Latihan Hitungan NLP (IF5153)

Materi, rumus, arti simbol, contoh soal, dan soal penuh
untuk ujian tanpa laptop

Dokumen ini mengumpulkan semua perhitungan dari materi UTS NLP (catatan 1, 2, 3, 5, 6, 7). Tujuannya agar kamu bisa belajar memahami *dan* mengerjakan soal hitungan dengan kertas, pensil, dan kalkulator scientific.

Susunan tiap topik. Pertama, kotak **Materi** menjelaskan konsepnya: apa masalahnya, apa idenya, dan bagaimana langkahnya. Kedua, kotak **Rumus** memuat rumus beserta tabel **arti simbol**. Ketiga, ada **contoh soal** pendek dengan **pembahasan** yang menjelaskan alasan tiap langkah. Keempat, tiap bab ditutup dengan **Soal Penuh**: soal bertahap seperti Exercise on Transformer Encoder, yang menggabungkan beberapa rumus dalam satu kasus.

Penulisan angka. Desimal ditulis dengan *titik* (0.5), dan elemen vektor dipisah *koma* (1, 0, 1), sama seperti soal latihan aslinya.

Peringatan

Isi ujian sebenarnya tidak diketahui. Daftar hitungan disusun dari rumus yang ada di catatan dan pola dua latihan (Exercise on Transformer Encoder dan Exercise for Sentence Comparison). Bagian bertanda **(tambahan)** adalah rumus atau angka yang *tidak* ada di slide, misalnya LayerNorm, Naive Bayes, dan ukuran vocabulary BERT. Bagian itu dimasukkan karena kemungkinan berguna, tetapi jangan dianggap pasti keluar. Semua angka jawaban sudah dihitung ulang dengan komputer.

Kebiasaan Kalkulator

Hal	Aturan
Mode sudut	**Radian** (RAD) untuk sin dan cos pada positional encoding. Mode derajat memberi angka salah.
log vs ln	Tombol `log` berbasis 10, `ln` berbasis e. Slide menulis $\log(N/\text{df})$ tanpa basis, jadi **tulis basis yang kamu pakai**. Loss (cross-entropy) memakai ln.
Log basis 2	$\log_2 x = \ln x / \ln 2$ (kalkulator biasanya tidak punya tombol $\log_2$).
Pembulatan	Simpan 4 angka di belakang koma selama menghitung, bulatkan di akhir. Hasilmu boleh berbeda di digit terakhir karena pembulatan.
Softmax	Hitung e^x tiap elemen, jumlahkan, lalu bagi. Cek: hasil berjumlah 1.
Dua kelas	Softmax dua kelas $= \dfrac{1}{1+e^{-\Delta}}$ dengan Δ selisih logit.
Hemat kerja	Jika yang ditanya hanya `[CLS]`, hitung **hanya baris Query `[CLS]`**; K dan V tetap semua baris.

Daftar Isi

1 Klasifikasi Teks **4**

1.1 Word Segmentation: Maximum Matching 4
1.2 TF-IDF 5
1.3 Mutual Information (pemilihan fitur) 5
1.4 Naive Bayes (tambahan, prasyarat) 6
1.5 CNN untuk Teks 7
1.6 Evaluasi: Precision, Recall, F1, Accuracy (tambahan) 8
1.7 Soal Penuh: Filter Spam Mini (preprocessing sampai evaluasi) 9

2 Word Representation **11**
2.1 Cosine Similarity 11
2.2 SVD dan Latent Semantic Indexing (LSI) 12
2.3 Soal Penuh: SVD dan LSI pada Matriks Kecil 13
2.4 HAL (Hyperspace Analogue to Language) 14
2.5 Soal Penuh: HAL dari Teks sampai Kemiripan Dokumen 15
2.6 LDA (Latent Dirichlet Allocation) 16
2.7 Soal Penuh: LDA: Peluang Kata, Vektor, dan Satu Langkah Gibbs 17
2.8 GloVe (Global Vectors) 18
2.9 Soal Penuh: GloVe: Loss dan Satu Update 19
2.10 Word2Vec (CBOW dan Skip-gram) 20
2.11 Soal Penuh: Word2Vec: Pasangan Latih, Forward Pass, Loss, dan Analogi 21

3 Transformer **22**
3.1 Softmax dan Cross-Entropy 23
3.2 Positional Encoding 23
3.3 Scaled Dot-Product Attention (satu head) 24
3.4 Masked Self-Attention 25
3.5 Multi-Head: Dimensi dan Jumlah Parameter 26
3.6 LayerNorm (tambahan) 26
3.7 Soal Penuh: Transformer Encoder-Only tanpa LayerNorm (Exercise) 27
3.8 Soal Penuh: Encoder yang Sama dengan LayerNorm (tambahan) 29
3.9 Soal Penuh: Satu Langkah Decoder (masked self-attention, cross-attention, keluaran) 30
3.10 Kompleksitas Self-Attention vs RNN 31

4 BERT **32**
4.1 Tokenisasi WordPiece 32
4.2 Hitungan Ukuran BERT 33
4.3 Soal Penuh: BERT dari Masukan sampai Kepala Klasifikasi 34

5 Decoder, Prompting, dan PEFT **35**
5.1 Self-Consistency 36

5.2 LoRA (Low-Rank Adaptation) 36
5.3 Adapter 37
5.4 $(IA)^3$, Prefix-Tuning, dan Selective PEFT 38
5.5 Soal Penuh: Membandingkan Biaya Metode PEFT 39

6 Sentence Comparison **39**
6.1 Cosine Leksikal 40
6.2 Pooling Sentence Embedding 40
6.3 Bi-encoder dengan Classifier (gaya SBERT) 41
6.4 Loss Kontrastif dan Temperatur τ 42
6.5 Skor Hybrid BGE-M3 42
6.6 Biaya Cross- vs Bi-encoder dan Zero-shot NLI 43
6.7 Soal Penuh: SimCSE dengan In-batch Negative 43
6.8 Soal Penuh: BGE-M3 Hybrid Retrieval (Exercise for Sentence Comparison) 44

7 Rumus Cepat **47**

8 Sengaja Tidak Dimasukkan **48**

1 Klasifikasi Teks

Materi: alur klasifikasi teks

Text classification memberi *satu label untuk seluruh teks*, misalnya email menjadi *spam* atau *ham* (bukan spam), atau ulasan menjadi *positif* atau *negatif*. Komputer tidak bisa langsung membaca kata, jadi teks harus diubah menjadi angka dulu. Karena itu alurnya selalu tiga tahap:

1. **Preprocessing**: membersihkan dan memecah teks. Contohnya *tokenization* (memecah kalimat menjadi token), *lowercase*, *stop word elimination* (membuang kata umum seperti "the", "is"), *stemming* (memotong kata ke stem, *writing*→*writ*), *lemmatization* (mengubah ke bentuk dasar yang valid, *writing*→*write*), dan *entity masking* (nomor telepon atau URL diganti penanda).
2. **Feature extraction**: mengubah token menjadi vektor angka. Cara paling sederhana adalah *bag of words*: tiap kata di vocabulary menjadi satu kolom, urutan kata diabaikan. Kumpulan vektor dokumen membentuk matriks *term* × *document*, dan cara memandang dokumen sebagai titik di ruang kata ini disebut *Vector Space Model*. Isi sel bisa *boolean* (1 jika kata ada), *TF* (berapa kali muncul), atau *TF-IDF* (diberi bobot kelangkaan). Karena kolomnya bisa ribuan, fitur dikurangi dengan memilih kata berskor tinggi (TF-IDF atau Mutual Information).
3. **Classification**: vektor fitur dimasukkan ke algoritma machine learning, misalnya Naive Bayes, SVM, Decision Tree, atau neural network seperti CNN dan RNN/LSTM.

Kelemahan bag of words: urutan kata hilang, dan kata bersinonim dianggap tidak berhubungan. Masalah ini yang nanti diselesaikan oleh word representation (Bab 2).

1.1 Word Segmentation: Maximum Matching

Materi

Bahasa seperti Mandarin tidak memakai spasi, sehingga tokenization butuh *word segmentation*. **Maximum Matching** adalah baseline yang *greedy* (rakus): dari posisi pointer, ambil kata kamus *terpanjang* yang cocok, pindahkan pointer ke akhir kata itu, lalu ulangi. Karena selalu memilih yang terpanjang tanpa melihat ke depan, hasilnya bisa salah.

Soal

Kamus: {the, theta, table, bled, down, own, there, he, re}. Segmentasikan `thetabledownthere` dengan Maximum Matching. Apakah hasilnya benar?

Pembahasan

1. Pointer di awal. Kata kamus yang cocok di awal: *the* (3 huruf) dan *theta* (5 huruf). Ambil yang terpanjang: **theta**.
2. Sisa `bledownthere`: yang cocok hanya **bled**.
3. Sisa `ownthere`: **own**.
4. Sisa `there`: **there** (lebih panjang dari *the*).

Hasil: *theta bled own there*. Hasil yang benar adalah *the table down there*. Jadi Maximum Matching **salah** di sini, karena memilih *theta* yang terpanjang di langkah pertama tanpa melihat bahwa sisanya jadi tidak bermakna.

1.2 TF-IDF

Materi

Ide TF-IDF: sebuah kata penting untuk sebuah dokumen jika kata itu *sering* muncul di dokumen tersebut (TF tinggi) tetapi *jarang* muncul di dokumen lain (IDF tinggi). Kata seperti "the" sering di semua dokumen, jadi IDF-nya kecil dan bobotnya rendah. TF-IDF dipakai sebagai isi sel matriks term × document dan untuk memilih fitur.

Rumus

$$w(x,y) = \mathrm{tf}(x,y) \times \mathrm{idf}(x), \qquad \mathrm{idf}(x) = \log \frac{N}{\mathrm{df}_x}$$

Arti simbol

Simbol	Arti	Contoh / ukuran
$w(x,y)$	bobot kata x di dokumen y	ball: $2 \times 0.301 = 0.602$
$\mathrm{tf}(x,y)$	*term frequency*: berapa kali kata x muncul di dokumen y	ball: 2
df_x	*document frequency*: berapa dokumen yang memuat kata x (bukan berapa kali muncul)	ball: 2 dokumen
N	jumlah seluruh dokumen di korpus	4
$\mathrm{idf}(x)$	*inverse document frequency*: makin langka, makin besar	ball: $\log_{10}(4/2) = 0.301$

Variasi idf di slide: $1/\mathrm{df}$ atau $\log(N/\mathrm{df})$. Kata yang ada di semua dokumen punya $\log(N/N) = 0$.

Soal

Korpus 4 dokumen. D_1 = "ball player goal ball". Kata *ball* muncul di 2 dokumen, *player* di 3 dokumen, *goal* di 1 dokumen. Hitung TF-IDF tiap kata di D_1 (basis 10).

Pembahasan

Hitung tf dengan membaca D_1, lalu idf dari df, lalu kalikan.

Kata	tf di D_1	df	$\mathrm{idf} = \log_{10}(4/\mathrm{df})$	TF-IDF
ball	2	2	$\log_{10} 2 = 0.3010$	$2 \times 0.3010 = \mathbf{0.6021}$
player	1	3	$\log_{10} 1.333 = 0.1249$	**0.1249**
goal	1	1	$\log_{10} 4 = 0.6021$	**0.6021**

Cek dan jebakan

Soal tanpa basis log. Slide menulis $\log(N/\mathrm{df})$ tanpa menyebut basisnya. Jika soal ujian juga tidak menyebut basis, tulis basis yang kamu pakai. Dengan ln, hasil soal ini menjadi 1.3863; 0.2877; 1.3863: angkanya berbeda, tetapi urutan pentingnya kata tetap sama.
player paling rendah karena paling umum (ada di 3 dari 4 dokumen). *ball* dan *goal* seri meski *ball* muncul dua kali, sebab *goal* lebih langka sehingga idf-nya menutup selisih tf. Jebakan umum: memakai jumlah kemunculan total sebagai df. df adalah jumlah *dokumen*.

1.3 Mutual Information (pemilihan fitur)

Materi

Mutual Information (MI) mengukur seberapa kuat *kemunculan sebuah kata* berkaitan dengan *sebuah label*. Jika kata dan label muncul bersama lebih sering daripada yang diharapkan secara kebetulan, MI positif, dan kata itu fitur yang bagus untuk label tersebut. Kata dengan MI

tertinggi dipilih sebagai fitur.

Rumus

$$\mathrm{MI}(t,l) = \log_2 \frac{p(t,l)}{p(t)\,p(l)} \approx \log_2 \frac{A \cdot N}{(A+B)(A+C)}$$

Arti simbol

Simbol	Arti	Contoh / ukuran
t, l	kata (*term*) dan label	"free", spam
A	jumlah dokumen yang memuat t *dan* berlabel l	2
B	dokumen yang memuat t tetapi *bukan* label l	1
C	dokumen berlabel l tetapi *tidak* memuat t	0
D	dokumen yang tidak memuat t dan bukan label l	1
N	jumlah seluruh dokumen, $N = A+B+C+D$ (slide: "jumlah dokumen berlabel")	4
$A+B$, $A+C$	semua dokumen yang memuat t; semua dokumen berlabel l	3; 2

Tafsiran: MI > 0 berkaitan, MI ≈ 0 tidak berkaitan, MI < 0 saling menjauh. Jika $A = 0$, MI tak terdefinisi ($\log 0$), artinya kata itu tidak pernah muncul bersama label tersebut.

Soal

$N = 100$, $A = 20$, $B = 5$, $C = 10$ (jadi $D = 65$). Hitung MI kata "gratis" terhadap label spam.

Pembahasan

$A + B = 25$ (dokumen berisi "gratis"), $A + C = 30$ (dokumen spam).

$$\mathrm{MI} = \log_2 \frac{20 \times 100}{25 \times 30} = \log_2 \frac{2000}{750} = \log_2 2.667 = \frac{\ln 2.667}{\ln 2} = \frac{0.9808}{0.6931} = \mathbf{1.415}$$

Positif, jadi "gratis" berkaitan kuat dengan spam.

Cek dan jebakan

Jika $A = (A+B)(A+C)/N$ (di sini 7.5), MI tepat 0, artinya kata dan label independen. $A = 20$ jauh di atas 7.5, jadi MI positif.

1.4 Naive Bayes (tambahan, prasyarat)

Materi

Naive Bayes ada di daftar classifier pada slide, tetapi cara hitungnya tidak diajarkan. Idenya: hitung peluang tiap kelas berdasarkan kata-kata di dokumen, dengan anggapan "naif" bahwa kata-kata saling bebas. Kelas dengan skor terbesar dipilih. Karena kata yang belum pernah muncul di sebuah kelas membuat peluangnya 0, dipakai *Laplace smoothing* (add-1).

Rumus

$$\mathrm{skor}(c) = P(c) \prod_i P(w_i \mid c)$$

$$P(w \mid c) = \frac{\mathrm{count}(w,c) + 1}{\mathrm{total}(c) + |V|}$$

Arti simbol

Simbol	Arti	Contoh / ukuran
c	kelas	spam, ham
$P(c)$	prior: proporsi dokumen latih berkelas c	$2/4 = 0.5$
w_i	kata ke-i di dokumen yang diklasifikasi	free
$\text{count}(w, c)$	berapa kali kata w muncul di semua dokumen latih kelas c	2
$\text{total}(c)$	jumlah seluruh kata di dokumen latih kelas c	8
$\lvert V \rvert$	ukuran vocabulary (jumlah kata unik di semua data latih)	10

Skor dinormalkan (dibagi jumlah skor semua kelas) agar menjadi peluang.

Soal

Data latih: spam = {"free discount", "free gift"}, ham = {"meeting today"}. V = {free, discount, gift, meeting, today}, $|V| = 5$. Klasifikasikan "free today".

Pembahasan

Total kata spam = 4 (free×2, discount, gift); ham = 2. Prior: spam 2/3, ham 1/3.

$$\text{spam} = \tfrac{2}{3} \cdot \tfrac{2+1}{4+5} \cdot \tfrac{0+1}{4+5} = \tfrac{2}{3} \cdot \tfrac{3}{9} \cdot \tfrac{1}{9} = 0.02469$$
$$\text{ham} = \tfrac{1}{3} \cdot \tfrac{0+1}{2+5} \cdot \tfrac{1+1}{2+5} = \tfrac{1}{3} \cdot \tfrac{1}{7} \cdot \tfrac{2}{7} = 0.01361$$

Normalisasi: $P(\text{spam}) = 0.02469/(0.02469+0.01361) = \mathbf{0.645}$, $P(\text{ham}) = \mathbf{0.355}$. Prediksi: **spam**.

Cek dan jebakan

Tanpa add-1, $P(\text{today} \mid \text{spam}) = 0$ dan skor spam langsung 0. Untuk banyak kata, perkalian peluang jadi sangat kecil; jumlahkan ln peluangnya saja (kelas dengan jumlah ln terbesar menang).

1.5 CNN untuk Teks

Materi

CNN (Yoon Kim, 2014) memperlakukan kalimat sebagai matriks $n \times k$: n kata, tiap baris vektor kata berdimensi k. Sebuah *filter* selebar h kata digeser dari atas ke bawah. Di tiap posisi, filter melihat h kata berurutan (seperti n-gram) dan menghasilkan satu angka. Deretan angka itu disebut *feature map*. Lalu *max-over-time pooling* mengambil nilai terbesar dari feature map, artinya "seberapa kuat pola filter ini muncul di mana pun dalam kalimat". Tiap filter menyumbang satu angka; semua angka digabung lalu masuk ke layer fully connected dan softmax.

