BAB 2
LANDASAN TEORI

2.1 Tanda Baca

Tanda baca merupakan simbol yang digunakan dalam sistem ejaan [17]. Simbol ini digunakan oleh penulis untuk memberikan jeda antarkalimat, membentuk struktur tulisan, serta membimbing intonasi pembaca agar tidak terjadi kesalahpahaman [18]. Dalam Ejaan Yang Disempurnakan (EYD) edisi kelima, bahasa Indonesia memiliki lima belas jenis tanda baca. Tanda baca tersebut meliputi:

1. tanda titik (.),
2. tanda koma (,),
3. tanda titik koma (;),
4. tanda titik dua (:),
5. tanda hubung (-),
6. tanda pisah (),
7. tanda tanya (?),
8. tanda seru (!),
9. tanda elipsis (...),
10. tanda petik ("..."),
11. tanda petik tunggal ('...'),
12. tanda kurung ((...)),
13. tanda kurung siku ([...]),
14. tanda garis miring (/), dan
15. tanda apostrof (').

Dari kelima belas tanda baca tersebut, empat tanda baca yang paling sering digunakan dalam penulisan adalah tanda titik, tanda koma, tanda tanya, dan tanda seru. Masing-masing memiliki fungsi yang berbeda tergantung pada konteks kalimat yang digunakan.

2.1.1 Tanda Titik (.)

Berikut merupakan fungsi dari penggunaan tanda titik sesuai dengan EYD V [19]:

1. Digunakan pada akhir kalimat pernyataan.
2. Digunakan untuk mengakhiri pernyataan lengkap yang diikuti perincian berupa kalimat baru, paragraf baru, atau subjudul baru.
3. Digunakan di belakang angka atau huruf dalam suatu daftar, perincian, tabel, atau bagan.
4. Tidak digunakan di belakang angka terakhir pada deret nomor dalam perincian.
5. Tidak digunakan pada angka atau huruf yang sudah bertanda kurung dalam perincian.
6. Tidak digunakan di belakang angka terakhir, baik satu digit maupun lebih, dalam judul tabel, bagan, grafik, atau gambar.
7. Digunakan untuk memisahkan angka jam, menit, dan detik yang menunjukkan waktu atau jangka waktu.
8. Digunakan untuk memisahkan bilangan ribuan atau kelipatannya yang menunjukkan jumlah.
9. Tidak digunakan untuk memisahkan bilangan ribuan atau kelipatannya yang tidak menunjukkan jumlah.
10. Tidak digunakan pada akhir judul dan subjudul.
11. Tidak digunakan di belakang alamat penerima surat serta tanggal surat.

2.1.2 Tanda Koma (,)

Berikut merupakan fungsi dari penggunaan tanda koma sesuai dengan EYD V [20]:

1. Digunakan di antara unsur-unsur dalam perincian berupa kata, frasa, atau bilangan.
2. Digunakan sebelum kata penghubung seperti *tetapi*, *melainkan*, dan *sedangkan*, dalam kalimat majemuk pertentangan.
3. Digunakan untuk memisahkan anak kalimat yang mendahului induk kalimat.
4. Tidak digunakan jika induk kalimat mendahului anak kalimat.
5. Digunakan di belakang kata atau ungkapan penghubung antarkalimat seperti *oleh karena itu*, *jadi*, *dengan demikian*, *sehubungan dengan itu*, dan *meskipun demikian*.
6. Digunakan sebelum dan/atau sesudah kata seru seperti *o*, *ya*, *wah*, *aduh*, atau *hai*, serta kata sapaan seperti *Bu*, *Dik*, atau *Nak*.
7. Digunakan untuk memisahkan petikan langsung dari bagian lain dalam kalimat.
8. Tidak digunakan untuk memisahkan petikan langsung yang diakhiri tanda tanya atau tanda seru dari bagian kalimat yang mengikutinya.
9. Digunakan di antara: (a) nama dan alamat, (b) bagian-bagian alamat, (c) tempat dan tanggal, serta (d) nama tempat dan wilayah yang ditulis berurutan.
10. Digunakan sesudah salam pembuka (misalnya *dengan hormat* atau *salam sejahtera*), salam penutup (*salam takzim*, *hormat kami*), dan nama jabatan penanda tangan surat.
11. Digunakan di antara nama orang dan singkatan gelar akademik yang mengikutinya untuk membedakannya dari singkatan nama diri, nama keluarga, atau nama marga.
12. Digunakan sebelum angka desimal atau di antara satuan uang (rupiah dan sen) yang dinyatakan dengan angka.

