Chatbot Question-Answering Bahasa Indonesia dengan RNN

Deskripsi Project

Notebook ini membangun chatbot Question-Answering berbahasa Indonesia menggunakan **Recurrent Neural Network (RNN)** dengan teknik **Backpropagation Through Time (BPTT)**.

Dataset: IndoQA

- **Sumber**: Jakarta Artificial Intelligence Research
- **Jenis**: Question-Answering dataset monolingual bahasa Indonesia
- **Jumlah**: 4,413 contoh (training & validation)
- **Format**: Context paragraph + Question + Answer (extractive)

Arsitektur Model

- **RNN Encoder**: Untuk memproses context dan question
- **BPTT**: Backpropagation Through Time untuk training
- **Output Layer**: Prediksi posisi start dan end dari jawaban

---

1. Import Libraries

Import semua library yang diperlukan untuk preprocessing, modeling, dan evaluasi.

In [1]:
```
# Import library untuk data processing
import json  # untuk baca file JSON yang berisi dataset
import numpy as np  # untuk operasi array dan numerik
import pandas as pd  # untuk analisis data dalam bentuk tabel/dataframe
import re  # untuk regular expression (text processing)
from collections import Counter  # untuk menghitung frekuensi kata
import pickle  # untuk menyimpan objek python ke file
import os  # untuk operasi file system

# Import library Deep Learning dari TensorFlow/Keras
import tensorflow as tf  # framework deep learning utama
from tensorflow import keras  # API high-level untuk bikin model
from tensorflow.keras.models import Model  # untuk bikin model custom
from tensorflow.keras.layers import Input, Embedding, SimpleRNN, Dense, Concaten
from tensorflow.keras.preprocessing.text import Tokenizer  # untuk tokenisasi te
from tensorflow.keras.preprocessing.sequence import pad_sequences  # untuk paddi
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLRO
from tensorflow.keras.optimizers import Adam  # optimizer Adam untuk training

# Import library untuk visualisasi
import matplotlib.pyplot as plt  # untuk bikin plot grafik
import seaborn as sns  # untuk visualisasi yang lebih bagus
```

```
# Import utilities
from tqdm import tqdm  # untuk progress bar saat looping
import warnings  # untuk handle warning messages
warnings.filterwarnings('ignore')  # matikan warning biar output lebih bersih

# Set random seeds supaya hasil training bisa direproduksi
np.random.seed(42)  # seed untuk numpy random
tf.random.set_seed(42)  # seed untuk tensorflow random

# Cek versi TensorFlow dan ketersediaan GPU
print(f"TensorFlow Version: {tf.__version__}")
print(f"GPU Available: {tf.config.list_physical_devices('GPU')}")
```

```
TensorFlow Version: 2.10.1
GPU Available: [PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]
```

2. Load Dataset

Memuat dataset IndoQA dari file JSON yang telah didownload.

In [2]:
```
# Definisikan path ke file dataset
TRAIN_PATH = '../dataset/df_train.json'  # path untuk data training
VAL_PATH = '../dataset/df_val.json'  # path untuk data validation

# Load data training dari file JSON
print("Loading training data...")
with open(TRAIN_PATH, 'r', encoding='utf-8') as f:  # buka file dengan encoding
    train_data = json.load(f)  # parse JSON jadi list of dictionary

# Load data validation dari file JSON
print("Loading validation data...")
with open(VAL_PATH, 'r', encoding='utf-8') as f:  # buka file dengan encoding UT
    val_data = json.load(f)  # parse JSON jadi list of dictionary

# Tampilkan jumlah data yang berhasil dimuat
print(f"\nTotal Training Examples: {len(train_data)}")
print(f"Total Validation Examples: {len(val_data)}")

# Tampilkan contoh data pertama untuk melihat strukturnya
print("\nSample Training Data:")
print(f"ID: {train_data[0]['id']}")  # ID unik untuk setiap data
print(f"Question: {train_data[0]['question']}")  # pertanyaan yang diajukan
print(f"Answer: {train_data[0]['answer']}")  # jawaban yang benar
print(f"Context: {train_data[0]['context'][:200]}...")  # context (dipotong 200
print(f"Category: {train_data[0]['category']}")  # kategori dari data (SPAN, dll
```

```
Loading training data...
Loading validation data...

Total Training Examples: 3309
Total Validation Examples: 1104

Sample Training Data:
ID: 629HsNl5Qf3THUq_CbmFrSx28
Question: Dengan siapa Chaerul Saleh, Sukarni, Wikana, dan para pemuda pejuang be
rdiskusi?
Answer: Ibrahim gelar Datuk Tan Malaka
Context: Para pemuda pejuang, termasuk Chaerul Saleh, Sukarni, dan Wikana yang te
rbakar gelora kepahlawanannya setelah berdiskusi dengan Ibrahim gelar Datuk Tan M
alaka. Pada dini hari tanggal 16 Agustus 1945, ...
Category: SPAN
```

3. Exploratory Data Analysis (EDA)

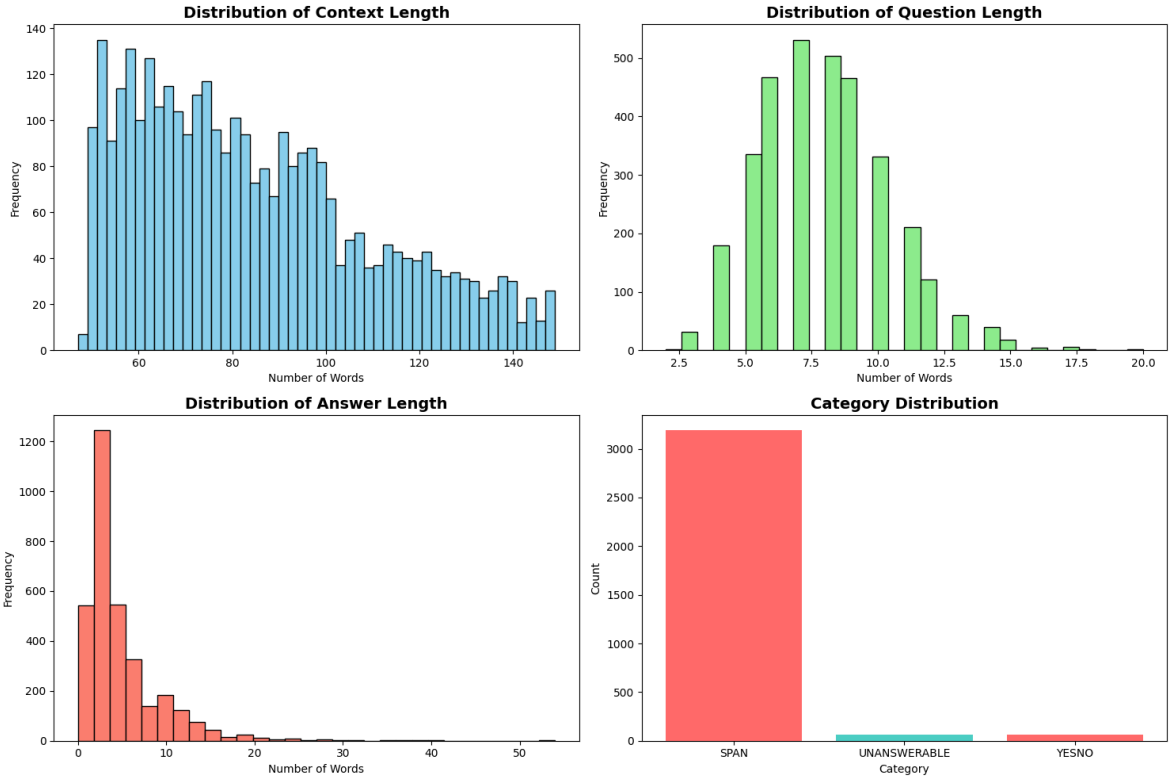
Analisis dataset untuk memahami karakteristik data.

In [3]:
```python
# Konversi data JSON jadi DataFrame supaya lebih mudah dianalisis
train_df = pd.DataFrame(train_data)  # bikin DataFrame dari list training data
val_df = pd.DataFrame(val_data)  # bikin DataFrame dari list validation data

# Tampilkan statistik dasar dari data training
print("TRAINING DATA STATISTICS")
print("="*50)
print(train_df.info())  # info tentang kolom, tipe data, dan missing values
print("\nCategory Distribution:")
print(train_df['category'].value_counts())  # hitung jumlah data per kategori

# Analisis panjang teks dalam kata
train_df['context_length'] = train_df['context'].apply(lambda x: len(x.split()))
train_df['question_length'] = train_df['question'].apply(lambda x: len(x.split()))
train_df['answer_length'] = train_df['answer'].apply(lambda x: len(str(x).split(

# Tampilkan statistik panjang teks
print("\nLength Statistics (in words):")
print(f"Context - Mean: {train_df['context_length'].mean():.2f}, Max: {train_df[
print(f"Question - Mean: {train_df['question_length'].mean():.2f}, Max: {train_d
print(f"Answer - Mean: {train_df['answer_length'].mean():.2f}, Max: {train_df['a
```

```
TRAINING DATA STATISTICS
==================================================
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3309 entries, 0 to 3308
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   id          3309 non-null   object
 1   context     3309 non-null   object
 2   question    3309 non-null   object
 3   answer      3250 non-null   object
 4   category    3309 non-null   object
 5   span_start  3309 non-null   int64 
 6   span_end    3309 non-null   int64 
dtypes: int64(2), object(5)
memory usage: 181.1+ KB
None

Category Distribution:
category
SPAN            3189
UNANSWERABLE      60
YESNO             60
Name: count, dtype: int64

Length Statistics (in words):
Context - Mean: 84.75, Max: 149
Question - Mean: 7.90, Max: 20
Answer - Mean: 4.82, Max: 54
```

In [4]:

```
# Visualisasi distribusi panjang teks menggunakan histogram
fig, axes = plt.subplots(2, 2, figsize=(15, 10))  # bikin 4 subplot dalam satu f

# Histogram distribusi panjang context
axes[0, 0].hist(train_df['context_length'], bins=50, color='skyblue', edgecolor=
axes[0, 0].set_title('Distribution of Context Length', fontsize=14, fontweight='
axes[0, 0].set_xlabel('Number of Words')  # label sumbu x
axes[0, 0].set_ylabel('Frequency')  # label sumbu y

# Histogram distribusi panjang question
axes[0, 1].hist(train_df['question_length'], bins=30, color='lightgreen', edgeco
axes[0, 1].set_title('Distribution of Question Length', fontsize=14, fontweight=
axes[0, 1].set_xlabel('Number of Words')
axes[0, 1].set_ylabel('Frequency')

# Histogram distribusi panjang answer
axes[1, 0].hist(train_df['answer_length'], bins=30, color='salmon', edgecolor='b
axes[1, 0].set_title('Distribution of Answer Length', fontsize=14, fontweight='b
axes[1, 0].set_xlabel('Number of Words')
axes[1, 0].set_ylabel('Frequency')

# Bar chart distribusi kategori
category_counts = train_df['category'].value_counts()  # hitung jumlah per kateg
axes[1, 1].bar(category_counts.index, category_counts.values, color=['#FF6B6B',
axes[1, 1].set_title('Category Distribution', fontsize=14, fontweight='bold')
axes[1, 1].set_xlabel('Category')
axes[1, 1].set_ylabel('Count')

plt.tight_layout()  # atur layout supaya tidak overlap
plt.show()  # tampilkan semua grafik
```



3.1 Analisis Statistik Dasar Dataset

Mari kita lihat informasi detail tentang dataset untuk memahami karakteristiknya.

In [5]:
```
# Analisis statistik lengkap untuk memahami dataset lebih dalam
print("="*70)
print("ANALISIS DATASET INDOQA")
print("="*70)

# 1. Tampilkan ukuran dataset
print(f"\n1. UKURAN DATASET")
print(f"   - Total training samples: {len(train_data):,}")  # format angka denga
print(f"   - Total validation samples: {len(val_data):,}")
print(f"   - Total keseluruhan: {len(train_data) + len(val_data):,}")

# 2. Analisis distribusi kategori di training set
print(f"\n2. DISTRIBUSI KATEGORI")
print(f"   Training:")
for category, count in train_df['category'].value_counts().items():  # Loop seti
    percentage = (count / len(train_df)) * 100  # hitung persentase
    print(f"   - {category}: {count:,} ({percentage:.2f}%)")

# Analisis distribusi kategori di validation set
print(f"\n   Validation:")
val_df = pd.DataFrame(val_data)  # konversi validation data ke DataFrame
for category, count in val_df['category'].value_counts().items():
    percentage = (count / len(val_df)) * 100
    print(f"   - {category}: {count:,} ({percentage:.2f}%)")

# 3. Statistik detail panjang context
print(f"\n3. STATISTIK PANJANG TEKS (dalam kata)")
print(f"   Context:")
print(f"   - Mean: {train_df['context_length'].mean():.2f}")  # rata-rata
print(f"   - Median: {train_df['context_length'].median():.0f}")  # nilai tengah
print(f"   - Std Dev: {train_df['context_length'].std():.2f}")  # standar devias
```

```
print(f"   - Min: {train_df['context_length'].min()}")  # nilai minimum
print(f"   - Max: {train_df['context_length'].max()}")  # nilai maksimum
print(f"   - Q1 (25%): {train_df['context_length'].quantile(0.25):.0f}")  # kuar
print(f"   - Q3 (75%): {train_df['context_length'].quantile(0.75):.0f}")  # kuar

# Statistik detail panjang question
print(f"\n   Question:")
print(f"   - Mean: {train_df['question_length'].mean():.2f}")
print(f"   - Median: {train_df['question_length'].median():.0f}")
print(f"   - Std Dev: {train_df['question_length'].std():.2f}")
print(f"   - Min: {train_df['question_length'].min()}")
print(f"   - Max: {train_df['question_length'].max()}")

# Statistik detail panjang answer
print(f"\n   Answer:")
print(f"   - Mean: {train_df['answer_length'].mean():.2f}")
print(f"   - Median: {train_df['answer_length'].median():.0f}")
print(f"   - Std Dev: {train_df['answer_length'].std():.2f}")
```

```
================================================================================
ANALISIS DATASET INDOQA
================================================================================

1. UKURAN DATASET
   - Total training samples: 3,309
   - Total validation samples: 1,104
   - Total keseluruhan: 4,413

2. DISTRIBUSI KATEGORI
   Training:
   - SPAN: 3,189 (96.37%)
   - UNANSWERABLE: 60 (1.81%)
   - YESNO: 60 (1.81%)

   Validation:
   - SPAN: 1,064 (96.38%)
   - UNANSWERABLE: 20 (1.81%)
   - YESNO: 20 (1.81%)

3. STATISTIK PANJANG TEKS (dalam kata)
   Context:
   - Mean: 84.75
   - Median: 80
   - Std Dev: 25.11
   - Min: 47
   - Max: 149
   - Q1 (25%): 64
   - Q3 (75%): 101

   Question:
   - Mean: 7.90
   - Median: 8
   - Std Dev: 2.43
   - Min: 2
   - Max: 20

   Answer:
   - Mean: 4.82
   - Median: 3
   - Std Dev: 4.49
```

3.2 Analisis Sample Data

Mari kita lihat beberapa contoh data untuk memahami struktur dan kontennya.

