UNIVERSITAS INDONESIA

PEMERINGKATAN TEKS BAHASA INDONESIA DENGAN BERT

SKRIPSI

CARLES OCTAVIANUS
2006568613

FAKULTAS FAKULTAS MATEMATIKA DAN ILMU PENGATAHUAN ALAM
PROGRAM STUDI MATEMATIKA
DEPOK
DESEMBER 2023

UNIVERSITAS INDONESIA

PEMERINGKATAN TEKS BAHASA INDONESIA DENGAN BERT

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar

Sarjana Sains

CARLES OCTAVIANUS

2006568613

FAKULTAS FAKULTAS MATEMATIKA DAN ILMU PENGATAHUAN ALAM

PROGRAM STUDI MATEMATIKA

DEPOK

DESEMBER 2023

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya saya sendiri,
dan semua sumber baik yang dikutip maupun dirujuk
telah saya nyatakan dengan benar.

Nama	:	**Carles Octavianus**
NPM	:	**2006568613**
Tanda Tangan	:	
Tanggal	:	**2 Desember 2023**

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh	:	
Nama	:	Carles Octavianus
NPM	:	2006568613
Program Studi	:	Matematika
Judul Skripsi	:	Pemeringkatan Teks Bahasa Indonesia Dengan BERT

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana pada Program Studi Matematika, Fakultas Fakultas Matematika dan Ilmu Pengatahuan Alam, Universitas Indonesia.

DEWAN PENGUJI

Pembimbing 1	:	Sarini Abdullah S.Si., M.Stats., Ph.D.	( )
Penguji 1	:	Penguji Pertama Anda	( )
Penguji 2	:	Penguji Kedua Anda	( )

Ditetapkan di	:	Depok
Tanggal	:	2 Desember 2023

KATA PENGANTAR

Template ini disediakan untuk orang-orang yang berencana menggunakan LaTeX untuk membuat dokumen tugas akhir.

@todo

Silakan ganti pesan ini dengan pendahuluan kata pengantar Anda.

Ucapan Terima Kasih:

1. Pembimbing.
2. Dosen.
3. Instansi.
4. Orang tua.
5. Sahabat.
6. Teman.

Penulis menyadari bahwa laporan Skripsi ini masih jauh dari sempurna. Oleh karena itu, apabila terdapat kesalahan atau kekurangan dalam laporan ini, Penulis memohon agar kritik dan saran bisa disampaikan langsung melalui *e-mail* `emailanda@mail.id`.

Terkait template ini, gambar lisensi di atas diambil dari `http://creativecommons.org/licenses/by-nc-sa/1.0/deed.en_CA`. Jika ingin mengentahui lebih lengkap mengenai *Creative Common License 1.0 Generic*, silahkan buka `http://creativecommons.org/licenses/by-nc-sa/1.0/legalcode`. Seluruh dokumen yang dibuat dengan menggunakan template ini sepenuhnya menjadi hak milik pembuat dokumen dan bebas didistribusikan sesuai dengan keperluan masing-masing. Lisensi hanya berlaku jika ada orang yang membuat template baru dengan menggunakan template ini sebagai dasarnya.

Penyusun template ingin berterima kasih kepada Andreas Febrian, Lia Sadita, Fahrurrozi Rahman, Andre Tampubolon, dan Erik Dominikus atas kontribusinya dalam template yang menjadi pendahulu template ini. Penyusun template juga ingin mengucapkan terima kasih kepada Azhar Kurnia atas kontribusinya dalam template yang menjadi pendahulu template ini.

Semoga template ini dapat membantu orang-orang yang ingin mencoba menggu-

nakan LaTeX. Semoga template ini juga tidak berhenti disini dengan ada kontribusi dari para penggunanya. Jika Anda memiliki perubahan yang dirasa penting untuk disertakan dalam template, silakan lakukan *fork* repositori Git template ini di `https://gitlab.com/ichlaffterlalu/latex-skripsi-ui-2017`, lalu lakukan *merge request* perubahan Anda terhadap *branch* `master`. Kami berharap agar *template* ini dapat terus diperbarui mengikuti perubahan ketentuan dari pihak Rektorat Universitas Indonesia, dan hal itu tidak mungkin terjadi tanpa kontribusi dari teman-teman sekalian.

Depok, 2 Desember 2023

Carles Octavianus

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

Nama	:	Carles Octavianus
NPM	:	2006568613
Program Studi	:	Matematika
Jenis Karya	:	Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif (*Non-exclusive Royalty Free Right*)** atas karya ilmiah saya yang berjudul:

Pemeringkatan Teks Bahasa Indonesia Dengan BERT

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan memublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok
Pada tanggal : 2 Desember 2023
Yang menyatakan

(Carles Octavianus)

ABSTRAK

Nama	:	Carles Octavianus
Program Studi	:	Matematika
Judul	:	Pemeringkatan Teks Bahasa Indonesia Dengan BERT
Pembimbing	:	Sarini Abdullah S.Si., M.Stats., Ph.D.

Isi abstrak.

Kata kunci:
Keyword satu, kata kunci dua

ABSTRACT

Name	:	Carles Octavianus
Study Program	:	Mathematics
Title	:	Text Ranking in Indonesian Using BERT
Counselor	:	Sarini Abdullah S.Si., M.Stats., Ph.D.

Abstract content.

Key words:
Keyword one, keyword two

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR KODE PROGRAM

DAFTAR LAMPIRAN

BAB 1
PENDAHULUAN

@todo
wew

BAB 2
LANDASAN TEORI

@todo
kasih contoh ndcg

2.1 Masalah Pemeringkatan Teks

2.1.1 Pemeringkatan Teks

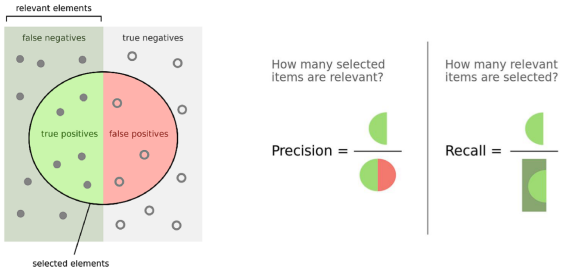
Permasalahan pemeringkatan teks adalah Permasalahan untuk menentukan urutan dokumen yang paling relevan dengan kueri q yang diberikan. Dalam bahasa yang lebih formal, diberikan kueri q dan himpunan dokumen terbatas $\mathcal{D} = \{d_1, d_2, ..., d_n\}$, keluaran yang diinginkan dari permasalahan ini adalah barisan dokumen $D_k = (d_{i_1}, d_{i_2}, ..., d_{i_k})$ yang merupakan k dokumen yang paling relevan dengan kueri q. Selain itu, biasanya nilai k akan lebih kecil dari banyaknya dokumen yang ada, sehingga permasalahan pemeringkatan sering juga disebut sebagai *top-k retrieval*. Untuk mengukur performa suatu model pemeringkatan, biasanya digunakan metrik evaluasi seperti presisi, *recall*, *reciprocal rank*, dan *normalized discounted cumulative gain* (nDCG) yang akan dijelaskan pada Subbab 2.1.3.

2.1.2 Bentuk Umum Dataset untuk Evaluasi Pemeringkatan Teks

2.1.2.1 *Judgements*

2.1.3 Metrik Evaluasi dalam Pemeringkatan Teks

2.1.3.1 *Recall* dan Presisi



Gambar 2.1: Ilustrasi *recall* dan presisi. Nilai *recall* dihitung sebagai rasio dokumen relevan yang diambil oleh model terhadap seluruh dokumen yang relevan dengan kueri q. Sedangkan nilai presisi dihitung sebagai rasio dokumen relevan yang diambil oleh model terhadap seluruh dokumen yang diambil oleh model.

Presisi dan *recall* adalah metrik yang paling sederhana untuk mengukur kemampuan dari suatu model pemeringkatan teks. Untuk suatu kueri q, kumpulan dokumen $\mathcal{D} = \{d_1, d_2, ..., d_n\}$, dan barisan k dokumen yang diambil oleh model, $D_k = (d_{i_1}, d_{i_2}, ..., d_{i_k})$, *recall* dan presisi dapat dihitung dengan Persamaan 2.1 dan Persamaan 2.4.

$$\mathcal{D} = \{d_1, d_2, \ldots, d_n\} \tag{2.1}$$

$$D_k = (d_{i_1}, d_{i_2}, \ldots, d_{i_k}) \tag{2.2}$$

$$\text{recall}(q, D_k)@\text{k} = \frac{\sum_{d \in D_k} \text{rel}(q,d)}{\sum_{d \in \mathcal{D}} \text{rel}(q,d)} \in [0,1] \tag{2.3}$$

$$\text{precision}(q, D_k)@\text{k} = \frac{\sum_{d \in D_k} \text{rel}(q,d)}{|D_k|} \in [0,1] \tag{2.4}$$

$$\text{dengan rel}(q,d) = \begin{cases} 1 & \text{jika } r > 1 \\ 0 & \text{jika } r = 0 \end{cases} \tag{2.5}$$

Sebagai Contoh, Jika terdapat 10 dokumen yang relevan dengan kueri q, dan model mengembalikan $k = 100$ dokumen, namun hanya terdapat 5 dokumen yang relevan pada D_k maka *recall* dan presisi dari model tersebut adalah 0.5 ($\frac{5}{10}$) dan 0.05 ($\frac{5}{100}$) masing-masing.