13. Digunakan untuk mengapit keterangan tambahan atau keterangan aposisi.
14. Dapat digunakan di belakang keterangan yang terdapat pada awal kalimat untuk menghindari salah pengertian.

2.1.3 Tanda Tanya (?)

Berikut merupakan fungsi dari penggunaan tanda tanya sesuai dengan EYD V [21]:

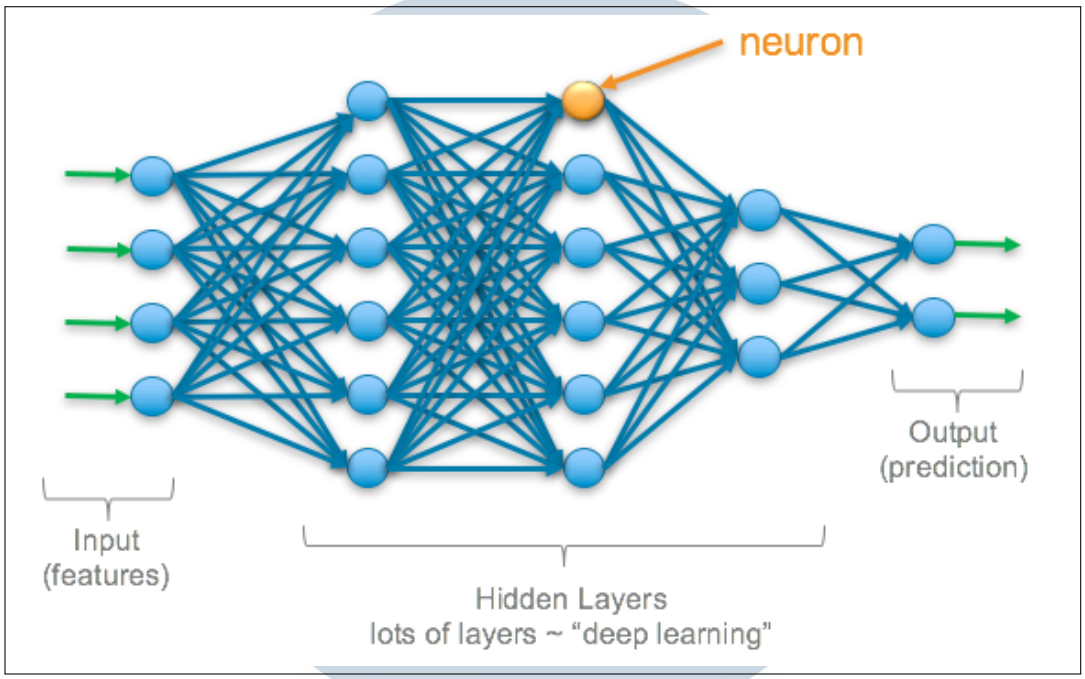
1. Digunakan pada akhir kalimat tanya.
2. Digunakan di dalam tanda kurung untuk menyatakan bagian kalimat yang diragukan atau yang kurang dapat dibuktikan kebenarannya.

2.1.4 Tanda Seru (!)

Berikut merupakan fungsi dari penggunaan tanda seru sesuai dengan EYD V [22]:

1. Digunakan untuk mengakhiri ungkapan yang menggambarkan kekaguman, kesungguhan, emosi yang kuat, seruan, atau perintah.

2.2 Deep Learning



Gambar 2.1. Ilustrasi konsep Deep Learning

Sumber: [23]

Deep Learning merupakan salah satu pendekatan modern dalam pembelajaran mesin yang memanfaatkan jaringan saraf tiruan dengan banyak lapisan. [24] Pendekatan ini memungkinkan komputer untuk mempelajari representasi data dalam berbagai tingkat kompleksitas, mulai dari fitur permukaan hingga pola abstrak yang tersembunyi. Melalui proses pelatihan berulang dan pemanfaatan sejumlah besar data, deep learning mampu menggeneralisasi pola dan menghasilkan prediksi yang akurat.