Rumus

$$c_i = f\big(\mathbf{w} \cdot \mathbf{x}_{i:i+h-1} + b\big), \qquad \hat{c} = \max_i c_i, \qquad \text{panjang feature map} = n - h + 1$$

Arti simbol

Simbol	Arti	Contoh / ukuran
$\mathbf{x}_{i:i+h-1}$	gabungan (konkatenasi) vektor kata ke-i sampai ke-$(i+h-1)$	$h=2$, $k=2$: 4 angka
$\mathbf{w}$, b	bobot filter (panjang $h \cdot k$) dan bias	$(1, 0, -1, 1)$
f	fungsi aktivasi, biasanya ReLU: $\max(0, x)$	
c_i	nilai feature map di posisi i	
$\hat{c}$	hasil max-over-time pooling (satu angka per filter)	
n, h	jumlah kata, lebar filter	$n=4$, $h=2$

Soal

Kalimat 4 kata dengan vektor $x_1 = (1, 0)$, $x_2 = (0, 1)$, $x_3 = (1, 1)$, $x_4 = (2, 0)$. Dua filter lebar $h = 2$, bias 0, ReLU: $\mathbf{w}_1 = (1, 0, -1, 1)$ dan $\mathbf{w}_2 = (0, 1, 1, 0)$. Hitung feature map, hasil pooling, lalu logits dengan $W = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$ dan softmax-nya.

Pembahasan

Langkah 1, susun jendela. Panjang feature map $= 4-2+1 = 3$. Jendela: $[x_1; x_2] = (1, 0, 0, 1)$, $[x_2; x_3] = (0, 1, 1, 1)$, $[x_3; x_4] = (1, 1, 2, 0)$.
Langkah 2, filter 1. $(1, 0, 0, 1) \cdot (1, 0, -1, 1) = 2$; $(0, 1, 1, 1) \cdot \mathbf{w}_1 = 0$; $(1, 1, 2, 0) \cdot \mathbf{w}_1 = 1-2 = -1$. Setelah ReLU: $(2, 0, 0)$. Pooling: $\hat{c}_1 = 2$.
Langkah 3, filter 2. Hasil dot: 0, 2, 3; ReLU tidak mengubah. Pooling: $\hat{c}_2 = 3$.
Langkah 4, klasifikasi. Fitur $(2, 3)$. Logits $= (2, 3)W = (2-3, -2+3) = (-1, 1)$. Softmax: $1/(1+e^2) =$ **0.119** dan **0.881**.

Cek dan jebakan

Soal ukuran sering muncul: kalimat $n = 7$ kata, filter lebar 3, 4, 5 masing-masing 100 buah. Panjang feature map: 5, 4, 3. Setelah pooling tiap filter memberi satu angka, jadi vektor akhir berukuran $3 \times 100 = 300$, tidak bergantung panjang kalimat.

1.6 Evaluasi: Precision, Recall, F1, Accuracy (tambahan)

Materi

Setelah classifier jadi, kinerjanya diukur dengan membandingkan prediksi dan label asli. Untuk kelas positif (mis. spam): TP = spam yang ditebak spam, FP = ham yang ditebak spam, FN = spam yang ditebak ham, TN = ham yang ditebak ham. *Precision* menjawab "dari yang ditebak spam, berapa yang benar?". *Recall* menjawab "dari semua spam, berapa yang tertangkap?". F1 menggabungkan keduanya.

Rumus

$$P = \frac{TP}{TP+FP}, \quad R = \frac{TP}{TP+FN}, \quad F_1 = \frac{2PR}{P+R}, \quad \text{Acc} = \frac{TP+TN}{TP+FP+FN+TN}$$

Soal

Filter spam: $TP = 40$, $FP = 10$, $FN = 20$, $TN = 30$. Hitung semuanya.

Pembahasan

$P = 40/50 = 0.80$; $R = 40/60 = 0.667$; $F_1 = \dfrac{2(0.80)(0.667)}{0.80 + 0.667} = 0.727$; Acc $= 70/100 = 0.70$.

Cek dan jebakan

F_1 selalu di antara P dan R, lebih dekat ke yang lebih kecil.

1.7 Soal Penuh: Filter Spam Mini (preprocessing sampai evaluasi)

Soal penuh

Empat SMS latih:

Dok	Teks	Label
D_1	"Win a FREE prize now"	spam
D_2	"Claim your free gift now"	spam
D_3	"Are you free for lunch today"	ham
D_4	"Meeting is moved to today"	ham

Stop word list: {a, your, are, you, for, is, to}.

1. Lakukan lowercase dan stop word elimination. Tuliskan vocabulary V.
2. Tuliskan matriks term × document (boolean).
3. Hitung df, idf ($\log_{10}$), dan vektor TF-IDF D_1 dan D_2. Kata apa yang paling penting di D_1?
4. Hitung cosine similarity D_1 dan D_2 dengan vektor boolean dan dengan vektor TF-IDF. Jelaskan bedanya.
5. Hitung MI kata "free" dan "now" terhadap label spam. Mana fitur yang lebih baik?
6. Dengan Naive Bayes (add-1), klasifikasikan "Free prize now!" dan "Free lunch today?".
7. Pada 6 SMS uji, label asli: S, S, S, H, H, H dan prediksi: S, S, H, S, H, H (S = spam, H = ham). Hitung precision, recall, F1, dan accuracy untuk kelas spam.

Pembahasan soal penuh

1. Preprocessing.
D_1: win free prize now. D_2: claim free gift now. D_3: free lunch today. D_4: meeting moved today.
V = {win, free, prize, now, claim, gift, lunch, today, meeting, moved}, $|V| = 10$.

2. Matriks term × document (1 = kata ada di dokumen).

	win	free	prize	now	claim	gift	lunch	today	meeting	moved
D_1	1	1	1	1	0	0	0	0	0	0
D_2	0	1	0	1	1	1	0	0	0	0
D_3	0	1	0	0	0	0	1	1	0	0
D_4	0	0	0	0	0	0	0	1	1	1
df	1	3	1	2	1	1	1	2	1	1

(Matriks ditulis dokumen × term agar muat; isinya sama.)

3. idf dan TF-IDF.
$N = 4$. idf: df $= 1 \Rightarrow \log_{10} 4 = 0.602$; df $= 2 \Rightarrow \log_{10} 2 = 0.301$; df $= 3 \Rightarrow \log_{10}(4/3) = 0.125$. Semua tf $= 1$, jadi TF-IDF $=$ idf.

- D_1: win 0.602, free 0.125, prize 0.602, now 0.301.
- D_2: claim 0.602, free 0.125, gift 0.602, now 0.301.

Kata terpenting di D_1: **win** dan **prize** (seri, 0.602) karena hanya muncul di D_1. "free" paling kecil karena ada di 3 dokumen.

4. Cosine.
Kedua dokumen hanya berbagi "free" dan "now".

- Boolean: $u \cdot v = 2$, $\|u\| = \|v\| = \sqrt{4} = 2$, jadi cos $= 2/4 =$ **0.5**.
- TF-IDF: $u \cdot v = 0.125^2 + 0.301^2 = 0.0156 + 0.0906 = 0.1062$. $\|u\|^2 = 0.602^2 + 0.125^2 + 0.602^2 + 0.301^2 = 0.8312 = \|v\|^2$. cos $= 0.1062/0.8312 =$ **0.128**.

TF-IDF menurunkan kemiripan karena kata yang dibagi (free, now) adalah kata umum yang bobotnya kecil, sedangkan kata khas tiap dokumen (win, prize vs claim, gift) berbeda.

5. Mutual Information terhadap spam.

- "free": $A = 2$ (D_1, D_2), $B = 1$ (D_3), $C = 0$, $N = 4$. MI $= \log_2 \frac{2 \cdot 4}{3 \cdot 2} = \log_2 1.333 =$ **0.415**.
- "now": $A = 2$, $B = 0$, $C = 0$. MI $= \log_2 \frac{2 \cdot 4}{2 \cdot 2} = \log_2 2 =$ **1.0**.

"now" lebih baik: selalu muncul di spam dan tidak pernah di ham, sedangkan "free" juga muncul di ham.

6. Naive Bayes.
Prior spam $=$ ham $= 2/4 = 0.5$. Total kata: spam $= 4 + 4 = 8$, ham $= 3 + 3 = 6$. $|V| = 10$, jadi penyebut spam $= 18$, ham $= 16$.
"free prize now":

$$\text{spam} = 0.5 \cdot \tfrac{2+1}{18} \cdot \tfrac{1+1}{18} \cdot \tfrac{2+1}{18} = 0.5 \cdot \tfrac{3 \cdot 2 \cdot 3}{18^3} = 0.001543$$
$$\text{ham} = 0.5 \cdot \tfrac{1+1}{16} \cdot \tfrac{0+1}{16} \cdot \tfrac{0+1}{16} = 0.5 \cdot \tfrac{2}{16^3} = 0.000244$$

$P(\text{spam}) = 0.001543/(0.001543 + 0.000244) =$ **0.863**, jadi **spam**.
"free lunch today":

$$\text{spam} = 0.5 \cdot \tfrac{3}{18} \cdot \tfrac{1}{18} \cdot \tfrac{1}{18} = 0.000257$$
$$\text{ham} = 0.5 \cdot \tfrac{2}{16} \cdot \tfrac{2}{16} \cdot \tfrac{3}{16} = 0.001465$$

$P(\text{ham}) = 0.001465/(0.000257 + 0.001465) =$ **0.851**, jadi **ham**.

7. Evaluasi (kelas positif = spam).
Bandingkan per pasangan: (S,S) TP, (S,S) TP, (S,H) FN, (H,S) FP, (H,H) TN, (H,H) TN. Jadi $TP = 2$, $FN = 1$, $FP = 1$, $TN = 2$.

$$P = \tfrac{2}{3} = 0.667, \quad R = \tfrac{2}{3} = 0.667, \quad F_1 = 0.667, \quad \text{Acc} = \tfrac{4}{6} = 0.667.$$

2 Word Representation

Materi: dari bag of words ke vektor kata

Pada bag of words tiap kata hanya menjadi *satu skor* di satu kolom. Akibatnya "laptop" dan "notebook" dianggap tidak berhubungan sama sekali. **Word representation** mengganti satu skor itu dengan *satu vektor angka per kata*, sehingga kata bermakna mirip punya vektor yang berdekatan.

Taksonomi.

- **One-hot**: vektor sepanjang vocabulary, hanya satu elemen bernilai 1. Tidak menangkap kemiripan (semua kata saling tegak lurus).
- **Distributed representation** (vektor padat), dibuat dengan dua cara:
  - **Count-based**: hitung seberapa sering kata muncul bersama kata atau dokumen lain, lalu (bila perlu) kecilkan dimensinya. Contoh: LSI (SVD), HAL.
  - **Predict-based**: pelajari vektor dengan tugas menebak kata. Contoh: Word2Vec. GloVe berada di tengah (hybrid).

Dasar semua metode ini adalah **distributional hypothesis**: kata yang muncul di konteks mirip cenderung bermakna mirip.

Metode	Sumber statistik	Cara memperoleh vektor
LSI	kata–dokumen (global)	dibaca dari SVD (closed-form)
HAL	kata–kata dalam window (lokal)	dibaca dari matriks R dan L
LDA	kata–dokumen (global)	inferensi probabilistik (Gibbs sampling)
GloVe	kata–kata (global)	optimisasi gradient atas loss
Word2Vec	kata–kata dalam window (lokal)	neural network, murni prediktif

2.1 Cosine Similarity

Materi

Setelah kata atau dokumen menjadi vektor, kemiripannya diukur dari *sudut* antar vektor, bukan panjangnya. Dua vektor searah berarti mirip, walaupun panjangnya beda (misalnya dokumen panjang vs pendek dengan proporsi kata sama).

Rumus

$$\cos(u, v) = \frac{u \cdot v}{\|u\| \, \|v\|}, \qquad u \cdot v = \sum_i u_i v_i, \qquad \|u\| = \sqrt{\textstyle\sum_i u_i^2}$$

Arti simbol

Simbol	Arti	Contoh / ukuran
u, v	dua vektor yang dibandingkan	$(0.6, 0.5, 0.4)$
$u \cdot v$	dot product: kalikan elemen sepasang-sepasang, jumlahkan	0.545
$\|u\|$	panjang (norma) vektor	$\sqrt{0.77}$
$\cos(u, v)$	1: searah; 0: tegak lurus; -1: berlawanan	0.769

Soal

$u = (0.6,\ 0.5,\ 0.4)$ dan $v = (0.8,\ 0.05,\ 0.1)$. Hitung cosine.

Pembahasan

Dot product: $0.6(0.8) + 0.5(0.05) + 0.4(0.1) = 0.48 + 0.025 + 0.04 = 0.545$.
Norma: $\|u\| = \sqrt{0.36 + 0.25 + 0.16} = \sqrt{0.77} = 0.8775$; $\|v\| = \sqrt{0.64 + 0.0025 + 0.01} = \sqrt{0.6525} = 0.8078$.

$$\cos = \frac{0.545}{0.8775 \times 0.8078} = \mathbf{0.769}$$

Cek dan jebakan

Hasil tidak boleh lebih dari 1. Vektor yang semua elemennya positif tidak bisa menghasilkan cosine negatif.

2.2 SVD dan Latent Semantic Indexing (LSI)

Materi

LSI (disebut juga LSA) memetakan kata dan dokumen ke ruang "makna tersembunyi" (latent) berdimensi kecil. Mulai dari matriks term × document A. **SVD** memecah A menjadi tiga matriks:

- U: tiap baris adalah koordinat sebuah **kata** di ruang latent (kolomnya eigenvector $AA^\top$).
- S: matriks diagonal berisi **singular value**, yaitu "seberapa penting" tiap dimensi latent, terurut dari terbesar.
- V: tiap baris adalah koordinat sebuah **dokumen** (kolomnya eigenvector $A^\top A$).

Dengan hanya menyimpan k singular value terbesar (*truncation*), noise dibuang dan kata yang sering muncul di dokumen yang sama menjadi berdekatan, walaupun tidak pernah muncul bersama. Dokumen atau query baru dipetakan ke ruang latent dengan *fold-in*.

Rumus

$$A = USV^\top, \qquad \sigma_i = \sqrt{\lambda_i}, \qquad u_i = \frac{Av_i}{\sigma_i}, \qquad \lambda^2 - (\mathrm{tr})\lambda + \det = 0 \ \ \text{(matriks } 2 \times 2\text{)}$$

$$\text{fold-in: } q = Q^\top US^{-1}, \qquad d = D^\top US^{-1}$$

Arti simbol

Simbol	Arti	Contoh / ukuran
A	matriks term × document (n kata, p dokumen)	3×2
λ_i	eigenvalue dari $A^\top A$, diurutkan menurun	3, 1
σ_i	singular value (akar eigenvalue)	$\sqrt{3}$, 1
v_i	eigenvector $A^\top A$ dengan panjang 1 (kolom ke-i dari V)	$(0.707, 0.707)$
u_i	kolom ke-i dari U	
tr, det	trace (jumlah diagonal) dan determinan matriks 2×2	4, 3
Q, D	vektor kolom query atau dokumen baru dalam ruang kata	$(1, 1, 1)$
k	jumlah dimensi latent yang disimpan	1 atau 2

2.3 Soal Penuh: SVD dan LSI pada Matriks Kecil

Soal penuh

$A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{pmatrix}$ (3 kata $\times$ 2 dokumen).

1. Hitung $A^\top A$, eigenvalue, dan singular value.
2. Cari V dan U.
3. Cek bahwa $USV^\top$ kembali menjadi A (cukup satu elemen).
4. Hitung aproksimasi rank-1 $A_1 = \sigma_1 u_1 v_1^\top$ dan persentase "energi" yang disimpan.
5. Petakan dokumen baru $D = (1, 1, 1)$ (berisi ketiga kata) ke ruang latent, lalu hitung kemiripannya dengan dokumen 1.

Pembahasan soal penuh

1. $A^\top A$ dan eigenvalue.
Kolom A adalah $a_1 = (1, 0, 1)$ dan $a_2 = (1, 1, 0)$. $A^\top A$ berisi dot product antar kolom:

$$A^\top A = \begin{pmatrix} a_1 \cdot a_1 & a_1 \cdot a_2 \\ a_2 \cdot a_1 & a_2 \cdot a_2 \end{pmatrix} = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$$

$\text{tr} = 4$, $\det = 2 \cdot 2 - 1 \cdot 1 = 3$, jadi $\lambda^2 - 4\lambda + 3 = 0 \Rightarrow (\lambda - 3)(\lambda - 1) = 0$, sehingga $\lambda_1 = 3$ dan $\lambda_2 = 1$. Singular value: $\sigma_1 = \sqrt{3} = 1.732$ dan $\sigma_2 = 1$.

2. V dan U.
Untuk $\lambda = 3$: $(A^\top A - 3I)v = 0$ memberi $-x + y = 0$, jadi $v_1 = (1, 1)/\sqrt{2} = (0.707,\ 0.707)$.
Untuk $\lambda = 1$: $x + y = 0$, jadi $v_2 = (1, -1)/\sqrt{2} = (0.707,\ -0.707)$.
$u_1 = Av_1/\sigma_1$: $Av_1 = (2, 1, 1)/\sqrt{2}$, dibagi $\sqrt{3}$ menjadi $(2, 1, 1)/\sqrt{6} = (0.816,\ 0.408,\ 0.408)$.
$u_2 = Av_2/\sigma_2 = (0, -1, 1)/\sqrt{2} = (0,\ -0.707,\ 0.707)$.

$$U = \begin{pmatrix} 0.816 & 0 \\ 0.408 & -0.707 \\ 0.408 & 0.707 \end{pmatrix}, \quad S = \begin{pmatrix} 1.732 & 0 \\ 0 & 1 \end{pmatrix}, \quad V = \begin{pmatrix} 0.707 & 0.707 \\ 0.707 & -0.707 \end{pmatrix}$$

3. Cek rekonstruksi.
Elemen A_{21} (kata 2, dokumen 1): $\sigma_1 u_{21} v_{11} + \sigma_2 u_{22} v_{12} = 1.732(0.408)(0.707) + 1(-0.707)(0.707) = 0.5 - 0.5 = 0$. Cocok dengan $A_{21} = 0$.

