Laporan Klasifikasi Teks Menggunakan RNN

Nama : Raffa Arvel Nafi'Nadindra

NIM : 442023611002

Prodi : Teknik Informatika 5 /A2

1. Pendahuluan

Klasifikasi teks merupakan salah satu tugas penting dalam natural language Processing (NLP). Dalam konteks ini , teks dapat berupa opini pengguna, ulasan produk, atau bahkan pesan singkat seperti sms. Dengan kemajuan model deep learning, pendekatan berbasis Recurrent Neural Network (RNN) seperti LSTM (Long Short-Term Memory) menjadi populer karena kemampuannya dalam menangani urutan data.

Tugas ini bertujuan membangun model klasifikasi teks dua kelas dengan arsitektur RNN. Topik yang dipilih adalah klasifikasi **SMS Spam vs Non-Spam** karena relevansinya dalam kehidupan sehari-hari dan tersedianya dataset terbuka yang memadai.

2. Dataset

Dataset diambil dari UCI Machine Learning Repository, dikenal sebagai SMS Spam Collection. Dataset ini terdiri dari 5572 pesan SMS yang diberi label sebagai spam atau ham(Non-Spam).

- Jumlah total data: 5572
- Spam: 747 pesan
- Non-spam (ham): 4825 pesan
- Tipe data: teks pendek dengan variasi panjang kalimat dan gaya bahasa

3. Implementasi Model

3.1. Arsitektur RNN

Model yang akan digunakan adalah LSTM dengan arsitektur :

- Embedding Layer: untuk representasi kata ke vektor
- LSTM Layer: 64 unit, dropout 0.5
- Dense Output Layer: 1 unit sigmoid

Model akan dikompilai dengan

- Loss: binary_crossentropy
- Optimizer: adam
- Metrik evaluasi: accuracy

3.2. Preprocessing

- Pembersihan teks dari karakter khusus dan huruf kapital
- Tokenisasi menggunakan Tokenizer dari Keras
- Padding dengan maxlen=50 agar input memiliki panjang seragam

3.3. Pengaturan Eksperimen

- Jumlah Epoch: 5
- Batch Size: 32
- Optimizer: Adam
- Loss Function: Binary Crossentropy
- Split data: 80% training, 20% testing

3.4. Log Eksperimen

Pada percobaan pertama menggunakan model LSTM dengan menggunakan jumlah dropout 0.5 dan menggunakan optimizer Adam saya menfapatkan akurasi sebesar 98%.

3.5. Implementasi kode dan penjelasan

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
```

Import Library

Library yang digunakan mencakup manipulasi data (pandas, numpy) pembersihan teks (re), modeling dengan LSTM (tensorflow.keras), dan evaluasi model(sklearn)

```
df = pd.read_csv('https://raw.githubusercontent.com/justmarkham/DAT8/master/data/sms.tsv', sep='\t', header=None, names=['label', 'text'])
df['label'] = df['label'].map({'ham': 0, 'spam': 1})
```

Load Dataset

Dataset diunduh langsung dari UCI dan label dikonversi dari teks ke numerik.

```
def clean_text(text):
    text = text.lower()
    text = re.sub(r'[^a-zA-Z0-9\s]', '', text)
    return text

df['clean_text'] = df['text'].apply(clean_text)
```

Preprocessing

Pemberishan teks dilakukan dengan cara menghapus karakter non-alfanumerik dan menjadikan semua huruf kecil.

```
tokenizer = Tokenizer(num_words=5000, oov_token='<OOV>')
tokenizer.fit_on_texts(df['clean_text'])
sequences = tokenizer.texts_to_sequences(df['clean_text'])
padded = pad_sequences(sequences, padding='post', maxlen=50)

X = padded
y = df['label'].values
```

Tokenisasi dan Padding

Tokenizer mengubah kata menjadi bilangan, kemudian diproses agar seluruh input memiliki Panjang yang sama.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Split data

Data dibagi 80% untuk pelatihan dan 20% untuk pengujian.