Struktur jaringan berlapis yang digunakan dalam deep learning dapat dilihat pada Gambar 2.1, yang menggambarkan bagaimana informasi diproses secara bertahap melalui lapisan-lapisan untuk membentuk pemahaman yang lebih kompleks terhadap data masukan.

Teknologi ini telah menjadi landasan bagi berbagai kemajuan dalam bidang kecerdasan buatan, termasuk pemrosesan bahasa alami, pengenalan suara, dan visi komputer. Pendekatan deep learning banyak digunakan dalam sistem yang menuntut pemahaman konteks dan struktur informasi secara utuh, terutama ketika pola-pola dalam data bersifat sekuensial atau kontekstual. Ini membuka

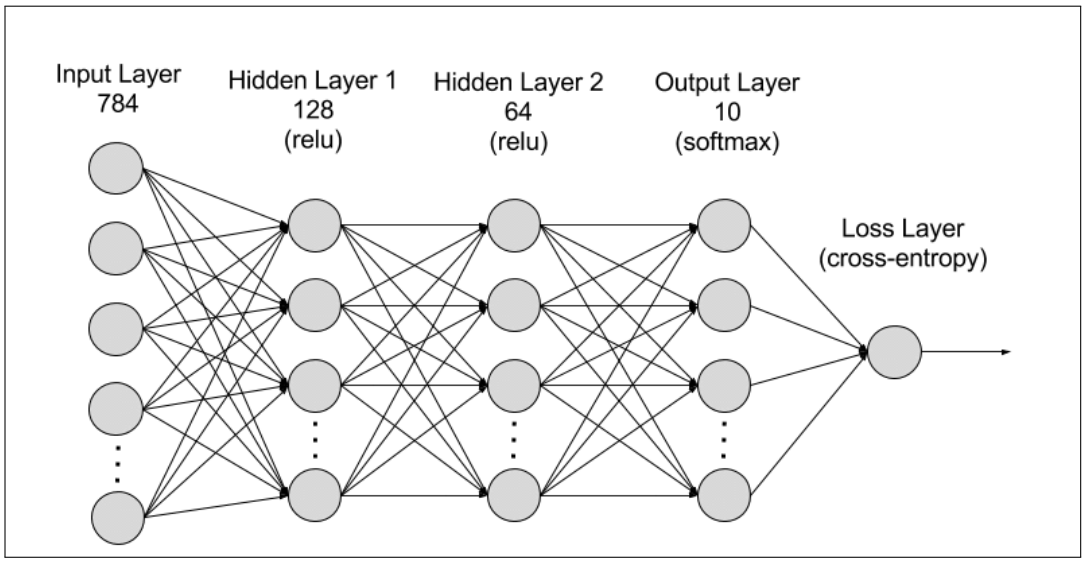
peluang untuk menerapkan model dalam berbagai bidang linguistik komputasional, termasuk analisis struktur kalimat secara mendalam.

2.3 Natural Language Processing (NLP)

Natural Language Processing (NLP) adalah bidang interdisipliner yang menggabungkan linguistik, ilmu komputer, dan pembelajaran mesin untuk memungkinkan komputer memahami dan menghasilkan bahasa manusia. [25] NLP bertumpu pada analisis struktur sintaksis dan semantik teks, termasuk segmentasi kata, pengenalan entitas, serta pemodelan konteks dalam kalimat.

Dengan kemajuan teknik pembelajaran mesin modern, pemrosesan teks kini dapat dilakukan secara otomatis dan efisien, baik pada data terstruktur maupun tidak terstruktur. Representasi linguistik seperti word embedding dan representasi kontekstual semakin memperkuat performa model-model berbasis jaringan saraf, terutama ketika berhadapan dengan nuansa bahasa yang kompleks dan bervariasi antar domain atau genre teks.

