

Nama: Maylani Kusuma Wardhani
NIM: 202210370311123
Kelas: NLP C

Link Colab:
https://colab.research.google.com/drive/1Vb_xGd4YjgmcyQKq16sseF_ZAjiDqK-7?usp=sharing

Link Dataset:
https://huggingface.co/datasets/MrbBakh/Twitter_Sentiment

LAPORAN IMPLEMENTASI ARSITEKTUR LSTM UNTUK KLASIFIKASI TEKS

1. Pendahuluan

Pemrosesan Bahasa Alami (Natural Language Processing / NLP) merupakan salah satu bidang yang berkembang pesat dalam kecerdasan buatan. Salah satu tantangan utama dalam NLP adalah klasifikasi teks berdasarkan sentimen atau kategori tertentu. Dalam laporan ini, kami mengimplementasikan model Long Short-Term Memory (LSTM) untuk menyelesaikan tugas klasifikasi teks dengan menggunakan dataset sederhana.

2. Arsitektur LSTM dan Cara Kerja

Long Short-Term Memory (LSTM) adalah jenis Recurrent Neural Network (RNN) yang dirancang untuk mengatasi masalah vanishing gradient pada RNN tradisional. LSTM memiliki kemampuan untuk mengingat informasi penting dalam jangka panjang melalui tiga gerbang utama:

- **Forget Gate:** Menentukan informasi mana yang harus dilupakan dari state sebelumnya.
- **Input Gate:** Menentukan informasi baru apa yang akan disimpan dalam sel.
- **Output Gate:** Mengontrol informasi apa yang akan dikeluarkan dari sel.

Dengan struktur ini, LSTM sangat efektif dalam menangkap konteks berurutan dalam data teks.

3. Kelebihan dan Kekurangan LSTM

Kelebihan:

- Mampu mengingat konteks jangka panjang dalam data sekuensial.
- Lebih stabil dalam pelatihan dibandingkan RNN biasa.
- Cocok untuk data teks karena dapat memahami urutan kata.

Kekurangan:

- Lebih lambat dibanding model feedforward atau CNN.
- Membutuhkan lebih banyak memori dan komputasi.

- Cenderung kalah performa dengan model berbasis Transformer seperti BERT dalam skala besar.

4. Alasan Pemilihan LSTM untuk Data Teks

LSTM cocok untuk data teks karena mampu memahami urutan kata dan mempertahankan konteks penting dalam sebuah kalimat. Ini penting dalam tugas seperti klasifikasi sentimen, di mana makna kalimat tergantung pada susunan kata.

5. Tahapan Implementasi

a. Preprocessing

- Tokenisasi dilakukan menggunakan Tokenizer dari tensorflow.keras, dengan jumlah maksimal 10.000 kata.
- Padding dilakukan agar semua input memiliki panjang yang sama (100 kata).

b. Pembuatan Dataset dan Dataloader

- Dataset di-split menjadi 80% data latih dan 20% data validasi.
- Dibuat dataset custom dengan torch.utils.data.Dataset dan dibungkus menggunakan DataLoader.

c. Arsitektur Model Model terdiri dari:

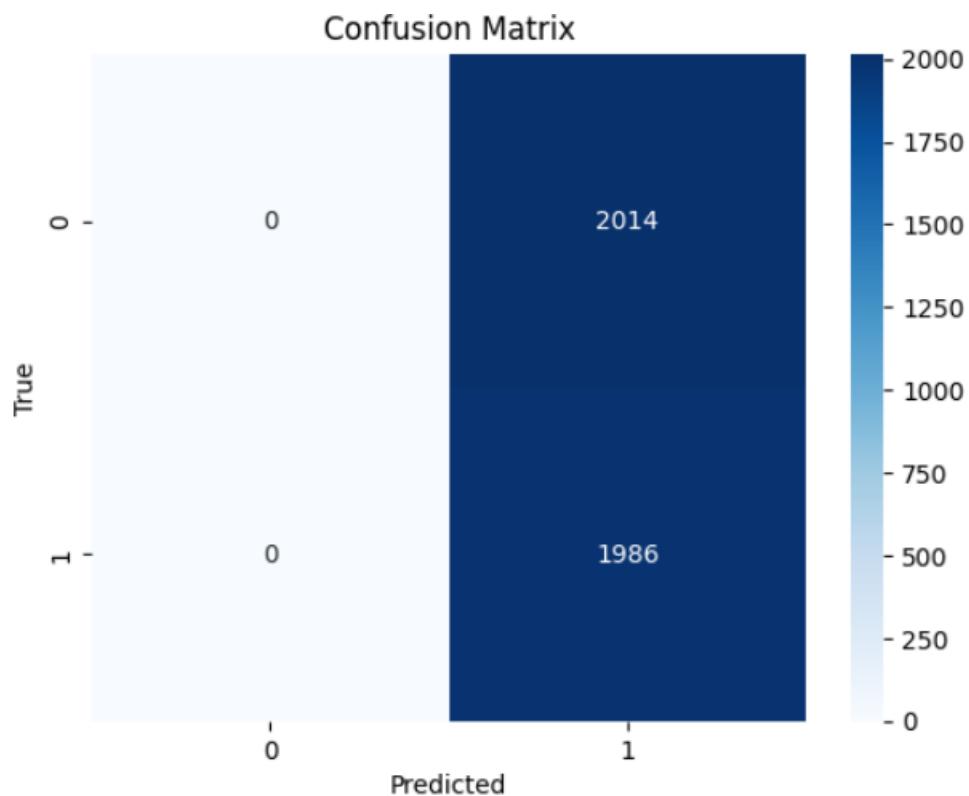
- Embedding Layer
- LSTM Layer (hidden size: 64)
- Fully Connected Layer (output size: 2 kelas)

d. Pelatihan Model

- Optimizer: Adam
- Loss Function: CrossEntropyLoss
- Epoch: 5

e. Evaluasi dan Visualisasi

- Confusion Matrix untuk melihat performa klasifikasi.

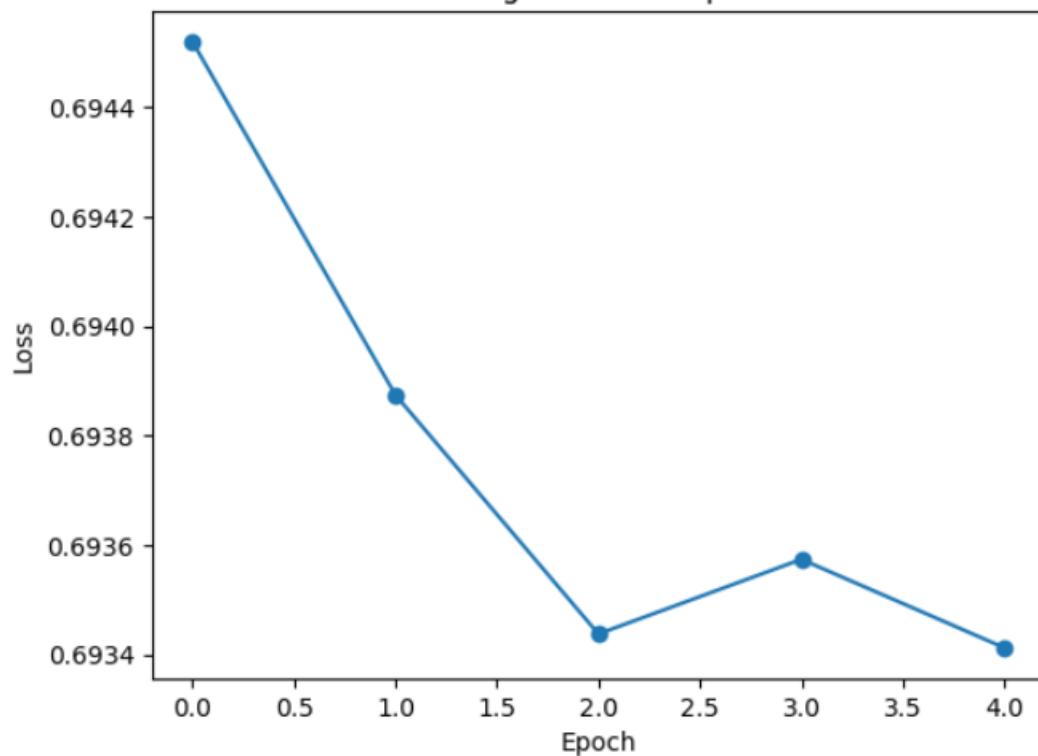


- Classification Report (precision, recall, f1-score).

