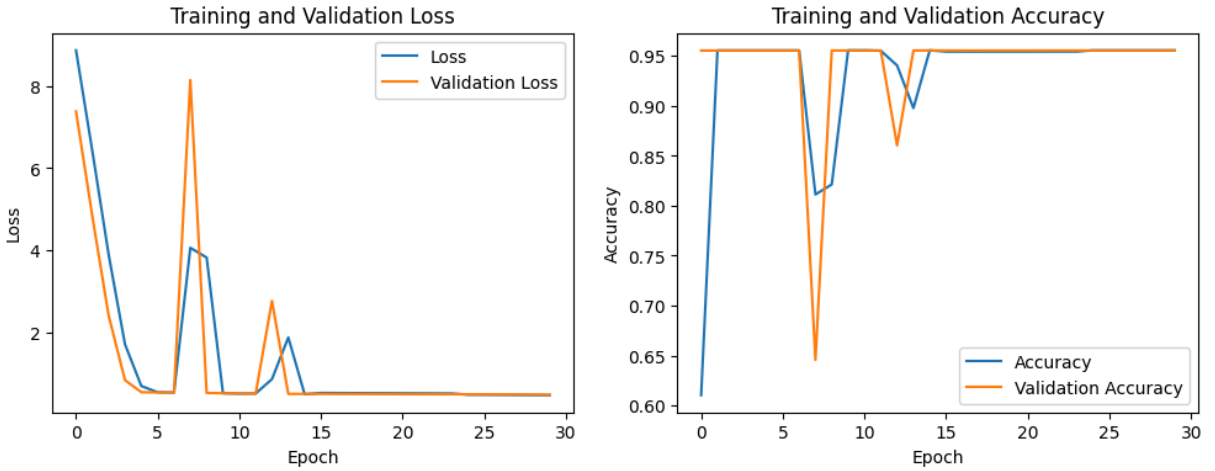
```
cy: 0.9555 - val_loss: 0.5164 - learning_rate: 0.0010
Epoch 20/30
5/5 ━━━━━━━━━━━━━━━━━━━━ 101s 20s/step - accuracy: 0.9551 - loss: 0.5226 - val_accur
acy: 0.9555 - val_loss: 0.5154 - learning_rate: 0.0010
Epoch 21/30
5/5 ━━━━━━━━━━━━━━━━━━━━ 109s 22s/step - accuracy: 0.9542 - loss: 0.5410 - val_accur
acy: 0.9555 - val_loss: 0.5143 - learning_rate: 0.0010
Epoch 22/30
5/5 ━━━━━━━━━━━━━━━━━━━━ 112s 23s/step - accuracy: 0.9551 - loss: 0.5170 - val_accur
acy: 0.9556 - val_loss: 0.5134 - learning_rate: 0.0010
Epoch 23/30
5/5 ━━━━━━━━━━━━━━━━━━━━ 112s 23s/step - accuracy: 0.9551 - loss: 0.5179 - val_accur
acy: 0.9556 - val_loss: 0.5125 - learning_rate: 0.0010
Epoch 24/30
5/5 ━━━━━━━━━━━━━━━━━━━━ 110s 22s/step - accuracy: 0.9543 - loss: 0.5360 - val_accur
acy: 0.9556 - val_loss: 0.5114 - learning_rate: 0.0010
Epoch 25/30
5/5 ━━━━━━━━━━━━━━━━━━━━ 113s 22s/step - accuracy: 0.9558 - loss: 0.5001 - val_accur
acy: 0.9556 - val_loss: 0.5103 - learning_rate: 0.0010
Epoch 26/30
5/5 ━━━━━━━━━━━━━━━━━━━━ 112s 23s/step - accuracy: 0.9560 - loss: 0.4962 - val_accur
acy: 0.9556 - val_loss: 0.5093 - learning_rate: 0.0010
Epoch 27/30
5/5 ━━━━━━━━━━━━━━━━━━━━ 115s 23s/step - accuracy: 0.9557 - loss: 0.4970 - val_accur
acy: 0.9556 - val_loss: 0.5080 - learning_rate: 0.0010
Epoch 28/30
5/5 ━━━━━━━━━━━━━━━━━━━━ 109s 22s/step - accuracy: 0.9559 - loss: 0.4938 - val_accur
acy: 0.9556 - val_loss: 0.5060 - learning_rate: 0.0010
Epoch 29/30
5/5 ━━━━━━━━━━━━━━━━━━━━ 111s 22s/step - accuracy: 0.9555 - loss: 0.4952 - val_accur
acy: 0.9556 - val_loss: 0.5034 - learning_rate: 0.0010
Epoch 30/30
5/5 ━━━━━━━━━━━━━━━━━━━━ 111s 22s/step - accuracy: 0.9557 - loss: 0.4897 - val_accur
acy: 0.9556 - val_loss: 0.5010 - learning_rate: 0.0010
```