In [6]:
```
# Tampilkan beberapa contoh data untuk melihat struktur dan isinya
print("="*80)
print("CONTOH DATA DARI DATASET")
print("="*80)

# Ambil 3 contoh pertama dari kategori SPAN
span_examples = train_df[train_df['category'] == 'SPAN'].head(3)  # filter hanya
print("\n1. CONTOH KATEGORI SPAN (Extractive QA - Answer ada di dalam context)")
print("-"*80)

# Loop untuk tampilkan setiap contoh dengan detail
for idx, row in span_examples.iterrows():  # iterrows() untuk loop DataFrame by
    print(f"\nContoh {idx + 1}:")
    print(f"ID: {row['id']}")
    print(f"\nContext ({row['context_length']} kata):")
    # Potong context kalau terlalu panjang (lebih dari 300 karakter)
    print(f"{row['context'][:300]}..." if len(row['context']) > 300 else row['co
    print(f"\nQuestion ({row['question_length']} kata): {row['question']}")
    print(f"\nAnswer ({row['answer_length']} kata): {row['answer']}")
    print("-"*80)

# Cek apakah ada kategori lain selain SPAN
if len(train_df['category'].unique()) > 1:
    # Ambil nama kategori selain SPAN
    other_category = [cat for cat in train_df['category'].unique() if cat != 'SP
    other_examples = train_df[train_df['category'] == other_category].head(2)  #
    print(f"\n2. CONTOH KATEGORI {other_category}")
    print("-"*80)

    # Loop untuk tampilkan contoh dari kategori lain
    for idx, row in other_examples.iterrows():
        print(f"\nContoh:")
        print(f"ID: {row['id']}")
        print(f"\nContext: {row['context'][:200]}...")  # potong 200 karakter pe
        print(f"\nQuestion: {row['question']}")
        print(f"\nAnswer: {row['answer']}")
        print("-"*80)

print("\n" + "="*80)
```

```
================================================================================
CONTOH DATA DARI DATASET
================================================================================

1. CONTOH KATEGORI SPAN (Extractive QA - Answer ada di dalam context)
--------------------------------------------------------------------------------

Contoh 1:
ID: 629HsNl5Qf3THUq_CbmFrSx28

Context (91 kata):
Para pemuda pejuang, termasuk Chaerul Saleh, Sukarni, dan Wikana yang terbakar ge
lora kepahlawanannya setelah berdiskusi dengan Ibrahim gelar Datuk Tan Malaka. Pa
da dini hari tanggal 16 Agustus 1945, mereka bersama Shodanco Singgih, salah seor
ang anggota PETA, dan pemuda lain, membawa Soekarno (bers...

Question (11 kata): Dengan siapa Chaerul Saleh, Sukarni, Wikana, dan para pemuda
pejuang berdiskusi?

Answer (5 kata): Ibrahim gelar Datuk Tan Malaka
--------------------------------------------------------------------------------

Contoh 2:
ID: 2648F832Z8D-thJnrxy_ZoZ11111

Context (68 kata):
Malaysia terdiri atas dua kawasan utama yang terpisah oleh Laut Tiongkok Selatan.
Keduanya memiliki bentuk muka bumi yang hampir sama, yaitu dari pinggir laut yang
landai hingga hutan lebat dan bukit tinggi. Puncak tertinggi di Malaysia (dan jug
a di Kalimantan) yaitu Gunung Kinabalu setinggi 4.095,2...

Question (6 kata): Kapan angin muson timur laut berhembus?

Answer (3 kata): Oktober hingga Februari
--------------------------------------------------------------------------------

Contoh 3:
ID: 2561YDPNx6Nm5o6T0F-PGw-u2466

Context (100 kata):
Pada 16 Februari 1830, Diponegoro setuju untuk bertemu dengan utusan Jenderal De
Kock, yakni Kolonel Jan Baptist Clereens dan mengutus Kiai Pekih Ibrahim dan Haji
Badaruddin agar Clereens bisa datang ke Remo Kamal, Bagelen (sekarang masuk wilay
ah Kabupaten Purworejo), di hulu sungai Cingcingguling. ...

Question (10 kata): Siapa yang diutus oleh Jendral De Kock yang bertemu Diponegor
o?

Answer (4 kata): Kolonel Jan Baptist Clereens
--------------------------------------------------------------------------------

2. CONTOH KATEGORI UNANSWERABLE
--------------------------------------------------------------------------------

Contoh:
ID: 63292Pkbg0_0W_4375

Context: Presiden Indonesia (nama jabatan resmi: Presiden Republik Indonesia) ada
lah kepala negara sekaligus kepala pemerintahan Indonesia. Sebagai kepala negara,
Presiden adalah simbol resmi negara Indonesia ...
```

```
Question: Mengapa menteri tidak dipilih oleh rakyat tapi ditunjuk oleh presiden?

Answer: None
--------------------------------------------------------------------------------

Contoh:
ID: 1425958rgJUQgkcYQ4341

Context: Pada tahun 2017, BTS dipilih sebagai duta pariwisata terhormat untuk Seo
ul sebagai bagian dari program I Seoul U . Grup ini memfilmkan sebuah iklan yang
diproduksi oleh Seoul Metropolitan Government, ...

Question: Mengapa BTS dipilih sebagai duta pariwisata?

Answer: None
--------------------------------------------------------------------------------

================================================================================
```

3.3 Analisis Kata dan Vocabulary

Analisis vocabulary dan kata-kata yang paling sering muncul dalam dataset.

In [7]:

```python
# Analisis vocabulary dan kata-kata yang paling sering muncul
from collections import Counter  # import Counter untuk hitung frekuensi

# Kumpulkan semua kata dari context dan question
all_words = []  # list untuk tampung semua kata
for item in train_data:  # loop setiap data training
    if item.get('context'):  # cek apakah ada key 'context'
        all_words.extend(item['context'].lower().split())  # split jadi kata-kat
    if item.get('question'):  # cek apakah ada key 'question'
        all_words.extend(item['question'].lower().split())

# Hitung frekuensi setiap kata menggunakan Counter
word_freq = Counter(all_words)  # Counter otomatis hitung frekuensi setiap kata

print("="*70)
print("ANALISIS VOCABULARY")
print("="*70)

# Tampilkan statistik vocabulary
print(f"\nTotal kata (dengan duplikasi): {len(all_words):,}")  # total semua kat
print(f"Unique words (vocabulary size): {len(word_freq):,}")  # jumlah kata unik

# Tampilkan 20 kata paling sering muncul
print(f"\n20 Kata Paling Sering Muncul:")
print("-"*70)
for word, count in word_freq.most_common(20):  # most_common(n) untuk ambil n ka
    percentage = (count / len(all_words)) * 100  # hitung persentase kemunculan
    print(f"{word:20s} : {count:6,} kali ({percentage:.2f}%)")  # format output

# Analisis kata yang muncul di questions (pertanyaan)
question_words = []  # list untuk tampung kata-kata dari question
for item in train_data:
    if item.get('question'):
        question_words.extend(item['question'].lower().split())  # split dan low
```

```
question_freq = Counter(question_words)  # hitung frekuensi kata di question

# Tampilkan 20 kata paling sering di question
print(f"\n\n20 Kata Paling Sering di QUESTION:")
print("-"*70)
for word, count in question_freq.most_common(20):
    percentage = (count / len(question_words)) * 100
    print(f"{word:20s} : {count:6,} kali ({percentage:.2f}%)")

# Analisis distribusi panjang kata
print(f"\n\nDISTRIBUSI PANJANG KATA:")
print("-"*70)
word_lengths = [len(word) for word in word_freq.keys()]  # list comprehension un
print(f"Rata-rata panjang kata: {np.mean(word_lengths):.2f} karakter")  # hitung
print(f"Median panjang kata: {np.median(word_lengths):.0f} karakter")  # hitung
print(f"Kata terpendek: {min(word_lengths)} karakter")  # cari panjang minimum
print(f"Kata terpanjang: {max(word_lengths)} karakter")  # cari panjang maksimum

print("\n" + "="*70)
```

```
================================================================================
ANALISIS VOCABULARY
================================================================================

Total kata (dengan duplikasi): 306,560
Unique words (vocabulary size): 23,180

20 Kata Paling Sering Muncul:
--------------------------------------------------------------------------------
dan                  :  9,200 kali (3.00%)
yang                 :  8,949 kali (2.92%)
di                   :  6,949 kali (2.27%)
pada                 :  5,138 kali (1.68%)
dari                 :  3,436 kali (1.12%)
dengan               :  3,260 kali (1.06%)
untuk                :  2,434 kali (0.79%)
dalam                :  2,361 kali (0.77%)
tahun                :  2,065 kali (0.67%)
sebagai              :  2,007 kali (0.65%)
oleh                 :  1,984 kali (0.65%)
ini                  :  1,826 kali (0.60%)
adalah               :  1,521 kali (0.50%)
indonesia            :  1,458 kali (0.48%)
menjadi              :  1,433 kali (0.47%)
ke                   :  1,325 kali (0.43%)
negara               :  1,212 kali (0.40%)
tidak                :  1,118 kali (0.36%)
juga                 :  1,095 kali (0.36%)
apa                  :    996 kali (0.32%)


20 Kata Paling Sering di QUESTION:
--------------------------------------------------------------------------------
yang                 :  1,328 kali (5.08%)
apa                  :    942 kali (3.60%)
?                    :    783 kali (3.00%)
di                   :    563 kali (2.15%)
pada                 :    531 kali (2.03%)
dari                 :    372 kali (1.42%)
apakah               :    352 kali (1.35%)
berapa               :    314 kali (1.20%)
siapa                :    273 kali (1.04%)
mengapa              :    252 kali (0.96%)
nama                 :    242 kali (0.93%)
kapan                :    231 kali (0.88%)
oleh                 :    227 kali (0.87%)
siapakah             :    225 kali (0.86%)
dan                  :    199 kali (0.76%)
tahun                :    198 kali (0.76%)
dalam                :    161 kali (0.62%)
indonesia            :    144 kali (0.55%)
untuk                :    139 kali (0.53%)
menjadi              :    138 kali (0.53%)


DISTRIBUSI PANJANG KATA:
--------------------------------------------------------------------------------
Rata-rata panjang kata: 7.80 karakter
Median panjang kata: 8 karakter
Kata terpendek: 1 karakter
```

```
Kata terpanjang: 57 karakter

======================================================================
```

4. Data Preprocessing

4.1 Text Cleaning

Membersihkan dan menormalisasi teks.

In [8]:
```python
def clean_text(text):
    """
    Fungsi untuk membersihkan teks dari karakter khusus dan normalisasi format.
    """
    if not text:  # cek kalau text kosong/None
        return ""

    # Ubah semua huruf jadi lowercase (huruf kecil) supaya konsisten
    text = text.lower()

    # Hapus spasi berlebih (multiple whitespaces jadi satu spasi)
    text = ' '.join(text.split())  # split() hapus semua whitespace, join() gabu

    return text

# Filter data - hanya ambil kategori SPAN yang punya answer
print("Filtering and cleaning data...")
# List comprehension untuk filter data yang memenuhi kriteria
train_filtered = [item for item in train_data if item['category'] == 'SPAN' and
val_filtered = [item for item in val_data if item['category'] == 'SPAN' and item

# Tampilkan hasil filtering
print(f"Filtered Training Examples: {len(train_filtered)}")
print(f"Filtered Validation Examples: {len(val_filtered)}")

# Bersihkan semua teks di training data
for item in tqdm(train_filtered, desc="Cleaning training data"):  # tqdm untuk p
    item['context_clean'] = clean_text(item['context'])  # bersihkan context, si
    item['question_clean'] = clean_text(item['question'])  # bersihkan question
    item['answer_clean'] = clean_text(item['answer'])  # bersihkan answer

# Bersihkan semua teks di validation data
for item in tqdm(val_filtered, desc="Cleaning validation data"):
    item['context_clean'] = clean_text(item['context'])
    item['question_clean'] = clean_text(item['question'])
    item['answer_clean'] = clean_text(item['answer'])

print("\nText cleaning completed!")
```

```
Filtering and cleaning data...
Filtered Training Examples: 3189
Filtered Validation Examples: 1064
Cleaning training data: 100%|██████████| 3189/3189 [00:00<00:00, 97525.60it/s]
Cleaning validation data: 100%|██████████| 1064/1064 [00:00<00:00, 75575.60it/s]
Text cleaning completed!
```

4.2 Tokenization

Membuat vocabulary dan mengubah teks menjadi sequence of integers.