Baik *recall* maupun presisi memiliki rentang nilai dari 0 hingga 1, dimana nilai 1 menunjukkan performa model yang terbaik. perhitungan *recall* biasanya dilakukan untuk k yang cukup besar ($k = 100, 1000$), sedangkan perhitungan presisi dilakukan untuk k yang kecil ($k = 1, 3, 5$) (Hofstätter, Althammer, Sertkan, & Hanbury, 2021).

2.1.3.2 *Reciprocal Rank*

Gambar 2.2: Ilustrasi *reciprocal rank*.

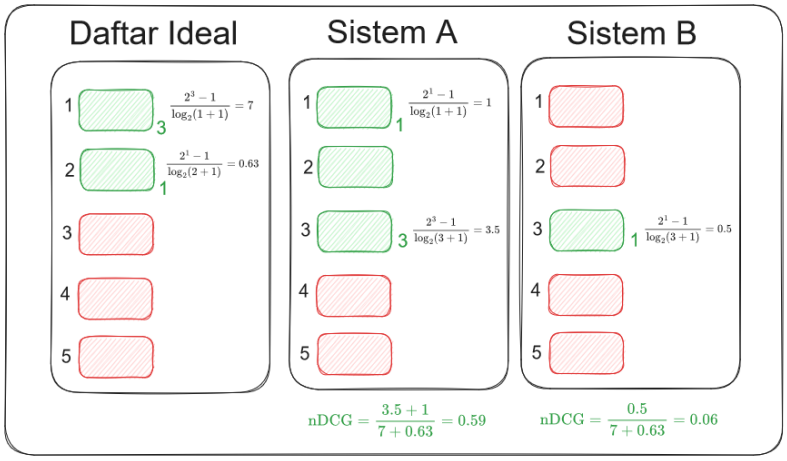
Metrik lainnya yang sering digunakan untuk mengukur performa model pemeringkatan adalah *reciprocal rank* (RR). Metrik RR menitikberatkan pada peringkat pertama dari dokumen yang relevan dengan kueri q. Semakin tinggi peringkat dari dokumen yang relevan dengan kueri q. Persamaan 2.6 hingga Persamaan 2.7 menunjukkan cara menghitung RR dari suatu kueri q dan barisan k dokumen yang diambil oleh model (Hofstätter et al., 2021; Lin, Nogueira, & Yates, 2020).

$$\text{RR}(q, D_k)@\text{k} = \begin{cases} \frac{1}{\text{FirstRank}(q, D_k)} & \text{jika } \exists d \in D_k \text{ dengan } \text{rel}(q,d) = 1 \\ 0 & \text{jika } \forall d \in D_k,\ \text{rel}(q,d) = 0 \end{cases} \in [0,1] \quad (2.6)$$

$$\text{FirstRank}(q, D_k) = \text{posisi dokumen relevan pertama } d \in D_k \text{ dengan } \text{rel}(q,d) = 1 \quad (2.7)$$

Gambar 2.2 mengilustrasikan metrik RR. Pada gambar tersebut, nilai RR dari sistem A adalah 1 $(\frac{1}{1})$ karena posisi dari dokumen yang relevan pertama adalah 1. Sedangkan nilai RR dari sistem B dan sistem C masing-masing adalah 0.33 $(\frac{1}{3})$ dan 0.5 $(\frac{1}{2})$ karena posisi dari dokumen yang relevan pertama adalah 3 dan 2. Selain itu, jika tidak terdapat dokumen yang relevan dengan kueri q pada D_k, maka nilai RR dari sistem tersebut adalah 0.

2.1.3.3 *Normalized Discounted Cumulative Gain* (nDCG)



Gambar 2.3: *Creative Common License 1.0 Generic*.

Normalized Discounted Cumulative Gain (nDCG) adalah metrik yang umumnya digunakan untuk mengukur kualitas dari pencarian situs web. Tidak seperti metrik yang telah disebutkan sebelumnya, nDCG dirancang untuk suatu *judgements r* yang tak biner. Fungsi rel(q,d) pada Persamaan 2.5 berubah menjadi rel(q,d) $= r$ ketika menghitung metrik nDCG. Persamaan 2.8 hingga Persamaan 2.10 menunjukkan cara menghitung nDCG dari suatu kueri q dan barisan k dokumen yang diambil oleh model.

$$\text{nDCG}(q,D_k)@\text{k} = \frac{\text{DCG}(q,D_k)@\text{k}}{\text{DCG}(q,D_k^{\text{ideal}})@\text{k}} \in [0,1] \tag{2.8}$$

$$\text{DCG}(q,D_k)@\text{k} = \sum_{d \in D_k} \frac{2^{\text{rel}(q,d)}-1}{\log_2(\text{rank}(d,D_k)+1)} \tag{2.9}$$

$$\text{rank}(d,D_k) = \text{Posisi } d \text{ dalam } D_k \tag{2.10}$$

$$\text{rel}(q,d) = r \tag{2.11}$$

Perhitungan *discounted cumulative gain* (DCG) pada Persamaan 2.9 dapat dijelaskan menjadi dua faktor, yaitu:

1. faktor $2^{\text{rel}(q,d)}-1$ menunjukkan bahwa dokumen yang lebih relevan akan memiliki nilai yang lebih tinggi dari dokumen yang kurang relevan.

2. faktor $\frac{1}{\log_2(\text{rank}(d,D_k)+1)}$ menunjukkan bahwa dokumen yang relevan yang muncul pada peringkat yang lebih tinggi akan memiliki nilai yang lebih tinggi dari dokumen

dengan relevansi yang sama, tetapi muncul pada peringkat yang lebih rendah.

nilai dari nDCG pada Persamaan 2.8 adalah nilai DCG pada barisan dokumen D_k yang dinormalisasi oleh nilai DCG pada barisan dokumen ideal D_k^{ideal}. Barisan dokumen ideal D_k^{ideal} adalah barisan dokumen yang diurutkan berdasarkan relevansinya dengan kueri q.

Selain itu, jika pada *datasets* memiliki *judgements* biner, faktor $2^{\text{rel}(q,d)} - 1$ pada Persamaan 2.9 dapat diubah menjadi $\text{rel}(q,d)$. Persamaan 2.9 akan menjadi Persamaan 2.12.

$$\text{DCG}(q, D_k)@\text{k} = \sum_{d \in D_k} \frac{\text{rel}(q,d)}{\log_2(\text{rank}(d, D_k) + 1)}. \tag{2.12}$$

2.2 Pemeringkatan Teks dengan Statistik

Untuk mengambil k dokumen dari kumpulan $\mathcal{D}$ diperlukan suatu fungsi skor $s(q,d,\mathcal{D})$ yang mengukur relevansi antara kueri q dan dokumen d. dengan mencari skor antar q terhadap semua dokumen pada $\mathcal{D}$, Barisan dokumen $D_k = (d_{i_1}, d_{i_2}, \ldots, d_{i_k})$ dapat dipilih sehingga $\text{score}(q,d_{i_1}) \geq \text{score}(q,d_{i_2}) \geq \cdots \geq \text{score}(q,d_{i_k})$ adalah k dokumen dengan skor tertinggi.

Pada bagian ini, akan dijelaskan beberapa fungsi skor stastistik sederhana yang sering digunakan dalam pemeringkatan teks. Subbab 2.2.1 menjelaskan fungsi skor statistik yang berdasarkan pada frekuensi kemunculan kata pada dokumen dan kueri. Selanjutnya, Subbab 2.2.2 membahas fungsi skor statistik yang menjadi standar *de facto* dalam pemeringkatan teks.

2.2.1 *Term Frequency - Inverse Document Frequency* (TF-IDF)

	doc_1	doc_2	doc_3	doc_4
A	10	10	10	10
B	10	10	10	0
C	10	10	0	0
D	0	0	0	1

⇓

TF

	doc_1	doc_2	doc_3	doc_4
A	0.33	0.33	0.50	0.91
B	0.33	0.33	0.50	0.00
C	0.33	0.33	0.00	0.00
D	0.00	0.00	0.00	0.09

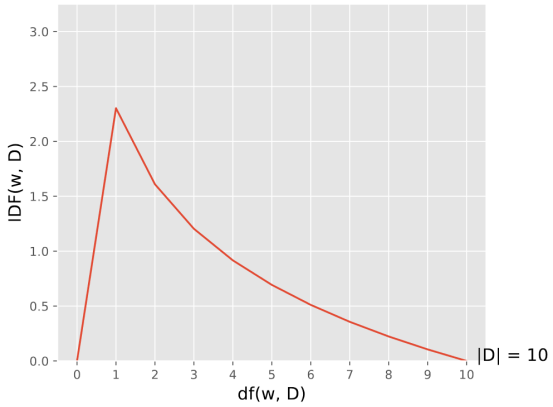
⇒

	IDF
A	0.00
B	0.29
C	0.69
D	1.39

TF-IDF

	doc_1	doc_2	doc_3	doc_4
A	0.00	0.00	0.00	0.00
B	0.10	0.10	0.14	0.00
C	0.23	0.23	0.00	0.00
D	0.00	0.00	0.00	0.13

Gambar 2.4: *Creative Common License 1.0 Generic*.



