

TEXT SUMMARIZATION

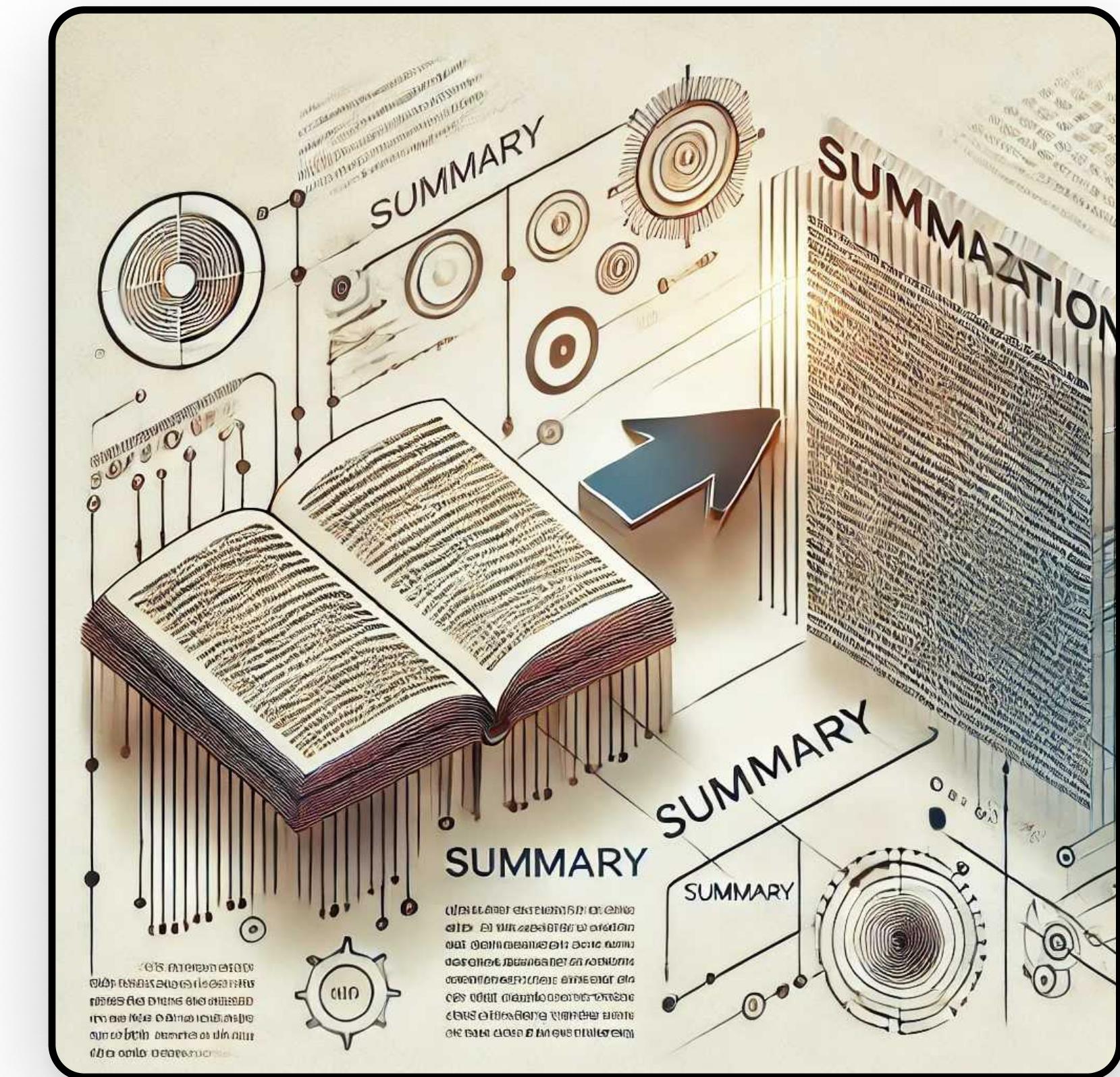
Artikel Berita Liputan 6

Defrfizal Yahdiyan Risyad
2206131
Ilmu Komputer C1

ABOUT PROJECT

Proyek ini adalah implementasi sederhana untuk membuat ringkasan otomatis menggunakan model Recurrent Neural Network (RNN). Model dilatih pada data berita untuk membuat ringkasan dari teks artikel. Proyek ini dijalankan di Google Colab dan memanfaatkan dataset berbentuk file CSV yang disimpan di Google Drive.

Ringkasan teks otomatis (text summarization) adalah teknik Natural Language Processing (NLP) untuk membuat versi pendek dari suatu teks sambil mempertahankan makna utamanya. Proyek ini menggunakan Simple RNN untuk menghasilkan ringkasan dari artikel berita.



ABOUT DATASET

Dataset Liputan6 adalah kumpulan data berskala besar yang berisi pasangan dokumen dan ringkasan artikel berita dalam Bahasa Indonesia. Dataset ini dirancang untuk mendukung model summarization baik dalam bentuk ekstraktif maupun abstraktif.