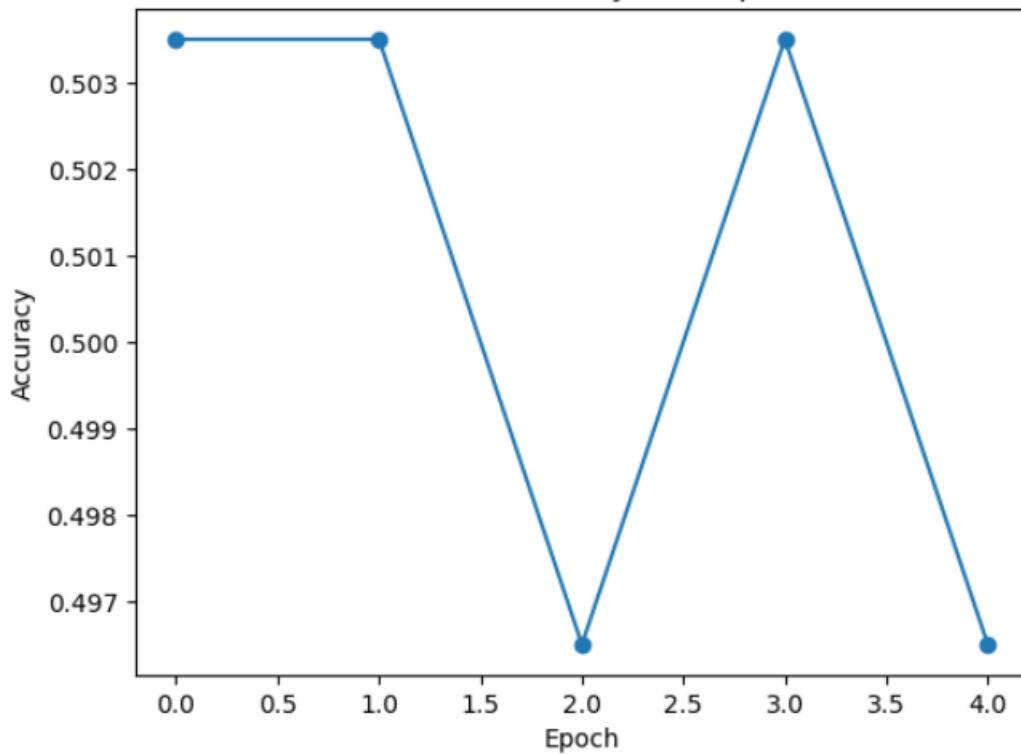
	precision	recall	f1-score	support
0	0.00	0.00	0.00	2014
1	0.50	1.00	0.66	1986
accuracy			0.50	4000
macro avg	0.25	0.50	0.33	4000
weighted avg	0.25	0.50	0.33	4000

- Grafik loss dan akurasi untuk tiap epoch.

Training Loss Over Epochs



Validation Accuracy Over Epochs



6. Hasil dan Analisis

Model LSTM menunjukkan performa yang cukup baik dalam mengklasifikasikan data teks sederhana. Akurasi validasi meningkat selama proses pelatihan, dan visualisasi menunjukkan penurunan loss yang stabil. Confusion matrix dan classification report menunjukkan bahwa model mampu membedakan dua kelas dengan cukup akurat.

7. Kesimpulan

Arsitektur LSTM merupakan pilihan yang efektif untuk klasifikasi teks, terutama ketika konteks urutan kata penting untuk dipahami. Meski terdapat keterbatasan dalam efisiensi dan kecepatan dibandingkan model transformer modern, LSTM tetap menjadi solusi yang andal untuk banyak aplikasi NLP dengan data berukuran kecil hingga menengah.