Gambar 2.5: *Creative Common License 1.0 Generic.*

fungsi skor TF-IDF adalah fungsi skor statistik yang mengukur relevansi antara kueri q dan dokumen d dengan menghitung frekuensi kemunculan kata pada dokumen dan kueri. Untuk suatu kueri q, misalkan $T_q = \{t_1, t_2, \ldots, t_{L_1}\}$ adalah himpunan kata yang terdapat pada kueri q. Selain itu, misalkan $T_d = \{t_1, t_2, \ldots, t_n\}$ adalah himpunan kata yang terdapat pada dokumen d. nilai skor antara q dan d diberikan oleh persamaan Persamaan 2.13 sampai Persamaan 2.21.

$$\mathcal{D} = \{d_1, d_2, \ldots, d_n\} \tag{2.13}$$

$$T_q = \{t_1, t_2, \ldots, t_{L_1}\} \tag{2.14}$$

$$T_d = \{t_1, t_2, \ldots, t_{L_2}\} \tag{2.15}$$

$$\text{tf}(t,d) = \frac{\text{Count}(t,d)}{|d|} \tag{2.16}$$

$$\text{Count}(t,d) = \text{jumlah kemunculan } t \text{ dalam } d \tag{2.17}$$

$$\text{df}(t,\mathcal{D}) = \text{jumlah dokumen pada } \mathcal{D} \text{ yang mengandung } t \tag{2.18}$$

$$\text{idf}(t,\mathcal{D}) = \begin{cases} \log_2\left(\frac{|\mathcal{D}|}{\text{df}(t,\mathcal{D})}\right) & \text{jika df}(t,\mathcal{D}) > 0 \\ 0 & \text{jika df}(t,\mathcal{D}) = 0 \end{cases} \tag{2.19}$$

$$\text{tf-idf}(t,d,\mathcal{D}) = \text{tf}(t,d) \times \text{idf}(t,\mathcal{D}) \tag{2.20}$$

$$\text{score}(q,d,\mathcal{D}) = \sum_{t \in T_q \cap T_d} \text{tf-idf}(t,d,\mathcal{D}) \tag{2.21}$$

skor untuk pasangan terurut (q,d) dihitung dengan menjumlahkan skor TF-IDF dari setiap kata yang terdapat pada kueri q dan dokumen d ($T_q \cap T_d$). skor TF-IDF dari suatu

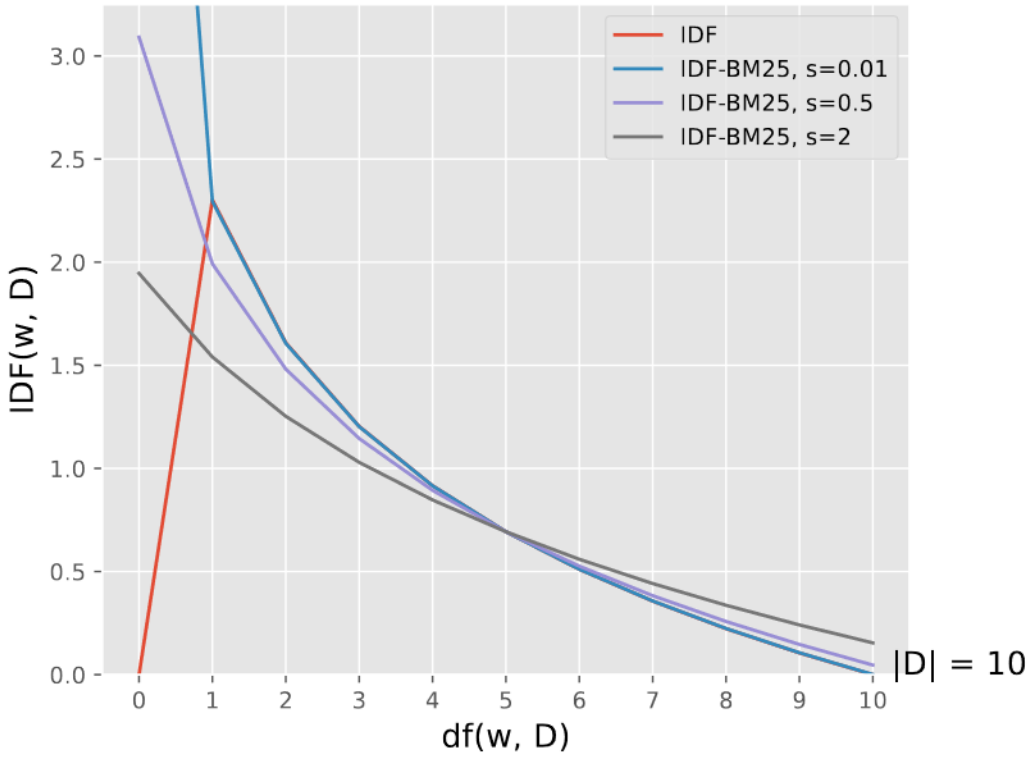
kata t adalah perkalian antar *term frequency* ($\text{tf}(q,d)$) dan *inverse document frequency* ($\text{idf}(t,\mathcal{D})$). fungsi skor pada Persamaan 2.21 dapat dijelaskan sebagai dua bagian utama, yaitu:

1. faktor $\text{tf}(t,d)$ menunjukkan bahwa nilai TF-IDF meningkat seiring dengan bertambahnya frekuensi kemunculan kata t pada dokumen d.

2. Faktor $\text{idf}(t,\mathcal{D})$ menunjukkan bahwa nilai TF-IDF meningkat seiring dengan *rarity* dari kata t pada himpunan dokumen $\mathcal{D}$. Akibatnya, kata yang jarang muncul pada himpunan dokumen $\mathcal{D}$ dan muncul pada suatu dokumen tertentu akan menghasilkan skor yang tinggi. Sementara itu, kata-kata yang sering muncul pada koleksi dokumen $\mathcal{D}$ memiliki nilai *downgraded*.

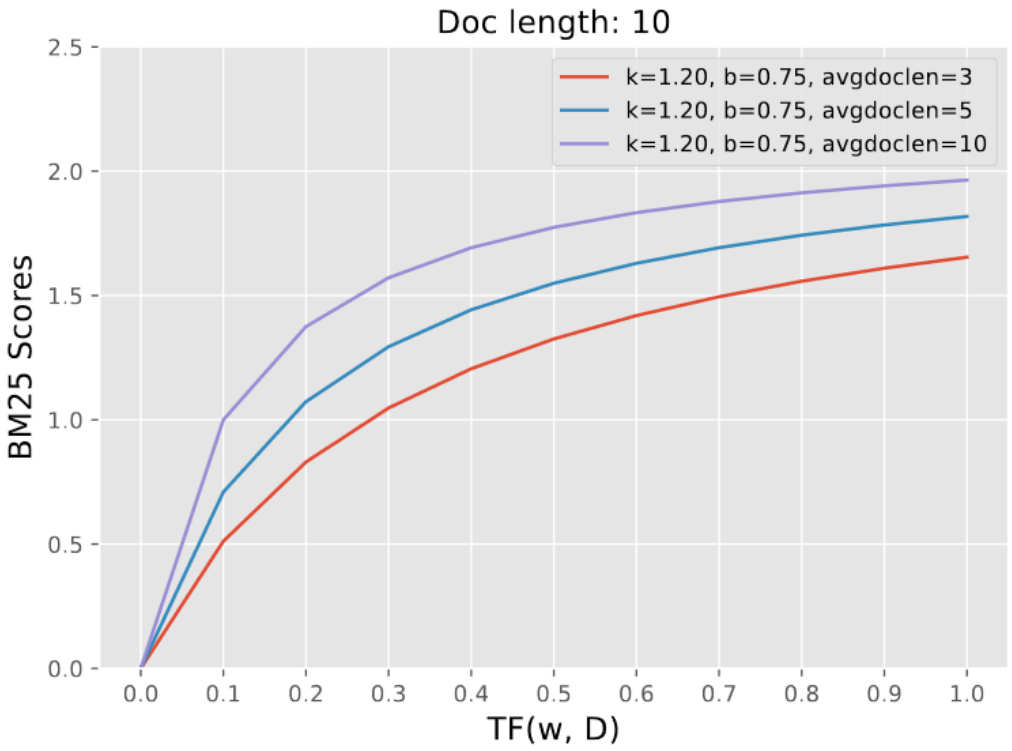
Kata-kata seperti preposisi atau kata ganti akan menghasilkan skor TF-IDF yang sangat rendah. Ini menyiratkan bahwa kata-kata tersebut memiliki sedikit relevansi dalam dokumen dan bisa diabaikan. Di sisi lain, kata-kata yang muncul secara berlebihan dalam satu dokumen tetapi jarang muncul dalam dokumen lainnya akan menghasilkan nilai $\text{tf}(t,d)$ dan $\log\left(\frac{\mathcal{D}}{\text{df}(t,\mathcal{D})}\right)$ yang relatif besar. Dampaknya adalah skor TF-IDF yang dihasilkan juga menjadi signifikan.