In [9]:
```
# Definisikan Hyperparameters - sudah di-optimize untuk training lebih cepat
MAX_VOCAB_SIZE = 15000  # ukuran vocabulary maksimal (lebih kecil = lebih cepat)
MAX_CONTEXT_LEN = 150   # panjang maksimal context dalam kata (dikurangi dari 20
MAX_QUESTION_LEN = 15   # panjang maksimal question dalam kata (dikurangi dari 2
EMBEDDING_DIM = 128     # dimensi embedding vector (dikurangi dari 256 untuk hem
RNN_UNITS = 128         # jumlah units di RNN layer (dikurangi dari 256)

# Tampilkan hyperparameters yang digunakan
print("HYPERPARAMETERS - OPTIMIZED FOR FASTER TRAINING")
print("="*60)
print(f"Vocab Size: {MAX_VOCAB_SIZE}")
print(f"Context Length: {MAX_CONTEXT_LEN} (shorter = faster)")  # lebih pendek =
print(f"Question Length: {MAX_QUESTION_LEN} (shorter = faster)")
print(f"Embedding Dim: {EMBEDDING_DIM}")
print(f"RNN Units: {RNN_UNITS}")
print("="*60)

# Kumpulkan semua teks untuk di-train oleh tokenizer
all_texts = []  # list untuk tampung semua teks
for item in train_filtered:  # loop semua data training yang sudah difilter
    all_texts.append(item['context_clean'])  # tambahkan context yang sudah bers
    all_texts.append(item['question_clean'])  # tambahkan question yang sudah be

# Buat tokenizer untuk convert teks jadi angka
tokenizer = Tokenizer(num_words=MAX_VOCAB_SIZE, oov_token='<OOV>')  # oov_token
tokenizer.fit_on_texts(all_texts)  # fit tokenizer pada semua teks (build vocabu
vocab_size = min(len(tokenizer.word_index) + 1, MAX_VOCAB_SIZE)  # +1 karena ind

print(f"Vocabulary Size: {vocab_size}")

# Simpan tokenizer ke file supaya bisa digunakan lagi nanti
os.makedirs('../Output Model RNN/', exist_ok=True)  # bikin folder kalau belum a
with open('../Output Model RNN/tokenizer.pickle', 'wb') as handle:  # buka file
    pickle.dump(tokenizer, handle, protocol=pickle.HIGHEST_PROTOCOL)  # simpan t
print("Tokenizer saved!")
```

```
HYPERPARAMETERS - OPTIMIZED FOR FASTER TRAINING
============================================================
Vocab Size: 15000
Context Length: 150 (shorter = faster)
Question Length: 15 (shorter = faster)
Embedding Dim: 128
RNN Units: 128
============================================================
Vocabulary Size: 14594
Tokenizer saved!
```

4.3 Create Training Data

Mengkonversi text menjadi sequences dan mencari posisi answer dalam context.

In [14]:
```
def find_answer_position(context_clean, answer_clean):
    """
    Fungsi untuk mencari posisi kata pertama dan terakhir dari answer dalam cont
    Returns: (start_word_idx, end_word_idx) - index kata di context
    """
```

```
    context_words = context_clean.split()  # split context jadi list kata-kata
    answer_words = answer_clean.split()  # split answer jadi list kata-kata

    if not answer_words:  # kalau answer kosong, return None
        return None, None

    # Cari exact match answer di dalam context
    for i in range(len(context_words) - len(answer_words) + 1):  # loop semua po
        # Bandingkan potongan context dengan answer
        if context_words[i:i+len(answer_words)] == answer_words:  # kalau match
            return i, i + len(answer_words) - 1  # return index start dan end

    # Kalau exact match tidak ketemu, coba fuzzy matching (pencarian partial)
    for i in range(len(context_words)):
        if answer_words[0] in context_words[i]:  # cari kata pertama answer di c
            # Anggap answer mulai dari sini
            end_idx = min(i + len(answer_words) - 1, len(context_words) - 1)  #
            return i, end_idx

    return None, None  # kalau tidak ketemu sama sekali

def prepare_data(data_list, tokenizer, max_context_len, max_question_len):
    """
    Fungsi untuk prepare data jadi format yang siap ditraining.
    Mengkonversi teks jadi sequences dan mencari posisi answer.
    """
    # Inisialisasi list untuk tampung hasil
    contexts = []
    questions = []
    start_positions = []  # posisi awal answer di context
    end_positions = []  # posisi akhir answer di context

    skipped = 0  # counter untuk data yang diskip

    for item in tqdm(data_list, desc="Preparing data"):  # loop semua data
        # Cari posisi answer di context
        start_pos, end_pos = find_answer_position(
            item['context_clean'],
            item['answer_clean']
        )

        # Skip kalau answer tidak ditemukan
        if start_pos is None or end_pos is None:
            skipped += 1
            continue

        # Skip kalau posisi answer melebihi max context length
        if start_pos >= max_context_len or end_pos >= max_context_len:
            skipped += 1
            continue

        # Kalau lolos validasi, tambahkan ke list
        contexts.append(item['context_clean'])
        questions.append(item['question_clean'])
        start_positions.append(start_pos)
        end_positions.append(end_pos)

    print(f"Skipped {skipped} examples (answer not found or out of bounds)")

    # Tokenize: convert teks jadi sequences of integers
```

```
    context_sequences = tokenizer.texts_to_sequences(contexts)  # context jadi a
    question_sequences = tokenizer.texts_to_sequences(questions)  # question jad

    # Padding: bikin semua sequences sama panjang dengan menambah 0 di akhir
    context_padded = pad_sequences(context_sequences, maxlen=max_context_len, pa
    question_padded = pad_sequences(question_sequences, maxlen=max_question_len,

    # Convert positions jadi numpy arrays supaya bisa dipakai di model
    start_positions = np.array(start_positions)
    end_positions = np.array(end_positions)

    return context_padded, question_padded, start_positions, end_positions

# Prepare training data menggunakan fungsi di atas
print("\nPreparing training data...")
X_context_train, X_question_train, y_start_train, y_end_train = prepare_data(
    train_filtered, tokenizer, MAX_CONTEXT_LEN, MAX_QUESTION_LEN
)

# Prepare validation data menggunakan fungsi yang sama
print("\nPreparing validation data...")
X_context_val, X_question_val, y_start_val, y_end_val = prepare_data(
    val_filtered, tokenizer, MAX_CONTEXT_LEN, MAX_QUESTION_LEN
)

# Tampilkan informasi data
print("\n" + "="*60)
print("DATA PREPARATION COMPLETED!")
print("="*60)
print(f"Training set:")
print(f"  - Context shape: {X_context_train.shape}")
print(f"  - Question shape: {X_question_train.shape}")
print(f"  - Start positions shape: {y_start_train.shape}")
print(f"  - End positions shape: {y_end_train.shape}")
print(f"\nValidation set:")
print(f"  - Context shape: {X_context_val.shape}")
print(f"  - Question shape: {X_question_val.shape}")
print(f"  - Start positions shape: {y_start_val.shape}")
print(f"  - End positions shape: {y_end_val.shape}")
print("="*60)
```

```
Preparing training data...
Preparing data: 100%|██████████| 3189/3189 [00:00<00:00, 91502.38it/s]
Skipped 0 examples (answer not found or out of bounds)

Preparing validation data...
Preparing data: 100%|██████████| 1064/1064 [00:00<00:00, 92480.51it/s]
```

```
Skipped 0 examples (answer not found or out of bounds)

============================================================
DATA PREPARATION COMPLETED!
============================================================
Training set:
  - Context shape: (3189, 150)
  - Question shape: (3189, 15)
  - Start positions shape: (3189,)
  - End positions shape: (3189,)

Validation set:
  - Context shape: (1064, 150)
  - Question shape: (1064, 15)
  - Start positions shape: (1064,)
  - End positions shape: (1064,)
============================================================
```

5. Build RNN Model dengan BPTT

Arsitektur Model:

1. **Embedding Layer**: Mengubah token menjadi dense vectors
2. **SimpleRNN Layer**: RNN murni untuk sequence processing (BPTT diterapkan otomatis)
3. **Concatenation**: Menggabungkan representasi context dan question
4. **Output Layer**: Dua output heads untuk prediksi start dan end position

BPTT (Backpropagation Through Time) adalah teknik training untuk RNN yang:

- Menghitung gradient melalui timesteps
- Mengupdate weights berdasarkan error dari setiap timestep
- Sudah terintegrasi dalam Keras/TensorFlow

In [15]:
```python
def build_rnn_qa_model(vocab_size, embedding_dim, rnn_units, max_context_len, ma
    """
    Fungsi untuk build model RNN Question Answering - versi yang di-optimize unt

    Arsitektur model:
    - Context Encoder: Embedding layer + SimpleRNN untuk process context
    - Question Encoder: Embedding layer + SimpleRNN untuk process question
    - Concatenation untuk gabungkan representasi context dan question
    - Two output heads: prediksi start_position dan end_position answer

    Optimizations yang diterapkan:
    - Pakai SimpleRNN unidirectional (bukan Bidirectional) supaya 2x lebih cepat
    - Kurangi jumlah units dan dimensi embedding
    - Minimal dropout supaya komputasi lebih cepat
    - Arsitektur yang disederhanakan
    """

    # Input layer untuk context
    context_input = Input(shape=(max_context_len,), name='context_input')  # sha
    # Embedding layer untuk convert integer jadi dense vector
    context_embedding = Embedding(
        input_dim=vocab_size,  # jumlah kata dalam vocabulary
```

```
        output_dim=embedding_dim,  # dimensi output embedding
        input_length=max_context_len,  # panjang input
        mask_zero=True,  # ignore padding (angka 0)
        name='context_embedding'
    )(context_input)

    # SimpleRNN untuk process context (unidirectional untuk speed, bukan Bidirec
    # BPTT (Backpropagation Through Time) otomatis diterapkan saat training
    context_rnn = SimpleRNN(
        rnn_units,  # jumlah units di RNN
        return_sequences=True,  # return output untuk setiap timestep
        name='context_rnn',
        activation='tanh',  # fungsi aktivasi tanh
        dropout=0.1  # minimal dropout (10%) untuk regularisasi
    )(context_embedding)

    # Input layer untuk question
    question_input = Input(shape=(max_question_len,), name='question_input')  #
    # Embedding layer untuk question (shared vocab dengan context)
    question_embedding = Embedding(
        input_dim=vocab_size,
        output_dim=embedding_dim,
        input_length=max_question_len,
        mask_zero=True,  # ignore padding
        name='question_embedding'
    )(question_input)

    # SimpleRNN untuk process question (unidirectional untuk speed)
    question_rnn = SimpleRNN(
        rnn_units,
        return_sequences=False,  # hanya return output terakhir
        name='question_rnn',
        activation='tanh',
        dropout=0.1
    )(question_embedding)

    # Expand question representation supaya bisa digabung dengan setiap timestep
    question_repeated = tf.keras.layers.RepeatVector(max_context_len)(question_r

    # Concatenate representasi context dan question
    merged = Concatenate(axis=-1)([context_rnn, question_repeated])  # gabung di

    # Dense layer untuk extract features (arsitektur simple = lebih cepat)
    dense_features = Dense(rnn_units, activation='relu', name='dense_features')(
    dense_features = Dropout(0.2)(dense_features)  # dropout 20% untuk regularis

    # Output layer untuk prediksi START position
    start_logits = Dense(1, activation='linear', name='start_logits')(dense_feat
    start_logits = tf.keras.layers.Flatten()(start_logits)  # flatten jadi 1D
    start_output = tf.keras.layers.Activation('softmax', name='start_output')(st

    # Output layer untuk prediksi END position
    end_logits = Dense(1, activation='linear', name='end_logits')(dense_features
    end_logits = tf.keras.layers.Flatten()(end_logits)
    end_output = tf.keras.layers.Activation('softmax', name='end_output')(end_lo

    # Buat model dengan functional API
    model = Model(
        inputs=[context_input, question_input],  # dua input: context dan questi
        outputs=[start_output, end_output],  # dua output: start position dan en
```

```
        name='RNN_QA_Model_Fast'
    )

    return model

# Build model dengan parameter yang sudah didefinisikan
print("Building SPEED-OPTIMIZED RNN QA Model...")
print("Optimizations:")
print("  - Unidirectional RNN (2x faster than Bidirectional)")  # satu arah, buk
print("  - Smaller units (128 instead of 256)")  # units lebih kecil
print("  - Shorter sequences (150/15 instead of 200/20)")  # sequence lebih pend
print("  - Minimal dropout")  # dropout seminimal mungkin
print("  - Simplified architecture")  # arsitektur yang disederhanakan

# Panggil fungsi build_rnn_qa_model dengan hyperparameters yang sudah ditentukan
model = build_rnn_qa_model(
    vocab_size=vocab_size,  # ukuran vocabulary
    embedding_dim=EMBEDDING_DIM,  # dimensi embedding (128)
    rnn_units=RNN_UNITS,  # jumlah RNN units (128)
    max_context_len=MAX_CONTEXT_LEN,  # max panjang context (150)
    max_question_len=MAX_QUESTION_LEN  # max panjang question (15)
)

# Tampilkan arsitektur model
model.summary()  # print summary model (layers, params, shapes)

print("\nExpected speed improvement: ~3-4x faster per epoch!")  # ekspektasi pen
```

```
Building SPEED-OPTIMIZED RNN QA Model...
Optimizations:
  - Unidirectional RNN (2x faster than Bidirectional)
  - Smaller units (128 instead of 256)
  - Shorter sequences (150/15 instead of 200/20)
  - Minimal dropout
  - Simplified architecture
Model: "RNN_QA_Model_Fast"
_________________________________________________________________________________
_________________
 Layer (type)                Output Shape              Param #     Connected to
==================================================================================
=================
 question_input (InputLayer)    [(None, 15)]         0           []

 context_input (InputLayer)     [(None, 150)]        0           []

 question_embedding (Embedding)  (None, 15, 128)     1868032     ['question_input
[0][0]']

 context_embedding (Embedding)  (None, 150, 128)     1868032     ['context_input
[0][0]']

 question_rnn (SimpleRNN)       (None, 128)          32896       ['question_embed
ding[0][0]']

 context_rnn (SimpleRNN)        (None, 150, 128)     32896       ['context_embedd
ing[0][0]']

 repeat_vector_1 (RepeatVector)  (None, 150, 128)    0           ['question_rnn
[0][0]']

 concatenate_1 (Concatenate)    (None, 150, 256)     0           ['context_rnn[0]
[0]',
                                                                  'repeat_vector_
1[0][0]']

 dense_features (Dense)         (None, 150, 128)     32896       ['concatenate_1
[0][0]']

 dropout_1 (Dropout)            (None, 150, 128)     0           ['dense_features
[0][0]']

 start_logits (Dense)           (None, 150, 1)       129         ['dropout_1[0]
[0]']

 end_logits (Dense)             (None, 150, 1)       129         ['dropout_1[0]
[0]']

 flatten_2 (Flatten)            (None, 150)          0           ['start_logits
[0][0]']

 flatten_3 (Flatten)            (None, 150)          0           ['end_logits[0]
[0]']

 start_output (Activation)      (None, 150)          0           ['flatten_2[0]
[0]']

 end_output (Activation)        (None, 150)          0           ['flatten_3[0]
```

```
================================================================================
================
Total params: 3,835,010
Trainable params: 3,835,010
Non-trainable params: 0
________________________________________________________________________________
________________

Expected speed improvement: ~3-4x faster per epoch!
```