2.4 Recurrent Neural Network (RNN)



Gambar 2.2. Arsitektur Recurrent Neural Network
Sumber: [26]

Recurrent Neural Network (RNN) yang sesuai dengan Gambar 2.2 dirancang untuk menangani data berurutan dengan memanfaatkan hubungan

antar waktu. [27] Arsitektur RNN menyimpan informasi sebelumnya dalam bentuk *hidden state* yang diperbarui pada setiap langkah waktu. Karakteristik ini memungkinkan RNN mengenali pola temporal dan ketergantungan dalam rangkaian input, menjadikannya efektif dalam pemodelan bahasa.

Persamaan umum dari jaringan Recurrent Neural Network (RNN) dapat dituliskan sebagai berikut:

$$h_t = \tanh(Wx_t + Uh_{t-1} + b) \tag{2.1}$$

Dengan rincian komponen sebagai berikut:

- $h_t \in \mathbb{R}^{n_h}$: vektor status tersembunyi (hidden state) pada waktu ke-t.
- $x_t \in \mathbb{R}^{n_x}$: vektor input pada waktu ke-t.
- $h_{t-1} \in \mathbb{R}^{n_h}$: vektor status tersembunyi dari waktu sebelumnya.
- $W \in \mathbb{R}^{n_h \times n_x}$: matriks bobot yang mengalikan input x_t.
- $U \in \mathbb{R}^{n_h \times n_h}$: matriks bobot yang mengalikan status tersembunyi sebelumnya h_{t-1}.
- $b \in \mathbb{R}^{n_h}$: vektor bias.
- tanh: fungsi aktivasi non-linear hiperbolik, dengan rentang output antara -1 hingga 1.

Sebagai contoh, misalkan diberikan parameter dan input sebagai berikut:

$$x_t = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad h_{t-1} = \begin{bmatrix} 0.5 \\ -0.5 \end{bmatrix}, \quad W = \begin{bmatrix} 0.1 & 0.3 \\ 0.2 & 0.4 \end{bmatrix}, \quad U = \begin{bmatrix} 0.5 & -0.1 \\ -0.3 & 0.2 \end{bmatrix}, \quad b = \begin{bmatrix} 0.1 \\ 0.2 \end{bmatrix}$$

Langkah-langkah perhitungan sebagai berikut:

1. Hitung Wx_t:

$$Wx_t = \begin{bmatrix} 0.1 & 0.3 \\ 0.2 & 0.4 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 0.1(1) + 0.3(2) \\ 0.2(1) + 0.4(2) \end{bmatrix} = \begin{bmatrix} 0.7 \\ 1.0 \end{bmatrix}$$

2. Hitung Uh_{t-1}:

$$Uh_{t-1} = \begin{bmatrix} 0.5 & -0.1 \\ -0.3 & 0.2 \end{bmatrix} \begin{bmatrix} 0.5 \\ -0.5 \end{bmatrix} = \begin{bmatrix} 0.25 + 0.05 \\ -0.15 - 0.1 \end{bmatrix} = \begin{bmatrix} 0.3 \\ -0.25 \end{bmatrix}$$

3. Jumlahkan dengan bias b:

$$Wx_t + Uh_{t-1} + b = \begin{bmatrix} 0.7 \\ 1.0 \end{bmatrix} + \begin{bmatrix} 0.3 \\ -0.25 \end{bmatrix} + \begin{bmatrix} 0.1 \\ 0.2 \end{bmatrix} = \begin{bmatrix} 1.1 \\ 0.95 \end{bmatrix}$$