2.2.2 *Best Match 25* (BM25)



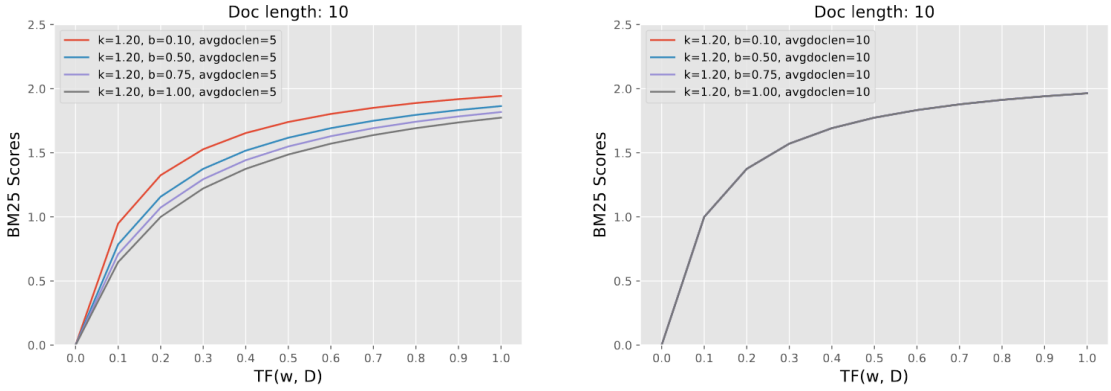
Gambar 2.6: *Creative Common License 1.0 Generic*.

;



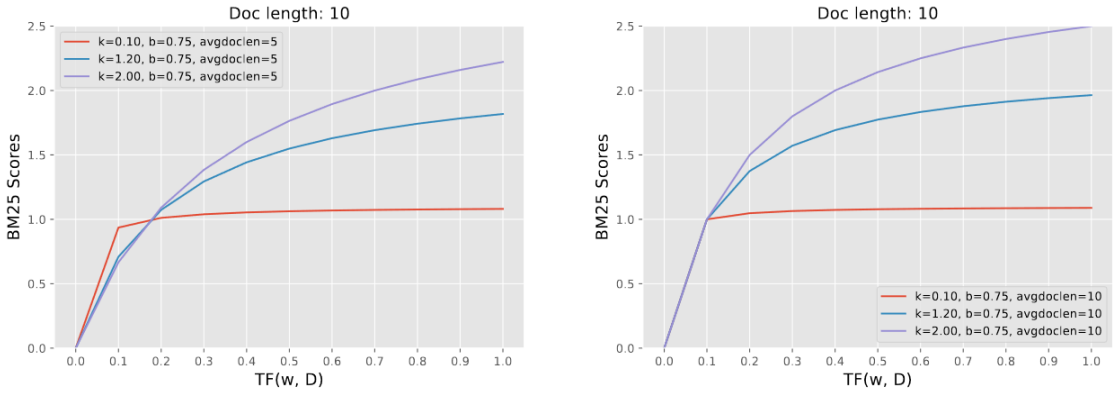
Gambar 2.7: *Creative Common License 1.0 Generic*.

;



Gambar 2.8: *Creative Common License 1.0 Generic*.

;



Gambar 2.9: *Creative Common License 1.0 Generic*.

;

BM25 (*Best Match attempt* 25) merupakan pengembangan dari fungsi skor TF-IDF dengan perbedaan utamanya adalah fungsi nilai menggunakan $\text{score}_{\text{BM25}}(q,d)$ (Persamaan 2.24) daripada $\text{tf}(q,d)$ (Persamaan 2.20). pada fungsi $\text{score}_{\text{BM25}(q,d)}$ terdapat 2 parameter yang dapat diatur, yaitu b, dan k_1. Setiap parameter mempunyai efek yang berbeda terhadap skor yang dihasilkan. Selain itu, Perbedaan *minor* lainnya ada di fungsi idf. Pada fungsi idf pada BM25, terdapat *smoothing* yang bertujuan untuk menghindari nilai idf yang bernilai 0 ketika kata t tidak muncul pada himpunan dokumen $\mathcal{D}$. Persamaan 2.22 hingga Persamaan 2.26 menunjukkan perhitungan skor relevansi dengan BM25.

$$\text{idf}_{\text{BM25}}(t,\mathcal{D}) = \log\left(1 + \frac{|\mathcal{D}| - \text{df}(t,\mathcal{D}) + 0.5}{\text{df}(t,\mathcal{D}) + 0.5}\right) \tag{2.22}$$

$$\text{score}_{\text{BM25}}(t,d) = \frac{\text{tf}(t,d) \times (k_1+1)}{\text{tf}(t,d) + k_1 \times (1 - b + b \times \frac{|d|}{\text{avgdl}})} \tag{2.23}$$

$$\text{BM25}(t,d,\mathcal{D}) = \text{idf}_{\text{BM25}}(t,\mathcal{D}) \times \text{score}_{\text{BM25}}(q,d,\mathcal{D}) \tag{2.24}$$

$$\text{avgdl} = \text{rata-rata panjang dokumen pada koleksi } \mathcal{D} \tag{2.25}$$

$$\text{score}(q,d,\mathcal{D}) = \sum_{t \in T_q \cap T_d} \text{BM25}(t,d,\mathcal{D}) \tag{2.26}$$

$$\tag{2.27}$$

Gambar 2.6 Menunjukkan Perbedaan antara idf_{BM25} dan idf. Perbedaan utamanya terjadi ketika $\text{df}(t,\mathcal{D}) = 0$, nilai dari idf_{BM25} tak nol dan mengikuti pola yang diharapkan, yaitu semakin jarang kata t muncul pada himpunan dokumen $\mathcal{D}$, semakin tinggi nilai

idf_{BM25} yang dihasilkan. Ketika $\text{df}(t,\mathcal{D}) > 0$, nilai dari idf_{BM25} dan idf hampir serupa.

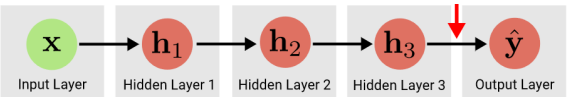
Perbedaan antar $\text{score}_{\text{BM25}}(t,d)$ dengan $\text{tf}(t,d)$ jauh lebih kompleks karena adanya parameter tambahan k_1, dan b serta faktor baru avgdl. Berikut adalah penjelasan dari masing-masing parameter dan efeknya terhadap skor yang dihasilkan:

1. faktor $\frac{|\text{d}|}{\text{avgdl}}$ pada $\frac{\text{tf}(t,d)\times(k_1+1)}{\text{tf}(t,d)+k_1\times\left(1-b+b\times\frac{|d|}{\text{avgdl}}\right)}$ men-*penalize* skor pada dokumen yang panjangnya lebih besar dari rata-rata panjang dokumen pada himpunan dokumen $\mathcal{D}$. Gambar 2.7 Menjukkan efek dari perbedaan nilai avgdl terhadap skor yang dihasilkan. Semakin besar rasio $\frac{|d|}{\text{avgdl}}$ semakin kecil skor yang dihasilkan,.

2. nilai b menentukan seberapa besar efek dari faktor $\frac{|d|}{\text{avgdl}}$ terhadap skor yang dihasilkan. Gambar 2.8 Menunjukkjan efek dari perbedaan nilai b terhadap skor yang dihasilkan. Untuk $\frac{|d|}{\text{avgdl}} = 1$, Faktor b tidak memiliki pengaruh terhadap skor.

3. nilai k_1 men-*penalize* kemunculan kata t pada dokumen d yang sering. Gambar 2.9 Menunjukkan efek dari perbedaan nilai k_1 terhadap skor yang dihasilkan. Perhatikan bahwa ketika kemunculan kata t pada dokumen d semakin sering, skor penigkatan skor yang dihasilkan semakin kecil.

2.3 *Deep Learning*



Gambar 2.10: *Creative Common License 1.0 Generic*.

Arsitektur *Deep learning* merujuk pada model *machine learning* yang tersusun dari fungsi-fungsi terturunkan (yang biasa disebut sebagai *layer*), dimana komposisi antar fungsi-fungsi tersebut dapat digambarkan sebagai *directed acyclic graph* (DAG) yang memetakan suatu *input* ke suatu *output*. Biasanya, setiap fungsi tersebut dalam Arsitektur *Deep learning* memiliki parameter yang ingin diestimasi atau dicari dengan data.

Gambar 2.10 menunjukkan arsitektur deep learning yang sederhana, yaitu *feed-forward neural network* (FFN). Pada Gambar 2.10, *input* $\mathbf{x}$ akan dipetakan ke *output* $\hat{y}$ melalui serangkaian fungsi f_1, f_2, f_3 yang disebut sebagai *layer*. Setiap *layer* f_i memiliki parameter θ_i yang akan diestimasi dengan data. Selain itu, *Output* dari *layer* f_i akan

menjadi *input* dari *layer* f_{i+1}. *Output* dari *layer* f_3 adalah *output* dari model. Model pada Gambar 2.10 dapat ditulis sebagai Persamaan 2.28.

$$\hat{y} = f_{\text{model}}(\mathbf{x};\boldsymbol{\theta}) = f_3(f_2(f_1(\mathbf{x};\boldsymbol{\theta}_1);\boldsymbol{\theta}_2);\boldsymbol{\theta}_3) \tag{2.28}$$

$$\text{dengan } \boldsymbol{\theta} = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3\} \tag{2.29}$$

2.3.1 *Feed-Forward Neural Network* (FFN)

Feed-Forward Neural Network (FFN) adalah salah satu arsitektur *deep learning* yang paling sederhana. Pada *feed-forward neural network*, setiap fungsi f_i adalah fungsi linear yang diikuti oleh fungsi aktivasi non-linear ϕ yang diterapkan *element-wise* pada setiap *output*-nya. *hyperparameter* lainnya selain fungsi aktivasi adalah kedalamaan model L, dan dimensi *output* dari setiap *layer* $d_1, d_2, \ldots, d_L$.