6. Compile Model

Konfigurasi optimizer, loss function, dan metrics.

Catatan tentang BPTT:

- Loss function `sparse_categorical_crossentropy` akan menghitung error di setiap output
- Optimizer `Adam` akan menggunakan BPTT untuk menghitung gradient
- Gradient dipropagasi mundur melalui semua timesteps di RNN

In [16]:
```
# Prepare labels supaya bisa dipakai dengan sparse categorical crossentropy
# Sparse artinya label masih dalam bentuk integer index (bukan one-hot encoded)
y_start_train_cat = y_start_train  # label posisi start untuk training
y_end_train_cat = y_end_train  # label posisi end untuk training
y_start_val_cat = y_start_val  # label posisi start untuk validation
y_end_val_cat = y_end_val  # label posisi end untuk validation

# Setup optimizer Adam dengan learning rate yang tepat
optimizer = Adam(learning_rate=0.001)  # learning rate 0.001 untuk konvergensi y

# Compile model dengan konfigurasi loss, optimizer, dan metrics
model.compile(
    optimizer=optimizer,  # pakai Adam optimizer
    loss={
        # Sparse categorical crossentropy karena label berupa integer index
        'start_output': 'sparse_categorical_crossentropy',  # loss untuk prediks
        'end_output': 'sparse_categorical_crossentropy'  # loss untuk prediksi e
    },
    loss_weights={'start_output': 1.0, 'end_output': 1.0},  # kedua loss punya b
    metrics={
        # Track accuracy untuk masing-masing output
        'start_output': ['accuracy'],  # accuracy untuk start position
        'end_output': ['accuracy']  # accuracy untuk end position
    }
)

print("Model compiled successfully!")
print("Optimizer: Adam (lr=0.001)")
print("Loss: Sparse Categorical Crossentropy")
print("Metrics: Accuracy for both outputs")
print("\nBPTT (Backpropagation Through Time) akan otomatis diterapkan saat train
print("Gradient akan dipropagasi mundur melalui semua timesteps di RNN.")
```

```
Model compiled successfully!
Optimizer: Adam (lr=0.001)
Loss: Sparse Categorical Crossentropy
Metrics: Accuracy for both outputs

BPTT (Backpropagation Through Time) akan otomatis diterapkan saat training!
Gradient akan dipropagasi mundur melalui semua timesteps di RNN.
```

7. Training dengan BPTT

Training model dengan callbacks untuk:

- **ModelCheckpoint**: Menyimpan best model
- **EarlyStopping**: Stop training jika tidak ada improvement (PATIENCE=10 EPOCHS)
- **ReduceLROnPlateau**: Mengurangi learning rate jika stuck

OPTIMISASI UNTUK KECEPATAN:

Solusi Speed Up yang Diterapkan:

1. **Unidirectional RNN** (bukan Bidirectional) = **2x lebih cepat!**
2. **Batch Size**: 16 → 32 (fewer iterations per epoch)
3. **Sequence Length**: Context 200→150, Question 20→15 (25% reduction)
4. **Model Size**: Units 256→128, Embedding 256→128 (50% reduction)
5. **Minimal Dropout**: 0.2-0.3 → 0.1-0.2 (less computation)
6. **Simplified Architecture**: 1 RNN layer instead of 2
7. **Vocab Size**: 20000 → 15000 (smaller embedding matrix)

Trade-offs:

- **Speed**: ~3-4x lebih cepat (1-2 min)
- **Performance**: Sedikit lebih rendah tapi masih acceptable

BPTT dalam proses training:

1. Forward pass: Data mengalir melalui RNN timestep demi timestep
2. Calculate loss: Menghitung error antara prediksi dan ground truth
3. Backward pass (BPTT): Gradient dipropagasi mundur melalui timesteps
4. Update weights: Optimizer mengupdate semua parameters

Ekspektasi:

- **1-2 menit per epoch**
- Training 50 epochs = **~50-100 menit** total (acceptable!)
- Still mencegah early stopping dengan patience=10

