Week 4

Persamaan Matematika

RNN Models

Embedding

$$x_t = \text{Embedding}(w_t) = W_e[w_t]$$

Mengubah kata menjadi vektor numerik

RNN Cell

$$h_t = \tanh(W_{xh}x_t + W_{hh}h_{t-1} + b)$$

Hidden state h_t pada waktu t dipengaruhi oleh input saat ini dan state sebelumnya.

Output

$$\hat{y} = \sigma(W_o h_t + b_o)$$

- Output akhir menggunakan sigmoid agar hasil antara 0 sampai 1 (probabilitas sentimen positif).

LSTM Models

Embedding Layer

$$x_t = \text{Embedding}(w_t) = W_e[w_t]$$

Mengubah kata menjadi vektor.

LSTM Cell

$$
\begin{aligned}
f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) && \text{(forget gate)} \\
i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) && \text{(input gate)} \\
\tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) && \text{(candidate cell state)} \\
C_t &= f_t * C_{t-1} + i_t * \tilde{C}_t && \text{(cell state update)} \\
o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) && \text{(output gate)} \\
h_t &= o_t * \tanh(C_t) && \text{(hidden state)}
\end{aligned}
$$

LSTM menyimpan memori jangka panjang dan pendek melalui cell state C_t.

Gate (input, forget, output) mengatur informasi yang disimpan atau dibuang.

GRU Models

$$
\begin{aligned}
z_t &= \sigma(W_z \cdot [h_{t-1}, x_t]) && \text{(update gate)} \\
r_t &= \sigma(W_r \cdot [h_{t-1}, x_t]) && \text{(reset gate)} \\
\tilde{h}_t &= \tanh(W \cdot [r_t * h_{t-1}, x_t]) && \text{(candidate hidden state)} \\
h_t &= (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t && \text{(final output)}
\end{aligned}
$$

GRU lebih sederhana dari LSTM, hanya punya dua gate: update dan reset.

zt mengontrol seberapa banyak ht mengambil dari memori sebelumnya atau dari \tilde{h}_t.

Perbandingkan hasil models RNN, LSTM, dan GRU

Model	Framework	Accuracy	Precision	Recall	F1 Score	$F1^2$	AUC
RNN	TensorFlow	0.8440	0.8412	0.8480	0.8446	0.7134	0.9152
LSTM	TensorFlow	0.8745	0.8710	0.8784	0.8747	0.7641	0.9367
GRU	TensorFlow	0.8692	0.8663	0.8720	0.8691	0.7553	0.9310
RNN	PyTorch	0.8418	0.8395	0.8450	0.8422	0.7093	0.9119
LSTM	PyTorch	0.8730	0.8702	0.8770	0.8736	0.7631	0.9355
GRU	PyTorch	0.8674	0.8647	0.8702	0.8674	0.7524	0.9301

Analisis Performa

Akurasi

- **LSTM** unggul tipis atas GRU dan RNN.
- RNN cenderung kurang stabil pada long-range dependencies (karena masalah vanishing gradient).

F1 Squared

- **LSTM** punya nilai $F1^2$ paling tinggi, menandakan konsistensi balance antara precision & recall.

AUC (Area Under Curve)

- AUC LSTM dan GRU di atas 0.93, menunjukkan kemampuan deteksi kelas positif sangat baik.
- RNN agak lebih rendah, tapi tetap layak.

Kelebihan dan Kekurangan

Model	Kelebihan	Kekurangan
RNN	Sederhana, cepat dilatih	Vanishing gradient, kurang akurat
LSTM	Paling akurat, powerful untuk long dependencies	Training agak lama, kompleks
GRU	Performa hampir setara LSTM, lebih ringan	Kadang kurang akurat untuk data yang sangat panjang

Kesimpulan

- **Butuh akurasi tinggi dan mampu menangani long sequence → LSTM**
- **Ingin lebih efisien dengan performa hampir LSTM → GRU**
- **Untuk baseline atau eksperimen awal → RNN**