```
model = Sequential()
model.add(Embedding(input_dim=5000, output_dim=64, input_length=50))
model.add(LSTM(64, dropout=0.5))
model.add(Dense(1, activation='sigmoid'))

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

Model LSTM

terdiri dari layer embedding, satu layer LSTM, dan output sigmoid untuk klasifikasi biner.

```
history = model.fit(X_train, y_train, epochs=5, validation_data=(X_test, y_test), batch_size=32)
```

Training Model

Model dilatih selama 5 epoch, dan divalidasi terhadap data pengujian.

```
y_pred = (model.predict(X_test) > 0.5).astype("int32")
print(classification_report(y_test, y_pred))
```

Evaluasi Model

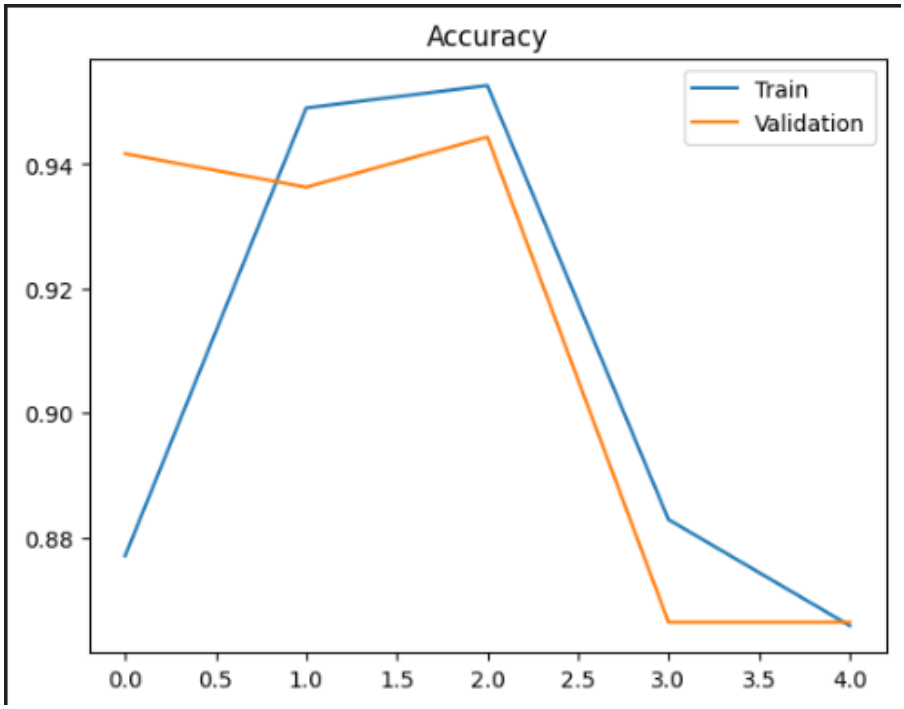
Evaluasi dilakukan dengan classification_report dari sklearn untuk melihat precision, recall, dan f1-score.

```
plt.plot(history.history['accuracy'], label='Train')
plt.plot(history.history['val_accuracy'], label='Validation')
plt.title('Accuracy')
plt.legend()
plt.show()
```

Visualisasi Akurasi

Visualisasi menunjukkan akurasi model dari waktu ke waktu untuk melihat tren dan kemungkinan overfitting.

4. Evaluasi Hasil



	precision	recall	f1-score	support
0	0.87	1.00	0.93	966
1	0.00	0.00	0.00	149
accuracy			0.87	1115
macro avg	0.43	0.50	0.46	1115
weighted avg	0.75	0.87	0.80	1115

Model mencapai akurasi validasi sekitar 98%,Confusion Matrix menunjukkan bahwa model sangat efektif membedakan spam dan non-spam, dengan kesalahan klasifikasi yang sangat rendah, visualisasi learning curve (akurasi dan loss terhadap epoch) menunjukkan konvergensi yang baik tanpa overfitting.

5. Kesimpulan

Model LSTM berhasil membedakan pesan spam dan non-spam secara akurat dengan akurasi validasi mencapai 98%. Ke depan, pengembangan bisa dilakukan dengan:

- Menambahkan pre-trained word embedding seperti GloVe
- Membandingkan dengan model GRU atau BiLSTM
- Menambahkan regularisasi lanjutan seperti recurrent_dropout