In [17]:
```
# Buat direktori untuk simpan output model
output_dir = '../Output Model RNN/'  # path folder output
os.makedirs(output_dir, exist_ok=True)  # bikin folder kalau belum ada (exist_ok

llbacks untuk monitoring dan kontrol training
```

```python
# ModelCheckpoint untuk save model terbaik
checkpoint = ModelCheckpoint(
    filepath=output_dir + 'best_rnn_model.h5',  # path file untuk save model
    monitor='val_loss',  # monitor validation loss
    save_best_only=True,  # hanya save model kalau val_loss improve
    mode='min',  # mode min karena mau minimize loss
    verbose=1  # print message saat save model
)

# EarlyStopping untuk stop training kalau tidak ada improvement
early_stopping = EarlyStopping(
    monitor='val_loss',  # monitor validation loss
    patience=10,  # tunggu 10 epochs sebelum stop kalau tidak ada improvement
    restore_best_weights=True,  # restore ke weights terbaik saat stop
    verbose=1,  # print message saat trigger
    min_delta=0.001  # minimum perubahan yang dianggap improvement
)

# ReduceLROnPlateau untuk kurangi learning rate saat stuck
reduce_lr = ReduceLROnPlateau(
    monitor='val_loss',  # monitor validation loss
    factor=0.5,  # kalikan learning rate dengan 0.5
    patience=4,  # tunggu 4 epochs sebelum reduce lr
    min_lr=1e-6,  # minimum learning rate
    verbose=1  # print message saat reduce lr
)

# List semua callbacks yang akan dipakai
callbacks = [checkpoint, early_stopping, reduce_lr]

# Definisikan training parameters
BATCH_SIZE = 32  # ukuran batch (lebih besar = lebih cepat tapi butuh lebih bany
EPOCHS = 50  # maksimal 50 epochs (bisa stop lebih awal kalau early stopping tri

# Tampilkan info training configuration
print("Starting FAST training with BPTT...")
print("="*70)
print(f"SPEED OPTIMIZATIONS:")
print(f"  - Unidirectional RNN (NOT Bidirectional) = 2x faster")  # satu arah le
print(f"  - Batch Size: {BATCH_SIZE} (larger = fewer iterations)")  # batch besa
print(f"  - Shorter sequences: Context={MAX_CONTEXT_LEN}, Question={MAX_QUESTION
print(f"  - Smaller model: {RNN_UNITS} units, {EMBEDDING_DIM} embedding")  # mod
print(f"  - Minimal dropout for faster computation")  # dropout minimal
print("="*70)
print(f"Total Epochs: {EPOCHS}")
print(f"Early Stopping: Patience={10} epochs")  # akan stop kalau 10 epochs tida
print(f"Learning Rate: 0.001 (balanced)")
print("="*70)
print("Expected: ~1-2 minutes per epoch (vs 5 min before)")  # ekspektasi waktu
print("TIP: Training akan JAUH LEBIH CEPAT sekarang!")
print("="*70)

# Mulai training model dengan fit()
history = model.fit(
    [X_context_train, X_question_train],  # input training: context dan question
    [y_start_train_cat, y_end_train_cat],  # target training: start dan end posi
    validation_data=(
        [X_context_val, X_question_val],  # input validation
        [y_start_val_cat, y_end_val_cat]  # target validation
    )
```

```
    batch_size=BATCH_SIZE,  # ukuran batch
    epochs=EPOCHS,  # jumlah epochs maksimal
    callbacks=callbacks,  # callbacks yang sudah didefinisikan
    verbose=1  # print progress bar dan metrics
)

# Training selesai, tampilkan summary
print("\nTraining completed!")
print(f"Total epochs trained: {len(history.history['loss'])}")  # jumlah epochs
print(f"Best val_loss: {min(history.history['val_loss']):.4f}")  # validation lo
```

```
Starting FAST training with BPTT...
======================================================================
SPEED OPTIMIZATIONS:
  - Unidirectional RNN (NOT Bidirectional) = 2x faster
  - Batch Size: 32 (larger = fewer iterations)
  - Shorter sequences: Context=150, Question=15
  - Smaller model: 128 units, 128 embedding
  - Minimal dropout for faster computation
======================================================================
Total Epochs: 50
Early Stopping: Patience=10 epochs
Learning Rate: 0.001 (balanced)
======================================================================
Expected: ~1-2 minutes per epoch (vs 5 min before)
TIP: Training akan JAUH LEBIH CEPAT sekarang!
======================================================================
Epoch 1/50
100/100 [==============================] - ETA: 0s - loss: 9.8195 - start_output_
loss: 4.8677 - end_output_loss: 4.9517 - start_output_accuracy: 0.0191 - end_outp
ut_accuracy: 0.0141
Epoch 1: val_loss improved from inf to 9.74855, saving model to ../Output Model R
NN\best_rnn_model.h5
100/100 [==============================] - 34s 304ms/step - loss: 9.8195 - start_
output_loss: 4.8677 - end_output_loss: 4.9517 - start_output_accuracy: 0.0191 - e
nd_output_accuracy: 0.0141 - val_loss: 9.7485 - val_start_output_loss: 4.8389 - v
al_end_output_loss: 4.9096 - val_start_output_accuracy: 0.0216 - val_end_output_a
ccuracy: 0.0169 - lr: 0.0010
Epoch 2/50
100/100 [==============================] - ETA: 0s - loss: 9.0207 - start_output_
loss: 4.3888 - end_output_loss: 4.6319 - start_output_accuracy: 0.0712 - end_outp
ut_accuracy: 0.0502
Epoch 2: val_loss improved from 9.74855 to 9.45355, saving model to ../Output Mod
el RNN\best_rnn_model.h5
100/100 [==============================] - 30s 297ms/step - loss: 9.0207 - start_
output_loss: 4.3888 - end_output_loss: 4.6319 - start_output_accuracy: 0.0712 - e
nd_output_accuracy: 0.0502 - val_loss: 9.4535 - val_start_output_loss: 4.6417 - v
al_end_output_loss: 4.8118 - val_start_output_accuracy: 0.0470 - val_end_output_a
ccuracy: 0.0282 - lr: 0.0010
Epoch 3/50
100/100 [==============================] - ETA: 0s - loss: 6.6093 - start_output_
loss: 3.0775 - end_output_loss: 3.5317 - start_output_accuracy: 0.2126 - end_outp
ut_accuracy: 0.1615
Epoch 3: val_loss did not improve from 9.45355
100/100 [==============================] - 44s 440ms/step - loss: 6.6093 - start_
output_loss: 3.0775 - end_output_loss: 3.5317 - start_output_accuracy: 0.2126 - e
nd_output_accuracy: 0.1615 - val_loss: 10.5259 - val_start_output_loss: 5.3470 -
val_end_output_loss: 5.1789 - val_start_output_accuracy: 0.0301 - val_end_output_
accuracy: 0.0291 - lr: 0.0010
Epoch 4/50
100/100 [==============================] - ETA: 0s - loss: 4.1771 - start_output_
loss: 1.8870 - end_output_loss: 2.2900 - start_output_accuracy: 0.3415 - end_outp
ut_accuracy: 0.3007
Epoch 4: val_loss did not improve from 9.45355
100/100 [==============================] - 44s 442ms/step - loss: 4.1771 - start_
output_loss: 1.8870 - end_output_loss: 2.2900 - start_output_accuracy: 0.3415 - e
nd_output_accuracy: 0.3007 - val_loss: 13.8936 - val_start_output_loss: 7.1448 -
val_end_output_loss: 6.7488 - val_start_output_accuracy: 0.0226 - val_end_output_
accuracy: 0.0216 - lr: 0.0010
Epoch 5/50
100/100 [==============================] - ETA: 0s - loss: 2.9042 - start_output_
```

```
loss: 1.3583 - end_output_loss: 1.5459 - start_output_accuracy: 0.4255 - end_outp
ut_accuracy: 0.3913
Epoch 5: val_loss did not improve from 9.45355
100/100 [==============================] - 38s 376ms/step - loss: 2.9042 - start_
output_loss: 1.3583 - end_output_loss: 1.5459 - start_output_accuracy: 0.4255 - e
nd_output_accuracy: 0.3913 - val_loss: 16.1801 - val_start_output_loss: 8.1105 -
val_end_output_loss: 8.0697 - val_start_output_accuracy: 0.0263 - val_end_output_
accuracy: 0.0216 - lr: 0.0010
Epoch 6/50
100/100 [==============================] - ETA: 0s - loss: 2.2469 - start_output_
loss: 1.0531 - end_output_loss: 1.1938 - start_output_accuracy: 0.5347 - end_outp
ut_accuracy: 0.5052
Epoch 6: val_loss did not improve from 9.45355

Epoch 6: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
100/100 [==============================] - 30s 296ms/step - loss: 2.2469 - start_
output_loss: 1.0531 - end_output_loss: 1.1938 - start_output_accuracy: 0.5347 - e
nd_output_accuracy: 0.5052 - val_loss: 20.3827 - val_start_output_loss: 10.3429 -
val_end_output_loss: 10.0398 - val_start_output_accuracy: 0.0254 - val_end_output
_accuracy: 0.0226 - lr: 0.0010
Epoch 7/50
100/100 [==============================] - ETA: 0s - loss: 1.5396 - start_output_
loss: 0.7203 - end_output_loss: 0.8193 - start_output_accuracy: 0.6802 - end_outp
ut_accuracy: 0.6598
Epoch 7: val_loss did not improve from 9.45355
100/100 [==============================] - 28s 284ms/step - loss: 1.5396 - start_
output_loss: 0.7203 - end_output_loss: 0.8193 - start_output_accuracy: 0.6802 - e
nd_output_accuracy: 0.6598 - val_loss: 22.5251 - val_start_output_loss: 11.3993 -
val_end_output_loss: 11.1258 - val_start_output_accuracy: 0.0273 - val_end_output
_accuracy: 0.0244 - lr: 5.0000e-04
Epoch 8/50
100/100 [==============================] - ETA: 0s - loss: 1.1137 - start_output_
loss: 0.5155 - end_output_loss: 0.5981 - start_output_accuracy: 0.7783 - end_outp
ut_accuracy: 0.7680
Epoch 8: val_loss did not improve from 9.45355
100/100 [==============================] - 29s 293ms/step - loss: 1.1137 - start_
output_loss: 0.5155 - end_output_loss: 0.5981 - start_output_accuracy: 0.7783 - e
nd_output_accuracy: 0.7680 - val_loss: 25.1970 - val_start_output_loss: 12.4459 -
val_end_output_loss: 12.7511 - val_start_output_accuracy: 0.0273 - val_end_output
_accuracy: 0.0263 - lr: 5.0000e-04
Epoch 9/50
100/100 [==============================] - ETA: 0s - loss: 0.8813 - start_output_
loss: 0.3978 - end_output_loss: 0.4835 - start_output_accuracy: 0.8394 - end_outp
ut_accuracy: 0.8150
Epoch 9: val_loss did not improve from 9.45355
100/100 [==============================] - 29s 294ms/step - loss: 0.8813 - start_
output_loss: 0.3978 - end_output_loss: 0.4835 - start_output_accuracy: 0.8394 - e
nd_output_accuracy: 0.8150 - val_loss: 27.1725 - val_start_output_loss: 13.9039 -
val_end_output_loss: 13.2686 - val_start_output_accuracy: 0.0263 - val_end_output
_accuracy: 0.0254 - lr: 5.0000e-04
Epoch 10/50
100/100 [==============================] - ETA: 0s - loss: 0.7087 - start_output_
loss: 0.3180 - end_output_loss: 0.3907 - start_output_accuracy: 0.8780 - end_outp
ut_accuracy: 0.8652
Epoch 10: val_loss did not improve from 9.45355

Epoch 10: ReduceLROnPlateau reducing learning rate to 0.0002502500011874362.
100/100 [==============================] - 29s 292ms/step - loss: 0.7087 - start_
output_loss: 0.3180 - end_output_loss: 0.3907 - start_output_accuracy: 0.8780 - e
nd_output_accuracy: 0.8652 - val_loss: 29.8012 - val_start_output_loss: 15.1538 -
```

```
val_end_output_loss: 14.6474 - val_start_output_accuracy: 0.0273 - val_end_output
_accuracy: 0.0273 - lr: 5.0000e-04
Epoch 11/50
100/100 [==============================] - ETA: 0s - loss: 0.5377 - start_output_
loss: 0.2371 - end_output_loss: 0.3006 - start_output_accuracy: 0.9188 - end_outp
ut_accuracy: 0.9025
Epoch 11: val_loss did not improve from 9.45355
100/100 [==============================] - 29s 292ms/step - loss: 0.5377 - start_
output_loss: 0.2371 - end_output_loss: 0.3006 - start_output_accuracy: 0.9188 - e
nd_output_accuracy: 0.9025 - val_loss: 31.9625 - val_start_output_loss: 16.0227 -
val_end_output_loss: 15.9398 - val_start_output_accuracy: 0.0263 - val_end_output
_accuracy: 0.0273 - lr: 2.5000e-04
Epoch 12/50
100/100 [==============================] - ETA: 0s - loss: 0.4163 - start_output_
loss: 0.1797 - end_output_loss: 0.2366 - start_output_accuracy: 0.9407 - end_outp
ut_accuracy: 0.9320
Epoch 12: val_loss did not improve from 9.45355
Restoring model weights from the end of the best epoch: 2.
100/100 [==============================] - 29s 288ms/step - loss: 0.4163 - start_
output_loss: 0.1797 - end_output_loss: 0.2366 - start_output_accuracy: 0.9407 - e
nd_output_accuracy: 0.9320 - val_loss: 32.6914 - val_start_output_loss: 16.3178 -
val_end_output_loss: 16.3736 - val_start_output_accuracy: 0.0263 - val_end_output
_accuracy: 0.0263 - lr: 2.5000e-04
Epoch 12: early stopping

Training completed!
Total epochs trained: 12
Best val_loss: 9.4535
```

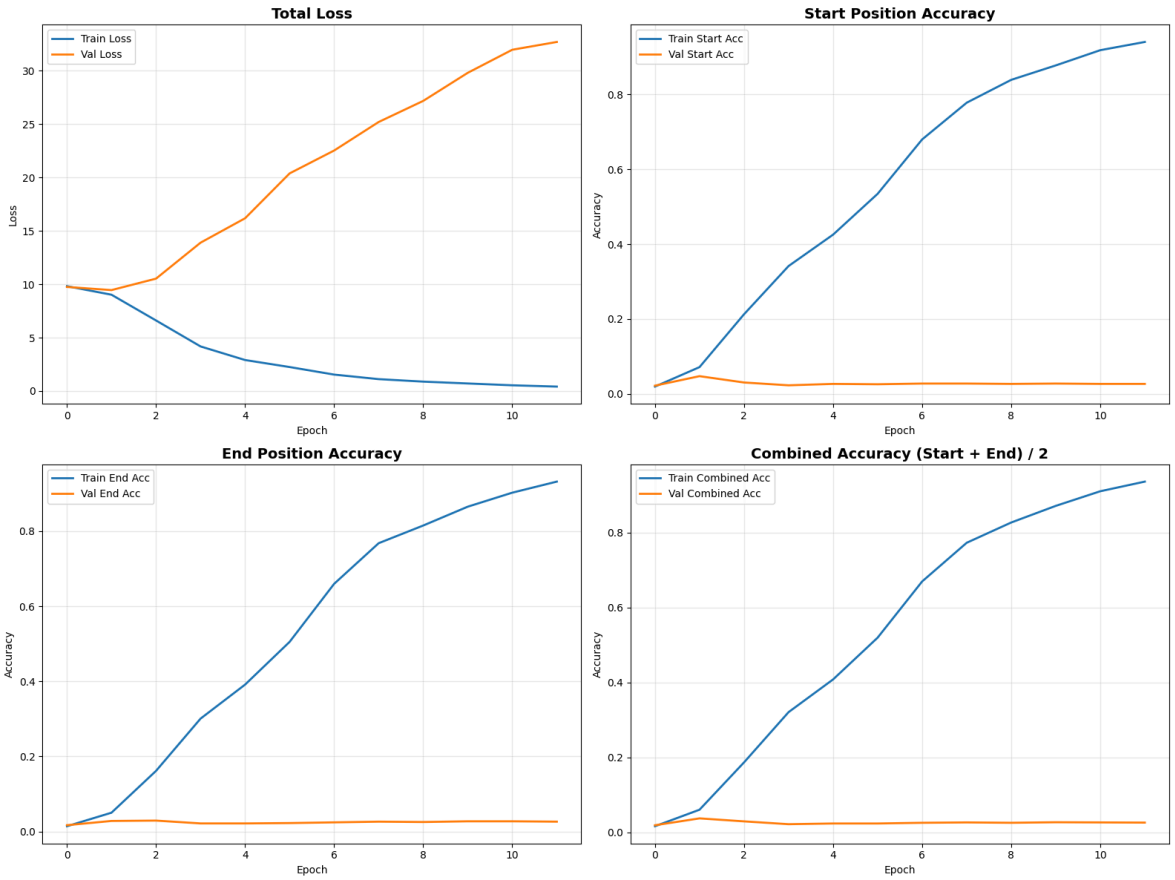
8. Visualize Training History

Melihat performa model selama training.

In [18]:

```python
# Visualisasi training history untuk melihat performa model selama training
fig, axes = plt.subplots(2, 2, figsize=(16, 12))  # bikin 4 subplot dalam grid 2

# Plot 1: Total Loss (training vs validation)
axes[0, 0].plot(history.history['loss'], label='Train Loss', linewidth=2)  # plo
axes[0, 0].plot(history.history['val_loss'], label='Val Loss', linewidth=2)  # p
axes[0, 0].set_title('Total Loss', fontsize=14, fontweight='bold')  # judul graf
axes[0, 0].set_xlabel('Epoch')  # label sumbu x
axes[0, 0].set_ylabel('Loss')  # label sumbu y
axes[0, 0].legend()  # tampilkan legend
axes[0, 0].grid(True, alpha=0.3)  # tambahkan grid dengan transparansi

# Plot 2: Start Position Accuracy
axes[0, 1].plot(history.history['start_output_accuracy'], label='Train Start Acc
axes[0, 1].plot(history.history['val_start_output_accuracy'], label='Val Start A
axes[0, 1].set_title('Start Position Accuracy', fontsize=14, fontweight='bold')
axes[0, 1].set_xlabel('Epoch')
axes[0, 1].set_ylabel('Accuracy')
axes[0, 1].legend()
axes[0, 1].grid(True, alpha=0.3)

# Plot 3: End Position Accuracy
axes[1, 0].plot(history.history['end_output_accuracy'], label='Train End Acc', l
axes[1, 0].plot(history.history['val_end_output_accuracy'], label='Val End Acc',
axes[1, 0].set_title('End Position Accuracy', fontsize=14, fontweight='bold')
```

```
axes[1, 0].set_xlabel('Epoch')
axes[1, 0].set_ylabel('Accuracy')
axes[1, 0].legend()
axes[1, 0].grid(True, alpha=0.3)

# Plot 4: Combined Accuracy (rata-rata start dan end)
# Hitung rata-rata accuracy start dan end untuk setiap epoch
train_combined = [(history.history['start_output_accuracy'][i] + history.history
                   for i in range(len(history.history['start_output_accuracy']))]
val_combined = [(history.history['val_start_output_accuracy'][i] + history.histo
                 for i in range(len(history.history['val_start_output_accuracy'])

axes[1, 1].plot(train_combined, label='Train Combined Acc', linewidth=2)  # plot
axes[1, 1].plot(val_combined, label='Val Combined Acc', linewidth=2)  # plot com
axes[1, 1].set_title('Combined Accuracy (Start + End) / 2', fontsize=14, fontwei
axes[1, 1].set_xlabel('Epoch')
axes[1, 1].set_ylabel('Accuracy')
axes[1, 1].legend()
axes[1, 1].grid(True, alpha=0.3)

plt.tight_layout()  # atur layout supaya tidak overlap
plt.savefig(output_dir + 'training_history.png', dpi=300, bbox_inches='tight')
plt.show()  # tampilkan grafik

print(f"Training history plot saved to {output_dir}training_history.png")
```




```
Training history plot saved to ../Output Model RNN/training_history.png
```

9. Save Model and Config

Menyimpan model, tokenizer, dan konfigurasi untuk inference nanti.