Untuk permasalahan regresi dengan *input* $\mathbf{x} \in \mathbb{R}^{d_0}$ dan *output* $\mathbf{y} \in \mathbb{R}^{d_L}$, Persamaan 2.30 hingga Persamaan 2.36 menunjukkan arsitektur *feed-forward neural network* untuk permasalahan regresi dengan kedalaman L.

$$f_l(\mathbf{x};\mathbf{W}_l, b_l) = \phi(\mathbf{x}\mathbf{W}_l + \mathbf{b}_l) \in \mathbb{R}^{d_l}, \quad l = 1, 2, \ldots, L-1 \tag{2.30}$$

$$f_L(\mathbf{x}) = \mathbf{x}\mathbf{W}_L + \mathbf{b}_L \in \mathbb{R}^{d_L} \tag{2.31}$$

$$f_{\text{model}}(\mathbf{x};\boldsymbol{\theta}) = f_L(f_{L-1}(\ldots f_1(\mathbf{x}))\ldots) \tag{2.32}$$

$$\phi(\mathbf{x}) = \text{fungsi aktivitasi non-linear} \tag{2.33}$$

$$\boldsymbol{\theta} = \{\mathbf{W}_1, \mathbf{b}_1, \mathbf{W}_2, \mathbf{b}_2, \ldots, \mathbf{W}_L, \mathbf{b}_L\} \tag{2.34}$$

$$\mathbf{W}_l = \text{matriks bobot} \in \mathbb{R}^{d_{l-1} \times d_l} \tag{2.35}$$

$$\mathbf{b}_l = \text{vektor bias} \in \mathbb{R}^{d_l} \tag{2.36}$$

Untuk Permasalahan Klasifikasi Biner dengan *input* $\mathbf{x} \in \mathbb{R}^{d_0}$ dan *output* $\mathbf{y} \in \{0, 1\}$, Persamaan 2.37 hingga Persamaan 2.41 menunjukkan arsitektur *feed-forward neural network* untuk permasalahan klasifikasi.

$$f_{\text{model}}(\mathbf{x};\boldsymbol{\theta}) = f_L(f_{L-1}(\dots f_1(\mathbf{x}))\dots), \tag{2.37}$$

$$f_L(\mathbf{x}) = \sigma(\mathbf{x}\mathbf{W}_L + \mathbf{b}_L \in \mathbb{R}), \tag{2.38}$$

$$\sigma(x) = \frac{1}{1+e^x} \in (0,1), \tag{2.39}$$

$$\text{decision}(\mathbf{x};\boldsymbol{\theta}) = \begin{cases} 1 & \text{jika } f(\mathbf{x};\boldsymbol{\theta}) \geq \text{threshold} \\ 0 & \text{jika } f(\mathbf{x};\boldsymbol{\theta}) < \text{threshold} \end{cases}, \tag{2.40}$$

$$\text{threshold} \in [0,1]. \tag{2.41}$$

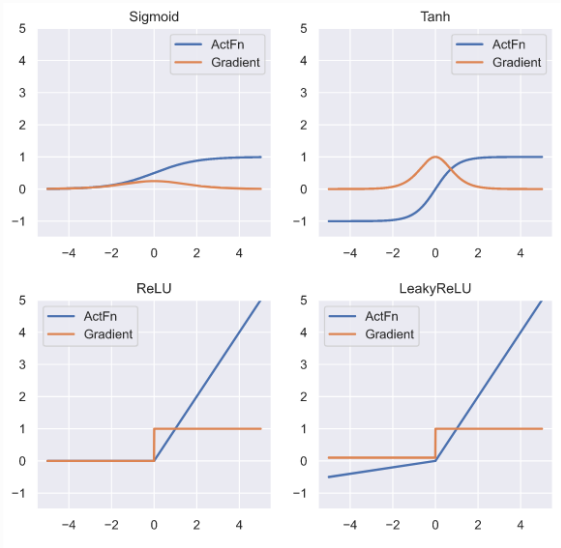
Perbedaan utama antara *feed-forward neural network* untuk permasalahan regresi dan klasifikasi adalah fungsi aktivasi pada *output layer*. Pada permasalahan regresi, fungsi aktivasi pada *output layer* adalah fungsi identitas, sedangkan pada permasalahan klasifikasi, fungsi aktivasi pada *output layer* adalah fungsi *sigmoid*. Tujuan digunakan fungsi *sigmoid* pada permasalahan klasifikasi adalah untuk memastikan bahwa *output* dari model berada pada rentang $[0,1]$, dimana nilai tersebut dapat diinterpretasikan sebagai probabilitas $\mathbf{x}$ termasuk pada kelas positif. Selain itu, *threshold* pada Persamaan 2.41 digunakan untuk menentukan apakah $\mathbf{x}$ termasuk pada kelas positif atau negatif.

2.3.2 Fungsi Aktivasi

Fungsi Aktivasi pada setiap *layer* f_i pada *feed-forward neural network* digunakan untuk menambahkan non-linearitas pada model. Sebab, tanpa adanya fungsi aktivasi non-linear, model *feed-forward neural network* akan menjadi model linear. Selain itu, fungsi aktivasi juga biasanya adalah fungsi yang terturunkan, meskipun tidak perlu terturunkan disetiap titik. Tabel 2.1 menunjukkan beberapa fungsi aktivasi yang sering digunakan pada *feed-forward neural network*. Gambar 2.11 menunjukkan grafik dari fungsi aktivasi pada Tabel 2.1 dan turunannya.

Tabel 2.1: Beberapa fungsi aktivasi yang sering digunakan pada *feed-forward neural network*.

Fungsi Aktivasi	Persamaan
Sigmoid	$\sigma(x) = \frac{1}{1+e^{-x}}$
Tanh	$\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$
ReLU	$\text{ReLU}(x) = \max(0,x)$
Leaky ReLU	$\text{LeakyReLU}(x) = \max(\alpha x, x), \alpha \in [0,1]$



Gambar 2.11: *Creative Common License 1.0 Generic*.

2.3.3 Fungsi *Loss*

Misalkan $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \ldots, (\mathbf{x}_n, \mathbf{y}_n)\}$ adalah *dataset* yang terdiri dari n pasangan *input* dan *output*. Parameter $\boldsymbol{\theta}$ pada f_{model} diestimasi dengan melakukan *fitting* pada *dataset* $\mathcal{D}$. Untuk melakukan *fitting* pada *dataset* $\mathcal{D}$, diperlukan suatu fungsi *loss* yang mengukur seberapa baik hasil pemetaan f_{model} dengan *input* $\mathbf{x}_i$ terhadap *output* $\mathbf{y}_i$. Meskipun sembarang fungsi yang terturunkan dapat digunakan sebagai fungsi *loss*, namun pemilihan fungsi *loss* berdasarkan *maximum likelihood estimation* (MLE) lebih disarankan.

Untuk permasalahan klasifikasi biner, fungsi *loss* yang sering digunakan adalah *binary cross entropy* (BCE) seperti yang ditinjukkan pada Persamaan 2.50. Penurunan fungsi *loss* BCE dengan mengikuti prinsip MLE yang akan dijelaskan pada bagian selanjutnya.

Misalkan $y_i \mid \mathbf{x}$ mengikuti distribusi bernouli dengan parameter $\text{p} = f_{\text{model}}(\mathbf{x}; \boldsymbol{\theta})$ yang saling independen antar satu sama lainnya. Persamaan 2.42 menunjukkan definisi dari

$y_i \mid \mathbf{x}$.

$$y_i \mid \mathbf{x} \overset{\text{iid}}{\sim} \text{Bernoulli}(f_{\text{model}}(\mathbf{x};\boldsymbol{\theta})) \tag{2.42}$$

$$p(y_i \mid \mathbf{x}) = f_{\text{model}}(\mathbf{x};\boldsymbol{\theta})^{y_i}(1 - f_{\text{model}}(\mathbf{x};\boldsymbol{\theta}))^{1-y_i} \tag{2.43}$$

Fungsi *likelihood* dari $\boldsymbol{\theta}$ terhadap *dataset* $\mathcal{D}$ dapat ditulis sebagai berikut:

$$\mathcal{L}(\boldsymbol{\theta}) = \prod_{i=1}^{N} p(y_i \mid \mathbf{x}_i;\boldsymbol{\theta}). \tag{2.44}$$

Dengan prinsip MLE, parameter $\boldsymbol{\theta}$ yang dicari adalah parameter $\boldsymbol{\theta}$ yang memaksimalkan fungsi *likelihood* $\mathcal{L}(\boldsymbol{\theta})$,

$$\boldsymbol{\theta}_{\text{MLE}} = \arg\max_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}). \tag{2.45}$$

Untuk mempermudah perhitungan, fungsi *likelihood* diubah menjadi negatif *log-likelihood* $\ell(\boldsymbol{\theta})$, sehingga permasalahan optimasi menjadi:

$$\ell(\boldsymbol{\theta}) = -\log \mathcal{L}(\boldsymbol{\theta}), \tag{2.46}$$

$$= -\sum_{i=1}^{N} \log\left(p(y_i \mid \mathbf{x}_i;\boldsymbol{\theta})\right), \tag{2.47}$$

$$\boldsymbol{\theta}_{\text{MLE}} = \arg\min_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}). \tag{2.48}$$