4. Rank-1.

$$A_1 = \sigma_1 u_1 v_1^\top = \sqrt{3} \cdot \frac{(2, 1, 1)^\top}{\sqrt{6}} \cdot \frac{(1, 1)}{\sqrt{2}} = \frac{\sqrt{3}}{\sqrt{12}} \begin{pmatrix} 2 & 2 \\ 1 & 1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 0.5 & 0.5 \\ 0.5 & 0.5 \end{pmatrix}$$

Energi yang disimpan: $\sigma_1^2/(\sigma_1^2 + \sigma_2^2) = 3/4 = \mathbf{75}\%$. Dalam rank-1, kata 2 dan kata 3 menjadi identik, karena perbedaannya hanya ada di dimensi kedua yang dibuang.

5. Fold-in dokumen baru.
$D^\top U = (0.816{+}0.408{+}0.408,\ 0{-}0.707{+}0.707) = (1.633,\ 0)$. Kali S^{-1}: $d = (1.633/1.732,\ 0/1) = (\mathbf{0.943},\ \mathbf{0})$.

Koordinat dokumen lama adalah baris V: dokumen 1 $= (0.707, 0.707)$, dokumen 2 $= (0.707, -0.707)$.

$$\cos(d, \text{dok}_1) = \frac{0.943(0.707) + 0}{0.943 \times 1} = \mathbf{0.707}$$

Hasil yang sama untuk dokumen 2: dokumen baru yang berisi semua kata sama miripnya dengan kedua dokumen.

Cek dan jebakan

$\sigma_1 \geq \sigma_2$. $v_1 \perp v_2$ (dot $= 0$). Tanda eigenvector boleh terbalik ($-v$ juga benar). $AA^\top$ (3×3) punya eigenvalue 3, 1, 0: eigenvalue taknol $AA^\top$ dan $A^\top A$ selalu sama. Contoh slide ($A = \begin{pmatrix} 1 & 2 \\ 0 & 1 \\ 1 & 1 \end{pmatrix}$, $\lambda = 4 \pm \sqrt{13}$, $\sigma = 2.7578$ dan 0.6281) memakai langkah yang sama.

2.4 HAL (Hyperspace Analogue to Language)

Materi

HAL membuat vektor kata dari *co-occurrence* lokal. Sebuah jendela (window) selebar W digeser sepanjang teks. Tiga sifatnya:

1. **Local context**: hanya kata dalam window yang dihitung.
2. **Distance-weighted**: kata yang lebih dekat diberi bobot lebih besar.
3. **Direction-sensitive**: kata yang muncul *sesudah* dan *sebelum* dicatat terpisah, di matriks R (right) dan L (left).

Vektor kata adalah gabungan barisnya di R dan L. Vektor dokumen adalah rata-rata vektor kata di dalamnya. Karena panjang vektor $2|V|$ bisa besar, dimensinya dikurangi dengan memilih kolom bervarians terbesar (*variance-based selection*) atau SVD.

Rumus

$$w(d) = W - d + 1, \qquad L = R^\top, \qquad v(w) = [\, R[w,:] \parallel L[w,:] \,]$$

$$v(\text{dok}) = \frac{1}{|\text{dok}|} \sum_w \text{count}(w, \text{dok})\, v(w)$$

Arti simbol

Simbol	Arti	Contoh / ukuran
W	lebar window	2
d	jarak dua kata (1 = bersebelahan)	1 atau 2
$w(d)$	bobot pasangan pada jarak d	$w(1) = 2$, $w(2) = 1$
$R[a][b]$	jumlah bobot saat b muncul *sesudah* a dalam window	
$L[a][b]$	jumlah bobot saat b muncul *sebelum* a; selalu $L = R^\top$	
$\parallel$	konkatenasi (sambung dua vektor)	panjang $2\|V\|$
$\|\text{dok}\|$	jumlah token di dokumen	3
$\text{count}(w, \text{dok})$	berapa kali kata w ada di dokumen	2

2.5 Soal Penuh: HAL dari Teks sampai Kemiripan Dokumen

Soal penuh

D_1 = "cat dog cat", D_2 = "bird dog". Stream gabungan: `cat dog cat bird dog`. V = {cat, dog, bird}, $W = 2$.

1. Tentukan bobot jarak dan semua pasangan berurutan dalam window.
2. Bangun matriks R dan L.
3. Tuliskan vektor kata ketiga kata.
4. Hitung $\cos(\text{cat}, \text{dog})$ dan $\cos(\text{cat}, \text{bird})$. Mana yang lebih mirip, dan kenapa?
5. Hitung vektor dokumen D_1, D_2 dan cosine keduanya.
6. Kurangi dimensi menjadi 2 dengan variance-based selection (varians populasi, bagi 3).

Pembahasan soal penuh

1. Bobot dan pasangan.
$w(1) = 2 - 1 + 1 = 2$, $w(2) = 1$. Beri nomor posisi: cat_0 dog_1 cat_2 bird_3 dog_4.

- Jarak 1 (bobot 2): (cat,dog), (dog,cat), (cat,bird), (bird,dog).
- Jarak 2 (bobot 1): (cat,cat), (dog,bird), (cat,dog).

Pasangan ditulis (kata kiri, kata kanan): kata kanan muncul *sesudah* kata kiri.

2. Matriks R dan L.
Jumlahkan bobot per pasangan. Contoh $R[\text{cat}][\text{dog}] = 2$ (posisi 0–1) $+1$ (posisi 2–4) $= 3$.

$$R = \begin{array}{c|ccc} & \text{cat} & \text{dog} & \text{bird} \\ \hline \text{cat} & 1 & 3 & 2 \\ \text{dog} & 2 & 0 & 1 \\ \text{bird} & 0 & 2 & 0 \end{array} \qquad L = R^\top = \begin{array}{c|ccc} & \text{cat} & \text{dog} & \text{bird} \\ \hline \text{cat} & 1 & 2 & 0 \\ \text{dog} & 3 & 0 & 2 \\ \text{bird} & 2 & 1 & 0 \end{array}$$

3. Vektor kata
$v(w) = [R \text{ baris } w \parallel L \text{ baris } w]$:

$$v(\text{cat}) = (1, 3, 2 \parallel 1, 2, 0), \quad v(\text{dog}) = (2, 0, 1 \parallel 3, 0, 2), \quad v(\text{bird}) = (0, 2, 0 \parallel 2, 1, 0)$$

4. Kemiripan kata.
$\|\text{cat}\| = \sqrt{1+9+4+1+4} = \sqrt{19} = 4.359$, $\|\text{dog}\| = \sqrt{18} = 4.243$, $\|\text{bird}\| = \sqrt{9} = 3$.

- cat·dog $= 2 + 0 + 2 + 3 + 0 + 0 = 7$, jadi cos $= 7/(4.359 \times 4.243) =$ **0.379**.
- cat·bird $= 0 + 6 + 0 + 2 + 2 + 0 = 10$, jadi cos $= 10/(4.359 \times 3) =$ **0.765**.

cat lebih mirip *bird*, karena keduanya sama-sama sering diikuti *dog* dan didahului *cat*/*dog*: konteksnya mirip. *cat* dan *dog* sering bersebelahan, tetapi konteks mereka (siapa di kiri dan kanannya) berbeda. HAL mengukur kemiripan *konteks*, bukan seberapa sering dua kata muncul berdekatan.

5. Vektor dokumen.

$$v(D_1) = \frac{2v(\text{cat}) + v(\text{dog})}{3} = \frac{(4, 6, 5 \parallel 5, 4, 2)}{3} = (1.333,\ 2,\ 1.667,\ 1.667,\ 1.333,\ 0.667)$$

$$v(D_2) = \frac{v(\text{bird}) + v(\text{dog})}{2} = \frac{(2, 2, 1 \parallel 5, 1, 2)}{2} = (1,\ 1,\ 0.5,\ 2.5,\ 0.5,\ 1)$$

Dot product: $1.333(1) + 2(1) + 1.667(0.5) + 1.667(2.5) + 1.333(0.5) + 0.667(1) = 9.667$. Norma: $\|v(D_1)\| = \sqrt{13.556} = 3.682$, $\|v(D_2)\| = \sqrt{9.75} = 3.122$.

$$\cos(D_1, D_2) = \frac{9.667}{3.682 \times 3.122} = \mathbf{0.841}$$

6. Variance-based selection.
Enam kolom (nilai cat, dog, bird) dan variansnya:

Kolom	R·cat	R·dog	R·bird	L·cat	L·dog	L·bird
Nilai	(1,2,0)	(3,0,2)	(2,1,0)	(1,3,2)	(2,0,1)	(0,2,0)
Varians	0.667	**1.556**	0.667	0.667	0.667	**0.889**

Contoh kolom R·dog: rata-rata $5/3 = 1.667$; varians $= [(3 - 1.667)^2 + (0 - 1.667)^2 + (2 - 1.667)^2]/3 = 1.556$.
Ambil dua kolom tertinggi: R·dog dan L·bird. Vektor baru: cat $= (3, 0)$, dog $= (0, 2)$, bird $= (2, 0)$.

Cek dan jebakan

Jumlah semua isi R = jumlah semua isi L (di sini 11). Diagonal R boleh tidak nol ("cat …cat" dalam window). Hati-hati arah: baris R = kata kiri, kolom = kata yang muncul sesudahnya.

2.6 LDA (Latent Dirichlet Allocation)

Materi

LDA adalah model *generatif probabilistik* yang menganggap tiap dokumen ditulis dengan cara berikut:

1. Pilih campuran topik dokumen θ_d dari distribusi Dirichlet, misalnya 80% olahraga dan 20% politik.
2. Untuk tiap kata: pilih satu topik z menurut θ_d, lalu pilih kata w menurut distribusi kata topik itu, φ_z.

Dua asumsi: **dokumen adalah campuran topik**, dan **topik adalah distribusi atas kata**. Saat training kita hanya melihat kata, lalu menebak topik tiap kata berulang kali (*Gibbs sampling*) sampai stabil. Hasilnya: φ (matriks topik × kata) dan θ (matriks dokumen × topik).
Vektor yang dihasilkan: *kolom* φ adalah vektor **kata** (berdimensi jumlah topik K), dan *baris* θ adalah vektor **dokumen**. Keduanya bisa dibandingkan dengan cosine.
Peran α: α kecil (mis. 0.3) membuat dokumen condong ke satu topik (sparse, dipakai LDA); $\alpha = 1$ netral; α besar (mis. 8) membuat campuran topik merata.

Rumus

$$P(w \mid d) = \sum_{k=1}^{K} \theta_{dk}\, \varphi_{kw}, \qquad P(z = k \mid w, d) = \frac{\theta_{dk}\varphi_{kw}}{P(w \mid d)}$$

Gibbs (bentuk slide): $P(z_{d,n} = k \mid \dots) \propto (n_{dk} + \alpha)(n_{kw} + \beta)$, lalu dinormalkan

Arti simbol

Simbol	Arti	Contoh / ukuran
K	jumlah topik	2
θ_{dk}	proporsi topik k di dokumen d (satu baris berjumlah 1)	0.8
φ_{kw}	peluang kata w di topik k (satu baris berjumlah 1)	0.5
$z_{d,n}$	topik yang ditugaskan ke kata ke-n di dokumen d	
n_{dk}	jumlah kata lain di dokumen d yang saat ini bertopik k	3
n_{kw}	jumlah kemunculan kata w di seluruh korpus yang saat ini bertopik k	4
α, β	parameter Dirichlet (prior) untuk θ dan φ	0.1, 0.01

2.7 Soal Penuh: LDA: Peluang Kata, Vektor, dan Satu Langkah Gibbs

Soal penuh

$K = 2$ topik (Sport, Politics), 4 kata (ball, goal, vote, law).

$\varphi =$

	ball	goal	vote	law
Sport	0.50	0.40	0.05	0.05
Politics	0.05	0.05	0.50	0.40

$\theta =$

	Sport	Politics
D_1	0.8	0.2
D_2	0.1	0.9
D_3	0.5	0.5

1. Hitung $P(w \mid D_1)$ untuk keempat kata dan cek jumlahnya.
2. Berapa peluang D_1 menghasilkan dua kata "ball goal"?
3. Kata "ball" muncul di D_1. Berapa peluang kata itu berasal dari topik Sport?
4. Hitung $\cos(\text{ball}, \text{goal})$ dan $\cos(\text{ball}, \text{vote})$ dari vektor kata LDA.
5. Hitung $\cos(D_1, D_2)$ dan $\cos(D_1, D_3)$.
6. Satu langkah Gibbs: untuk sebuah kata "goal" di D_3, $n_{dk} = (3, 2)$ dan $n_{kw} = (4, 1)$ untuk (Sport, Politics), $\alpha = 0.1$, $\beta = 0.01$. Hitung peluang topiknya.

Pembahasan soal penuh

1. Peluang kata di D_1.
$P(w \mid D_1) = 0.8\,\varphi_{\text{Sport},w} + 0.2\,\varphi_{\text{Pol},w}$:

- ball: $0.8(0.50) + 0.2(0.05) = 0.40 + 0.01 = \mathbf{0.41}$
- goal: $0.32 + 0.01 = \mathbf{0.33}$
- vote: $0.04 + 0.10 = \mathbf{0.14}$
- law: $0.04 + 0.08 = \mathbf{0.12}$

Jumlah $= 1.00$ (cek lolos).

2. "ball goal".
Kata dianggap independen (bag of words): $0.41 \times 0.33 = \mathbf{0.1353}$.

3. Topik untuk "ball" di D_1.
$P(\text{Sport} \mid \text{ball}, D_1) = \dfrac{0.8 \times 0.50}{0.41} = \dfrac{0.40}{0.41} = \mathbf{0.976}$.

4. Kemiripan kata

(kolom φ): ball $= (0.50, 0.05)$, goal $= (0.40, 0.05)$, vote $= (0.05, 0.50)$.

$$\cos(\text{ball}, \text{goal}) = \frac{0.20 + 0.0025}{\sqrt{0.2525}\sqrt{0.1625}} = \frac{0.2025}{0.5025 \times 0.4031} = \mathbf{0.9997}$$

$$\cos(\text{ball}, \text{vote}) = \frac{0.025 + 0.025}{0.5025 \times 0.5025} = \frac{0.05}{0.2525} = \mathbf{0.198}$$

5. Kemiripan dokumen
(baris θ):

$$\cos(D_1, D_2) = \frac{0.08 + 0.18}{\sqrt{0.68}\sqrt{0.82}} = \frac{0.26}{0.8246 \times 0.9055} = \mathbf{0.348}$$

$$\cos(D_1, D_3) = \frac{0.4 + 0.1}{\sqrt{0.68}\sqrt{0.5}} = \frac{0.5}{0.8246 \times 0.7071} = \mathbf{0.857}$$

6. Gibbs.
Sport: $(3+0.1)(4+0.01) = 3.1 \times 4.01 = 12.431$. Politics: $(2+0.1)(1+0.01) = 2.1 \times 1.01 = 2.121$. Jumlah 14.552.

$$P(\text{Sport}) = 12.431/14.552 = \mathbf{0.854}, \qquad P(\text{Politics}) = \mathbf{0.146}$$

Topik baru untuk kata itu kemudian *diundi* dengan peluang ini (bukan langsung diambil yang terbesar), lalu hitungan n diperbarui dan diulang untuk kata lain.

Cek dan jebakan

Rumus Gibbs di slide tidak membagi dengan $n_k + V\beta$; ikuti bentuk slide kecuali soal memberi bentuk lengkap. Jangan tertukar: kata = *kolom* φ, dokumen = *baris* θ.

2.8 GloVe (Global Vectors)

Materi

GloVe disebut *hybrid* karena menggabungkan dua ide. Pertama, seperti HAL dan LSI, ia memakai *statistik global*: matriks co-occurrence X dari seluruh korpus. Kedua, seperti Word2Vec, vektornya *dipelajari* dengan gradient, bukan dibaca langsung.
Langkahnya:

1. Bangun matriks co-occurrence simetris X ($X_{ij} = X_{ji}$), misalnya $X = R + L$ dari HAL.
2. Tetapkan target: hasil kali vektor kata dan vektor konteks (plus bias) harus mendekati $\log X_{ij}$. Alasannya, rasio co-occurrence membawa makna ("ice" vs "steam": kata "solid" rasionya besar, "gas" kecil).
3. Ukur kesalahan dengan loss J. Pasangan yang terlalu sering muncul (seperti "the") diberi bobot f yang dibatasi agar tidak mendominasi.
4. Kecilkan J dengan SGD atau AdaGrad. Vektor dimulai acak. Satu *epoch* adalah satu putaran atas semua pasangan dengan $X_{ij} > 0$.
5. Setelah selesai, vektor akhir kata $= w + c$.

Rumus

$$\text{pred}_{ij} = w_i \cdot c_j + b_i + \tilde{b}_j, \qquad \text{err}_{ij} = \text{pred}_{ij} - \ln X_{ij}, \qquad J = \sum_{i,j} f(X_{ij})\, \text{err}_{ij}^2$$

$$f(x) = \begin{cases} (x/x_{\max})^{\alpha} & x < x_{\max} \\ 1 & x \geq x_{\max} \end{cases} \qquad x_{\max} = 100,\ \alpha = 0.75$$

Update (bentuk slide): $g = f \cdot \text{err}, \quad w_i \leftarrow w_i - \eta\, g\, c_j, \quad c_j \leftarrow c_j - \eta\, g\, w_i, \quad b_i, \tilde{b}_j \leftarrow b - \eta\, g$

Arti simbol

Simbol	Arti	Contoh / ukuran
X_{ij}	berapa kali kata j muncul dekat kata i	4
w_i	vektor kata i sebagai *target*	$(0.2, -0.1)$
c_j	vektor kata j sebagai *konteks*	$(0.3, -0.1)$
b_i, $\tilde{b}_j$	bias target dan bias konteks	0.1
$f(x)$	bobot pasangan; dibatasi maksimal 1	$f(4) = 0.0894$
err_{ij}	selisih prediksi dengan target $\ln X_{ij}$	-1.116
J	loss total (makin kecil makin baik)	
η	learning rate	0.1
g	"sinyal kesalahan" $f \cdot \text{err}$	-0.0998

Turunan sebenarnya punya faktor 2 ($2f \cdot \text{err}$); slide membuangnya (dianggap masuk ke η), jadi **ikuti soal**.