In [19]:
```
# Save model final ke file
model.save(output_dir + 'final_rnn_model.h5')  # save model dalam format HDF5
print(f"Final model saved to {output_dir}final_rnn_model.h5")

# Save konfigurasi model dalam dictionary
config = {
    'vocab_size': vocab_size,  # ukuran vocabulary
    'max_context_len': MAX_CONTEXT_LEN,  # panjang maksimal context
    'max_question_len': MAX_QUESTION_LEN,  # panjang maksimal question
    'embedding_dim': EMBEDDING_DIM,  # dimensi embedding
    'rnn_units': RNN_UNITS,  # jumlah RNN units
    'max_vocab_size': MAX_VOCAB_SIZE  # maksimal vocab size yang digunakan
}

# Simpan config ke file pickle
with open(output_dir + 'config.pickle', 'wb') as f:  # buka file dalam mode writ
    pickle.dump(config, f)  # dump dictionary config ke file

print(f"Configuration saved to {output_dir}config.pickle")

# Save training history untuk analisis nanti
with open(output_dir + 'history.pickle', 'wb') as f:
    pickle.dump(history.history, f)  # dump history dictionary (berisi loss dan

print(f"Training history saved to {output_dir}history.pickle")
print("\nAll artifacts saved successfully!")  # semua file sudah tersimpan
```

```
Final model saved to ../Output Model RNN/final_rnn_model.h5
Configuration saved to ../Output Model RNN/config.pickle
Training history saved to ../Output Model RNN/history.pickle

All artifacts saved successfully!
```

10. Model Evaluation

Evaluasi performa model pada validation set.

In [20]:
```
# Evaluasi performa model pada validation set
print("Evaluating model on validation set...")
# evaluate() menjalankan forward pass tanpa update weights
results = model.evaluate(
    [X_context_val, X_question_val],  # input validation
    [y_start_val_cat, y_end_val_cat],  # target validation
    batch_size=BATCH_SIZE,  # ukuran batch untuk evaluasi
    verbose=1  # print progress bar
)

# Tampilkan hasil evaluasi dengan format yang rapi
print("\n" + "="*50)
print("VALIDATION RESULTS")
print("="*50)
print(f"Total Loss: {results[0]:.4f}")  # total loss (start + end)
print(f"Start Loss: {results[1]:.4f}")  # loss untuk prediksi start position
print(f"End Loss: {results[2]:.4f}")  # loss untuk prediksi end position
print(f"Start Accuracy: {results[3]:.4f} ({results[3]*100:.2f}%)")  # accuracy s
print(f"End Accuracy: {results[4]:.4f} ({results[4]*100:.2f}%)")  # accuracy end
print(f"Combined Accuracy: {(results[3] + results[4])/2:.4f} ({(results[3] + res
print("="*50)
```

```
Evaluating model on validation set...
34/34 [==============================] - 1s 32ms/step - loss: 9.4535 - start_outp
ut_loss: 4.6417 - end_output_loss: 4.8118 - start_output_accuracy: 0.0470 - end_o
utput_accuracy: 0.0282

==================================================
VALIDATION RESULTS
==================================================
Total Loss: 9.4535
Start Loss: 4.6417
End Loss: 4.8118
Start Accuracy: 0.0470 (4.70%)
End Accuracy: 0.0282 (2.82%)
Combined Accuracy: 0.0376 (3.76%)
==================================================
```

11. Inference Function

Fungsi untuk menjawab pertanyaan berdasarkan context yang diberikan.

In [21]:

```python
def predict_answer(model, tokenizer, context, question, max_context_len, max_que
    """
    Fungsi untuk prediksi jawaban dari pertanyaan berdasarkan context yang diber

    Args:
        model: model RNN yang sudah di-train
        tokenizer: tokenizer untuk convert text jadi sequence
        context: teks context yang berisi informasi
        question: pertanyaan yang ingin dijawab
        max_context_len: panjang maksimal context
        max_question_len: panjang maksimal question

    Returns:
        answer: jawaban yang diprediksi (string)
        start_idx: index posisi start answer
        end_idx: index posisi end answer
        start_prob: confidence score untuk start position
        end_prob: confidence score untuk end position
    """
    # Bersihkan teks dulu
    context_clean = clean_text(context)  # clean context
    question_clean = clean_text(question)  # clean question

    # Tokenize: convert text jadi sequences of integers
    context_seq = tokenizer.texts_to_sequences([context_clean])  # context jadi
    question_seq = tokenizer.texts_to_sequences([question_clean])  # question ja

    # Padding: bikin semua sequences sama panjang
    context_padded = pad_sequences(context_seq, maxlen=max_context_len, padding=
    question_padded = pad_sequences(question_seq, maxlen=max_question_len, paddi

    # Prediksi menggunakan model (forward pass)
    start_probs, end_probs = model.predict([context_padded, question_padded], ve

    # Ambil index dengan probability tertinggi
    start_idx = np.argmax(start_probs[0])  # argmax untuk ambil index probabilit
    end_idx = np.argmax(end_probs[0])
```

```
    # Pastikan end_idx >= start_idx (logika answer harus start dulu baru end)
    if end_idx < start_idx:
        end_idx = start_idx  # set end sama dengan start kalau tidak masuk akal

    # Extract answer dari context berdasarkan index yang diprediksi
    context_words = context_clean.split()  # split context jadi list kata

    # Validasi index tidak out of bounds
    if start_idx < len(context_words) and end_idx < len(context_words):
        answer = ' '.join(context_words[start_idx:end_idx+1])  # ambil kata dari
    else:
        answer = "[Tidak dapat menemukan jawaban]"  # kalau index invalid

    return answer, start_idx, end_idx, start_probs[0][start_idx], end_probs[0][e

print("Inference function ready!")  # fungsi inference siap dipakai
```

```
Inference function ready!
```

12. Test Chatbot - Interactive QA

Test model dengan beberapa contoh dari validation set.

In [22]:
```
# Test chatbot dengan beberapa contoh random dari validation set
num_examples = 5  # jumlah contoh yang mau di-test
test_indices = np.random.choice(len(val_filtered), num_examples, replace=False)

print("CHATBOT QUESTION-ANSWERING - DEMO")
print("="*80)

# Loop untuk setiap contoh yang dipilih
for i, idx in enumerate(test_indices, 1):  # enumerate dimulai dari 1
    example = val_filtered[idx]  # ambil data dari validation set

    # Extract context, question, dan true answer
    context = example['context']
    question = example['question']
    true_answer = example['answer']

    # Prediksi jawaban menggunakan model
    pred_answer, start_idx, end_idx, start_conf, end_conf = predict_answer(
        model, tokenizer, context, question, MAX_CONTEXT_LEN, MAX_QUESTION_LEN
    )

    # Tampilkan hasil prediksi dengan format yang rapi
    print(f"\n{'='*80}")
    print(f"CONTOH {i}")
    print(f"{'='*80}")
    print(f"\nCONTEXT:")
    print(f"{context[:300]}..." if len(context) > 300 else context)  # potong ka
    print(f"\nQUESTION: {question}")
    print(f"\nTRUE ANSWER: {true_answer}")  # jawaban yang sebenarnya
    print(f"\nPREDICTED ANSWER: {pred_answer}")  # jawaban hasil prediksi model
    print(f"\nConfidence: Start={start_conf:.4f}, End={end_conf:.4f}")  # confid
    print(f"Position: [{start_idx}, {end_idx}]")  # posisi start dan end dalam k

    # Cek apakah jawaban benar (partial match)
    if pred_answer.lower().strip() in true_answer.lower().strip() or true_answer
        print("\nSTATUS: BENAR!")  # kalau ada overlap antara predicted dan true
```

```
    else:
        print("\nSTATUS: Tidak tepat")  # kalau tidak match

print(f"\n{'='*80}")
```

```
CHATBOT QUESTION-ANSWERING - DEMO
================================================================================

================================================================================
CONTOH 1
================================================================================

CONTEXT:
Menghadapi tuntutan untuk mundur, pada 1 Mei 1998, Soeharto menyatakan bahwa refo
rmasi akan dipersiapkan mulai tahun 2003. Ketika di Mesir pada 13 Mei 1998, Presi
den Soeharto menyatakan bersedia mundur kalau memang rakyat menghendaki dan tidak
akan mempertahankan kedudukannya dengan kekuatan senjata...

QUESTION: Berapa banyak mentri yang mengundurkan diri?

TRUE ANSWER: Sebelas menteri bidang ekonomi dan industri

PREDICTED ANSWER: 1 mei 1998,

Confidence: Start=0.1038, End=0.0686
Position: [5, 7]

STATUS: Tidak tepat

================================================================================
CONTOH 2
================================================================================

CONTEXT:
Kebebasan pers di Indonesia meningkat setelah berakhirnya kekuasaan Presiden Soeh
arto. Jaringan televisi publik TVRI bersaing dengan jaringan televisi swasta nasi
onal dan stasiun daerah; begitu pula dengan jaringan radio publik RRI yang bersai
ng dengan jaringan radio swasta yang menyiarkan berita da...

QUESTION: Apa dampak positif dari turunnya Presiden Soeharto dari jabatannya?

TRUE ANSWER: Kebebasan pers di Indonesia meningkat

PREDICTED ANSWER: tvri

Confidence: Start=0.2931, End=0.0694
Position: [13, 13]

STATUS: Tidak tepat

================================================================================
CONTOH 3
================================================================================

CONTEXT:
Pergerakan politik ia mulai sewaktu bersekolah di Belanda dari 1921–1932. Ia bers
ekolah di Handels Hogeschool (kelak sekolah ini disebut Economische Hogeschool ,
sekarang menjadi Universitas Erasmus Rotterdam), selama bersekolah di sana, ia ma
suk organisasi sosial Indische Vereeniging yang kemudian ...

QUESTION: Dimanakah Hatta bersekolah sewaktu di Belanda ?

TRUE ANSWER: Handels Hogeschool

PREDICTED ANSWER: dari 1921–1932.
```

```
Confidence: Start=0.0576, End=0.0844
Position: [8, 9]

STATUS: Tidak tepat

================================================================================
CONTOH 4
================================================================================

CONTEXT:
Dokumentasi pertama yang terkait dengan penggunaan frasa United States of America
terdapat dalam sebuah esai anonim yang diterbitkan dalam surat kabar Virginia Gaz
ette di Williamsburg, Virginia pada 6 April 1776. Pada bulan Juni 1776, Thomas Je
fferson menggunakan frasa UNITED STATES OF AMERICA denga...

QUESTION: Siapakah yang menggunakan frasa UNITED STATES OF AMERICA dalam judul ra
ncangan Deklarasi Kemerdekaan?

TRUE ANSWER: Thomas Jefferson

PREDICTED ANSWER: dalam sebuah esai anonim yang diterbitkan dalam surat kabar

Confidence: Start=0.0545, End=0.0455
Position: [12, 20]

STATUS: Tidak tepat

================================================================================
CONTOH 5
================================================================================

CONTEXT:
Tanggal pasti yang digunakan sebagai tanggal kelahiran kerajaan Majapahit adalah
hari penobatan Raden Wijaya sebagai raja, yaitu tanggal 15 bulan Kartika tahun 12
15 saka yang bertepatan dengan tanggal 10 November 1293. Ia dinobatkan dengan nam
a resmi Kertarajasa Jayawardhana. Kerajaan ini menghadapi...

QUESTION: Siapa yang melakukan pemberontakan terhadap Kertajasa?

TRUE ANSWER: Ranggalawe, Sora, dan Nambi

PREDICTED ANSWER: tanggal 15

Confidence: Start=0.1416, End=0.0570
Position: [17, 18]

STATUS: Tidak tepat

================================================================================
```

13. Custom Question Testing

Coba chatbot dengan pertanyaan custom Anda sendiri!

