# News Summarization SimpleRNN

Hestina Dwi H - 2108077

# 1. Import Dataset

Dataset didapatkan dari hasil scraping berita.

LINK = Di sini

# 2. Import Library

```
import re
import pandas as pd
import numpy as np
import datetime as dt
import string
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.graph_objects as go
from textblob import TextBlob
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, recall_score, precision_score
from sklearn.metrics import confusion_matrix, classification_report
# from google_play_scraper import Sort, reviews_all, reviews
import nltk
# import calendar
nltk.download('punkt')
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist
```

# 3. Preprocessing Data

1. Ubah teks menjadi huruf kecil semua

```python
def lowercase(review_text):
    if isinstance(review_text, str):
        return review_text.lower()
    else:
        return review_text  # Return the original value if it's not a string

# Apply the lowercase function to both columns
df['berita'] = df['berita'].apply(lowercase)
df['title'] = df['title'].apply(lowercase)
```

2. Hapus extra whitespaces

```python
import re

def remove_extra_whitespaces(review_text):
    if isinstance(review_text, str):  # Check if it's a string
        return re.sub(r'\s+', ' ', review_text)
    else:
        return review_text  # Return the original value if it's not a string

# Apply the function to both columns
df['berita'] = df['berita'].apply(remove_extra_whitespaces)
df['title'] = df['title'].apply(remove_extra_whitespaces)
```

3. Hapus Karakter Spesial

```python
def remove_special_characters(text):
    if isinstance(text, str):  # Check if it's a string
        return re.sub(r'[^A-Za-z0-9\s]', '', text)
    else:
        return text  # Return the original value if it's not a string

# Apply the function to both columns
df['berita'] = df['berita'].apply(remove_special_characters)
df['title'] = df['title'].apply(remove_special_characters)
```

# 4. Define Token

```
df['title']=df['title'].apply(lambda x : 'sostok ' + str(x) + ' eostok')
```

# 5. Hitung persentil value

```
for i in range(90, 100):
    var = df['word_count_text'].values
    var = np.sort(var, axis=None)
    print("{} percentile value is {}".format(i, var[int(len(var) * (float(i) / 100))]))

print("100 percentile value is", var[-1])
```

```
90 percentile value is 536
91 percentile value is 557
92 percentile value is 581
93 percentile value is 600
94 percentile value is 620
95 percentile value is 641
96 percentile value is 690
97 percentile value is 760
98 percentile value is 865
99 percentile value is 1032
100 percentile value is 1569
```

# 6. Ambil persentil ke 95 buat set max_len_text

```
max_len_text=641
max_len_title=20
```

# 7. Bagi data untuk training dan testing

```
from sklearn.model_selection import train_test_split

x_tr, x_val, y_tr, y_val = train_test_split(
    np.array(df["berita"]),
    np.array(df["title"]),
    test_size=0.1,
    random_state=0,
    shuffle=True,
)
```

# 8. Tokenize teks buat hitung jumlah kata

```
# Tokenize the text to get the vocab count
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

# Prepare a tokenizer on training data
x_tokenizer = Tokenizer()
x_tokenizer.fit_on_texts(list(x_tr))
```

# 7. Bagi data untuk training dan testing

```
from sklearn.model_selection import train_test_split

x_tr, x_val, y_tr, y_val = train_test_split(
    np.array(df["berita"]),
    np.array(df["title"]),
    test_size=0.1,
    random_state=0,
    shuffle=True,
)
```

# 8. Tokenize teks buat hitung jumlah kata

```
# Tokenize the text to get the vocab count
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

# Prepare a tokenizer on training data
x_tokenizer = Tokenizer()
x_tokenizer.fit_on_texts(list(x_tr))
```

# 7. Bagi data untuk training dan testing

```
from sklearn.model_selection import train_test_split

x_tr, x_val, y_tr, y_val = train_test_split(
    np.array(df["berita"]),
    np.array(df["title"]),
    test_size=0.1,
    random_state=0,
    shuffle=True,
)
```

# 8. Tokenize teks buat hitung jumlah kata

```
# Tokenize the text to get the vocab count
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

# Prepare a tokenizer on training data
x_tokenizer = Tokenizer()
x_tokenizer.fit_on_texts(list(x_tr))
```

# 9. Define SimpleRNN-nya

```
latent_dim = 300  # Size of the RNN layer
embedding_dim = 200  # Size of the embedding layer

# Encoder
encoder_inputs = Input(shape=(max_len_text,))

# Embedding layer
enc_emb = Embedding(x_voc, embedding_dim, trainable=True)(encoder_inputs)

# Encoder SimpleRNN 1
encoder_rnn1 = SimpleRNN(latent_dim, return_sequences=True, return_state=True, dropout=0.4)
encoder_output1, state_h1 = encoder_rnn1(enc_emb)

# Encoder SimpleRNN 2
encoder_rnn2 = SimpleRNN(latent_dim, return_sequences=True, return_state=True, dropout=0.4)
encoder_output2, state_h2 = encoder_rnn2(encoder_output1)

# Set up the decoder, using encoder_states as the initial state
decoder_inputs = Input(shape=(None,))

# Embedding layer for the decoder
dec_emb_layer = Embedding(y_voc, embedding_dim, trainable=True)
dec_emb = dec_emb_layer(decoder_inputs)

# Decoder SimpleRNN
decoder_rnn = SimpleRNN(latent_dim, return_sequences=True, return_state=True, dropout=0.4)
decoder_outputs, decoder_state = decoder_rnn(dec_emb, initial_state=[state_h2])

# Dense layer
decoder_dense = TimeDistributed(Dense(y_voc, activation='softmax'))
decoder_outputs = decoder_dense(decoder_outputs)
```