2.9 Soal Penuh: GloVe: Loss dan Satu Update

Soal penuh

Tiga kata a, b, c dengan $X_{ab} = 4$, $X_{ac} = 1$, $X_{bc} = 2$. Parameter awal (dimensi 2):

Kata	w (target)	c (konteks)	b / $\tilde{b}$
a	(0.2, −0.1)	(0.1, 0.2)	0.1 / 0
b	(0.1, 0.3)	(0.3, −0.1)	0 / 0.1
c	(−0.2, 0.2)	(0, 0.1)	−0.1 / 0.05

1. Hitung $f(X)$ untuk ketiga pasangan (a, b), (a, c), (b, c).
2. Hitung prediksi, error, dan loss tiap pasangan, lalu J total untuk ketiga pasangan itu (satu arah saja).
3. Untuk pasangan (a, b) dengan $\eta = 0.1$, hitung gradient lalu perbarui w_a, c_b, b_a, $\tilde{b}_b$.
4. Hitung ulang loss pasangan (a, b) dengan parameter baru. Apakah turun?
5. Berapa vektor akhir kata a jika training berhenti sebelum update?

Pembahasan soal penuh

1. Bobot f.
Semua $X < 100$: $f(4) = 0.04^{0.75} = e^{0.75 \ln 0.04} = e^{-2.414} = \mathbf{0.0894}$; $f(1) = 0.01^{0.75} = \mathbf{0.0316}$; $f(2) = 0.02^{0.75} = \mathbf{0.0532}$.

2. Prediksi, error, loss.
Pasangan (i, j) memakai w_i, c_j, b_i, $\tilde{b}_j$.

Pasangan	$w_i \cdot c_j$	pred	$\ln X$	err	f	$f \cdot \text{err}^2$
(a, b)	$0.06 + 0.01 = 0.07$	$0.07 + 0.1 + 0.1 = 0.27$	1.3863	-1.1163	0.0894	0.1115
(a, c)	$0 - 0.01 = -0.01$	$-0.01 + 0.1 + 0.05 = 0.14$	0	0.14	0.0316	0.0006
(b, c)	$0 + 0.03 = 0.03$	$0.03 + 0 + 0.05 = 0.08$	0.6931	-0.6131	0.0532	0.0200

$J = 0.1115 + 0.0006 + 0.0200 = \mathbf{0.1321}$.

3. Update pasangan (a, b).
$g = f \cdot \text{err} = 0.0894 \times (-1.1163) = -0.0998$.

- $g_{w_a} = g\, c_b = -0.0998\,(0.3, -0.1) = (-0.0300, 0.0100) \Rightarrow w_a = (0.2, -0.1) - 0.1(-0.0300, 0.0100) = (\mathbf{0.2030}, \mathbf{-0.1010})$
- $g_{c_b} = g\, w_a = -0.0998\,(0.2, -0.1) = (-0.0200, 0.0100) \Rightarrow c_b = (0.3, -0.1) - 0.1(-0.0200, 0.0100) = (\mathbf{0.3020}, \mathbf{-0.1010})$
- $b_a = 0.1 - 0.1(-0.0998) = \mathbf{0.1100}$; $\tilde{b}_b = 0.1 + 0.00998 = \mathbf{0.1100}$

4. Loss baru.
pred $= 0.2030(0.3020) + (-0.1010)(-0.1010) + 0.1100 + 0.1100 = 0.2915$; err $= 0.2915 - 1.3863 = -1.0948$; loss $= 0.0894 \times 1.0948^2 = \mathbf{0.1072}$. Turun dari 0.1115, jadi update berjalan ke arah yang benar.

5. Vektor akhir.
$v(a) = w_a + c_a = (0.2 + 0.1,\ -0.1 + 0.2) = (\mathbf{0.3}, \mathbf{0.1})$.

Cek dan jebakan

Error negatif (prediksi di bawah target) membuat w bergerak searah c, sehingga dot product naik. $\ln 1 = 0$, jadi pasangan dengan $X = 1$ menargetkan prediksi 0. $f(x) = 1$ untuk $x \geq 100$. Karena X simetris, J penuh menjumlahkan *kedua* arah (i, j) dan (j, i); arah sebaliknya memakai w_j dan c_i sehingga nilainya berbeda. Ikuti pasangan yang diminta soal.

2.10 Word2Vec (CBOW dan Skip-gram)

Materi

Word2Vec (Mikolov dkk., 2013) belajar vektor kata dengan tugas *menebak kata di sekitar*. Jaringannya sangat sederhana:

1. Masukan berupa one-hot kata (panjang $|V|$).
2. Dikalikan matriks W ($|V| \times N$). Karena masukannya one-hot, hasilnya hanya mengambil *satu baris* W, yaitu vektor kata itu (h).
3. h dikalikan matriks W' ($N \times |V|$), menghasilkan skor untuk tiap kata di vocabulary.
4. Softmax mengubah skor menjadi peluang; loss $= -\ln$ peluang kata yang benar.

Dua varian: **Skip-gram** menebak kata konteks dari kata tengah; **CBOW** menebak kata tengah dari rata-rata vektor kata konteks. Setelah training, W menjadi *lookup table* vektor kata, dan relasi aljabar muncul: $V_{\text{king}} - V_{\text{man}} + V_{\text{woman}} \approx V_{\text{queen}}$.

Rumus

$$h = W[\text{kata tengah}]\ \text{(skip-gram)} \quad \text{atau} \quad h = \tfrac{1}{C} \sum_c W[\text{konteks}_c]\ \text{(CBOW)}$$

$$u = hW', \qquad p = \text{softmax}(u), \qquad \text{loss} = -\ln p_{\text{kata benar}}, \qquad \#\text{parameter} = 2|V|N$$

Arti simbol

Simbol	Arti	Contoh / ukuran
$\|V\|$, N	ukuran vocabulary dan dimensi vektor (hidden)	5 dan 2
W	matriks input→hidden ($\|V\| \times N$); baris = vektor kata	
W'	matriks hidden→output ($N \times \|V\|$)	
h	vektor hidden	$(0.2, 0.4)$
u	skor (logit) untuk tiap kata vocabulary	
C	jumlah kata konteks	2
window	jarak maksimum kata konteks dari kata tengah	2

2.11 Soal Penuh: Word2Vec: Pasangan Latih, Forward Pass, Loss, dan Analogi

Soal penuh

Kalimat: "I love NLP very much". Vocabulary urut (I, love, NLP, very, much), $N = 2$.

$$W = \begin{array}{l|cc} \text{I} & 0.1 & 0.2 \\ \text{love} & 0.3 & -0.1 \\ \text{NLP} & 0.2 & 0.4 \\ \text{very} & -0.2 & 0.1 \\ \text{much} & 0 & 0.3 \end{array} \qquad W' = \begin{pmatrix} 0.1 & 0.2 & -0.1 & 0.3 & 0 \\ 0.2 & -0.1 & 0.4 & 0 & 0.1 \end{pmatrix}$$

1. Dengan window 2, berapa pasangan (tengah, konteks) skip-gram yang terbentuk?
2. Skip-gram dengan kata tengah "NLP": hitung h, skor u, dan softmax.
3. Hitung loss untuk konteks "love" dan loss total untuk keempat konteks NLP.
4. CBOW (window 1): tebak kata tengah dari konteks "love" dan "very". Hitung h, peluang, dan loss untuk target "NLP".
5. Berapa jumlah parameter model ini? Berapa jika $|V| = 10{,}000$, $N = 300$?
6. Analogi: king $= (0.9, 0.8)$, man $= (0.5, 0.1)$, woman $= (0.4, 0.6)$, queen $= (0.8, 1.3)$, prince $= (0.9, 0.3)$, apple $= (-0.2, 0.5)$. Cari hasil king $-$ man $+$ woman.

Pembahasan soal penuh

1. Pasangan latih.
Tiap kata dipasangkan dengan kata dalam jarak ≤ 2: I: (love, NLP) $\to$ 2; love: (I, NLP, very) $\to$ 3; NLP: (I, love, very, much) $\to$ 4; very: (love, NLP, much) $\to$ 3; much: (NLP, very) $\to$ 2. Total $=$ **14** pasangan.

2. Forward skip-gram.
$h = W[\text{NLP}] = (0.2,\ 0.4)$.

$$u = hW' = (0.2(0.1) + 0.4(0.2),\ 0.2(0.2) + 0.4(-0.1),\ \ldots) = (0.10,\ 0.00,\ 0.14,\ 0.06,\ 0.04)$$

$e^u = (1.1052,\ 1,\ 1.1503,\ 1.0618,\ 1.0408)$, jumlah 5.3581. Softmax:

$$p = (0.2063,\ 0.1866,\ 0.2147,\ 0.1982,\ 0.1943)$$

3. Loss.

Konteks “love”: $-\ln 0.1866 = \mathbf{1.6786}$. Keempat konteks (I, love, very, much): $1.5786 + 1.6786 + 1.6186 + 1.6386 = \mathbf{6.5144}$. Training akan mengubah W dan W' agar peluang kata konteks yang benar naik dan loss turun.

4. CBOW.
$h = \frac{1}{2}\big(W[\text{love}] + W[\text{very}]\big) = \frac{1}{2}\big((0.3, -0.1) + (-0.2, 0.1)\big) = (0.05,\ 0)$.
$u = hW' = 0.05 \times (0.1,\ 0.2,\ -0.1,\ 0.3,\ 0) = (0.005,\ 0.010,\ -0.005,\ 0.015,\ 0)$.
Softmax $\approx (0.2000,\ 0.2010,\ 0.1980,\ 0.2020,\ 0.1990)$. Loss target NLP $= -\ln 0.1980 = \mathbf{1.6195}$. Model yang belum dilatih memberi peluang hampir rata (sekitar 1/5).

5. Parameter.
$2|V|N = 2 \times 5 \times 2 = \mathbf{20}$. Untuk $|V| = 10{,}000$, $N = 300$: $2 \times 10{,}000 \times 300 = \mathbf{6{,}000{,}000}$ (tanpa bias).

6. Analogi.
$t = (0.9 - 0.5 + 0.4,\ 0.8 - 0.1 + 0.6) = (0.8,\ 1.3)$. Cosine t terhadap kandidat: queen 1.000, woman 0.999, king 0.958, prince 0.767, man 0.681, apple 0.596. Kata masukan (king, man, woman) dikecualikan, jadi jawabannya **queen**.

Cek dan jebakan

Kata masukan dikecualikan karena pada embedding nyata hasil analogi sering paling dekat dengan salah satu kata masukan. Di skip-gram, satu kata tengah menghasilkan beberapa pasangan (satu per konteks); di CBOW, satu kata tengah hanya satu contoh latih.

3 Transformer

Materi: cara kerja Transformer

Masalah yang diselesaikan. RNN/LSTM membaca kata satu per satu, sehingga training tidak bisa diparalelkan dan hubungan antar kata yang berjauhan sulit ditangkap. Transformer membuang komputasi berurutan itu: semua kata diproses *sekaligus* memakai **attention**.
Dari teks ke matriks X. Teks dipecah tokenizer (WordPiece untuk BERT, BPE untuk GPT-2, SentencePiece untuk Flan-T5). Tiap token dipetakan ke *ID* lewat vocabulary; ID hanya nomor baris tanpa makna. ID dipakai mengambil baris *embedding matrix* (nilainya dipelajari saat training). Karena attention tidak tahu urutan, ditambahkan **positional encoding** (dijumlahkan, bukan disambung). Hasilnya matriks X: satu baris per token.
Self-attention. Tiap token membuat tiga vektor dari X lewat matriks bobot yang dipelajari:

- **Query** q: “informasi apa yang saya cari?”
- **Key** k: “informasi apa yang saya tawarkan?”
- **Value** v: “isi yang akan diambil dari saya”.

Skor relevansi token i terhadap token j adalah $q_i \cdot k_j$, dibagi $\sqrt{d_k}$ agar tidak terlalu besar, lalu di-softmax menjadi bobot. Keluaran token i adalah *rata-rata berbobot* semua value. Contoh: pada “The animal didn’t cross the street because it was too tired”, kata “it” memberi bobot besar ke “animal”.
Multi-head. Attention dihitung beberapa kali paralel (h head), tiap head dengan W^Q, W^K, W^V sendiri sehingga bisa fokus ke relasi berbeda. Hasil semua head dikonkatenasi lalu dikali W^O.
Blok encoder (ditumpuk N kali): Multi-Head Attention → Add & Norm → Feed-Forward → Add & Norm. *Add* adalah residual connection (masukan blok ditambahkan ke keluarannya agar informasi tidak hilang); *Norm* adalah LayerNorm. Feed-forward network (FFN) adalah dua layer linear dengan ReLU, diterapkan ke tiap token terpisah.
Blok decoder menulis keluaran token demi token (autoregressive):

- **Masked self-attention**: token hanya boleh melihat token sebelumnya; skor ke token di depan diganti $-\infty$.
- **Cross-attention**: Query dari decoder, Key dan Value dari keluaran encoder. Di sinilah decoder "membaca" kalimat sumber.
- Keluaran akhir: Linear (ke ukuran vocabulary, menghasilkan logits) $\rightarrow$ Softmax $\rightarrow$ argmax $\rightarrow$ token berikutnya. Encoder hanya dijalankan sekali.

3.1 Softmax dan Cross-Entropy

Materi

Model mengeluarkan *logits*: skor mentah yang bisa negatif dan tidak berjumlah 1. **Softmax** mengubahnya menjadi peluang. Saat training, kesalahan diukur dengan **cross-entropy** (prasyarat): $-\ln$ peluang kelas yang benar. Jika model yakin dan benar, loss kecil; jika yakin tetapi salah, loss besar.

Rumus

$$\text{softmax}(z)_i = \frac{e^{z_i}}{\sum_j e^{z_j}}, \qquad L = -\ln p_y$$

Arti simbol

Simbol	Arti	Contoh / ukuran
z_i	logit kelas ke-i	2.0
p_y	peluang yang diberikan ke kelas benar y	0.659
L	loss cross-entropy (pakai ln)	0.417

Soal

Logits (2.0, 1.0, 0.1). Hitung peluang dan loss jika kelas benar adalah kelas 1.

Pembahasan

$e^2 = 7.389$, $e^1 = 2.718$, $e^{0.1} = 1.105$; jumlah 11.212. Peluang: $7.389/11.212 =$ **0.659**, **0.242**, **0.099**. Loss $= -\ln 0.659 =$ **0.417**.

Cek dan jebakan

Jumlah peluang $= 1$. Menambah konstanta yang sama ke semua logit tidak mengubah softmax. Perbandingan: $-\ln 0.7 = 0.357$, sedangkan $-\ln 0.05 = 2.996$.

3.2 Positional Encoding

Materi

Karena semua token diproses sekaligus, Transformer tidak tahu urutan. Positional encoding memberi tiap posisi t sebuah vektor unik dari gelombang sin dan cos dengan frekuensi berbeda per dimensi, lalu vektor itu *dijumlahkan* ke embedding token. Dimensi awal berubah cepat (frekuensi tinggi), dimensi akhir berubah lambat.

Rumus

$$PE(t,\,2k) = \sin(\omega_k t), \qquad PE(t,\,2k+1) = \cos(\omega_k t), \qquad \omega_k = \frac{1}{10000^{2k/d}}$$

Arti simbol

Simbol	Arti	Contoh / ukuran
t	posisi token (mulai 0)	1
d	dimensi embedding	4
$2k$, $2k+1$	indeks dimensi genap (pakai sin) dan ganjil (pakai cos)	$k = 0, 1$
ω_k	frekuensi untuk pasangan dimensi ke-k	1; 0.01

Soal

$d = 4$. Hitung vektor posisi untuk $t = 1$ dan $t = 2$.

Pembahasan

Dimensi 0 dan 1 memakai $k = 0$: $\omega_0 = 1/10000^0 = 1$. Dimensi 2 dan 3 memakai $k = 1$: $\omega_1 = 1/10000^{2/4} = 1/100 = 0.01$.

$$PE(1) = (\sin 1,\ \cos 1,\ \sin 0.01,\ \cos 0.01) = (\mathbf{0.8415},\ \mathbf{0.5403},\ \mathbf{0.0100},\ \mathbf{1.0000})$$

$$PE(2) = (\sin 2,\ \cos 2,\ \sin 0.02,\ \cos 0.02) = (\mathbf{0.9093},\ \mathbf{-0.4161},\ \mathbf{0.0200},\ \mathbf{0.9998})$$

Cek dan jebakan

Kalkulator harus mode **radian** ($\sin 1$ dalam derajat $= 0.0175$, salah). $PE(0) = (0, 1, 0, 1)$. BERT tidak memakai rumus ini: position embedding BERT *dipelajari*.

3.3 Scaled Dot-Product Attention (satu head)

Rumus

$$Q = XW^Q, \quad K = XW^K, \quad V = XW^V, \qquad \text{Attention}(Q,K,V) = \text{softmax}\Big(\frac{QK^\top}{\sqrt{d_k}}\Big)V$$

Arti simbol

Simbol	Arti	Contoh / ukuran
X	matriks masukan, satu baris per token ($n \times d_{\text{model}}$)	4×4
W^Q, W^K, W^V	matriks bobot yang dipelajari ($d_{\text{model}} \times d_k$)	4×2
Q, K, V	query, key, value; satu baris per token ($n \times d_k$)	4×2
n	jumlah token	3 atau 4
d_k	dimensi query/key per head	2
$QK^\top$	skor relevansi; baris i kolom $j = q_i \cdot k_j$ ($n \times n$)	
softmax per baris	mengubah skor tiap query jadi bobot berjumlah 1	
keluaran	baris $i = \sum_j (\text{bobot}_{ij})\, v_j$	$n \times d_v$

Soal

Tiga token, $d_k = 2$, sudah diberikan $Q = \begin{pmatrix} 1 & 0 \\ 0 & 2 \\ 1 & 1 \end{pmatrix}$, $K = \begin{pmatrix} 1 & 1 \\ 0 & 1 \\ 2 & 0 \end{pmatrix}$, $V = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix}$. Hitung keluaran attention untuk token 1.