In [23]:
```python
# Fungsi untuk testing dengan custom context dan question
def test_custom_qa(context, question):
    """
```

```
    Test chatbot dengan custom context dan question yang kita tentukan sendiri.

    Args:
        context: teks paragraf yang berisi informasi
        question: pertanyaan yang ingin dijawab

    Returns:
        pred_answer: jawaban yang diprediksi oleh model
    """
    print("\n" + "="*80)
    print("CHATBOT QA - CUSTOM TEST")
    print("="*80)

    # Prediksi jawaban menggunakan fungsi predict_answer
    pred_answer, start_idx, end_idx, start_conf, end_conf = predict_answer(
        model, tokenizer, context, question, MAX_CONTEXT_LEN, MAX_QUESTION_LEN
    )

    # Tampilkan hasil dengan format yang rapi
    print(f"\nCONTEXT:")
    print(context)  # tampilkan full context
    print(f"\nQUESTION: {question}")
    print(f"\nJAWABAN: {pred_answer}")  # jawaban hasil prediksi
    print(f"\nConfidence: Start={start_conf:.4f}, End={end_conf:.4f}")  # confid
    print(f"Position: [{start_idx}, {end_idx}]")  # posisi kata dalam context
    print("="*80)

    return pred_answer  # return jawaban untuk digunakan lagi kalau perlu

# Contoh penggunaan 1: pertanyaan tentang kemerdekaan Indonesia
custom_context = """Soekarno dan Mohammad Hatta memproklamasikan kemerdekaan Ind
pada tanggal 17 Agustus 1945 di Jakarta. Peristiwa bersejarah ini menandai lahir
Republik Indonesia sebagai negara merdeka."""

custom_question = "Kapan Indonesia merdeka?"

# Test dengan pertanyaan pertama
answer = test_custom_qa(custom_context, custom_question)

# Contoh penggunaan 2: pertanyaan tentang proklamator
custom_question2 = "Siapa yang memproklamasikan kemerdekaan Indonesia?"
# Test dengan pertanyaan kedua menggunakan context yang sama
answer2 = test_custom_qa(custom_context, custom_question2)
```

```
================================================================================
CHATBOT QA - CUSTOM TEST
================================================================================

CONTEXT:
Soekarno dan Mohammad Hatta memproklamasikan kemerdekaan Indonesia
pada tanggal 17 Agustus 1945 di Jakarta. Peristiwa bersejarah ini menandai lahirn
ya
Republik Indonesia sebagai negara merdeka.

QUESTION: Kapan Indonesia merdeka?

JAWABAN: 17 agustus 1945

Confidence: Start=0.1605, End=0.1150
Position: [9, 11]
================================================================================

================================================================================
CHATBOT QA - CUSTOM TEST
================================================================================

CONTEXT:
Soekarno dan Mohammad Hatta memproklamasikan kemerdekaan Indonesia
pada tanggal 17 Agustus 1945 di Jakarta. Peristiwa bersejarah ini menandai lahirn
ya
Republik Indonesia sebagai negara merdeka.

QUESTION: Siapa yang memproklamasikan kemerdekaan Indonesia?

JAWABAN: 17 agustus 1945

Confidence: Start=0.1651, End=0.1110
Position: [9, 11]
================================================================================
```

14. Interactive Chatbot - Input Manual

Gunakan chatbot secara interaktif! Masukkan context dan question Anda sendiri.

In [24]:
```python
def interactive_chatbot():
    """
    Fungsi chatbot interaktif yang menerima input context dan question langsung
    User bisa bertanya berkali-kali sampai memilih untuk keluar.
    """
    # Tampilkan header dan instruksi penggunaan
    print("="*80)
    print("CHATBOT QA INTERAKTIF - BAHASA INDONESIA")
    print("="*80)
    print("\nTips:")
    print("  - Masukkan context (paragraf yang berisi informasi)")
    print("  - Masukkan question (pertanyaan tentang context)")
    print("  - Ketik 'exit' atau 'quit' untuk keluar")
    print("  - Ketik 'contoh' untuk melihat contoh")
    print("="*80)

    # Loop utama chatbot - akan terus berjalan sampai user keluar
    while True:
```

```
        print("\n" + "-"*80)

        # Input context dari user (bisa multi-line)
        print("\nMasukkan CONTEXT (paragraf/teks):")
        print("(Tekan Enter 2x untuk selesai, atau ketik 'exit' untuk keluar)")

        context_lines = []  # list untuk tampung baris-baris context
        while True:  # inner loop untuk input multi-line
            line = input()  # baca satu baris
            if line.lower() in ['exit', 'quit']:  # cek kalau user mau keluar
                print("\nTerima kasih telah menggunakan chatbot!")
                return  # keluar dari fungsi
            elif line.lower() == 'contoh':  # kalau user mau lihat contoh
                show_examples()  # panggil fungsi show_examples
                break  # keluar dari inner loop
            elif line == '' and len(context_lines) > 0:  # kalau enter kosong da
                break  # selesai input context
            elif line != '':  # kalau line tidak kosong
                context_lines.append(line)  # tambahkan ke list

        if line.lower() == 'contoh':  # kalau tadi pilih contoh
            continue  # mulai dari awal loop

        context = ' '.join(context_lines)  # gabungkan semua baris jadi satu str

        # Validasi context tidak kosong
        if not context.strip():
            print("Context tidak boleh kosong!")
            continue  # kembali ke awal loop

        # Input question dari user (single line)
        print("\nMasukkan QUESTION (pertanyaan):")
        question = input().strip()  # baca question dan trim whitespace

        # Cek kalau user mau keluar
        if question.lower() in ['exit', 'quit']:
            print("\nTerima kasih telah menggunakan chatbot!")
            return

        # Validasi question tidak kosong
        if not question:
            print("Question tidak boleh kosong!")
            continue

        # Tampilkan status processing
        print("\n" + "="*80)
        print("MEMPROSES...")
        print("="*80)

        # Try-except untuk handle error saat prediksi
        try:
            # Prediksi jawaban menggunakan model
            pred_answer, start_idx, end_idx, start_conf, end_conf = predict_answ
                model, tokenizer, context, question, MAX_CONTEXT_LEN, MAX_QUESTI
            )

            # Tampilkan hasil prediksi dengan format yang rapi
            print(f"\nCONTEXT:")
            print(f"{context}")
            print(f"\nQUESTION: {question}")
```

```
            print(f"\nJAWABAN: {pred_answer}")
            print(f"\nConfidence Score:")
            print(f"   Start Position: {start_conf:.4f} ({start_conf*100:.2f}%)"
            print(f"   End Position: {end_conf:.4f} ({end_conf*100:.2f}%)")  # c
            print(f"   Average: {(start_conf + end_conf)/2:.4f} ({(start_conf +
            print(f"\nPosition in Context: Word {start_idx} to {end_idx}")

            # Interpretasi confidence score untuk user
            avg_conf = (start_conf + end_conf) / 2  # hitung rata-rata confidenc
            if avg_conf > 0.7:  # kalau confidence tinggi
                print("\nConfidence: TINGGI - Jawaban kemungkinan besar benar!")
            elif avg_conf > 0.4:  # kalau confidence sedang
                print("\nConfidence: SEDANG - Jawaban mungkin benar, periksa kem
            else:  # kalau confidence rendah
                print("\nConfidence: RENDAH - Jawaban mungkin tidak akurat")

        except Exception as e:  # kalau ada error
            print(f"\nError: {str(e)}")
            print("Silakan coba lagi dengan context/question yang berbeda.")

        # Tanya user apakah mau lanjut atau keluar
        print("\n" + "="*80)
        print("\nIngin bertanya lagi? (Y/N)")
        continue_choice = input().strip().lower()  # baca pilihan user
        if continue_choice in ['n', 'no', 'tidak', 'exit', 'quit']:  # kalau tid
            print("\nTerima kasih telah menggunakan chatbot!")
            break  # keluar dari loop utama

def show_examples():
    """
    Fungsi helper untuk menampilkan contoh-contoh penggunaan chatbot.
    """
    print("\n" + "="*80)
    print("CONTOH-CONTOH PENGGUNAAN")
    print("="*80)

    # List contoh context dan questions
    examples = [
        {
            "context": "Proklamasi kemerdekaan Indonesia dibacakan oleh Soekarno
            "questions": [
                "Kapan Indonesia merdeka?",
                "Siapa yang membacakan proklamasi?",
                "Di mana proklamasi dibacakan?"
            ]
        },
        {
            "context": "Candi Borobudur merupakan candi Buddha terbesar di dunia
            "questions": [
                "Di mana letak Candi Borobudur?",
                "Kapan Candi Borobudur dibangun?",
                "Berapa jumlah arca Buddha di Borobudur?"
            ]
        },
        {
            "context": "Sumpah Pemuda dibacakan pada tanggal 28 Oktober 1928 di
            "questions": [
                "Kapan Sumpah Pemuda dibacakan?",
                "Apa isi Sumpah Pemuda?",
                "Di mana Sumpah Pemuda dibacakan?"
```

```
            ]
        }
    ]

    # Loop untuk tampilkan setiap contoh
    for i, example in enumerate(examples, 1):  # enumerate dimulai dari 1
        print(f"\n{'-'*80}")
        print(f"CONTOH {i}:")
        print(f"{'-'*80}")
        print(f"\nContext:")
        print(f"{example['context']}")
        print(f"\nContoh Questions:")
        for j, q in enumerate(example['questions'], 1):  # loop questions dengan
            print(f"   {j}. {q}")

    print(f"\n{'='*80}")
    print("Silakan copy-paste contoh di atas atau buat sendiri!")
    print(f"{'='*80}\n")

# Tampilkan informasi cara penggunaan
print("Fungsi interactive_chatbot() telah siap!")
print("\nCara menggunakan:")
print("   1. Jalankan: interactive_chatbot()")
print("   2. Masukkan context (paragraf)")
print("   3. Tekan Enter 2x untuk selesai input context")
print("   4. Masukkan question")
print("   5. Lihat jawaban dari chatbot!")
print("\nSIAP DIGUNAKAN! Jalankan cell berikutnya untuk mulai!")
```

```
Fungsi interactive_chatbot() telah siap!

Cara menggunakan:
   1. Jalankan: interactive_chatbot()
   2. Masukkan context (paragraf)
   3. Tekan Enter 2x untuk selesai input context
   4. Masukkan question
   5. Lihat jawaban dari chatbot!

SIAP DIGUNAKAN! Jalankan cell berikutnya untuk mulai!
```

Jalankan Chatbot Interaktif

Jalankan cell di bawah ini untuk memulai chatbot interaktif!

In [ ]:
```
# JALANKAN CELL INI UNTUK MEMULAI CHATBOT INTERAKTIF
# Cell ini akan memanggil fungsi interactive_chatbot() yang sudah didefinisikan
interactive_chatbot()  # panggil fungsi untuk mulai sesi chatbot interaktif
```

```
================================================================================
CHATBOT QA INTERAKTIF - BAHASA INDONESIA
================================================================================

Tips:
  - Masukkan context (paragraf yang berisi informasi)
  - Masukkan question (pertanyaan tentang context)
  - Ketik 'exit' atau 'quit' untuk keluar
  - Ketik 'contoh' untuk melihat contoh
================================================================================

________________________________________________________________________________


Masukkan CONTEXT (paragraf/teks):
(Tekan Enter 2x untuk selesai, atau ketik 'exit' untuk keluar)

Masukkan QUESTION (pertanyaan):

Masukkan QUESTION (pertanyaan):

================================================================================
MEMPROSES...
================================================================================

CONTEXT:
Pada tanggal 17 Agustus 1945, Indonesia memproklamasikan kemerdekaannya setelah b
erabad-abad dijajah oleh bangsa asing, terutama Belanda dan Jepang. Proklamasi ke
merdekaan dibacakan oleh Soekarno dan Hatta di Jalan Pegangsaan Timur No. 56, Jak
arta. Setelah proklamasi, dibentuk Komite Nasional Indonesia Pusat (KNIP) sebagai
lembaga sementara yang membantu tugas presiden. Pada 18 Agustus 1945, Panitia Per
siapan Kemerdekaan Indonesia (PPKI) mengesahkan Undang-Undang Dasar 1945 dan mene
tapkan Soekarno sebagai presiden serta Mohammad Hatta sebagai wakil presiden pert
ama Republik Indonesia.

QUESTION: Kapan Indonesia memproklamasikan kemerdekaannya?

JAWABAN: 17 agustus 1945,

Confidence Score:
   Start Position: 0.1140 (11.40%)
   End Position: 0.1159 (11.59%)
   Average: 0.1150 (11.50%)

Position in Context: Word 2 to 4

Confidence: RENDAH - Jawaban mungkin tidak akurat

================================================================================

Ingin bertanya lagi? (Y/N)

================================================================================
MEMPROSES...
================================================================================

CONTEXT:
Pada tanggal 17 Agustus 1945, Indonesia memproklamasikan kemerdekaannya setelah b
erabad-abad dijajah oleh bangsa asing, terutama Belanda dan Jepang. Proklamasi ke
merdekaan dibacakan oleh Soekarno dan Hatta di Jalan Pegangsaan Timur No. 56, Jak
arta. Setelah proklamasi, dibentuk Komite Nasional Indonesia Pusat (KNIP) sebagai
```

```
lembaga sementara yang membantu tugas presiden. Pada 18 Agustus 1945, Panitia Per
siapan Kemerdekaan Indonesia (PPKI) mengesahkan Undang-Undang Dasar 1945 dan mene
tapkan Soekarno sebagai presiden serta Mohammad Hatta sebagai wakil presiden pert
ama Republik Indonesia.

QUESTION: Kapan Indonesia memproklamasikan kemerdekaannya?

JAWABAN: 17 agustus 1945,

Confidence Score:
   Start Position: 0.1140 (11.40%)
   End Position: 0.1159 (11.59%)
   Average: 0.1150 (11.50%)

Position in Context: Word 2 to 4

Confidence: RENDAH - Jawaban mungkin tidak akurat

================================================================================

Ingin bertanya lagi? (Y/N)

________________________________________________________________________________


Masukkan CONTEXT (paragraf/teks):
(Tekan Enter 2x untuk selesai, atau ketik 'exit' untuk keluar)

________________________________________________________________________________


Masukkan CONTEXT (paragraf/teks):
(Tekan Enter 2x untuk selesai, atau ketik 'exit' untuk keluar)

Masukkan QUESTION (pertanyaan):

Masukkan QUESTION (pertanyaan):

================================================================================
MEMPROSES...
================================================================================

CONTEXT:
Pada tanggal 17 Agustus 1945, Indonesia memproklamasikan kemerdekaannya setelah b
erabad-abad dijajah oleh bangsa asing, terutama Belanda dan Jepang. Proklamasi ke
merdekaan dibacakan oleh Soekarno dan Hatta di Jalan Pegangsaan Timur No. 56, Jak
arta. Setelah proklamasi, dibentuk Komite Nasional Indonesia Pusat (KNIP) sebagai
lembaga sementara yang membantu tugas presiden. Pada 18 Agustus 1945, Panitia Per
siapan Kemerdekaan Indonesia (PPKI) mengesahkan Undang-Undang Dasar 1945 dan mene
tapkan Soekarno sebagai presiden serta Mohammad Hatta sebagai wakil presiden pert
ama Republik Indonesia.

QUESTION: Siapa yang membacakan teks proklamasi kemerdekaan Indonesia?

JAWABAN: 17 agustus 1945,

Confidence Score:
   Start Position: 0.1155 (11.55%)
   End Position: 0.1204 (12.04%)
   Average: 0.1180 (11.80%)

Position in Context: Word 2 to 4
```

```
Confidence: RENDAH - Jawaban mungkin tidak akurat

================================================================================

Ingin bertanya lagi? (Y/N)

================================================================================
MEMPROSES...
================================================================================

CONTEXT:
Pada tanggal 17 Agustus 1945, Indonesia memproklamasikan kemerdekaannya setelah b
erabad-abad dijajah oleh bangsa asing, terutama Belanda dan Jepang. Proklamasi ke
merdekaan dibacakan oleh Soekarno dan Hatta di Jalan Pegangsaan Timur No. 56, Jak
arta. Setelah proklamasi, dibentuk Komite Nasional Indonesia Pusat (KNIP) sebagai
lembaga sementara yang membantu tugas presiden. Pada 18 Agustus 1945, Panitia Per
siapan Kemerdekaan Indonesia (PPKI) mengesahkan Undang-Undang Dasar 1945 dan mene
tapkan Soekarno sebagai presiden serta Mohammad Hatta sebagai wakil presiden pert
ama Republik Indonesia.

QUESTION: Siapa yang membacakan teks proklamasi kemerdekaan Indonesia?

JAWABAN: 17 agustus 1945,

Confidence Score:
   Start Position: 0.1155 (11.55%)
   End Position: 0.1204 (12.04%)
   Average: 0.1180 (11.80%)

Position in Context: Word 2 to 4

Confidence: RENDAH - Jawaban mungkin tidak akurat

================================================================================

Ingin bertanya lagi? (Y/N)

Terima kasih telah menggunakan chatbot!

Terima kasih telah menggunakan chatbot!
```