4. Terapkan fungsi aktivasi tanh**:**

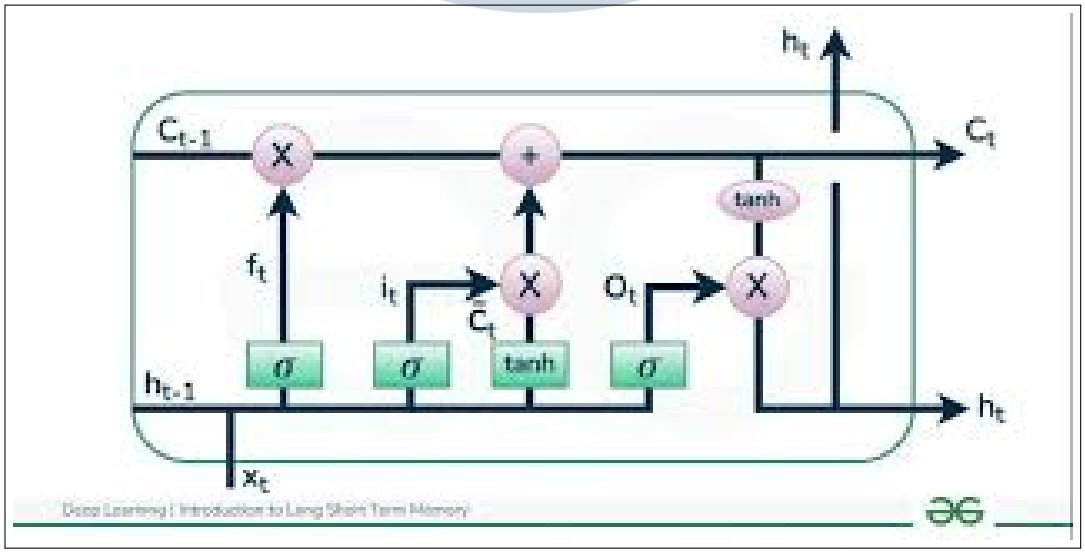
$$h_t = \tanh\left(\begin{bmatrix} 1.1 \\ 0.95 \end{bmatrix}\right) \approx \begin{bmatrix} 0.800 \\ 0.740 \end{bmatrix}$$

Jadi, nilai akhir hidden state h_t adalah:

$$h_t \approx \begin{bmatrix} 0.800 \\ 0.740 \end{bmatrix}$$

2.5 Long Short-Term Memory (LSTM)

2.6 Long Short-Term Memory (LSTM)



Gambar 2.3. Ilustrasi arsitektur Long Short-Term Memory (LSTM)
Sumber: [28]

Long Short-Term Memory (LSTM) merupakan pengembangan dari RNN yang mampu mempertahankan konteks dalam rentang waktu yang lebih panjang. [29] Dengan memperkenalkan tiga gerbang pengontrol (input, forget, dan output

gate), LSTM memungkinkan pemeliharaan informasi penting sambil membuang informasi yang kurang relevan seiring waktu.

Struktur dan mekanisme kerja ketiga gerbang utama tersebut dapat dilihat pada Gambar 2.3, yang menggambarkan bagaimana aliran data dikendalikan dalam unit LSTM untuk mengatasi permasalahan *vanishing gradient* dan mempertahankan informasi jangka panjang secara efektif.

Arsitektur Long Short-Term Memory (LSTM) dirancang untuk mengatasi kelemahan RNN dalam mengingat informasi jangka panjang. LSTM menggunakan struktur internal yang lebih kompleks dengan *cell state* C_t, yang memungkinkan informasi untuk dipertahankan atau dilupakan secara selektif. Persamaan logika internal LSTM dirumuskan sebagai berikut:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{2.2}$$

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{2.3}$$

$$\tilde{C}_t = \tanh(W_C[h_{t-1}, x_t] + b_C) \tag{2.4}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{2.5}$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{2.6}$$

$$h_t = o_t * \tanh(C_t) \tag{2.7}$$

Dengan penjelasan sebagai berikut:

- x_t: input saat waktu ke-t
- h_{t-1}: status tersembunyi dari waktu sebelumnya
- C_{t-1}: status sel sebelumnya (*cell state*)
- f_t: **forget gate**, menentukan informasi mana yang dilupakan dari C_{t-1}
- i_t: **input gate**, menentukan informasi baru mana yang akan disimpan
- $\tilde{C}_t$: kandidat nilai *cell state* baru
- C_t: *cell state* saat ini
- o_t: **output gate**, menentukan bagian mana dari C_t yang menjadi output
- h_t: status tersembunyi saat ini (output LSTM)

- σ: fungsi sigmoid, memberi output antara 0 dan 1
- tanh: fungsi aktivasi hiperbolik
- $[h_{t-1}, x_t]$: konkatenasi (penggabungan) dari h_{t-1} dan x_t
- W_f, W_i, W_C, W_o: bobot untuk masing-masing gate
- b_f, b_i, b_C, b_o: bias masing-masing gate