Pembahasan

Langkah 1, skor. Token 1 sebagai query: $q_1 = (1, 0)$. Dot dengan tiap key: $q_1 \cdot k_1 = 1$, $q_1 \cdot k_2 = 0$, $q_1 \cdot k_3 = 2$.
Langkah 2, skala. Bagi $\sqrt{2} = 1.414$: $(0.707,\ 0,\ 1.414)$.
Langkah 3, softmax. $e^{0.707} = 2.028$, $e^0 = 1$, $e^{1.414} = 4.113$; jumlah 7.141. Bobot (**0.284**, **0.140**, **0.576**): token 1 paling memperhatikan token 3.
Langkah 4, rata-rata berbobot value. $z_1 = 0.284(1, 0) + 0.140(0, 1) + 0.576(1, 1) =$ (**0.860**, **0.716**).
Matriks bobot lengkap (untuk cek): $\begin{pmatrix} 0.284 & 0.140 & 0.576 \\ 0.446 & 0.446 & 0.108 \\ 0.401 & 0.198 & 0.401 \end{pmatrix}$, keluaran token 2 = (0.554, 0.554), token 3 = (0.802, 0.599).

Cek dan jebakan

Tiap baris bobot berjumlah 1. Keluaran z adalah campuran baris V, jadi tiap elemennya selalu di antara minimum dan maksimum kolom V. Jangan lupa membagi $\sqrt{d_k}$ sebelum softmax.

3.4 Masked Self-Attention

Rumus

$$e_{ij} = \begin{cases} q_i \cdot k_j / \sqrt{d_k} & j \le i \\ -\infty & j > i \end{cases} \qquad e^{-\infty} = 0 \ \Rightarrow \ \text{bobot ke token depan } = 0$$

Soal

Pakai Q, K, V dari subbab sebelumnya. Hitung keluaran masked attention ketiga token.

Pembahasan

- Token 1 hanya melihat dirinya: bobot $(1, 0, 0)$, $z_1 = v_1 =$ (**1**, **0**).
- Token 2 melihat token 1–2: skor $(1.414,\ 1.414,\ -\infty)$, bobot $(0.5,\ 0.5,\ 0)$, $z_2 = 0.5(1, 0) + 0.5(0, 1) =$ (**0.5**, **0.5**).
- Token 3 melihat semua: sama seperti tanpa mask, $z_3 =$ (**0.802**, **0.599**).

Cek dan jebakan

Token terakhir tidak berubah karena tidak ada token di depannya. Matriks bobot berbentuk segitiga bawah. Masking tetap satu operasi matriks, jadi tetap paralel.

3.5 Multi-Head: Dimensi dan Jumlah Parameter

Rumus

$$d_k = d_v = \frac{d_{\text{model}}}{h}, \qquad \text{Concat}(Z_1, \ldots, Z_h) \in \mathbb{R}^{n \times d_{\text{model}}}, \qquad \#\text{param attention} = 4\, d_{\text{model}}^2$$

Arti simbol

Simbol	Arti	Contoh / ukuran
h	jumlah head	8
d_{model}	dimensi model (embedding)	512
W^O	proyeksi keluaran setelah konkatenasi ($d_{\text{model}} \times d_{\text{model}}$)	
$4d_{\text{model}}^2$	W^Q, W^K, W^V (gabungan semua head) dan W^O, masing-masing d_{model}^2, tanpa bias	
d_{ff}	dimensi tengah FFN (tambahan; Transformer asli $4d_{\text{model}}$)	2048

Soal

$d_{\text{model}} = 512$, $h = 8$, $d_{ff} = 2048$. Tentukan d_k, ukuran konkatenasi, parameter attention, dan parameter FFN.

Pembahasan

$d_k = d_v = 512/8 =$ **64**. Konkatenasi $8 \times 64 = 512$ kolom (kembali ke d_{model}, sehingga residual bisa dijumlahkan). Parameter attention: $4 \times 512^2 = 4 \times 262{,}144 =$ **1,048,576**. FFN (tambahan): $2 \times 512 \times 2048 =$ **2,097,152** (tanpa bias).

3.6 LayerNorm (tambahan)

Materi

LayerNorm menormalkan *satu vektor token* (satu baris) agar rata-ratanya 0 dan variansnya 1, lalu dikali γ dan ditambah β (keduanya dipelajari; di soal biasanya $\gamma = 1$, $\beta = 0$). Tujuannya agar nilai tidak membesar atau mengecil terus selama training. Rumusnya tidak ada di slide, tetapi blok "Add & Norm" memakainya, dan latihan dosen secara eksplisit menyebut versi *tanpa* normalisasi, sehingga versi *dengan* normalisasi mungkin muncul.

Rumus

$$\mu = \frac{1}{d}\sum_{i=1}^{d} x_i, \qquad \sigma^2 = \frac{1}{d}\sum_{i=1}^{d}(x_i - \mu)^2, \qquad \text{LayerNorm}(x)_i = \gamma\,\frac{x_i - \mu}{\sqrt{\sigma^2 + \varepsilon}} + \beta$$

Arti simbol

Simbol	Arti	Contoh / ukuran
x	satu baris (satu token), panjang d	$(2, 4, 6, 8)$
μ, σ^2	rata-rata dan varians *populasi* (bagi d, bukan $d-1$)	5; 5
ε	angka kecil agar tidak membagi nol	10^{-5}
γ, β	skala dan geser yang dipelajari	1; 0

Soal

Normalkan $x = (2, 4, 6, 8)$ dengan $\gamma = 1$, $\beta = 0$.

Pembahasan

$\mu = 20/4 = 5$. $\sigma^2 = (9+1+1+9)/4 = 5$, $\sqrt{5} = 2.236$. Hasil: $((2-5)/2.236, \ldots) = (\mathbf{-1.342}, \mathbf{-0.447}, \mathbf{0.447}, \mathbf{1.342})$.

Cek dan jebakan

Dengan $\gamma = 1$ dan $\beta = 0$, hasil LayerNorm selalu berjumlah 0 (dengan γ, β lain belum tentu). Normalisasi dilakukan per *baris* (per token), bukan per kolom.

3.7 Soal Penuh: Transformer Encoder-Only tanpa LayerNorm (Exercise)

Ini Exercise on Transformer Encoder dari dosen (assignment ditutup 2 Oktober 2026; matriks sudah dicocokkan dengan PDF aslinya).

Soal penuh

Klasifikasi biner spam vs not spam. Masukan `[CLS]`, give, discount, `[SEP]` di posisi 0–3. Target: "give discount" $\rightarrow$ spam. Pakai representasi akhir `[CLS]` sebagai masukan classification head.

1. $d_{\text{model}} = 4$, 2 head, $d_k = d_v$. Tentukan d_k dan d_v.
2. Token embedding: `[CLS]`$= (1,0,1,0)$, give $= (0,1,1,0)$, discount $= (1,1,0,0)$, `[SEP]`$= (0,0,1,1)$. Positional embedding: Pos-0 $= (0,0,0,0)$, Pos-1 $= (0,0,1,0)$, Pos-2 $= (0,1,0,0)$, Pos-3 $= (0,0,0,1)$. Tuliskan X.
3. Head 1: $W_Q^{(1)} = W_V^{(1)} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$, $W_K^{(1)} = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix}$. Head 2: $W_Q^{(2)} = W_K^{(2)} = W_V^{(2)} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{pmatrix}$. $W_O = I_4$. Hitung multi-head self-attention lengkap tanpa LayerNorm (hanya residual).
4. FFN dengan ReLU: $W_1 = I_4$, $b_1 = 0$, $W_2 = \begin{pmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \\ 0 & -1 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{pmatrix}$, $b_2 = 0$. Tunjukkan keluaran FFN dan residual.
5. Classification head $W_{cls} = \begin{pmatrix} 1 & -1 \\ 0 & 1 \\ 1 & 0 \\ -1 & 1 \end{pmatrix}$ (kolom 1 spam, kolom 2 not spam), $b_{cls} = 0$. Hitung logits dan softmax.
6. Nyatakan prediksi untuk "give discount".

Pembahasan soal penuh

1. Dimensi.
Tiap head mendapat bagian yang sama dari d_{model}: $d_k = d_v = 4/2 = \mathbf{2}$.

2. Matriks X.

Jumlahkan token embedding dan positional embedding baris demi baris:

$$X = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 2 & 0 \\ 1 & 2 & 0 & 0 \\ 0 & 0 & 1 & 2 \end{pmatrix}$$

3. Multi-head self-attention.
Tiap head menghitung $Q = XW_Q$, $K = XW_K$, $V = XW_V$, skor $QK^\top/\sqrt{2}$, softmax per baris, lalu kali V. Contoh satu elemen: baris `[CLS]` pada head 1, $q = (1,0,1,0)W_Q^{(1)} = (1+1,\ 0+0) = (2,0)$.
Head 1.

$$Q = V = \begin{pmatrix} 2 & 0 \\ 2 & 1 \\ 1 & 2 \\ 1 & 2 \end{pmatrix}, \quad K = \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 3 & 0 \\ 0 & 3 \end{pmatrix}, \quad QK^\top = \begin{pmatrix} 2 & 2 & 6 & 0 \\ 3 & 4 & 6 & 3 \\ 3 & 5 & 3 & 6 \\ 3 & 5 & 3 & 6 \end{pmatrix}$$

$$\frac{QK^\top}{\sqrt{2}} = \begin{pmatrix} 1.414 & 1.414 & 4.243 & 0.000 \\ 2.121 & 2.828 & 4.243 & 2.121 \\ 2.121 & 3.536 & 2.121 & 4.243 \\ 2.121 & 3.536 & 2.121 & 4.243 \end{pmatrix}, \quad \text{bobot} = \begin{pmatrix} 0.052 & 0.052 & 0.883 & 0.013 \\ 0.081 & 0.164 & 0.674 & 0.081 \\ 0.069 & 0.285 & 0.069 & 0.577 \\ 0.069 & 0.285 & 0.069 & 0.577 \end{pmatrix}$$

Contoh softmax baris `[CLS]`: $e^{1.414} = 4.113$, $e^{1.414} = 4.113$, $e^{4.243} = 69.59$, $e^0 = 1$; jumlah 78.82; bobot $(0.052, 0.052, 0.883, 0.013)$. Lalu $z = 0.052(2,0) + 0.052(2,1) + 0.883(1,2) + 0.013(1,2) = (1.104,\ 1.843)$.

$$Z_1 = \begin{pmatrix} 1.104 & 1.843 \\ 1.245 & 1.674 \\ 1.354 & 1.577 \\ 1.354 & 1.577 \end{pmatrix}$$

Head 2.

$$Q = K = V = \begin{pmatrix} 0 & 2 \\ 1 & 2 \\ 2 & 1 \\ 2 & 1 \end{pmatrix}, \quad QK^\top = \begin{pmatrix} 4 & 4 & 2 & 2 \\ 4 & 5 & 4 & 4 \\ 2 & 4 & 5 & 5 \\ 2 & 4 & 5 & 5 \end{pmatrix}, \quad \text{bobot} = \begin{pmatrix} 0.402 & 0.402 & 0.098 & 0.098 \\ 0.199 & 0.403 & 0.199 & 0.199 \\ 0.046 & 0.189 & 0.383 & 0.383 \\ 0.046 & 0.189 & 0.383 & 0.383 \end{pmatrix}$$

$$\frac{QK^\top}{\sqrt{2}} = \begin{pmatrix} 2.828 & 2.828 & 1.414 & 1.414 \\ 2.828 & 3.536 & 2.828 & 2.828 \\ 1.414 & 2.828 & 3.536 & 3.536 \\ 1.414 & 2.828 & 3.536 & 3.536 \end{pmatrix}, \quad Z_2 = \begin{pmatrix} 0.793 & 1.804 \\ 1.199 & 1.602 \\ 1.720 & 1.235 \\ 1.720 & 1.235 \end{pmatrix}$$

Konkatenasi dan residual. $W_O = I$ sehingga keluaran attention = konkatenasi $[Z_1\ Z_2]$. Residual: $Y = X + \text{Concat}$.

$$\text{Concat} = \begin{pmatrix} 1.104 & 1.843 & 0.793 & 1.804 \\ 1.245 & 1.674 & 1.199 & 1.602 \\ 1.354 & 1.577 & 1.720 & 1.235 \\ 1.354 & 1.577 & 1.720 & 1.235 \end{pmatrix}, \quad Y = \begin{pmatrix} 2.104 & 1.843 & 1.793 & 1.804 \\ 1.245 & 2.674 & 3.199 & 1.602 \\ 2.354 & 3.577 & 1.720 & 1.235 \\ 1.354 & 1.577 & 2.720 & 3.235 \end{pmatrix}$$

4. FFN dan residual.
$W_1 = I$ dan $b_1 = 0$, jadi $\mathrm{ReLU}(YW_1) = \mathrm{ReLU}(Y) = Y$ (semua elemen Y positif). Lalu FFN $= YW_2$. Contoh baris pertama: $(2.104 - 1.804,\ 1.843 - 1.793,\ -1.843 + 1.793,\ -2.104 + 1.804) = (0.300,\ 0.050,\ -0.050,\ -0.300)$.

$$\mathrm{FFN} = \begin{pmatrix} 0.300 & 0.050 & -0.050 & -0.300 \\ -0.357 & -0.525 & 0.525 & 0.357 \\ 1.119 & 1.858 & -1.858 & -1.119 \\ -1.881 & -1.142 & 1.142 & 1.881 \end{pmatrix}$$

$$O = \mathrm{FFN} + Y = \begin{pmatrix} 2.404 & 1.894 & 1.743 & 1.504 \\ 0.887 & 2.150 & 3.723 & 1.960 \\ 3.473 & 5.435 & -0.138 & 0.115 \\ -0.527 & 0.435 & 3.862 & 5.115 \end{pmatrix}$$

5. Logits dan softmax.
Vektor `[CLS]` akhir = baris pertama $O = (2.404,\ 1.894,\ 1.743,\ 1.504)$.

$$\text{logit spam} = 2.404(1) + 1.894(0) + 1.743(1) + 1.504(-1) = \mathbf{2.643}$$
$$\text{logit not spam} = 2.404(-1) + 1.894(1) + 1.743(0) + 1.504(1) = \mathbf{0.994}$$

Selisih 1.649: $P(\text{spam}) = 1/(1 + e^{-1.649}) = \mathbf{0.839}$, $P(\text{not spam}) = \mathbf{0.161}$.

6. Prediksi.
Spam, sesuai target.

Cek dan jebakan

Hemat waktu: karena hanya `[CLS]` yang dipakai di akhir (satu layer), cukup hitung q baris `[CLS]`, tetapi K dan V semua baris. FFN dan classifier bekerja per baris. Tiap baris bobot berjumlah 1. Semua matriks dihitung dengan presisi penuh; jika kamu menghitung FFN dan O dari Y yang sudah dibulatkan 3 desimal, selisih 0.001 di beberapa sel (mis. 1.857 vs 1.858) wajar dan tidak mengubah prediksi. Vektor `[CLS]` akhir sama dengan contoh O_{CLS} di catatan Sentence Comparison.

3.8 Soal Penuh: Encoder yang Sama dengan LayerNorm (tambahan)

Varian buatan dari soal sebelumnya: semua data sama, tetapi setelah tiap residual diterapkan LayerNorm (post-norm: $\mathrm{LayerNorm}(X + \mathrm{Attn})$ lalu $\mathrm{LayerNorm}(H + \mathrm{FFN}(H))$), dengan $\gamma = 1$, $\beta = 0$, $\varepsilon = 10^{-5}$. Karena LayerNorm dan FFN bekerja per baris, cukup hitung baris `[CLS]`.

Soal penuh

1. Ambil baris `[CLS]` setelah residual attention, Y_0. Terapkan LayerNorm.
2. Terapkan FFN (ReLU) ke hasilnya, lalu residual dan LayerNorm lagi.
3. Hitung logits, softmax, dan prediksi. Apa beda perannya ReLU dibanding versi tanpa LayerNorm?

Pembahasan soal penuh

1. LayerNorm pertama.
Dari soal sebelumnya, $Y_0 = (2.1044, 1.8434, 1.7934, 1.8044)$ (4 desimal).
$\mu = 7.5456/4 = 1.8864$. Selisih: $(0.2180, -0.0430, -0.0930, -0.0820)$. $\sigma^2 = (0.04752 + 0.00185 + 0.00865 + 0.00672)/4 = 0.01619$; $\sqrt{\sigma^2 + \varepsilon} = 0.1273$.

$$H = \text{LayerNorm}(Y_0) = (\mathbf{1.713},\ \mathbf{-0.338},\ \mathbf{-0.731},\ \mathbf{-0.644})$$

2. FFN, residual, LayerNorm kedua.
$\text{ReLU}(HW_1) = \text{ReLU}(H) = (1.713, 0, 0, 0)$: tiga elemen negatif *dinolkan*. $\text{FFN} = (1.713, 0, 0, 0)\, W_2 = 1.713 \times (1, 0, 0, -1) = (1.713, 0, 0, -1.713)$.
Residual: $H + \text{FFN} = (3.426, -0.338, -0.731, -2.357)$. Rata-rata $= 0$; $\sigma^2 = (11.737 + 0.114 + 0.535 + 5.555)/4 = 4.485$; $\sqrt{4.485} = 2.118$.

$$O_0 = \text{LayerNorm}(H + \text{FFN}) = (\mathbf{1.6176},\ \mathbf{-0.1594},\ \mathbf{-0.3452},\ \mathbf{-1.1130})$$

3. Klasifikasi.
logit spam $= 1.6176 + 0 - 0.3452 + 1.1130 = \mathbf{2.3854}$; logit not spam $= -1.6176 - 0.1594 + 0 - 1.1130 = \mathbf{-2.8900}$. Selisih 5.2754: $P(\text{spam}) = 1/(1 + e^{-5.275}) = \mathbf{0.995}$. Prediksi tetap **spam**, tetapi lebih yakin.
Perbedaan ReLU: tanpa LayerNorm semua nilai positif sehingga ReLU tidak berbuat apa-apa. Setelah LayerNorm, nilai berpusat di 0 sehingga sebagian negatif dan ReLU benar-benar memotongnya.