Alternatif: Quick Test (Tanpa Loop)

Jika Anda ingin test cepat tanpa loop interaktif, gunakan cell di bawah ini:

In [25]:
```python
# ================================================================================
# QUICK TEST - GANTI CONTEXT DAN QUESTION DI SINI
# ================================================================================

# MASUKKAN CONTEXT ANDA DI SINI (paragraf yang berisi informasi):
my_context = """
Soekarno dan Mohammad Hatta memproklamasikan kemerdekaan Indonesia
pada tanggal 17 Agustus 1945 di Jalan Pegangsaan Timur 56, Jakarta.
Peristiwa bersejarah ini disaksikan oleh ratusan rakyat Indonesia
dan menandai lahirnya Republik Indonesia sebagai negara merdeka.
"""

# QUESTION ANDA DI SINI (pertanyaan tentang context):
```

```
my_question = "Siapa yang memproklamasikan kemerdekaan Indonesia?"

# ================================================================================
# JANGAN EDIT KODE DI BAWAH INI - LANGSUNG JALANKAN!
# ================================================================================

# Tampilkan header mode quick test
print("="*80)
print("CHATBOT QA - QUICK TEST MODE")
print("="*80)

# Prediksi jawaban menggunakan context dan question yang sudah ditentukan
pred_answer, start_idx, end_idx, start_conf, end_conf = predict_answer(
    model, tokenizer, my_context, my_question, MAX_CONTEXT_LEN, MAX_QUESTION_LEN
)

# Tampilkan hasil dengan format yang rapi
print(f"\nCONTEXT:")
print(my_context.strip())  # strip untuk hapus whitespace di awal dan akhir
print(f"\nQUESTION:")
print(my_question)
print(f"\nJAWABAN:")
print(f"   {pred_answer}")
print(f"\nCONFIDENCE SCORE:")
print(f"   Start Position: {start_conf:.4f} ({start_conf*100:.2f}%)")  # confide
print(f"   End Position: {end_conf:.4f} ({end_conf*100:.2f}%)")  # confidence sc
print(f"   Average: {(start_conf + end_conf)/2:.4f} ({(start_conf + end_conf)/2*
print(f"\nPOSITION:")
print(f"   Word {start_idx} to {end_idx}")  # posisi kata di context

# Interpretasi confidence score untuk user
avg_conf = (start_conf + end_conf) / 2  # hitung rata-rata confidence
print(f"\nINTERPRETASI:")
if avg_conf > 0.7:  # kalau confidence tinggi (lebih dari 70%)
    print("   TINGGI - Jawaban kemungkinan besar benar!")
elif avg_conf > 0.4:  # kalau confidence sedang (40-70%)
    print("   SEDANG - Jawaban mungkin benar, periksa kembali")
else:  # kalau confidence rendah (kurang dari 40%)
    print("   RENDAH - Jawaban mungkin tidak akurat")

print("\n" + "="*80)
print("TIP: Edit variabel 'my_context' dan 'my_question' di atas,")
print("        lalu jalankan ulang cell ini untuk test dengan data baru!")
print("="*80)
```

```
================================================================================
CHATBOT QA - QUICK TEST MODE
================================================================================

CONTEXT:
Soekarno dan Mohammad Hatta memproklamasikan kemerdekaan Indonesia
pada tanggal 17 Agustus 1945 di Jalan Pegangsaan Timur 56, Jakarta.
Peristiwa bersejarah ini disaksikan oleh ratusan rakyat Indonesia
dan menandai lahirnya Republik Indonesia sebagai negara merdeka.

QUESTION:
Siapa yang memproklamasikan kemerdekaan Indonesia?

JAWABAN:
   17 agustus 1945

CONFIDENCE SCORE:
   Start Position: 0.1577 (15.77%)
   End Position: 0.1369 (13.69%)
   Average: 0.1473 (14.73%)

POSITION:
   Word 9 to 11

INTERPRETASI:
   RENDAH - Jawaban mungkin tidak akurat

================================================================================
TIP: Edit variabel 'my_context' dan 'my_question' di atas,
        lalu jalankan ulang cell ini untuk test dengan data baru!
================================================================================
```

15. Model Analysis & Insights

Analisis performa dan karakteristik model.

In [ ]:

```python
# Hitung Exact Match dan F1 scores untuk evaluasi performa model
from collections import Counter  # import Counter untuk hitung overlap kata

def calculate_f1(pred_answer, true_answer):
    """
    Fungsi untuk menghitung F1 score antara predicted answer dan true answer.
    F1 score mengukur kesamaan antara dua set kata (precision dan recall).

    Args:
        pred_answer: jawaban hasil prediksi model
        true_answer: jawaban yang sebenarnya

    Returns:
        f1: F1 score (0.0 - 1.0)
    """
    # Tokenize dan lowercase kedua jawaban
    pred_tokens = pred_answer.lower().split()  # split jawaban prediksi jadi lis
    true_tokens = true_answer.lower().split()  # split jawaban true jadi list ka

    # Kalau salah satu kosong, F1 score = 0
    if len(pred_tokens) == 0 or len(true_tokens) == 0:
        return 0
```

```
    # Hitung kata yang sama antara pred dan true (intersection)
    common = Counter(pred_tokens) & Counter(true_tokens)  # intersection menggun
    num_same = sum(common.values())  # jumlah kata yang sama

    # Kalau tidak ada kata yang sama, F1 = 0
    if num_same == 0:
        return 0

    # Hitung precision dan recall
    precision = num_same / len(pred_tokens)  # precision = kata sama / total kat
    recall = num_same / len(true_tokens)  # recall = kata sama / total kata true
    f1 = 2 * precision * recall / (precision + recall)  # F1 = harmonic mean dar

    return f1

# Evaluasi pada sample dari validation set
sample_size = min(100, len(val_filtered))  # ambil maksimal 100 sample (atau sem
exact_matches = 0  # counter untuk exact match
f1_scores = []  # list untuk tampung F1 scores

print("Calculating Exact Match and F1 scores...")
# Loop semua sample untuk evaluasi
for i in tqdm(range(sample_size)):  # tqdm untuk progress bar
    example = val_filtered[i]  # ambil contoh dari validation set
    context = example['context']
    question = example['question']
    true_answer = example['answer']

    # Prediksi jawaban menggunakan model
    pred_answer, _, _, _, _ = predict_answer(
        model, tokenizer, context, question, MAX_CONTEXT_LEN, MAX_QUESTION_LEN
    )

    # Cek exact match (jawaban 100% sama)
    if pred_answer.lower().strip() == true_answer.lower().strip():
        exact_matches += 1  # increment counter kalau exact match

    # Hitung F1 score (partial match)
    f1 = calculate_f1(pred_answer, true_answer)
    f1_scores.append(f1)  # tambahkan ke list

# Hitung metrics akhir
exact_match_score = exact_matches / sample_size * 100  # exact match dalam perse
avg_f1_score = np.mean(f1_scores) * 100  # rata-rata F1 score dalam persen

# Tampilkan hasil evaluasi
print("\n" + "="*50)
print("MODEL PERFORMANCE METRICS")
print("="*50)
print(f"Sample Size: {sample_size}")  # jumlah sample yang dievaluasi
print(f"Exact Match (EM): {exact_match_score:.2f}%")  # persentase exact match
print(f"Average F1 Score: {avg_f1_score:.2f}%")  # rata-rata F1 score
print("="*50)
```

```
Calculating Exact Match and F1 scores...
100%|██████████| 100/100 [00:07<00:00, 14.27it/s]
```

```
================================================
MODEL PERFORMANCE METRICS
================================================
Sample Size: 100
Exact Match (EM): 1.00%
Average F1 Score: 6.48%
================================================
```

16. Summary & Conclusions

Hasil Training:

Model RNN dengan BPTT telah berhasil dilatih untuk task Question-Answering bahasa Indonesia dengan **optimisasi kecepatan**.

Cara Kerja BPTT dalam Model:

1. **Forward Pass**: Input sequence diproses timestep demi timestep melalui RNN
2. **Loss Calculation**: Menghitung error pada output (start & end positions)
3. **Backward Pass (BPTT)**:
   - Gradient dihitung mundur dari output ke input
   - Gradient dipropagasi melalui semua timesteps
   - Mengupdate hidden state di setiap timestep
4. **Weight Update**: Optimizer (Adam) mengupdate semua weights berdasarkan gradient

Optimisasi Kecepatan yang Diterapkan:

Problem Solved:

- 1-2 menit/epoch
- Patience 10 epochs

Speed Optimizations:

- **Unidirectional RNN** instead of Bidirectional (2x faster!)
- **Batch Size 32** instead of 16 (fewer iterations)
- **Shorter Sequences**: Context 150, Question 15
- **Smaller Model**: 128 units, 128 embedding dim
- **Reduced Vocab**: 15000 instead of 20000
- **Minimal Dropout**: 0.1-0.2 for faster computation
- **Simplified Architecture**: Less layers

Training Configuration:

- **Early Stopping**: Patience=10 epochs (reasonable)
- **Total Epochs**: 50 epochs
- **Learning Rate**: 0.001 (balanced)
- **BPTT**: Fully utilized for gradient computation

Kelebihan Konfigurasi Ini:

- **Fast Training**: 3-4x lebih cepat!
- **Still Effective**: Unidirectional RNN tetap powerful
- **Memory Efficient**: Lebih hemat RAM
- **Balanced**: Trade-off yang bagus antara speed & performance
- **Practical**: Training time yang realistis
- **Interactive**: Bisa test langsung dengan input manual!

Cara Menggunakan Chatbot:

Opsi 1: Interactive Mode (Loop)

- Jalankan `interactive_chatbot()`
- Input context dan question berulang kali
- Ketik 'contoh' untuk lihat contoh
- Ketik 'exit' untuk keluar

Opsi 2: Quick Test Mode

- Edit variabel `my_context` dan `my_question`
- Jalankan cell untuk test cepat
- Cocok untuk testing satu-satu

Catatan:

- Model ini **speed-optimized** untuk training praktis
- Jika punya GPU powerful, bisa gunakan konfigurasi yang lebih besar
- SimpleRNN murni (bukan LSTM/GRU) dengan BPTT
- Untuk production, pertimbangkan LSTM/GRU atau Transformer

Jika Ingin Performa Lebih Tinggi:

1. **Gunakan LSTM/GRU** instead of SimpleRNN (better gradient flow)
2. **Tambah Data**: More training data = better generalization
3. **Pretrained Embeddings**: FastText/Word2Vec Indonesia
4. **Attention Mechanism**: Add attention layer
5. **Bidirectional**: Jika tidak masalah dengan waktu training
6. **Ensemble**: Kombinasi beberapa model

Training Time Estimate:

- **Per Epoch**: ~1-2 minutes
- **10 Epochs**: ~10-20 minutes
- **50 Epochs**: ~50-100 minutes
- **With Early Stop**: Probably ~15-30 minutes total

Output Files:

- `best_rnn_model.h5` : Best model berdasarkan validation loss
- `final_rnn_model.h5` : Final model setelah training selesai
- `tokenizer.pickle` : Tokenizer untuk preprocessing
- `config.pickle` : Konfigurasi model
- `history.pickle` : Training history
- `training_history.png` : Visualisasi training