Sebagai contoh, diberikan input dan parameter sebagai berikut:

$$x_t = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad h_{t-1} = \begin{bmatrix} 0.5 \\ 0.1 \end{bmatrix}, \quad C_{t-1} = \begin{bmatrix} 0.4 \\ -0.3 \end{bmatrix}$$

Anggap $W_f = W_i = W_C = W_o = I$ (identitas), dan semua bias $b_* = \vec{0}$ untuk menyederhanakan:
Gabungkan input dan hidden state:

$$[h_{t-1}, x_t] = \begin{bmatrix} 0.5 \\ 0.1 \\ 1 \\ 0 \end{bmatrix}$$

Untuk mempermudah, kita anggap bobot melakukan pemetaan langsung ke dua dimensi output (2 unit LSTM). Maka:

$$f_t = \sigma([0.5, 0.1]) \approx \begin{bmatrix} 0.622 \\ 0.525 \end{bmatrix}$$

$$i_t = \sigma([0.5, 0.1]) \approx \begin{bmatrix} 0.622 \\ 0.525 \end{bmatrix}$$

$$\tilde{C}_t = \tanh([0.5, 0.1]) \approx \begin{bmatrix} 0.462 \\ 0.100 \end{bmatrix}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

$$= \begin{bmatrix} 0.622 \\ 0.525 \end{bmatrix} * \begin{bmatrix} 0.4 \\ -0.3 \end{bmatrix} + \begin{bmatrix} 0.622 \\ 0.525 \end{bmatrix} * \begin{bmatrix} 0.462 \\ 0.100 \end{bmatrix}$$

$$= \begin{bmatrix} 0.2488 \\ -0.1575 \end{bmatrix} + \begin{bmatrix} 0.2875 \\ 0.0525 \end{bmatrix}$$

$$
\begin{aligned}
&= \begin{bmatrix} 0.5363 \\ -0.1050 \end{bmatrix} \\
o_t &= \sigma([0.5, 0.1]) \approx \begin{bmatrix} 0.622 \\ 0.525 \end{bmatrix} \\
h_t &= o_t * \tanh(C_t) \\
&\approx \begin{bmatrix} 0.622 \\ 0.525 \end{bmatrix} * \tanh\left(\begin{bmatrix} 0.5363 \\ -0.1050 \end{bmatrix}\right) \\
&\approx \begin{bmatrix} 0.622 \\ 0.525 \end{bmatrix} * \begin{bmatrix} 0.489 \\ -0.104 \end{bmatrix} \\
&= \begin{bmatrix} 0.304 \\ -0.055 \end{bmatrix}
\end{aligned} \tag{2.8}
$$

Jadi, hasil akhir hidden state h_t adalah:

$$
h_t \approx \begin{bmatrix} 0.304 \\ -0.055 \end{bmatrix}, \quad C_t \approx \begin{bmatrix} 0.5363 \\ -0.1050 \end{bmatrix}
$$

LSTM unggul dalam memahami konteks yang tersebar luas dalam suatu rangkaian teks, terutama ketika sinyal penting tidak terletak berdekatan satu sama lain. Hal ini menjadikannya pilihan utama dalam pemodelan bahasa yang kompleks.

2.7 Dot Product

Dot product secara umum didefinisikan sebagai:

$$
\mathbf{a} \cdot \mathbf{b} = |\mathbf{a}| \times |\mathbf{b}| \times \cos(\theta) \tag{2.9}
$$

Dalam persamaan tersebut, **a** dan **b** merupakan vektor dalam ruang berdimensi n, sedangkan $|\mathbf{a}|$ dan $|\mathbf{b}|$ masing-masing menyatakan panjang (norma) dari vektor tersebut. Simbol θ menyatakan sudut antara kedua vektor, dan operasi $\cdot$ menunjukkan perkalian dot product. Rumus ini menunjukkan bahwa nilai dot product mencapai maksimum apabila kedua vektor memiliki arah yang sama ($\theta = 0^\circ$) karena $\cos(0^\circ) = 1$. Sebaliknya, apabila kedua vektor saling tegak lurus ($\theta = 90^\circ$), nilai dot product menjadi nol karena $\cos(90^\circ) = 0$. Oleh karena itu, dalam banyak aplikasi seperti pemrosesan bahasa alami atau klasifikasi berbasis

vektor, dot product digunakan sebagai ukuran kemiripan arah antar dua vektor.