Cek dan jebakan

Angka di atas dihitung tanpa pembulatan antara. Jika kamu membulatkan di tiap langkah, selisih di digit ketiga wajar (mis. -0.159 vs -0.160). Hasil LayerNorm harus berjumlah ≈ 0.

3.9 Soal Penuh: Satu Langkah Decoder (masked self-attention, cross-attention, keluaran)

Soal penuh

Menerjemahkan "I eat rice" ke bahasa Indonesia. Decoder sudah menghasilkan "<s> saya" dan akan menebak token berikutnya. Asumsi penyederhanaan: satu head, $d_{\text{model}} = d_k = 2$, semua W^Q, W^K, W^V adalah matriks identitas (sehingga $Q = K = V =$ masukan), tanpa LayerNorm dan tanpa FFN.

- Masukan decoder (embedding + posisi): <s> $= (1, 0)$, saya $= (0, 1)$.
- Keluaran encoder: $h_I = (1, 1)$, $h_{eat} = (0, 2)$, $h_{rice} = (2, 0)$.
- Proyeksi ke vocabulary (saya, makan, nasi, </s>): $W_{out} = \begin{pmatrix} 1 & 0 & -1 & 0.5 \\ 0 & 1 & 1 & -0.5 \end{pmatrix}$.

1. Hitung masked self-attention untuk kedua token, lalu residual.
2. Hitung cross-attention untuk posisi terakhir ("saya"): bobot ke tiap kata sumber, context vector, dan residual.
3. Hitung logits, softmax, dan token berikutnya.
4. Apa yang terjadi di langkah berikutnya?

Pembahasan soal penuh

1. Masked self-attention.
$Y = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, jadi $YY^\top/\sqrt{2} = \begin{pmatrix} 0.707 & 0 \\ 0 & 0.707 \end{pmatrix}$.

- `<s>` hanya boleh melihat dirinya: bobot $(1, 0)$, $z = (1, 0)$.
- saya melihat `<s>` dan dirinya: skor $(0,\ 0.707)$. $e^0 = 1$, $e^{0.707} = 2.028$; bobot $(0.330,\ 0.670)$. $z = 0.330(1, 0) + 0.670(0, 1) = (0.330,\ 0.670)$.

Residual: `<s>`: $(1, 0) + (1, 0) = (2, 0)$; saya: $(0, 1) + (0.330, 0.670) = (\mathbf{0.330},\ \mathbf{1.670})$.

2. Cross-attention (posisi "saya").
Query dari decoder $q = (0.330,\ 1.670)$; Key = Value = keluaran encoder.

- Skor: $q \cdot h_I = 0.330 + 1.670 = 2.000$; $q \cdot h_{eat} = 3.340$; $q \cdot h_{rice} = 0.660$. Dibagi $\sqrt{2}$: $(1.414,\ 2.361,\ 0.467)$.
- Softmax: $e^{1.414} = 4.113$, $e^{2.361} = 10.60$, $e^{0.467} = 1.595$; jumlah 16.31. Bobot $(\mathbf{0.252},\ \mathbf{0.650},\ \mathbf{0.098})$: decoder paling memperhatikan "eat".
- Context: $0.252(1, 1) + 0.650(0, 2) + 0.098(2, 0) = (\mathbf{0.448},\ \mathbf{1.552})$.
- Residual: $s = q + \text{context} = (0.330 + 0.448,\ 1.670 + 1.552) = (\mathbf{0.778},\ \mathbf{3.222})$.

3. Keluaran.
Logits $= sW_{out} = (0.778,\ 3.222,\ -0.778+3.222,\ 0.389-1.611) = (0.778,\ 3.222,\ 2.444,\ -1.222)$.
Softmax: $(\mathbf{0.056},\ \mathbf{0.642},\ \mathbf{0.295},\ \mathbf{0.008})$ untuk (saya, makan, nasi, `</s>`). Argmax: **"makan"**.

4. Langkah berikutnya.
"makan" ditambahkan ke masukan decoder ("`<s>` saya makan"), lalu proses diulang. Keluaran encoder (h_I, h_{eat}, h_{rice}) dipakai ulang tanpa dihitung lagi. Proses berhenti saat model mengeluarkan `</s>`.

Cek dan jebakan

Di cross-attention, **Query dari decoder, Key dan Value dari encoder**. Anotasi slide "K, Q from encoder; V from decoder" adalah salah cetak. Jumlah bobot cross-attention = 1.

3.10 Kompleksitas Self-Attention vs RNN

Rumus

Layer	Kompleksitas per layer	Operasi berurutan	Panjang jalur maksimum
Self-attention	$O(n^2 d)$	$O(1)$	$O(1)$
Recurrent	$O(nd^2)$	$O(n)$	$O(n)$
Convolutional	$O(knd^2)$	$O(1)$	$O(\log_k n)$

n: panjang sequence; d: dimensi; k: ukuran kernel. Self-attention lebih murah saat $n < d$, dan selalu paralel penuh.

Soal

$n = 100$, $d = 512$. Mana yang lebih sedikit operasinya per layer? Bagaimana jika $n = 1000$?

Pembahasan

$n = 100$: self-attention $100^2 \times 512 = 5{,}120{,}000$; RNN $100 \times 512^2 = 26{,}214{,}400$. Self-attention sekitar **5 kali lebih sedikit**.
$n = 1000$: self-attention $1000^2 \times 512 = 512{,}000{,}000$; RNN $1000 \times 512^2 = 262{,}144{,}000$. Sekarang **RNN lebih sedikit**, karena $n > d$. Inilah harga kompleksitas kuadratik terhadap panjang sequence.

4 BERT

Materi: apa itu BERT dan bagaimana dilatih

BERT (Bidirectional Encoder Representations from Transformers) hanya memakai bagian **encoder** Transformer. Karena encoder melihat seluruh kalimat sekaligus (kiri dan kanan), BERT cocok untuk *memahami* teks (NLU): klasifikasi, NLI, question answering, tagging.
Pre-training (tanpa label manual), dua tugas sekaligus:

1. **Masked Language Model (MLM)**: 15% token dipilih untuk ditebak. Model menebak kata aslinya dari konteks kiri dan kanan, lewat FFNN + softmax atas seluruh vocabulary. Inilah yang membuat BERT *bidirectional*. (Tambahan, dari paper BERT: dari token terpilih, 80% diganti `[MASK]`, 10% diganti token acak, dan 10% dibiarkan, supaya model tidak hanya belajar dari token `[MASK]`.)
2. **Next Sentence Prediction (NSP)**: dua kalimat A dan B digabung (`[CLS]` A `[SEP]` B `[SEP]`); dari keluaran `[CLS]`, model menebak apakah B memang kalimat lanjutan A (*IsNext*) atau acak (*NotNext*).

Representasi masukan adalah *jumlah* tiga embedding: **token embedding** (WordPiece), **segment embedding** (E_A untuk kalimat A, E_B untuk B), dan **position embedding** (dipelajari, bukan sin-cos).
Dua cara memakai BERT:

- **Fine-tuning**: tambahkan classifier di atas `[CLS]`, lalu latih ulang *seluruh* model pada data tugas.
- **Feature extraction**: BERT dibekukan; ambil vektornya (dari satu atau beberapa layer) sebagai fitur untuk algoritma lain, mis. SVM.

Filosofinya *pretrain once, finetune many times*: pre-training mahal (64 TPU, 4 hari), fine-tuning cukup satu GPU. Ukuran: **BERT-Base** 12 layer, hidden 768, 12 head, ∼110 juta parameter; **BERT-Large** 24 layer, hidden 1024, 16 head, ∼340 juta. Varian: RoBERTa (tanpa NSP, data lebih besar), ALBERT (berbagi parameter), SpanBERT (mask potongan berurutan), DistilBERT (distillation).

4.1 Tokenisasi WordPiece

Materi

WordPiece memecah kata yang tidak ada di vocabulary menjadi potongan (subword). Potongan lanjutan diberi awalan `##`, misalnya *capitalization* → capital + ##ization. Cara memotongnya (tambahan, tidak dirinci di slide) adalah *greedy longest-match-first*: mulai dari awal kata, ambil potongan terpanjang yang ada di vocabulary, lalu ulangi pada sisanya dengan awalan `##`. Mirip Maximum Matching di Bab 1. Jika tidak ada potongan yang cocok, kata menjadi `[UNK]`.

Soal

Vocabulary memuat: un, ##afford, ##able, ##aff, ##ord, afford, able. Tokenisasi "unaffordable".

Pembahasan

1. Dari awal: "unaffordable", "unaffordabl", … tidak ada di vocabulary; yang pertama cocok saat dipendekkan adalah **un**.
2. Sisa "affordable" dicari sebagai "##…". Potongan terpanjang yang ada: **##afford** (lebih panjang dari ##aff).
3. Sisa "able": **##able**. Perhatikan bahwa yang dipakai ##able, bukan able, karena potongan ini bukan awal kata.

Hasil: [un, ##afford, ##able].

4.2 Hitungan Ukuran BERT

Rumus

$$\#\text{token di-mask} = 0.15 \times \text{panjang sequence}$$

$$\text{epoch} = \frac{\text{step} \times \text{token per batch}}{\text{token korpus}}$$

$$\#\text{param per layer encoder} \approx 12d^2 \quad (4d^2 \text{ attention} + 8d^2 \text{ FFN dengan } d_{ff} = 4d) \text{ (tambahan)}$$

$$\#\text{param classifier head} = d \times C + C$$

Arti simbol

Simbol	Arti	Contoh / ukuran
d	hidden size	768 (Base), 1024 (Large)
C	jumlah kelas	2 atau 3
step	jumlah update training	10^6
token per batch	di slide 131,072 (1024×128 atau 256×512)	
token korpus	Wikipedia 2.5 miliar + BookCorpus 0.8 miliar kata	3.3×10^9
vocabulary	ukuran WordPiece BERT (tambahan, bukan dari slide)	$\approx$ 30,522

Soal

a. Sequence 512 token: berapa token di-mask?

b. Dengan angka di slide, berapa epoch pre-training BERT?

c. Perkirakan parameter BERT-Base dan BERT-Large.

Pembahasan

a. $0.15 \times 512 = 76.8 \approx$ **77** token dipilih. (Tambahan: dengan aturan 80/10/10, sekitar 61 menjadi `[MASK]`, 8 diganti token acak, dan 8 dibiarkan.)
b. $10^6 \times 131{,}072 = 1.31 \times 10^{11}$ token; dibagi 3.3×10^9 memberi $39.7 \approx$ **40** epoch (cocok dengan slide).
c. Base: $12 \times 768^2 = 7{,}077{,}888$ per layer, $\times 12$ layer $= 84{,}934{,}656$. Token embedding $30{,}522 \times 768 = 23{,}440{,}896$. Jumlah $= 108{,}375{,}552 \approx$ **108.4** juta. Ditambah position ($512 \times 768 = 393{,}216$) dan

segment ($2 \times 768 = 1{,}536$) embedding menjadi 108,770,304 $\approx$ 108.8 juta. Bias, LayerNorm, dan pooler tidak dihitung, jadi mendekati "$\sim$110 juta".
Large: $12 \times 1024^2 \times 24 = 301{,}989{,}888$; token embedding $30{,}522 \times 1024 = 31{,}254{,}528$; jumlah $\approx$ **333.2** juta, mendekati "$\sim$340 juta".

4.3 Soal Penuh: BERT dari Masukan sampai Kepala Klasifikasi

Soal penuh

Kalimat A = "my dog", kalimat B = "it barks".

1. Susun deretan token masukan BERT, segment id, dan position id.
2. Dengan vektor 2 dimensi berikut, hitung vektor masukan untuk tiga token pertama. Token: `[CLS]`$= (0.1, 0.2)$, my $= (0.3, 0.1)$, dog $= (0.5, 0.4)$. Segment: $E_A = (0, 0.1)$, $E_B = (0.1, 0)$. Posisi: $P_0 = (0.01, 0.02)$, $P_1 = (0.02, 0)$, $P_2 = (0, 0.03)$.
3. MLM: kata "barks" di-mask. Keluaran BERT di posisi itu $h = (0.5, -0.2)$, dan lapisan prediksi atas vocabulary kecil (barks, dog, cat, run) adalah $W = \begin{pmatrix} 2 & 1 & 0 & -1 \\ 0 & -1 & 1 & 2 \end{pmatrix}$. Hitung logits, softmax, prediksi, dan loss.
4. NSP: keluaran `[CLS]` $= (1.2, -0.4)$, bobot NSP $W_{nsp} = \begin{pmatrix} 1 & -1 \\ 0.5 & 0.5 \end{pmatrix}$ (kolom 1 IsNext, kolom 2 NotNext). Hitung P(IsNext).
5. Untuk fine-tuning sentimen 3 kelas pada BERT-Base, berapa parameter classifier baru? Berapa parameter yang dilatih pada fine-tuning vs feature extraction?
6. Berapa jumlah skor attention yang dihitung untuk sequence 512 token di seluruh BERT-Base?

Pembahasan soal penuh

1. Token, segment, posisi.

Token	`[CLS]`	my	dog	`[SEP]`	it	barks	`[SEP]`
Segment	A	A	A	A	B	B	B
Posisi	0	1	2	3	4	5	6

`[SEP]` pertama masih milik kalimat A. Total 7 token.

2. Vektor masukan
= token + segment + posisi:

- `[CLS]`: $(0.1, 0.2) + (0, 0.1) + (0.01, 0.02) = (\mathbf{0.11},\ \mathbf{0.32})$
- my: $(0.3, 0.1) + (0, 0.1) + (0.02, 0) = (\mathbf{0.32},\ \mathbf{0.20})$
- dog: $(0.5, 0.4) + (0, 0.1) + (0, 0.03) = (\mathbf{0.50},\ \mathbf{0.53})$

3. MLM.
Logits $= hW = (0.5 \cdot 2 + (-0.2) \cdot 0,\ 0.5 \cdot 1 + (-0.2)(-1),\ 0 + (-0.2)(1),\ -0.5 + (-0.2)(2)) = (1.0,\ 0.7,\ -0.2,\ -0.9)$.
$e^{1.0} = 2.718$, $e^{0.7} = 2.014$, $e^{-0.2} = 0.819$, $e^{-0.9} = 0.407$; jumlah 5.957. Softmax: (**0.456**, **0.338**, **0.137**, **0.068**).
Prediksi: **barks** (benar). Loss $= -\ln 0.456 =$ **0.785**.

4. NSP.
Logits $= (1.2(1) + (-0.4)(0.5),\ 1.2(-1) + (-0.4)(0.5)) = (1.0,\ -1.4)$. Selisih 2.4: $P(\text{IsNext}) = 1/(1+e^{-2.4}) = \mathbf{0.917}$.

5. Classifier.
$768 \times 3 + 3 = \mathbf{2{,}307}$ parameter baru. Fine-tuning melatih *semua* parameter (≈ 110 juta $+2{,}307$). Feature extraction hanya melatih classifier di atasnya (mis. 2,307 jika classifier-nya linear), karena BERT dibekukan.

6. Skor attention.
Satu head satu layer: $512^2 = 262{,}144$ skor. BERT-Base 12 head $\times$ 12 layer: $262{,}144 \times 144 = \mathbf{37{,}748{,}736}$. Kuadratik terhadap panjang sequence; itu sebabnya batas panjang BERT 512.

Cek dan jebakan

Segment embedding hanya dua vektor (E_A, E_B). Position embedding BERT dipelajari, sehingga maksimal 512 posisi. Loss MLM hanya dihitung di posisi yang *dipilih* (15%), bukan di semua token; termasuk posisi terpilih yang diganti token acak atau dibiarkan.

5 Decoder, Prompting, dan PEFT

Materi: dari prompting sampai fine-tuning hemat

LLM dan prompting. Model AI tradisional dibuat satu model untuk satu tugas. LLM (decoder seperti GPT) memakai *satu model untuk banyak tugas*, cukup dengan mengganti instruksi (*prompt*). Model hasil pretraining murni (*base model*) belum menjadi asisten: ia hanya melanjutkan teks. Prompt yang baik punya empat elemen: *instruction* (tugas), *context* (informasi tambahan), *input data* (pertanyaan), dan *output indicator* (format jawaban).
Teknik prompting.

- **Zero-shot**: instruksi tanpa contoh. **Few-shot**: beberapa contoh berlabel di dalam prompt. Few-shot disebut *in-context learning* karena model "belajar" dari contoh *tanpa* update bobot (tanpa gradient).
- **Chain-of-Thought (CoT)**: minta langkah penalaran sebelum jawaban (few-shot CoT, atau zero-shot CoT dengan "Let's think step by step"). **Auto-CoT**: kelompokkan pertanyaan, ambil satu pertanyaan wakil per kelompok, lalu bangkitkan penalarannya otomatis.
- **Self-Consistency**: ambil beberapa rantai penalaran untuk soal yang sama, lalu pilih jawaban yang paling sering muncul.
- **Generate Knowledge Prompting**: model menulis fakta relevan dulu, lalu memakainya untuk menjawab.