Dengan mengganti $p(y_i \mid \mathbf{x}_i;\boldsymbol{\theta})$ dengan fungsi distribusi-nya, maka fungsi *loss* yang digunakan untuk permasalahan klasifikasi biner adalah *binary cross entropy* (BCE) seperti pada Persamaan 2.50.

$$\boldsymbol{\theta}_{\text{MLE}} = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{N} \underbrace{-y_i \log\left(f_{\text{model}}(\mathbf{x}_i;\boldsymbol{\theta})\right) - (1-y_i)\log\left(1-f_{\text{model}}(\mathbf{x}_i;\boldsymbol{\theta})\right)}_{\text{Binary Cross Entropy Loss } L(y_i, f_{\text{model}}(\mathbf{x}_i;\boldsymbol{\theta}))}, \tag{2.49}$$

$$L(y_i, f_{\text{model}}(\mathbf{x}_i;\boldsymbol{\theta})) = -y_i \log\left(f_{\text{model}}(\mathbf{x}_i;\boldsymbol{\theta})\right) - (1-y_i)\log\left(1-f_{\text{model}}(\mathbf{x}_i;\boldsymbol{\theta})\right). \tag{2.50}$$

Untuk mendapatkan f_{model} dengan performa yang baik, dibutuhkan model dengan nilai $\ell(\boldsymbol{\theta})$ seminimum mungkin. Namun, pencarian $\boldsymbol{\theta}$ sehingga $\ell(\boldsymbol{\theta})$ minumum secara analitik tidak dapat dilakukan karena non-linearitas yang ada pada model, dengan kata lain solusi dari $\nabla_{\boldsymbol{\theta}}\ell(\boldsymbol{\theta}) = 0$ tidak dapat dicari secara analitik. Sebagai gantinya, pencarian $\boldsymbol{\theta}$ dilakukan secara numerik dengan menggunakan metode *gradient descent* yang akan dijelaskan pada bagian selanjutnya.

2.3.4 *Backpropagation*

2.3.5 *Optimisasi*

Gradient Descent adalah metode numerik yang digunakan untuk mencari nilai $\boldsymbol{\theta}$ yang meminimalkan fungsi *loss* $\ell(\boldsymbol{\theta})$. Pada metode *gradient descent*, nilai $\boldsymbol{\theta}$ diupdate secara iteratif dengan mengikuti arah negatif dari *gradient* $\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})$ yang menunjukkan arah dari penurunan fungsi *loss* $\mathcal{L}(\boldsymbol{\theta})$. Persamaan 2.52 menunjukkan algoritma *gradient descent*.

$$\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\} \tag{2.51}$$

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta \frac{1}{n} \sum_{i=1}^{n} \nabla_{\boldsymbol{\theta}} L(y_i, f_{\text{model}}(\mathbf{x}_i;\boldsymbol{\theta}^{(t)})), \tag{2.52}$$

$$\text{dengan } \eta \in \mathbb{R}^{+} \text{ adalah } \textit{learning rate}. \tag{2.53}$$

Perlu diketahui bahwa pada metode *gradient descent* memperbarui parameter dengan mengambil rata-rata *gradient* dari semua data pada *dataset* pelatihan $\mathcal{D}$. Hal ini menciptakan masalah ketika model menggunakan banyak parameter dan jumlah data pada *datasets* latih besar, yaitu komputasi *forward pass* dan *backward pass* menjadi sangat mahal dan diperlukan memori yang besar untuk menyimpan gradient dari semua data pada *dataset* latih. Untuk mengatasi masalah tersebut, digunakan metode *stochastic gra-*

dient descent (SGD) dimana setiap *update* dari parameter $\boldsymbol{\theta}$ dihitung dengan mengambil rata-rata *gradient* dari sebagian data pada *dataset* $\mathcal{B} \subseteq \mathcal{D}$. Persamaan 2.56 menunjukkan algoritma *stochastic gradient descent*.

$$\mathcal{B} = \{(\mathbf{x}_{i_1}, y_{i_1}), (\mathbf{x}_{i_2}, y_{i_2}), \ldots, (\mathbf{x}_{i_b}, y_{i_b})\} \subseteq \mathcal{D}, \mid \mathcal{B} \mid \ll \mid \mathcal{D} \mid, j \tag{2.54}$$

$$\frac{1}{n}\sum_{i=1}^{n} \nabla_{\boldsymbol{\theta}} L(y_i, f_{\text{model}}(\mathbf{x}_i; \boldsymbol{\theta})) \approx \frac{1}{b}\sum_{i=1}^{b} \nabla_{\boldsymbol{\theta}} L(y_i, f_{\text{model}}(\mathbf{x}_i; \boldsymbol{\theta})), \tag{2.55}$$

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta \frac{1}{b}\sum_{i=1}^{b} \nabla_{\boldsymbol{\theta}} L(y_i, f_{\text{model}}(\mathbf{x}_i; \boldsymbol{\theta}^{(t)})). \tag{2.56}$$

2.3.6 Inisialisasi Bobot

2.4 Pembelajaran Representasi

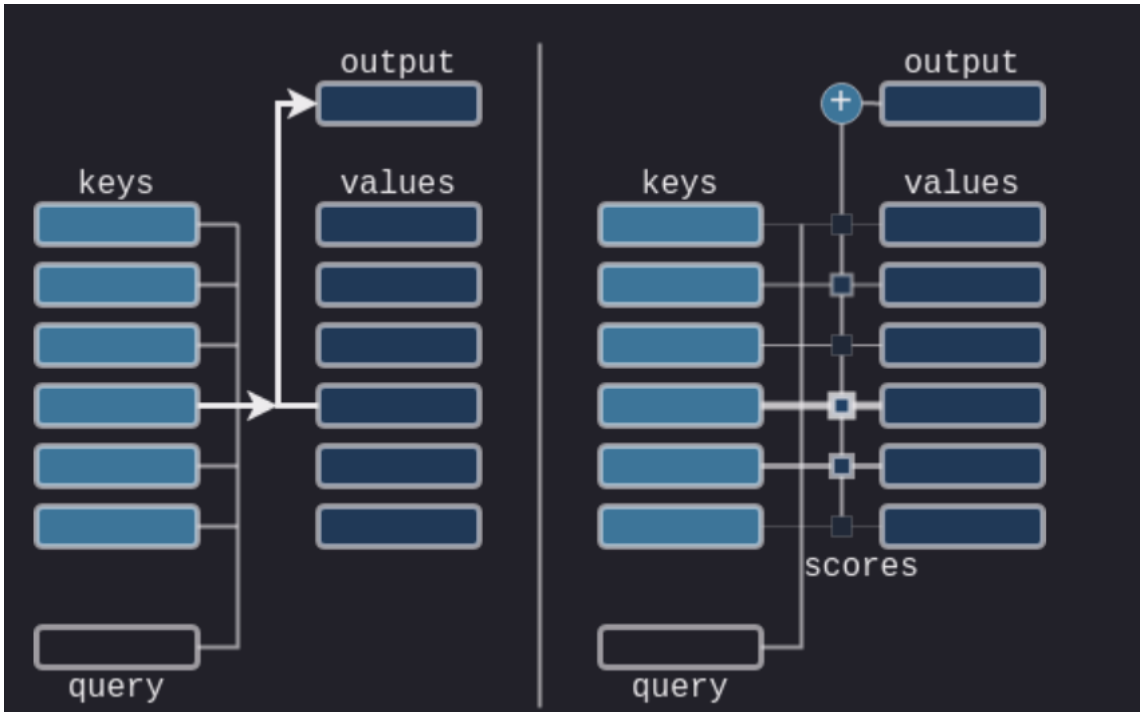
2.4.1 Fungsi *Loss* pada Pembelajaran Representasi

BAB 3
BIDIRECTIONAL ENCODER REPRESENTATION FROM TRANSFORMER (BERT) UNTUK PEMERINGKATAN TEKS

@todo
jabarin sih isinya mau gmna

3.1 Mekanisme *Attention*

3.1.1 Attention sebagai *Dictionary Lookup*



Gambar 3.1: Perbedaan mekanisme *hard attention* dan *soft attention* (pi tau, 2023)

Mekanisme *Attention* dapat ditinjau sebagai *Dictinoary Lookup*, yaitu untuk sebuah vektor kueri $\mathbf{q}$ dan sekumpulan pasangan terurut vektor $\mathcal{KV} = \{(\mathbf{k}_1, \mathbf{v}_2), (\mathbf{k}_2, \mathbf{v}_2), \ldots, (\mathbf{k}_n, \mathbf{v}_n)\}$, mekanisme *attention* akan mengembalikan vektor

nilai $\mathbf{v}_i$ yang memiliki vektor kunci $\mathbf{k}_i$ yang cocok dengan vektor kueri $\mathbf{q}$. Persamaan 3.1 hingga Persamaan 3.6 menunjukkan bagaimana mekanisme *attention* dilakukan.