Namun demikian, dalam konteks penelitian ini, dot product disesuaikan untuk hanya memperhitungkan pasangan vektor yang berasal dari dua klaster berbeda. Aturannya adalah sebagai berikut: pertama, hanya pasangan antara klaster 1 dan klaster 2 yang diperbolehkan untuk dihitung; kedua, urutan perhitungan harus berupa $a \in$ klaster 1 dan $b \in$ klaster 2; dan ketiga, pasangan dari klaster yang sama, seperti $(\text{klaster 1}, \text{klaster 1})$ atau $(\text{klaster 2}, \text{klaster 2})$, tidak diperhitungkan dalam analisis.

Dengan menyesuaikan ketentuan tersebut, maka operasi dot product antar klaster dituliskan sebagai berikut:

$$A \odot B = a_1 \times b_2, \quad \text{dengan } a_1 \in \text{klaster 1},\ b_2 \in \text{klaster 2} \tag{2.10}$$

Dalam hal ini, $A = [a_1, a_2]$ merupakan vektor representasi dengan elemen yang berasal dari dua klaster, sementara $B = [b_1, b_2]$ adalah vektor pembanding yang juga terdiri dari elemen yang mewakili dua klaster. Simbol $\odot$ digunakan untuk menunjukkan bahwa ini bukan dot product standar, melainkan khusus untuk konteks antar klaster. Hanya elemen a_1 yang berasal dari klaster 1 dan b_2 dari klaster 2 yang digunakan dalam perhitungan, sedangkan elemen lain diabaikan. Sebagai ilustrasi, misalkan diberikan vektor:

$$A = [3, 3], \quad B = [3, 3]$$

di mana elemen berwarna hitam berasal dari klaster yang berbeda dan elemen abu-abu berasal dari klaster yang sama. Maka, sesuai aturan yang telah ditetapkan, hanya elemen dari klaster berbeda yang diperhitungkan. Oleh karena itu, hasil perhitungan dot product antar klaster adalah sebagai berikut:

$$A \odot B = a_1 \times b_2 = 3 \times 3 = 9 \tag{2.11}$$

Dengan demikian, nilai akhir dari operasi dot product antar klaster pada contoh tersebut adalah sebesar 9.

2.8 Random Search

Random Search merupakan teknik eksplorasi hyperparameter yang dilakukan dengan memilih kombinasi nilai-nilai parameter secara acak dalam ruang pencarian yang telah ditentukan. [30] Teknik ini dinilai lebih efisien daripada *Grid Search* karena tidak membuang waktu untuk mengevaluasi kombinasi yang kurang informatif.

Random search bekerja dengan prinsip probabilistiksemakin luas ruang pencarian dan semakin banyak iterasi, semakin besar kemungkinan menemukan konfigurasi optimal. Dalam penerapannya, pendekatan ini sering digunakan pada proses tuning model berbasis jaringan saraf karena jumlah hyperparameter dan interaksinya yang kompleks.

2.9 Rectified Linear Unit (ReLU)

Rectified Linear Unit (ReLU) adalah fungsi aktivasi non-linear yang umum digunakan dalam jaringan saraf. Fungsi ini memetakan nilai negatif menjadi nol dan mempertahankan nilai positif, dituliskan sebagai:

$$f(x) = \max(0, x) \tag{2.12}$$

ReLU memiliki kelebihan dalam kecepatan komputasi dan efisiensi gradien selama proses pembelajaran. Berkat sifatnya yang tidak menyebabkan saturasi untuk input positif, ReLU mendorong konvergensi lebih cepat dan mencegah permasalahan vanishing gradient yang kerap menghambat pelatihan jaringan saraf dalam arsitektur yang dalam.