In [18]:

```python
# Plotting hasil training
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
```

```
plt.legend()
plt.show()
```



8. Evaluasi Model Dengan Data Testing dan Menampilkan Evaluasi Metriknya

In [30]:
```
# Evaluasi pada data test
chunk_size = 500  # Sesuaikan ukuran chunk untuk mencegah penggunaan memori yang be
y_test_pred = []

# Prediksi data dalam batch kecil
for i in range(0, len(X_test), chunk_size):
    batch_X = X_test[i:i+chunk_size]
    batch_pred_prob = model.predict(batch_X, batch_size=32)  # batch_size lebih kec
    batch_pred = np.argmax(batch_pred_prob, axis=1)
    y_test_pred.extend(batch_pred)

y_test_true = np.argmax(y_test, axis=-1)  # Argmax untuk label sebenarnya
y_test_pred = np.argmax(y_test_pred, axis=-1)  # Prediksi juga diubah

# Pastikan keduanya memiliki bentuk yang sama
print(f"Shape y_test_true: {y_test_true.shape}, y_test_pred: {y_test_pred.shape}")

# Evaluasi
test_accuracy = accuracy_score(y_test_true.flatten(), y_test_pred.flatten())
print(f"Test Accuracy: {test_accuracy:.4f}")
print("\nTest Classification Report:\n", classification_report(y_test_true.flatten(


# Membuat confusion matrix
conf_matrix_test = confusion_matrix(y_test_true, y_test_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix_test, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix - Test Data')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

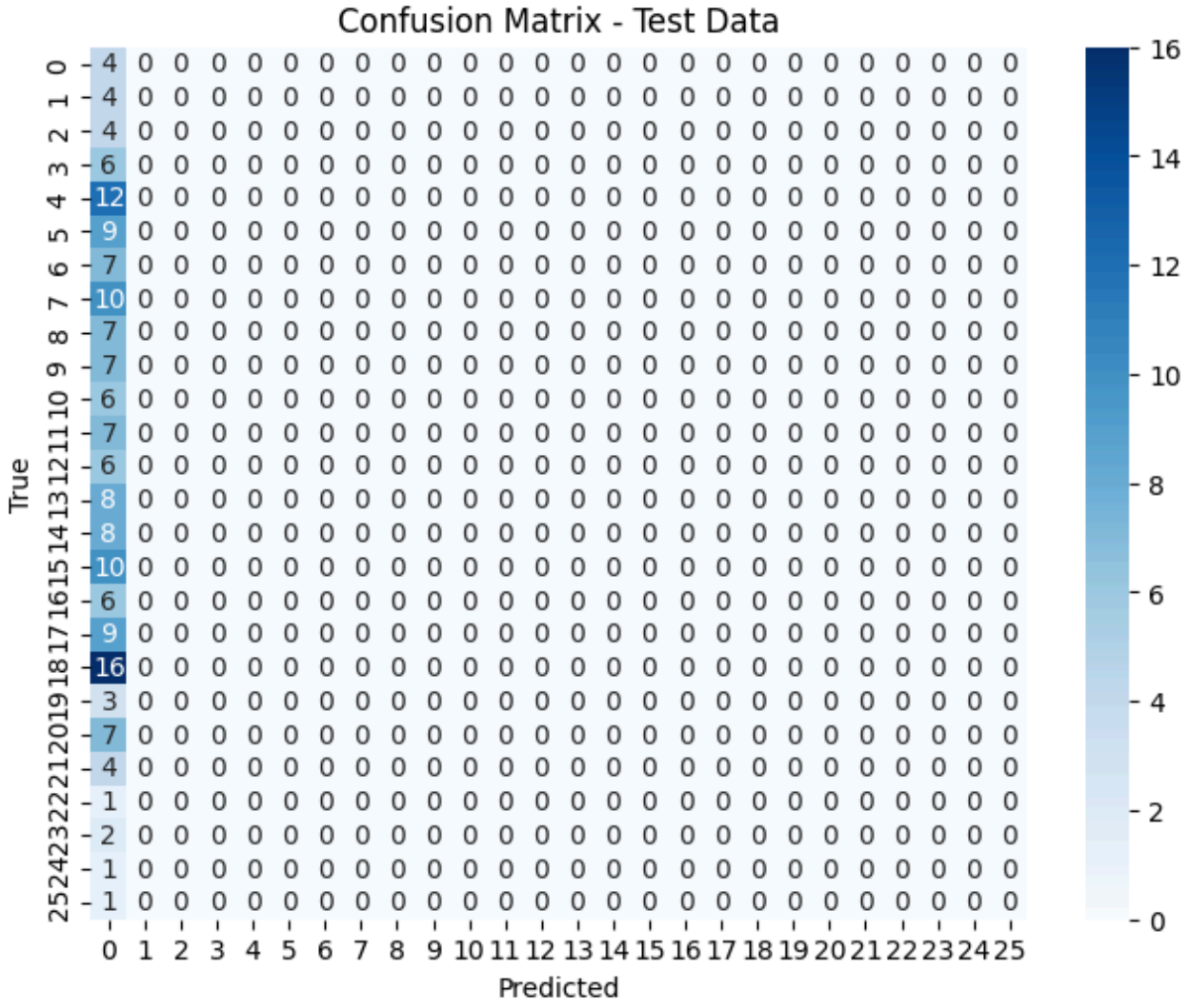
```
6/6 ━━━━━━━━━━━━━━━━━━━━ 5s 843ms/step
Shape y_test_true: (165,), y_test_pred: (165,)
Test Accuracy: 0.0242

