Laporan Tugas Individu: Klasifikasi Teks Menggunakan RNN

1. Pendahuluan

Klasifikasi teks merupakan salah satu tugas penting dalam bidang pemrosesan bahasa alami (Natural Language Processing/NLP). Dengan klasifikasi teks, kita dapat secara otomatis mengelompokkan dokumen atau pesan ke dalam kategori tertentu, seperti opini positif/negatif, spam/non-spam, atau topik-topik tertentu seperti berita politik dan olahraga. Tugas ini tidak hanya bermanfaat untuk keperluan komersial (misalnya, filter spam dan analisis sentimen pelanggan), tetapi juga penting dalam otomatisasi dan penyaringan informasi secara efisien.

Dalam tugas ini, saya membangun model klasifikasi teks menggunakan arsitektur Recurrent Neural Network (RNN), khususnya Long Short-Term Memory (LSTM), untuk memisahkan pesan spam dan non-spam. Fokus utama tidak hanya pada akurasi akhir, tetapi juga pada eksplorasi ide, refleksi, serta pengambilan keputusan berdasarkan eksperimen.

2. Dataset

Sumber Data

Data diambil dari dataset publik **SMS Spam Collection Dataset** yang tersedia di Kaggle. Dataset ini terdiri dari pesan SMS yang telah dilabeli sebagai “spam” atau “ham” (bukan spam).

Deskripsi Dataset

- Total data: 200 (100 spam, 100 ham)
- Contoh data:
    - Ham: “Hey, are we still meeting today?”
    - Spam: “Congratulations! You won a free ticket to Bahamas. Reply YES to claim.”
- Variasi:
    - Panjang teks bervariasi dari 5 kata hingga lebih dari 30 kata.
    - Gaya bahasa mencakup percakapan informal hingga promosi otomatis.

Alasan Pemilihan Dataset

Topik spam vs non-spam dipilih karena:

- Relevansi dengan aplikasi nyata (filter email, pesan SMS).
- Dataset tersedia secara terbuka dan memiliki keseimbangan kelas yang baik.
- Tantangan klasifikasi cukup kompleks karena beberapa spam menggunakan bahasa menyerupai pesan normal.

3. Implementasi Model

3.1 Arsitektur RNN

Model yang digunakan adalah **LSTM** dengan arsitektur sebagai berikut:

- **Embedding Layer**: memetakan kata ke vektor berdimensi 64.
- **LSTM Layer**: 64 unit dengan output berupa vektor.
- **Dense Layer**: aktivasi sigmoid untuk klasifikasi biner.
- **Dropout Layer**: digunakan untuk mengurangi overfitting setelah eksperimen awal menunjukkan peningkatan akurasi tetapi validasi menurun.

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

Dropout secara acak menonaktifkan neuron selama training, sehingga memaksa jaringan belajar representasi yang lebih umum dan mencegah model menghafal data training.

3.2 Preprocessing

Langkah preprocessing:

- **Tokenisasi**: mengubah kata menjadi urutan angka.
- **Padding**: menyamakan panjang input agar cocok dengan input layer.
- **Lowercasing**: mengurangi variasi kata karena huruf kapital.

```
tokenizer = Tokenizer(num_words=5000)
tokenizer.fit_on_texts(df['text'])
X = tokenizer.texts_to_sequences(df['text'])
X = pad_sequences(X, maxlen=50)
```

3.3 Pengaturan Eksperimen

- Epoch: 10
- Batch Size: 16
- Optimizer: Adam
- Loss: Binary Crossentropy
- Validation Split: 20%

3.4 Log Eksperimen

Percobaan	Model	Dropout	Optimizer	Akurasi Validasi	Catatan
#1	LSTM(64)	0	Adam	65.6%	Underfitting pada awal epoch
#2	LSTM(64)	0.5	Adam	87.5%	Hasil lebih stabil, tidak overfit
#3	LSTM(64)+DO	0.5	Adam	90.6%	Akurasi tinggi dan stabil

4. Evaluasi Hasil

Metrik dan Visualisasi

- Akurasi training terakhir: **96.75%**
- Akurasi validasi terbaik: **90.62%**
- Grafik loss dan akurasi menunjukkan model stabil tanpa overfitting.
- Confusion matrix menunjukkan prediksi cukup seimbang antara spam dan non-spam.

Analisis

Model berhasil mengklasifikasikan dengan baik setelah ditambahkan dropout dan embedding yang cukup besar. Evaluasi dilakukan tidak hanya dari akurasi, tetapi juga kurva learning yang menunjukkan bahwa model mencapai konvergensi tanpa gejala overfitting. Kesalahan masih terjadi pada beberapa pesan spam pendek atau ambigu.

5. Refleksi Pribadi

Tantangan Utama

- Menentukan parameter terbaik (maxlen, embedding dimensi, dropout)
- Memahami perilaku model LSTM saat underfitting atau overfitting

Solusi yang Dicoba

- Menambahkan dropout
- Menyesuaikan jumlah neuron
- Meningkatkan embedding dimensi

Bantuan dari AI

Saya menggunakan ChatGPT untuk:

- Menyusun struktur kode

- Menjelaskan parameter
- Memberi saran visualisasi dan evaluasi

Saya memverifikasi hasil dengan menginterpretasikan grafik dan metrik evaluasi secara mandiri.

Pelajaran Penting

- Eksperimen bertahap penting dalam machine learning.
- Visualisasi membantu mengenali over/underfitting.
- Dokumentasi proses dan pemikiran membuat hasil lebih terarah.

6. Kesimpulan dan Saran

Model LSTM berhasil memisahkan pesan spam dan non-spam dengan akurasi validasi di atas 90%. Dengan pengaturan yang tepat, RNN sangat efektif untuk klasifikasi teks pendek.

Saran pengembangan:

- Gunakan dataset lebih besar dan lebih kompleks
- Coba model BiLSTM atau GRU untuk efisiensi
- Gunakan embedding pre-trained seperti GloVe atau BERT
- Coba stacking layer RNN untuk menangkap pola jangka panjang

7. Referensi

- SMS Spam Collection Dataset - Kaggle
- Chollet, F. (2017). *Deep Learning with Python*. Manning Publications.
- TensorFlow Keras Documentation: https://keras.io