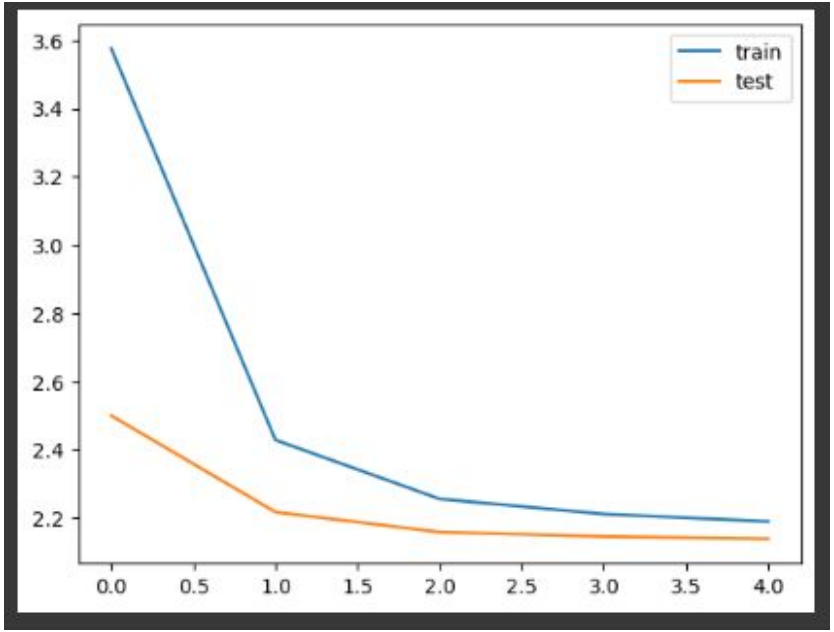
```
# Define the model
model = Model([encoder_inputs, decoder_inputs], decoder_outputs)

model.summary()
```

# 10. Training Modelnya

```
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping

checkpoint = ModelCheckpoint('best_model.keras', monitor='val_loss', save_best_only=True, mode='min', verbose=1)

early_stopping = EarlyStopping(monitor='val_loss', patience=3, mode='min', verbose=1)

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

history = model.fit(
    [x_tr, y_tr[:, :-1]],
    y_tr[:, 1:],
    epochs=5,
    callbacks=[early_stopping],
    batch_size=128,
    validation_data=([x_val, y_val[:, :-1]], y_val[:, 1:])
    )
```

# 11. Lihat Grafik Hasil Trainingnya

```
from matplotlib import pyplot

pyplot.plot(history.history['loss'], label='train')
pyplot.plot(history.history['val_loss'], label='test')
pyplot.legend()
pyplot.show()
```



# 12. Coba generate summarynya

```python
import numpy as np
from tensorflow.keras.preprocessing.sequence import pad_sequences

# Define start and end tokens using sostok and eostok
start_token = x_tokenizer.word_index.get('sostok', 1)
end_token = x_tokenizer.word_index.get('eostok', 0)

# Prepare encoder input by tokenizing and padding
encoder_input = x_tokenizer.texts_to_sequences(df['berita'])
encoder_input = pad_sequences(encoder_input, maxlen=max_len_text)

summaries = []

# Function for sampling with a temperature parameter
def sample_with_temperature(predictions, temperature=0.8):
    predictions = np.asarray(predictions).astype("float64")
    predictions = np.log(predictions + 1e-7) / temperature
    exp_preds = np.exp(predictions)
    predictions = exp_preds / np.sum(exp_preds)
    return np.random.choice(len(predictions), p=predictions)
```

```python
for i in range(10):
    decoder_input = np.array([[start_token]])
    generated_tokens = []

    for _ in range(max_len_title):
        predictions = model.predict([encoder_input[i:i+1], decoder_input])

        # Use sampling to add variability and avoid repetition
        predicted_token = sample_with_temperature(predictions[0, -1, :])

        if predicted_token == end_token:
            break

        generated_tokens.append(predicted_token)
        decoder_input = np.array([[predicted_token]])

    # Decode generated tokens to readable text
    generated_summary = ' '.join([x_tokenizer.index_word.get(token, '') for token in generated_tokens])
    summaries.append(generated_summary)

# Display results
result_df = pd.DataFrame({
    'title': df['title'][:10],
    'generated_summary': summaries
})

print(result_df)
```

# 13. Lihat Hasil Output Generate-nya

```
Article 1:
Title: sostok pramono janji lanjutkan program boti demi merangkul umat beragama eostok
Generated Summary: kemungkinan ini kpu ada sah masingmasing baru pada dalam mengambil adalah tangerang badan jalan indonesia jadwal jakarta melawan masingmasing kesehatan


Article 2:
Title: sostok mpr tetapkan susunan fraksi periode 20242029 eostok
Generated Summary: paslon pasar beberapa i menyebut maupun ketua jawab tanggal hak paslon agar jawa 2023 tak paslon pemilihan cianjur menegaskan anies


Article 3:
Title: sostok manchester city menang telak 40 atas slovan bratislava eostok
Generated Summary: menyebut pilkada antara membawa senin jakarta dpr pada i pendukung antara jakarta agar depan jakarta jakarta anggota liga jakarta pada


Article 4:
Title: sostok kpk periksa anggota pokja pengadaan truk basarnas eostok
Generated Summary: pak jawab mendukung provinsi kami juga surabaya with khofifah termasuk ketua menegaskan nanti pencalonan partai ingin meraih ketua andika akan


Article 5:
Title: sostok sultan najamudin terpilih jadi ketua dpd 20242029 eostok
Generated Summary: solo politik dico pilkada apakah negara jakarta gubernur data pembangunan malam jakarta lagi agustus ingin khofifah dico pilkada pilkada meraih
```