Test Classification Report:
              precision    recall  f1-score   support

           0       0.02      1.00      0.05         4
           1       0.00      0.00      0.00         4
           2       0.00      0.00      0.00         4
           3       0.00      0.00      0.00         6
           4       0.00      0.00      0.00        12
           5       0.00      0.00      0.00         9
           6       0.00      0.00      0.00         7
           7       0.00      0.00      0.00        10
           8       0.00      0.00      0.00         7
           9       0.00      0.00      0.00         7
          10       0.00      0.00      0.00         6
          11       0.00      0.00      0.00         7
          12       0.00      0.00      0.00         6
          13       0.00      0.00      0.00         8
          14       0.00      0.00      0.00         8
          15       0.00      0.00      0.00        10
          16       0.00      0.00      0.00         6
          17       0.00      0.00      0.00         9
          18       0.00      0.00      0.00        16
          19       0.00      0.00      0.00         3
          20       0.00      0.00      0.00         7
          21       0.00      0.00      0.00         4
          22       0.00      0.00      0.00         1
          23       0.00      0.00      0.00         2
          24       0.00      0.00      0.00         1
          25       0.00      0.00      0.00         1

    accuracy                           0.02       165
   macro avg       0.00      0.04      0.00       165
weighted avg       0.00      0.02      0.00       165
```

```
e:\Software\Python\lib\site-packages\sklearn\metrics\_classification.py:1531: Undefi
nedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no pr
edicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
e:\Software\Python\lib\site-packages\sklearn\metrics\_classification.py:1531: Undefi
nedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no pr
edicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
e:\Software\Python\lib\site-packages\sklearn\metrics\_classification.py:1531: Undefi
nedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no pr
edicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```



In [ ]:
```
# Teks input langsung di dalam kode
input_text = """
Liputan6. com, Surabaya : Radiogram Direktorat Jenderal Pemerintahan Umum dan Otono
Fadilah Budiono diabaikan Gubernur Jawa Timur Imam Utomo. Kendati begitu, Imam buka
Penegasan hal itu disampaikan Imam, baru-baru ini, di Surabaya. Imam mengaku memang
radiogram Dirjen PUOD Depdagri pun sudah luluh lewat surat perintah Presiden. Seben
radiogram Dirjen Puod meluncur ke Imam. Namun lagi-lagi Imam berkeras. Menurut dia,
tingkat kabupaten. Namun sampai saat ini, tak ada penyelesaian baik dari DPRD Sampa
mewakili kiai se-Madura diprotes keras dari kiai unsur Partai Kebangkitan Bangsa. B
mewakili kiai asal Pulau Garam, namun hanya mewakili orang-orang PPP. ( BMI/Hasan S
"""
# Meringkas Teks
def summarize_text(input_text):
    # Proses teks input
    input_seq = tokenizer.texts_to_sequences([input_text])
    input_seq = pad_sequences(input_seq, maxlen=max_input_len, padding='post')

    # Prediksi output
    predicted_seq = model.predict(input_seq)
    predicted_seq = np.argmax(predicted_seq, axis=-1)[0]  # Ambil indeks dengan pro
```

```
    # Konversi output menjadi teks
    summary = ' '.join(tokenizer.index_word[idx] for idx in predicted_seq if idx >
    return summary

# Menampilkan ringkasan
ringkasan = summarize_text(input_text)
print("Ringkasan:", ringkasan)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 116ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 116ms/step
Ringkasan: sejumlah pemerintah
```