2.10 Sigmoid

Sigmoid adalah fungsi aktivasi yang memetakan nilai input menjadi rentang antara 0 dan 1, sering digunakan dalam output layer untuk tugas klasifikasi biner. Fungsi ini dirumuskan sebagai:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.13}$$

Meskipun sigmoid bermanfaat dalam mengekspresikan probabilitas, fungsi ini memiliki kelemahan dalam bentuk gradien yang sangat kecil pada nilai ekstrem. Oleh karena itu, penggunaannya sering dibatasi pada lapisan keluaran, sementara lapisan tersembunyi lebih sering menggunakan ReLU atau fungsi aktivasi lain yang lebih stabil.

2.11 *Receiver Operating Characteristic* (ROC) Curve

Receiver Operating Characteristic (ROC) Curve adalah salah satu metode evaluasi yang digunakan untuk mengukur performa model klasifikasi, terutama dalam konteks binary classification. ROC curve menggambarkan hubungan antara *True Positive Rate* (TPR) dan *False Positive Rate* (FPR) pada berbagai nilai ambang (*threshold*) yang berbeda.

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.14}$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{2.15}$$

Dalam hal ini, TP (True Positive) adalah jumlah data positif yang diklasifikasikan dengan benar sebagai positif, FN (False Negative) adalah data positif yang salah diklasifikasikan sebagai negatif, FP (False Positive) adalah data negatif yang salah diklasifikasikan sebagai positif, dan TN (True Negative) adalah data negatif yang benar diklasifikasikan.

ROC curve dibentuk dengan memplot TPR terhadap FPR untuk setiap kemungkinan nilai ambang. Model yang ideal akan menghasilkan kurva yang menjulur ke sudut kiri atas grafik, di mana TPR mendekati 1 dan FPR mendekati 0.

2.11.1 Luas Area di Bawah Kurva (AUC)

Salah satu indikator kuantitatif dari performa ROC curve adalah nilai AUC (Area Under Curve). Nilai AUC berkisar antara 0 hingga 1, di mana:

- AUC = 1 menandakan model sempurna,
- AUC = 0.5 menandakan model sama baiknya dengan tebakan acak,
- AUC ¡ 0.5 berarti performa lebih buruk dari acak.

2.11.2 Keunggulan ROC Curve Menurut Penelitian Sebelumnya

Penggunaan ROC curve sebagai metrik evaluasi telah dibuktikan lebih robust oleh sejumlah penelitian terdahulu. Dalam penelitian NLP [31], ROC curve juga digunakan untuk mengevaluasi kemampuan model dalam membedakan antar kelas dalam situasi yang tidak seimbang. Model yang memiliki akurasi tinggi belum tentu memiliki performa ROC yang baik, terutama ketika dataset didominasi oleh satu kelas. Oleh karena itu, AUC ROC menjadi indikator penting dalam mengukur kualitas prediksi secara menyeluruh.

2.11.3 Implementasi pada Penelitian Ini

Dalam penelitian ini, ROC curve digunakan untuk mengevaluasi kemampuan model dalam mendeteksi kesalahan tanda baca. Karena proporsi antara kelas benar dan kelas salah tidak selalu seimbang, metrik ROC dan AUC dipilih agar dapat memberikan gambaran yang lebih adil terhadap kinerja klasifikasi, dibandingkan hanya mengandalkan akurasi atau presisi saja.

2.12 Classification Report

Classification Report adalah representasi ringkas dari performa model klasifikasi terhadap berbagai kelas dalam data. [32] Laporan ini menampilkan metrik penting berikut:

- **Precision**:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.16}$$

  Menunjukkan ketepatan model dalam memprediksi kelas positif.

- **Recall**:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.17}$$

  Menggambarkan seberapa banyak data aktual kelas positif yang berhasil ditemukan oleh model.

- **F1-Score**:

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.18}$$

  Merupakan harmoni dari precision dan recall.

- **Support**: jumlah total data aktual dari masing-masing kelas.

Penggunaan classification report membantu dalam mengevaluasi ketidakseimbangan antar kelas serta mengidentifikasi kelemahan model dalam mengenali kategori tertentu. Analisis ini menjadi penting untuk memahami apakah suatu sistem cenderung mengabaikan atau terlalu sering salah mengklasifikasikan kondisi tertentu yang berdampak pada keakuratan prediksi secara keseluruhan.