$$\mathcal{KV} = \{(\mathbf{k}_1, \mathbf{v}_2), (\mathbf{k}_2, \mathbf{v}_2), \ldots, (\mathbf{k}_n, \mathbf{v}_n)\}, \tag{3.1}$$

$$\text{tulis kembali } \mathbf{K} = \begin{bmatrix} \mathbf{k}_1 \\ \mathbf{k}_2 \\ \vdots \\ \mathbf{k}_n \end{bmatrix} \in \mathbb{R}^{n \times d_k}, \tag{3.2}$$

$$\mathbf{V} = \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_n \end{bmatrix} \in \mathbb{R}^{n \times d_v}, \tag{3.3}$$

$$\text{Attention}(\mathbf{q}, \mathbf{K}, \mathbf{V}) = \boldsymbol{\alpha}\mathbf{V} \in \mathbb{R}^{d_v}, \tag{3.4}$$

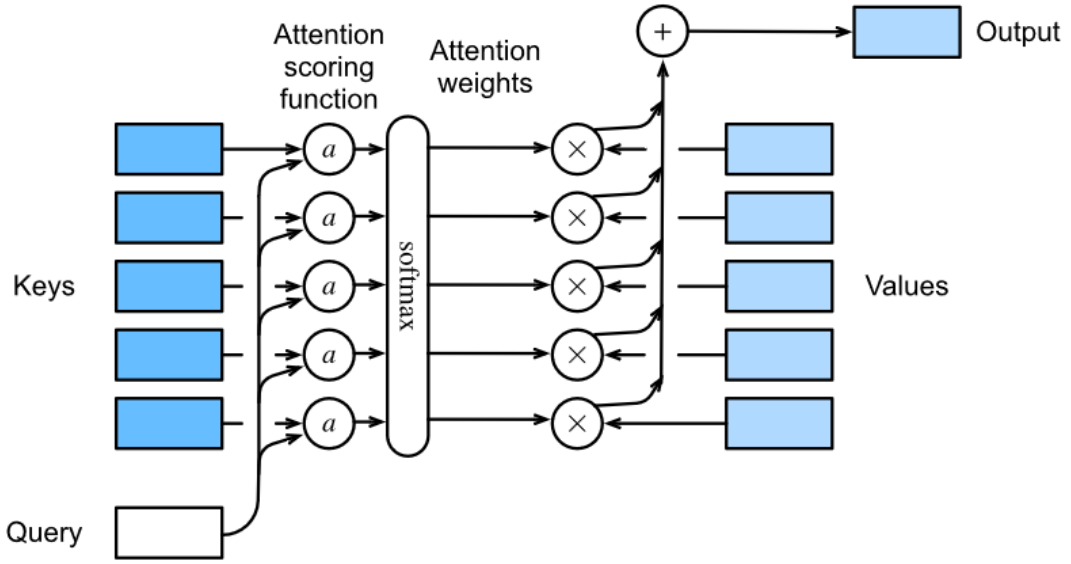
$$\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_n], \tag{3.5}$$

$$\text{dengan } \alpha_i = \begin{cases} 1, & \text{jika } i = \arg\max_j f_{attn}(\mathbf{q}, \mathbf{k}_j) \\ 0, & \text{lainnya} \end{cases}. \tag{3.6}$$

$f_{attn}(\mathbf{q}, \mathbf{k})$ adalah fungsi atensi yang menghitung nilai atensi antara vektor kueri $\mathbf{q}$ dan vektor kunci $\mathbf{k}$. α_i pada persamaan di atas disebut sebagai bobot atensi dan nilai $f_{attn}(\mathbf{q}, \mathbf{k})$ disebut sebagai nilai atensi.

Sebagai contoh, untuk $\mathbf{q} = [1,2]$, $\mathcal{KV} = \{([2,1],[1,0]),([1,2],[0,1])\}$ serta fungsi $f_{\text{attn}}(\mathbf{q}, \mathbf{k}) = \mathbf{q} \cdot \mathbf{k}$, nilai dari $\text{Attention}(\mathbf{q}, \mathbf{K}, \mathbf{V})$ adalah $[0,1]$, karena nilai maksimal f_{attn} terjadi ketika $\mathbf{k} = [1,2]$.

Mekansime *attention* pada Persamaan 3.1 hingga Persamaan 3.6 disebut sebagai *hard attention* karena hanya satu vektor nilai $\mathbf{v}_i$ yang dipilih dari sekumpulan vektor nilai $\mathbf{V}$. Berbeda dengan *hard attention* yang tidak terturunkan, *soft attention* mengambil seluruh vektor nilai $\mathbf{V}$ dan menghitung bobot α_i untuk setiap vektor nilai $\mathbf{v}_i$ dengan fungsi *softmax*. Hasil dari *soft attention* adalah rata-rata terbobot dari seluruh vektor nilai $\mathbf{V}$. Persamaan 3.7 hingga Persamaan 3.11 menunjukkan bagaimana mekanisme *soft attention* dilakukan.



Gambar 3.2: Ilustrasi dari mekanisme *soft attention* (Zhang et al., 2023)

$$\text{Attention}(\mathbf{q},\mathbf{K},\mathbf{V}) = \alpha\mathbf{V} \in \mathbb{R}^{d_v}, \tag{3.7}$$

$$\text{dengan } \boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_n], \tag{3.8}$$

$$\text{dan } \alpha_i(\mathbf{q},\mathbf{k}_i) = \text{Softmax}_i(\boldsymbol{\alpha}) = \frac{\exp(f_{attn}(\mathbf{q},\mathbf{k}_i))}{\sum_{j=1}^{n}\exp(f_{attn}(\mathbf{q},\mathbf{k}_j))}, \tag{3.9}$$

$$\sum_{i=1}^{n}\alpha_i = 1, \tag{3.10}$$

$$0 \leq \alpha_i \leq 1. \tag{3.11}$$

Dengan rata-rata terbobot dari $\mathbf{V}$, *soft attention* dapat dicari turunannya dengan *back-propagation* yang merupakan syarat *fundamental* yang harus dimiliki oleh sebuah model *deep learning*.

Sebagai contoh, hasil dari $\text{Attention}(\mathbf{q},\mathbf{K},\mathbf{V})$ untuk $\mathbf{q} = [1,2]$, $\mathcal{KV} = \{([2,1],[0,1]),([1,2],[1,0])\}$ serta fungsi $f_{\text{attn}}(\mathbf{q},\mathbf{k}) = \mathbf{q}\cdot\mathbf{k}$ adalah $0.268[0,1] + 0.732[1,0] = [0.732, 0.268]$ dengan $\alpha_1 = \frac{\exp(4)}{\exp(4)+\exp(5)} \approx 0.268$ dan $\alpha_2 = \frac{\exp(5)}{\exp(4)+\exp(5)} \approx 0.732$.

Pada kasus kumpulan kueri $Q = \{\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_m\}$, mekanisme *attention* untuk setiap triplet $(\mathbf{q}_i,\mathbf{K},\mathbf{V})$ dapat dihitung secara bersamaan seperti yang ditunjukkan pada Persamaan 3.12 hingga Persamaan 3.15.

$$\text{tulis } \mathbf{Q} = \begin{bmatrix} \mathbf{q}_1 \\ \mathbf{q}_2 \\ \vdots \\ \mathbf{q}_m \end{bmatrix} \in \mathbb{R}^{m \times d_k}, \tag{3.12}$$

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathbf{AV} \in \mathbb{R}^{m \times d_v}, \tag{3.13}$$

$$\mathbf{A} = \begin{bmatrix} \boldsymbol{\alpha}_1 \\ \boldsymbol{\alpha}_2 \\ \vdots \\ \boldsymbol{\alpha}_m \end{bmatrix} = \begin{bmatrix} \alpha_{11} & \alpha_{12} & \dots & \alpha_{1n} \\ \alpha_{21} & \alpha_{22} & \dots & \alpha_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{m1} & \alpha_{m2} & \dots & \alpha_{mn} \end{bmatrix} \in \mathbb{R}^{m \times n}, \tag{3.14}$$

$$\alpha_{ij}(\mathbf{q}_i, \mathbf{k}_j) = \text{Softmax}_j(\alpha_i) = \frac{\exp(f_{attn}(\mathbf{q}_i, \mathbf{k}_j))}{\sum_{k=1}^{n} \exp(f_{attn}(\mathbf{q}_i, \mathbf{k}_k))}, \tag{3.15}$$

dengan α_{ij} adalah bobot yang menunjukkan bobot atensi antara vektor kueri $\mathbf{q}_i$ dengan vektor kunci $\mathbf{k_j}$.