Ringkasan teks otomatis (text summarization) adalah teknik Natural Language Processing (NLP) untuk membuat versi pendek dari suatu teks sambil mempertahankan makna utamanya. Proyek ini menggunakan Simple RNN untuk menghasilkan ringkasan dari artikel berita.



ABOUT DATASET

Dataset ini dikembangkan oleh Fajri Koto et al. (2020) dan berisi lebih dari 215,827 pasangan dokumen-ringkasan yang diperoleh dari portal berita Liputan6.

The screenshot shows the Hugging Face dataset card for 'id_liputan6'. At the top, it displays basic information: Tasks (Summarization), Sub-tasks (news-articles-summarization), Languages (Indonesian), Size (100K<n<1M), ArXiv (arxiv:2011.00679), and Tags (extractive-summarization). Below this, there are tabs for 'Dataset card' (selected), 'Files and versions', and 'Community'. The 'Dataset Viewer' section is currently disabled. The 'Dataset Summary' section contains a detailed description of the dataset, mentioning its use for large-scale Indonesian summarization and its development over 215,827 document-summary pairs. It also notes the inclusion of error analysis and various summarization models. The bottom of the card indicates two variants: 'canonical' and 'xtreme', with the 'xtreme' variant being discarded for development and testing.

The screenshot shows the profile page for Fajri Koto, PhD, Assistant Professor at NLP Department, MBZUAI, United Arab Emirates. The page features a large circular portrait of Fajri Koto. His name is prominently displayed in bold text. Below his name, his title and affiliation are listed. A summary of his academic background and research interests is provided, mentioning his tenure track position, postdoctoral research at MBZUAI, and previous work at Samsung Research and Amazon Australia. His research interests include efficient multilingual NLP methods, dialogue systems, human-centric NLP, text and image generation, and computational social science and cultural analytics. The page also includes links to his publications, talks, teaching, research service, CV, and people.

DETAIL DATASET

STRUKTUR DATA

- Jenis Ringkasan: Ekstraktif dan Abstraktif
 - Bahasa: Indonesia
 - Ukuran Dataset: 16.5 MB (17,354,387 bytes)
 - Tugas yang Didukung: Text summarization, khususnya untuk artikel berita dalam Bahasa Indonesia
 - Referensi: arxiv:2011.00679

- id: ID unik untuk setiap artikel.
 - url: URL artikel asli di portal berita Liputan6.
 - clean_article: Artikel yang telah dibersihkan dari teks asli.
 - clean_summary: Ringkasan abstraktif dari artikel.

```
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
```

Info Dataset:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 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: 171.6+ KB
```

PRAPROCESSING

```
[ ] # 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

# Padding sequences untuk memastikan panjangnya seragam
input_sequences = pad_sequences(input_sequences, maxlen=max_input_len, padding='post')
target_sequences = pad_sequences(target_sequences, maxlen=max_target_len, padding='post')