Kelemahan prompting: tidak efisien (prompt diproses ulang tiap prediksi), umumnya kalah dari fine-tuning, dan sensitif terhadap kata-kata serta urutan contoh.
Fine-tuning. Pretraining (teks tanpa label, belajar bahasa umum) berfungsi sebagai inisialisasi; fine-tuning (sedikit data berlabel) menyesuaikan ke tugas. **Instruction finetuning** (mis. FLAN-T5) melatih banyak pasangan (instruksi, jawaban) dari banyak tugas, lalu dievaluasi pada *unseen tasks*.
PEFT (Parameter-Efficient Fine-Tuning). Melatih semua parameter model besar terlalu mahal, dan model besar *over-parameterized*. PEFT membekukan (*freeze*) model dasar dan hanya melatih sebagian kecil parameter:

Kategori	Prinsip	Contoh
Additive	menambah komponen baru	Adapter, prefix, $(IA)^3$, SSF
Selective	memilih sebagian parameter lama	BitFit (hanya bias), masking
Reparameterized	struktur low-rank, bisa di-merge kembali	LoRA, DyLoRA, DoRA
Hybrid	gabungan	UniPELT, MAM Adapter

Letak di blok (menurut gambar slide): LoRA paralel pada proyeksi Q (dan biasanya V), Adapter setelah sub-layer FFN, soft prompt sebagai token yang dipelajari di depan masukan. (Adapter versi asli Houlsby memasang dua adapter per layer, setelah attention dan setelah FFN.)

5.1 Self-Consistency

Soal

Lima rantai penalaran untuk soal yang sama menghasilkan jawaban 18, 18, 20, 18, 26. Apa jawaban akhir Self-Consistency dan seberapa besar kesepakatannya?

Pembahasan

Hitung frekuensi: 18 muncul 3 kali, 20 sekali, 26 sekali. Ambil yang terbanyak: **18**, dengan kesepakatan $3/5 = \mathbf{60}\%$. Jawaban dari satu rantai saja (mis. 26) bisa salah; voting mengurangi pengaruh rantai yang keliru.

5.2 LoRA (Low-Rank Adaptation)

Materi

LoRA membekukan bobot asli W_0 dan menambahkan "koreksi" ΔW yang dibuat dari dua matriks kecil: W_{down} (memperkecil dimensi ke r) dan W_{up} (memperbesar kembali). Karena r sangat kecil, parameter yang dilatih jauh lebih sedikit. Setelah training, ΔW bisa *digabung* ke W_0, sehingga saat inference tidak ada tambahan waktu.

Rumus

$$h = W_0 x + W_{up}(W_{down} x), \qquad \Delta W = W_{up} W_{down}, \qquad W_{merge} = W_0 + \Delta W$$

$$\#\text{param LoRA} = r(d+k) \quad \text{vs} \quad \#\text{param penuh} = d \cdot k$$

Arti simbol

Simbol	Arti	Contoh / ukuran
W_0	bobot asli yang dibekukan $(d \times k)$	4×4
W_{down}	matriks penurun dimensi $(r \times k)$, dilatih	1×4
W_{up}	matriks penaik dimensi $(d \times r)$, dilatih	4×1
r	rank; $r \ll \min(d, k)$	1, 8
x, h	masukan dan keluaran layer	

Soal

$W_0 = \begin{pmatrix} 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 2 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$, $x = (1, 2, 0, -1)^\top$, $r = 1$, $W_{down} = (1, 0, 1, 0)$, $W_{up} = (0.5,\ 0,\ -0.5,\ 1)^\top$.

a. Hitung h lewat dua jalur (beku + LoRA).

b. Hitung W_{merge} dan cek bahwa $W_{merge}x = h$.

c. Berapa parameter yang dilatih dibanding fine-tuning penuh matriks ini?

Pembahasan

a. Jalur beku: $W_0x = (1+0+0+0,\ 0+2+0-2,\ 1+2+0+0,\ 0+0+0-1) = (1,\ 0,\ 3,\ -1)$.
Jalur LoRA: $W_{down}x = 1 + 0 = 1$ (satu angka, karena $r = 1$); $W_{up} \cdot 1 = (0.5,\ 0,\ -0.5,\ 1)$.
$h = (1.5,\ 0,\ 2.5,\ 0)$.
b. $\Delta W = W_{up}W_{down}$ (vektor kolom kali vektor baris):

$$\Delta W = \begin{pmatrix} 0.5 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 \\ -0.5 & 0 & -0.5 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix}, \quad W_{merge} = \begin{pmatrix} 1.5 & 0 & 2.5 & 0 \\ 0 & 1 & 0 & 2 \\ 0.5 & 1 & -0.5 & 0 \\ 1 & 0 & 2 & 1 \end{pmatrix}$$

$W_{merge}x = (1.5,\ 0,\ 0.5 + 2,\ 1 - 1) = (1.5,\ 0,\ 2.5,\ 0) = h$. Sama, jadi merge tidak mengubah hasil.
c. LoRA: $r(d + k) = 1 \times (4 + 4) = \mathbf{8}$; penuh: $4 \times 4 = \mathbf{16}$. Pada ukuran nyata ($d = k = 1024$, $r = 8$): $8 \times 2048 = 16{,}384$ vs $1{,}048{,}576$, yaitu **1.56%**.

Cek dan jebakan

ΔW dari LoRA selalu ber-rank paling banyak r (di sini semua barisnya kelipatan $(1, 0, 1, 0)$). DyLoRA melatih beberapa nilai r sekaligus; DoRA memisah bobot menjadi besaran dan arah: $W = m \cdot V/\|V\|_c$, misalnya kolom $V = (3, 4)$ dengan $m = 10$ menjadi $10 \times (3, 4)/5 = (6, 8)$.

5.3 Adapter

Rumus

$$\text{Adapter}(x) = W_{up}\,\sigma(W_{down}x) + x, \qquad \#\text{param} = 2dm\ (+m + d\ \text{bias})$$

Arti simbol

Simbol	Arti	Contoh / ukuran
W_{down}	$m \times d$: memperkecil ke bottleneck m	2×4
σ	aktivasi, mis. ReLU	
W_{up}	$d \times m$: mengembalikan ke dimensi d	4×2
$+x$	residual: adapter hanya menambah koreksi	
m	ukuran bottleneck, $m \ll d$	64

Soal

$x = (1, -1, 2, 0)$, $W_{down} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$, ReLU, $W_{up} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 0.5 & 0 \end{pmatrix}$. Hitung keluaran adapter dan jumlah parameternya.

Pembahasan

$W_{down}x = (1+2,\ -1+0) = (3, -1)$; ReLU $\rightarrow$ $(3, 0)$; $W_{up}(3, 0) = (3, 0, 3, 1.5)$; tambah x: $(\mathbf{4},\ \mathbf{-1},\ \mathbf{5},\ \mathbf{1.5})$.
Parameter: $2 \times 4 \times 2 = \mathbf{16}$ (tanpa bias). Untuk $d = 768$, $m = 64$: $2 \times 768 \times 64 = 98{,}304$, dengan bias 99,136.

5.4 $(IA)^3$, Prefix-Tuning, dan Selective PEFT

Rumus

$$(IA)^3: \quad \mathrm{SA}(x) = \mathrm{softmax}\Big(\frac{Q\,(l_k \odot K^\top)}{\sqrt{d_{head}}}\Big)(l_v \odot V), \qquad \mathrm{FFN}(x) = W_{up}\big(l_{ff} \odot \sigma(W_{down}x)\big)$$

$$\text{Prefix: } X^{(l)} = [s_1^{(l)}, \ldots, s_{N_S}^{(l)}, x_1^{(l)}, \ldots, x_{N_X}^{(l)}], \qquad \text{Selective: } \theta_i' = \theta_i - \eta\, m_i\, \frac{\partial L}{\partial \theta_i}$$

Arti simbol

Simbol	Arti	Contoh / ukuran
l_k, l_v, l_{ff}	vektor skala yang dilatih untuk key, value, aktivasi FFN	$(1, 0.5, 2)$
$\odot$	perkalian elemen demi elemen	$(1,2,3) \odot (1,0.5,2) = (1,1,6)$
$s_i^{(l)}$	vektor prefix yang dilatih di layer l, ditempel di depan key dan value	
N_S	jumlah vektor prefix	20
m_i	mask biner: 1 dilatih, 0 dibekukan	$(1, 0, 1)$
η	learning rate	0.1

Soal

a. Key $k = (1, 2, 3)$ diskalakan $l_k = (1, 0.5, 2)$. Hasilnya?
b. Selective: $\theta = (0.5, -0.2, 0.8)$, gradient $(0.1, 0.4, -0.3)$, $m = (1, 0, 1)$, $\eta = 0.1$. Hitung θ'.

Pembahasan

a. $(1 \cdot 1,\ 2 \cdot 0.5,\ 3 \cdot 2) = (\mathbf{1},\ \mathbf{1},\ \mathbf{6})$.
b. $\theta_1' = 0.5 - 0.1(1)(0.1) = 0.49$; $\theta_2' = -0.2$ (mask 0, tidak berubah); $\theta_3' = 0.8 - 0.1(1)(-0.3) = 0.83$. Hasil $(\mathbf{0.49},\ \mathbf{-0.20},\ \mathbf{0.83})$.

5.5 Soal Penuh: Membandingkan Biaya Metode PEFT

Soal penuh

Model encoder seukuran BERT-Base: $L = 12$ layer, $d = 768$, $d_{ff} = 3072$. Anggap parameter inti per layer $12d^2$ (abaikan embedding).

1. Berapa parameter yang dilatih pada full fine-tuning (bagian layer saja)?
2. Hitung parameter yang dilatih untuk: LoRA $r = 8$ di W^Q dan W^V; Adapter $m = 64$ (satu per layer, dengan bias); $(IA)^3$; BitFit (bias Q, K, V, O dan dua bias FFN, tanpa LayerNorm); Prefix-tuning $N_S = 20$ (di key dan value tiap layer, tanpa MLP reparameterisasi).
3. Nyatakan tiap angka sebagai persen dari jawaban nomor 1.
4. Ada 10 tugas. Bandingkan total parameter yang disimpan: full fine-tuning (10 salinan model) vs LoRA (1 model dasar + 10 set LoRA).

Pembahasan soal penuh

1. Full fine-tuning.
$12 \times 768^2 \times 12 = \mathbf{84{,}934{,}656}$.

2 dan 3. Per metode.

Metode	Cara menghitung	Parameter	Persen
LoRA	$12 \times 2 \times r(d+d) = 12 \times 2 \times 8 \times 1536$	294,912	0.347%
Adapter	$12 \times (2 \cdot 768 \cdot 64 + 64 + 768)$	1,189,632	1.401%
$(IA)^3$	$12 \times (768 + 768 + 3072)$	55,296	0.065%
BitFit	$12 \times (4 \cdot 768 + 3072 + 768)$	82,944	0.098%
Prefix	$12 \times 2 \times 20 \times 768$	368,640	0.434%

Persen = parameter /84,934,656 $\times$ 100%. Contoh LoRA: 294,912/84,934,656 = 0.00347 = 0.347%.

4. Penyimpanan untuk 10 tugas.
Full fine-tuning: $10 \times 84{,}934{,}656 = \mathbf{849{,}346{,}560}$. LoRA: $84{,}934{,}656 + 10 \times 294{,}912 = \mathbf{87{,}883{,}776}$, sekitar 10 kali lebih hemat. Inilah alasan PEFT: satu model dasar dipakai bersama, dan tiap tugas hanya menyimpan bagian kecilnya.

Cek dan jebakan

Angka BitFit dan rincian bias adalah perkiraan tambahan (slide hanya menyebut "BitFit hanya melatih bias"). Pada soal ujian, ikuti asumsi yang diberikan soal.

6 Sentence Comparison

Materi: membandingkan dua kalimat

Dua jenis kemiripan. *Lexical*: kalimat jadi vektor *sparse* (bag of words), dibandingkan dengan cosine, sehingga yang diukur adalah kesamaan *kata*. *Semantic*: kalimat jadi vektor *dense* (sentence embedding), sehingga yang diukur adalah kesamaan *makna*. "Institut Teknologi Bandung" dan "Bandung Institute of Technology" hampir tidak berbagi kata, tetapi maknanya sama.
Sentence embedding dari encoder (BERT). Tiga cara: (1) ambil hidden state `[CLS]` sebelum classifier; (2) rata-rata semua hidden state termasuk `[CLS]` dan `[SEP]`; (3) rata-rata hidden

state token kata saja. Cara meringkas banyak vektor menjadi satu ini disebut *pooling*.
Melatih encoder untuk perbandingan. BERT dasar hanya dilatih MLM dan NSP, jadi embedding-nya belum bagus untuk cosine. Dataset yang dipakai:

- **NLI**: pasangan (premise, hypothesis) berlabel *entail*, *contradict*, atau *neutral*.
- **STS**: pasangan kalimat dengan skor kemiripan 0–5.

SBERT di-fine-tune pada NLI lalu STS. Variasinya: **SimCSE** (unsupervised: kalimat yang sama diproses dua kali dengan dropout berbeda sebagai pasangan positif; supervised: NLI, kalimat contradiction sebagai negatif); **E5** (pre-training contrastive dengan data weakly supervised seperti judul+paragraf, lalu fine-tuning dengan hard negative; awalan `query:`/`passage:`); **BGE-M3** (multilingual >100 bahasa, tiga cara retrieval: dense, sparse, multi-vector, hingga 8192 token, dengan self-knowledge distillation).
Tiga arsitektur.

- **Cross-encoder**: dua kalimat digabung (`[CLS]` S1 `[SEP]` S2 `[SEP]`) dan diproses bersama. Paling akurat, tetapi mahal dan tidak bisa disimpan sebelumnya. Hanya model neural sequence yang bisa memprosesnya.
- **Bi-encoder + cosine**: tiap kalimat di-encode terpisah (bisa disimpan), lalu dibandingkan dengan cosine dan threshold. Cepat untuk skala besar.
- **Bi-encoder + classifier**: fitur u, v, $|u-v|$ dimasukkan ke classifier. Karena fiturnya vektor berukuran tetap, algoritma ML klasik apa pun bisa dipakai.

Tugas dua kalimat: paraphrase identification, NLI, QNLI/RTE, fact verification, stance detection. **Zero-shot classification dengan NLI**: kalimat masukan jadi premise, tiap label jadi hypothesis "this text is {label}", dan label dengan $P(\text{entailment})$ tertinggi dipilih.

6.1 Cosine Leksikal

Soal

"I love NLP" vs "I love AI". Hitung cosine bag-of-words.

Pembahasan

Vocabulary (I, love, NLP, AI): $u = (1,1,1,0)$, $v = (1,1,0,1)$. $u \cdot v = 2$; $\|u\| = \|v\| = \sqrt{3}$. $\cos = 2/(\sqrt{3} \cdot \sqrt{3}) = 2/3 = \mathbf{0.667}$.

Peringatan

Angka slide yang tidak cocok. Contoh "Institut Teknologi Bandung" vs "Bandung Institute of Technology" ditulis $\cos = 1/(2\sqrt{3}) = 0.1667$. Itu tidak konsisten: $1/(2\sqrt{3}) = 0.2887$, dan rumus yang tertulis $(\sqrt{3} \times \sqrt{3})$ memberi $1/3 = 0.333$. Dengan kosakata (Institut, Teknologi, Bandung, Institute, of, Technology), kalimat 1 punya 3 token dan kalimat 2 punya 4 token, dengan satu kata sama (*Bandung*): $\cos = 1/(\sqrt{3} \times 2) = 0.2887$. Jangan hafal 0.1667.

6.2 Pooling Sentence Embedding

Rumus

$$s_{\texttt{[CLS]}} = h_{\texttt{[CLS]}}, \qquad s_{\text{mean}} = \frac{1}{n}\sum_{i=1}^{n} h_i$$

Arti simbol

Simbol	Arti	Contoh / ukuran
h_i	hidden state token ke-i dari layer terakhir encoder	$(1, 1)$
n	jumlah token yang dirata-ratakan (dengan atau tanpa `[CLS]` / `[SEP]`)	5 atau 3

Soal

Kalimat 1 "I love NLP": $h_{\texttt{[CLS]}} = (2, 0)$, $h_I = (1, 1)$, $h_{love} = (0, 2)$, $h_{NLP} = (1, 3)$, $h_{\texttt{[SEP]}} = (0, 0)$. Hitung tiga jenis sentence embedding. Lalu kalimat 2 "I like AI" punya token $h_I = (1, 1)$, $h_{like} = (0, 1)$, $h_{AI} = (2, 2)$; hitung cosine kedua kalimat dengan cara 3.

Pembahasan

1. `[CLS]`: $(\mathbf{2},\ \mathbf{0})$.
2. Rata-rata 5 token: $((2+1+0+1+0)/5,\ (0+1+2+3+0)/5) = (\mathbf{0.8},\ \mathbf{1.2})$.
3. Rata-rata token kata saja: $((1+0+1)/3,\ (1+2+3)/3) = (\mathbf{0.667},\ \mathbf{2.0})$.

Kalimat 2 (cara 3): $((1+0+2)/3,\ (1+1+2)/3) = (1,\ 1.333)$.

$$\cos = \frac{0.667(1) + 2(1.333)}{\sqrt{0.444 + 4}\,\sqrt{1 + 1.778}} = \frac{3.333}{2.108 \times 1.667} = \mathbf{0.949}.$$

6.3 Bi-encoder dengan Classifier (gaya SBERT)

Rumus

$$\text{fitur} = (u,\ v,\ |u - v|), \qquad o = \text{softmax}(\text{fitur} \cdot W_t), \qquad \#\text{param} = 3n \cdot k\ (+k)$$

Arti simbol

Simbol	Arti	Contoh / ukuran
u, v	sentence embedding kalimat 1 dan 2 (dimensi n)	$(1, 0.5)$
$\lvert u - v \rvert$	selisih mutlak per elemen	$(0.5, 0.5)$
W_t	bobot classifier $(3n \times k)$	6×3
k	jumlah kelas (NLI: entail, neutral, contradict)	3

Soal

$u = (1,\ 0.5)$, $v = (0.5,\ 1)$. Kolom W_t berurutan (entail, neutral, contradict):

$$W_t = \begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & 1 & 0 \\ -2 & 0 & 2 \\ -2 & 0 & 2 \end{pmatrix}$$

Hitung fitur, logits, softmax, dan label. Berapa cosine u dan v? Berapa parameter classifier jika $n = 768$?