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

MODEL FITTING

```
# Periksa panjang input dan target sequences yang sudah diproses
print("Panjang input sequences:", max_input_len)
print("Panjang target sequences:", max_target_len)

# Model dengan validasi
X_train, X_val, y_train, y_val = train_test_split(input_sequences, target_sequences, test_size=0.2, random_state=42)

# Menggunakan `max_input_len` untuk padding
y_train = pad_sequences(y_train, maxlen=max_input_len, padding='post')
y_val = pad_sequences(y_val, maxlen=max_input_len, padding='post')

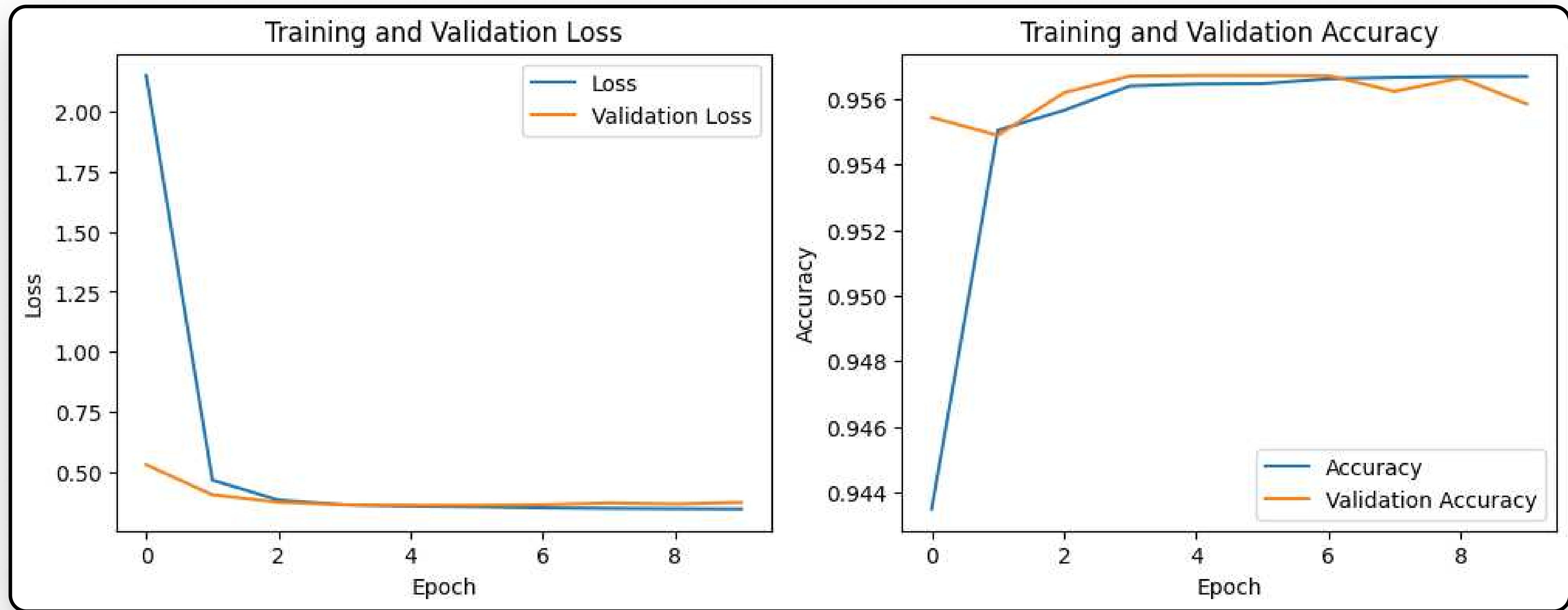
# Model
model = Sequential()
model.add(Embedding(input_dim=vocab_size, output_dim=64))
model.add(SimpleRNN(64, return_sequences=True))
model.add(TimeDistributed(Dense(vocab_size, activation='softmax')))
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Melatih model dengan validasi
history = model.fit(X_train, y_train, epochs=10, batch_size=32, validation_data=(X_val, y_val))
```

MODEL FITTING

```
Panjang input sequences: 500
Panjang target sequences: 50
Epoch 1/10
275/275 ━━━━━━━━ 803s 2s/step - accuracy: 0.8922 - loss: 4.8159 - val_accuracy: 0.9554 - val_loss: 0.5292
Epoch 2/10
275/275 ━━━━━━ 236s 777ms/step - accuracy: 0.9548 - loss: 0.5095 - val_accuracy: 0.9549 - val_loss: 0.4037
Epoch 3/10
275/275 ━━━━━━ 257s 759ms/step - accuracy: 0.9550 - loss: 0.3940 - val_accuracy: 0.9562 - val_loss: 0.3730
Epoch 4/10
275/275 ━━━━━━ 266s 775ms/step - accuracy: 0.9563 - loss: 0.3630 - val_accuracy: 0.9567 - val_loss: 0.3620
Epoch 5/10
275/275 ━━━━━━ 213s 775ms/step - accuracy: 0.9565 - loss: 0.3553 - val_accuracy: 0.9567 - val_loss: 0.3601
Epoch 6/10
275/275 ━━━━━━ 257s 759ms/step - accuracy: 0.9564 - loss: 0.3535 - val_accuracy: 0.9567 - val_loss: 0.3605
Epoch 7/10
275/275 ━━━━━━ 266s 775ms/step - accuracy: 0.9565 - loss: 0.3493 - val_accuracy: 0.9567 - val_loss: 0.3625
Epoch 8/10
275/275 ━━━━━━ 262s 775ms/step - accuracy: 0.9568 - loss: 0.3458 - val_accuracy: 0.9562 - val_loss: 0.3699
Epoch 9/10
275/275 ━━━━━━ 262s 775ms/step - accuracy: 0.9566 - loss: 0.3454 - val_accuracy: 0.9566 - val_loss: 0.3655
Epoch 10/10
275/275 ━━━━━━ 257s 759ms/step - accuracy: 0.9567 - loss: 0.3443 - val_accuracy: 0.9558 - val_loss: 0.3722
```

ACCURACY



IMPLEMENTASI MODEL UNTUK RINGKASAN TEKS OTOMATIS

```
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 probabilitas tertinggi

    # Konversi output menjadi teks
    summary = ' '.join(tokenizer.index_word[idx] for idx in predicted_seq if idx > 0)
    return summary

# Mengambil input dari pengguna dan menampilkan ringkasan
input_text = input("Masukkan teks yang ingin diringkas: ")
ringkasan = summarize_text(input_text)
print("Ringkasan:", ringkasan)
```

HASIL

Masukkan teks yang ingin diringkas: halo nama saya admin
1/1 ————— 0s 34ms/step

Ringkasan: pemerintah dan

TERIMA KASIH