Pembahasan

$|u - v| = (0.5,\ 0.5)$, jadi fitur $= (1,\ 0.5,\ 0.5,\ 1,\ 0.5,\ 0.5)$.
Logits: entail $= 1+0+0.5+0-1-1 = -0.5$; neutral $= 0+0.5+0+1+0+0 = 1.5$; contradict $= -1+0-0.5+0+1+1 = 0.5$.
Softmax: (**0.090**, **0.665**, **0.245**), jadi label **neutral**.
$\cos(u, v) = \dfrac{0.5+0.5}{\sqrt{1.25}\sqrt{1.25}} = \mathbf{0.8}$.
Parameter untuk $n = 768$, $k = 3$: $3 \times 768 \times 3 + 3 = \mathbf{6{,}915}$.

6.4 Loss Kontrastif dan Temperatur τ

Rumus

$$L = -\ln \frac{e^{\cos(u,v^+)/\tau}}{\sum_i e^{\cos(u,v_i)/\tau}}$$

Arti simbol

Simbol	Arti	Contoh / ukuran
u	vektor anchor (kalimat acuan)	$(1, 0)$
v^+	vektor positif (seharusnya mirip)	$(0.8, 0.6)$
v_i	semua kandidat: positif *dan* semua negatif	
τ	temperatur; makin kecil makin tajam	0.1
hard negative	negatif yang mirip di permukaan, sengaja dipilih agar model teliti	"Lyon …"

Soal

Anchor $(1, 0)$, positif $(0.8, 0.6)$, negatif $(0, 1)$ dan $(0.6, -0.8)$. Hitung loss dengan $\tau = 0.1$ dan $\tau = 1$.

Pembahasan

Cosine ke anchor (semua vektor panjangnya 1): positif 0.8, negatif 1 0, negatif 2 0.6.

- $\tau = 0.1$: eksponen (8, 0, 6): $e^8 = 2981$, $e^0 = 1$, $e^6 = 403.4$; jumlah 3385.4; $p^+ = 0.8805$; $L = -\ln 0.8805 = \mathbf{0.127}$.
- $\tau = 1$: eksponen (0.8, 0, 0.6): 2.226, 1, 1.822; jumlah 5.048; $p^+ = 0.441$; $L = \mathbf{0.819}$.

τ kecil memperbesar jarak antar skor sehingga positif (yang memang paling mirip) mendominasi dan loss kecil.

6.5 Skor Hybrid BGE-M3

Rumus

$$\begin{aligned} s_{\text{dense}} &= \cos(u_q, v_d) \\ s_{\text{sparse}} &= \sum_{t \in Q \cap D} w_t^Q \, w_t^D \\ s_{\text{multi}} &= \sum_{q_i \in Q} \max_{d_j \in D} \cos(q_i, d_j) \\ s_{\text{rank}} &= w_1 s_{\text{dense}} + w_2 s_{\text{sparse}} + w_3 s_{\text{multi}} \end{aligned}$$

Arti simbol

Simbol	Arti	Contoh / ukuran
u_q, v_d	vektor dense query dan dokumen (hasil pooling)	
$t \in Q \cap D$	token yang ada di query *dan* dokumen (persis sama)	"for"
w_t^Q, w_t^D	bobot token t dari layer linear ("kepentingan" kata)	0.05
q_i, d_j	vektor tiap token query dan dokumen	
$\max_{d_j}$	untuk tiap token query, ambil token dokumen yang paling mirip	0.93
w_1, w_2, w_3	bobot gabungan; paper memakai (1, 0.3, 1)	

Soal

Skor sudah dihitung: D_1: dense 0.90, sparse 0.10, multi 2.20; D_2: dense 0.40, sparse 0.50, multi 2.00. Bobot (1, 0.3, 1). Hitung s_{rank} dan cari w_2 agar keduanya seri.

Pembahasan

D_1: $0.90 + 0.3(0.10) + 2.20 = \mathbf{3.13}$. D_2: $0.40 + 0.3(0.50) + 2.00 = \mathbf{2.55}$. D_1 menang.
Seri: $0.90 + 2.20 + 0.10w_2 = 0.40 + 2.00 + 0.50w_2 \Rightarrow 0.70 = 0.40w_2 \Rightarrow \mathbf{w_2 = 1.75}$. Pada $w_2 = 1.75$ skornya seri; untuk $w_2 > 1.75$, D_2 menang karena sinyal leksikal terlalu dominan.

6.6 Biaya Cross- vs Bi-encoder dan Zero-shot NLI

Soal

a. Satu query dibandingkan dengan 10,000 kalimat. Berapa forward pass encoder untuk cross-encoder dan bi-encoder? Bagaimana jika 10,000 kalimat itu ingin dikelompokkan (semua pasangan)?
b. Zero-shot NLI untuk "I just adopted a puppy": P(entailment) untuk hypothesis "this text is pets" = 0.94, "…politics" = 0.02, "…sports" = 0.03. Label apa yang dipilih?

Pembahasan

a. Cross-encoder: setiap pasangan diproses penuh, jadi **10,000** forward pass untuk satu query. Bi-encoder: 10,000 kalimat di-encode sekali dan disimpan, lalu tiap query cukup 1 forward pass dan 10,000 cosine. Untuk semua pasangan: cross-encoder butuh $n(n-1)/2 = 10{,}000 \times 9{,}999/2 =$ **49,995,000** forward pass; bi-encoder tetap **10,000**.
b. Ambil P(entailment) terbesar: **pets**. Tiap pasangan dijalankan terpisah, jadi angkanya tidak harus berjumlah 1.

6.7 Soal Penuh: SimCSE dengan In-batch Negative

Soal penuh

SimCSE (unsupervised) memproses 3 kalimat dalam satu batch. Tiap kalimat dilewatkan dua kali dengan dropout berbeda, menghasilkan anchor a_i dan positif p_i:

$$a_1 = (1, 0),\ a_2 = (0, 1),\ a_3 = (0.6, 0.8); \qquad p_1 = (0.8, 0.6),\ p_2 = (0, 1),\ p_3 = (0.6, 0.8)$$

Positif milik kalimat lain diperlakukan sebagai negatif (*in-batch negative*). $\tau = 0.2$.

1. Hitung matriks kemiripan $S_{ij} = \cos(a_i, p_j)$.
2. Bagi dengan τ dan hitung softmax per baris.

3. Hitung loss tiap anchor dan rata-ratanya.

4. Anchor mana yang loss-nya paling besar, dan kenapa?

Pembahasan soal penuh

1. Matriks kemiripan.
Semua vektor panjangnya 1, jadi cosine = dot product.

$$S = \begin{pmatrix} 0.8 & 0 & 0.6 \\ 0.6 & 1 & 0.8 \\ 0.96 & 0.8 & 1 \end{pmatrix}$$

Contoh: $S_{31} = a_3 \cdot p_1 = 0.6(0.8) + 0.8(0.6) = 0.96$. Diagonal ($S_{ii}$) adalah pasangan positif.

2. Bagi τ dan softmax.
$S/\tau = 5S$.

- Baris 1: (4, 0, 3), $e^x = (54.60,\ 1,\ 20.09)$, jumlah 75.68; softmax $(0.7214,\ 0.0132,\ 0.2654)$.
- Baris 2: (3, 5, 4), $e^x = (20.09,\ 148.41,\ 54.60)$, jumlah 223.10; softmax $(0.0900,\ 0.6652,\ 0.2447)$.
- Baris 3: (4.8, 4, 5), $e^x = (121.51,\ 54.60,\ 148.41)$, jumlah 324.52; softmax $(0.3744,\ 0.1682,\ 0.4573)$.

3. Loss.
Loss anchor $i = -\ln$ (softmax di diagonal):

$$L_1 = -\ln 0.7214 = 0.3266, \quad L_2 = -\ln 0.6652 = 0.4076, \quad L_3 = -\ln 0.4573 = 0.7824$$

Rata-rata: $(0.3266 + 0.4076 + 0.7824)/3 = \mathbf{0.5055}$.

4. Analisis.
Anchor 3 paling besar loss-nya. Walaupun positifnya sendiri sempurna ($S_{33} = 1$), negatif p_1 juga sangat mirip (0.96), sehingga peluang positif hanya 0.457. Training akan mendorong a_3 menjauh dari p_1. Pada SimCSE supervised, negatif seperti ini diambil sengaja dari kalimat contradiction (hard negative).

6.8 Soal Penuh: BGE-M3 Hybrid Retrieval (Exercise for Sentence Comparison)

Soal penuh

Query: "laptop for programming" (3 token). D_1 = "best notebook computer for software developers". D_2 = "laptop sleeve and carrying case for programming books".

1. Sebelum menghitung: dokumen mana yang lebih relevan, dan metode mana yang paling mungkin salah?

2. Dense: $u_q = (0.6, 0.5, 0.4)$, $v_{D1} = (0.58, 0.52, 0.42)$, $v_{D2} = (0.8, 0.05, 0.1)$. Hitung cosine.

3. Sparse: bobot token (hanya token yang ada di kedua sisi):

Token	w^Q	w^{D1}	w^{D2}
laptop	0.50	–	0.40
for	0.05	0.04	0.05
programming	0.45	–	0.30

("laptop" dan "notebook" adalah token berbeda.)

4. Multi-vector: cosine tiap token query (baris) dengan tiap token D_1 (kolom):

	best	notebook	computer	for	software	developers
laptop	0.35	0.93	0.80	0.10	0.20	0.15
for	0.08	0.05	0.06	0.99	0.04	0.07
programming	0.12	0.18	0.40	0.06	0.90	0.85

   Untuk D_2, maksimum per token query sudah diberikan: laptop 0.90, for 0.95, programming 0.85.

5. Gabungkan dengan $(w_1, w_2, w_3) = (1, 0.3, 1)$. Dokumen mana peringkat pertama?

6. Jika hanya satu metode dipakai, siapa pemenangnya? Pada w_2 berapa D_2 mulai menang? Kenapa paper BGE-M3 memberi bobot sparse kecil?

Pembahasan soal penuh

1. Prediksi.
D_1 lebih relevan secara makna (notebook ≈ laptop, software developers ≈ programming). D_2 berbagi tiga kata persis, tetapi isinya sarung laptop dan buku. Metode **sparse** paling mungkin salah karena hanya mencocokkan kata persis.

2. Dense.
$\|u_q\| = \sqrt{0.36 + 0.25 + 0.16} = \sqrt{0.77} = 0.8775$.

- D_1: $u \cdot v = 0.348 + 0.260 + 0.168 = 0.776$; $\|v_{D1}\| = \sqrt{0.7832} = 0.8850$; cos $= 0.776/(0.8775 \times 0.8850) =$ **0.999**.
- D_2: $u \cdot v = 0.48 + 0.025 + 0.04 = 0.545$; $\|v_{D2}\| = \sqrt{0.6525} = 0.8078$; cos $= 0.545/(0.8775 \times 0.8078) =$ **0.769**.

3. Sparse.
Hanya token yang ada di kedua sisi yang dijumlahkan.

- D_1: hanya "for": $0.05 \times 0.04 =$ **0.002**.
- D_2: $0.50(0.40) + 0.05(0.05) + 0.45(0.30) = 0.200 + 0.0025 + 0.135 =$ **0.3375**.

4. Multi-vector.
Ambil maksimum tiap baris, lalu jumlahkan.

- D_1: laptop→notebook 0.93, for→for 0.99, programming→software 0.90; total **2.82**.
- D_2: $0.90 + 0.95 + 0.85 =$ **2.70**.

5. Gabungan.

$$s(D_1) = 1(0.999) + 0.3(0.002) + 1(2.82) = \mathbf{3.820}$$
$$s(D_2) = 1(0.769) + 0.3(0.3375) + 1(2.70) = \mathbf{3.570}$$

D_1 peringkat pertama, sesuai prediksi.

6. Analisis.

- Dense saja: D_1 ($0.999 > 0.769$), benar. Sparse saja: D_2 ($0.3375 > 0.002$), **salah**, tertipu kecocokan kata. Multi-vector saja: D_1 ($2.82 > 2.70$), benar tetapi tipis.
- Titik seri: $0.999 + 0.002w_2 + 2.82 = 0.769 + 0.3375w_2 + 2.70 \Rightarrow 0.350 = 0.3355w_2 \Rightarrow w_2 \approx \mathbf{1.04}$. Untuk $w_2 > 1.04$, D_2 menang.
- Di awal training BGE-M3, matriks bobot sparse (W_{lex}) diinisialisasi acak sehingga skor sparse masih buruk. Karena itu bobotnya dibuat kecil ($w_2 = 0.3$) agar tidak merusak gabungan.

Cek dan jebakan

Skor multi-vector di sini mengikuti slide dan soal latihan: *jumlah* maksimum per token query. Paper BGE-M3 memakai *rata-rata* ($\frac{1}{N}\sum$); dengan versi itu multi $D_1 = 0.94$, $D_2 = 0.90$, gabungan 1.940 vs 1.770, titik seri $w_2 \approx 0.81$, dan D_1 tetap menang. Ikuti rumus yang diberikan soal.

Sparse unggul untuk kata yang sangat spesifik, misalnya kode error “0x80070057”, nama produk, atau istilah teknis: kata persis itu langka di data latih sehingga vektor dense-nya kurang informatif.

7 Rumus Cepat

Topik	Rumus
TF-IDF	$\mathrm{tf} \times \log(N/\mathrm{df})$
Mutual Information	$\log_2 \dfrac{A \cdot N}{(A+B)(A+C)}$
Naive Bayes	$P(c)\prod P(w \mid c), \quad P(w \mid c) = \dfrac{\text{count}+1}{\text{total}(c)+\lvert V\rvert}$
CNN teks	$c_i = \mathrm{ReLU}(\mathbf{w} \cdot \mathbf{x}_{i:i+h-1} + b)$; panjang $n-h+1$; pooling $\max_i c_i$
P, R, F1	$\dfrac{TP}{TP+FP}, \quad \dfrac{TP}{TP+FN}, \quad \dfrac{2PR}{P+R}$
Cosine	$\dfrac{u \cdot v}{\lVert u\rVert\,\lVert v\rVert}$
SVD / LSI	$\sigma = \sqrt{\lambda(A^\top A)}$; $u = Av/\sigma$; energi $\sigma_1^2/\sum\sigma_i^2$; fold-in $d = D^\top U S^{-1}$
HAL	$w(d) = W - d + 1$; $v(w) = [R \,\|\, L]$, $L = R^\top$; dokumen = rata-rata vektor kata
LDA	$P(w \mid d) = \sum_k \theta_{dk}\varphi_{kw}$; $P(z \mid w,d) = \theta\varphi/P(w \mid d)$; Gibbs $(n_{dk}+\alpha)(n_{kw}+\beta)$
GloVe	$J = \sum f(X)(w \cdot c + b + \tilde{b} - \ln X)^2$; $f = \min\big(1, (x/100)^{0.75}\big)$; $w \leftarrow w - \eta f \,\mathrm{err}\, c$
Word2Vec	$h = W[\text{kata}]$ (skip-gram) atau rata-rata konteks (CBOW); $p = \mathrm{softmax}(hW')$; param $2\lvert V\rvert N$
Softmax, CE	$e^{z_i}/\sum e^z$; dua kelas $1/(1+e^{-\Delta})$; loss $-\ln p_{\text{benar}}$
Positional enc.	$\sin(\omega_k t)$ (indeks $2k$), $\cos(\omega_k t)$ (indeks $2k+1$); $\omega_k = 1/10000^{2k/d}$
Attention	$\mathrm{softmax}(QK^\top/\sqrt{d_k})V$; mask: skor ke depan $= -\infty$
Multi-head	$d_k = d_{\text{model}}/h$; param attention $4d_{\text{model}}^2$; FFN $2d_{\text{model}}d_{ff}$
LayerNorm	$(x_i - \mu)/\sqrt{\sigma^2+\varepsilon}$, per baris, varians populasi
Kompleksitas	self-attention $O(n^2 d)$, RNN $O(nd^2)$
BERT	masukan = token + segment + posisi; mask 15%; head $d \cdot C + C$; layer $\approx 12d^2$
LoRA	$h = W_0 x + W_{up}W_{down}x$; param $r(d+k)$ vs dk
Adapter	$W_{up}\sigma(W_{down}x) + x$; param $2dm$
Selective	$\theta' = \theta - \eta\, m\, \partial L/\partial\theta$
Pooling	`[CLS]` atau rata-rata hidden state
SBERT head	$\mathrm{softmax}([u; v; \lvert u-v\rvert]\, W_t)$; param $3n \cdot k + k$
Kontrastif	$-\ln \dfrac{e^{s^+/\tau}}{\sum_i e^{s_i/\tau}}, \quad s = \cos$
BGE-M3	$w_1 \cos + w_2 \sum w^Q w^D + w_3 \sum \max \cos$

8 Sengaja Tidak Dimasukkan

- Backpropagation dan turunan penuh (tidak diminta di slide, terlalu panjang untuk kertas).
- Eigen-dekomposisi matriks 3×3 atau lebih (tidak praktis tanpa laptop).
- Gibbs sampling berulang sampai konvergen (hanya satu langkah yang dimasukkan).
- Topik BEAM (di luar cakupan UTS NLP).
- Perhitungan RNN/LSTM langkah demi langkah (slide hanya membahasnya secara konsep).