3.1.2 *Attention* Parametrik

Mekanisme *attention* yang dilakukan oleh Vaswani et al. (2017) merupakan mekanisme *attention* parametrik. Pada mekanisme *attention* parametrik, nilai vektor kueri $\mathbf{q}$ dan $\mathbf{v}$ dibandingkan pada ruang vektor yang akan dipelajari (*learned embedding space*) daripada ruang vektor aslinya. Sebagai contoh, untuk suatu kueri $\mathbf{q} \in \mathbb{R}^{d_q}$, dan vektor kunci $\mathbf{k} \in \mathbb{R}^{d_k}$, *additive attention* yang diperkenalkan oleh Bahdanau, Cho, dan Bengio (2016) menghitung nilai keserupaan antara $\mathbf{q}$ dan $\mathbf{k}$ seperti pada Persamaan 3.16

$$f_{attn}(\mathbf{q}\mathbf{W}^q, \mathbf{k}\mathbf{W}^k) = (\mathbf{q}\mathbf{W}^q + \mathbf{k}\mathbf{W}^k)\mathbf{W}^{\text{out}} \in \mathbb{R}, \tag{3.16}$$

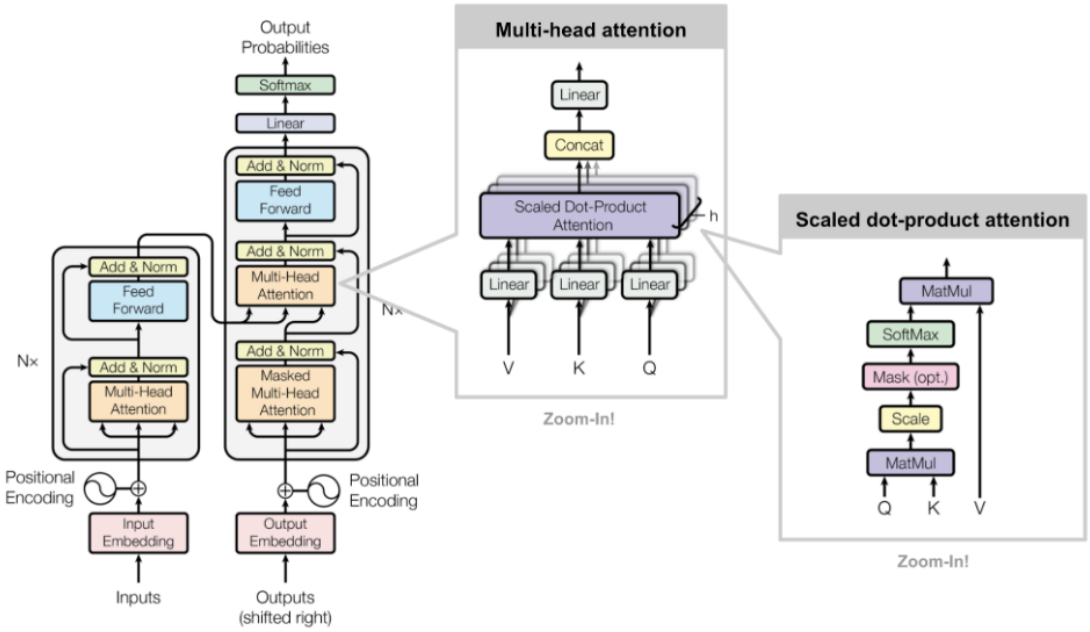
$$\text{dengan } \mathbf{W}^q \in \mathbb{R}^{d_q \times d_{\text{attn}}}, \mathbf{W}^k \in \mathbb{R}^{d_k \times d_{\text{attn}}}, \mathbf{W}_{\text{out}} \in \mathbb{R}^{d_{\text{attn}} \times 1}, \tag{3.17}$$

Dengan $\mathbf{W}^q$, $\mathbf{W}^k$, dan $\mathbf{W}^{\text{out}}$ adalah matriks parameter bobot yang akan diestimasi selama proses pelatihan. Contoh *attention* parametrik yang lebih sederhana adalah *dot-product attention*. Fungsi f_{attn} yang digunakan adalah perkalian titik antara $\mathbf{q}$ dan $\mathbf{k}$ di ruang vektor yang dipelajari (*learned embedding space*). Persamaan 3.18 menunjukkan bagaimana *dot-product attention* dihitung.

$$f_{attn}(\mathbf{q}\mathbf{W}^q, \mathbf{k}\mathbf{W}^k) = (\mathbf{q}\mathbf{W}^q)(\mathbf{k}\mathbf{W}^k)^\top \tag{3.18}$$

$$\text{dengan } \mathbf{W}^q \in \mathbb{R}^{d_q \times d_{\text{attn}}}, \mathbf{W}^k \in \mathbb{R}^{d_k \times d_{\text{attn}}}. \tag{3.19}$$

3.2 Transformer



Gambar 3.3: Arsitektur *transformer* (Weng, 2018).

Transformers merupakan Arsitektur *deep learning* yang pertama kali diperkenalkan oleh Vaswani et al. (2017). Awalnya Transformers merupakan model *sequance to sequance* yang diperuntukkan untuk permasalahan mesin translasi neural (*neural machine translation*). Namun, sekarang *transformer* juga digunakan untuk permasalahan pemrosesan bahasa alami lainnya. model-model yang berarsitektur *transformer* menjadi model *state-of-the-art* untuk permasalahan pemrosesan bahasa alami lainnya, seperti *question answering*, *sentiment analysis*, dan *named entity recognition*.

Berbeda dengan arsitektur mesin translasi terdahulu, transformer tidak mengunakan *recurrent neural network* (RNN) atau *convolutional neural network* (CNN), melainkan transformer adalah model *feed foward network* yang dapat memproses seluruh *input* pada barisan secara paralel. Untuk menggantikan kemampuan RNN dalam mempelajari ketergantungan antar *input* yang berurutan dan kemampuan CNN dalam mempelajari fitur

lokal, transformer bergantung pada mekanisme *attention*.

Terdapat tiga jenis *attention* yang digunakan dalam model *transformer* (Vaswani et al., 2017):

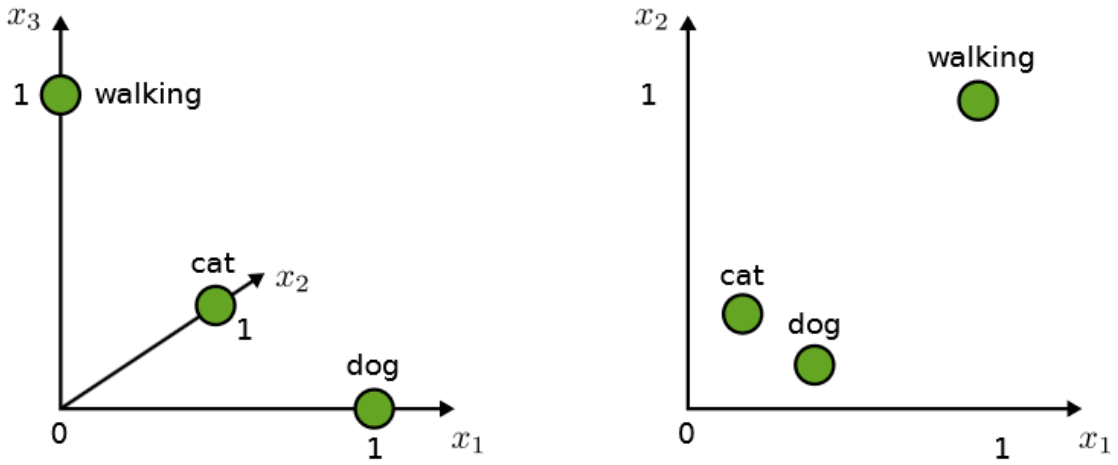
1. *Encoder self-attention*: menggunakan barisan *input* yang berupa barisan token atau kata sebagai masukan untuk menghasilkan barisan representasi kontekstual, berupa vektor, dari *input*. Setiap representasi token tersebut memiliki ketergantungan dengan token lainnya dari barisan *input*.

2. *Decoder self-attention*: menggunakan barisan *target* yang berupa kalimat terjemahan parsial, barisan token, sebagai masukan untuk menghasilkan barisan representasi kontekstual (vektor) dari *target*. Setiap representasi token tersebut memiliki ketergantungan dengan token sebelumnya dalam urutan masukan.

3. *Decoder-encoder attention*: menggunakan barisan representasi kontekstual dari *input*, dan barisan representasi kontekstual dari *target* untuk menghasilkan token berikutnya yang merupakan hasil prediksi dari model. Barisan *target* yang digabung dengan token hasil prediksi tersebut akan menjadi barisan *target* untuk prediksi selanjutnya.

Arsitektur dari *transformer* mengimplementasikan struktur pasangan encoder-decoder. Aristektur dari *transformer* dapat dilihat pada Gambar 3.3. Lapisan*encoder* berfungsi untuk memahami konteks suatu kata dalam dokumen atau kalimat, sementara lapisan *decoder* digunakan untuk menyelesaikan masalah translasi menuju bahasa berbeda. Subbab 3.2.1 hingga Subbab 3.2.8 menjelaskan arsitektur model *transformer* dan berbagai mekanisme yang menyusun model *transformer*, khususnya *transformer encoder* yang menjadi *building block* dari *Biderectional Encoder Representations from Transformers* (BERT).

3.2.1 *Token Embedding (Input Embedding)*



Gambar 3.4: Ilustrasi dari representasi token. Gambar kiri menunjukkan representasi token dengan *one-hot encoding*, sedangkan gambar kanan menunjukkan representasi token dengan *token embedding* (Geiger et al., 2022)

Perlu diingat kembali bahwa *input* dari *Attention* (dan tentunya *transformer*) adalah barisan vektor. Jika *Attention* ingin dapat digunakan pada permasalahan bahasa, barisan kata atau subkata (selanjutnya disebut token) harus terlebih dahulu diubah menjadi barisan vektor.

Representasi vektor dari token yang paling sederhana adalah dengan *one-hot encoding*. Andaikan $\mathcal{T} = \{t_1, t_2, \ldots, t_{|\mathcal{T}|}\}$ adalah semua kemungkinan token yang mungkin muncul dalam permasalahan bahasa yang ingin diselesaikan. Untuk sembarang barisan token $t = (t_{i_1}, t_{i_2}, \ldots, t_{i_L})$, representasi vektor dari token t_{i_j} adalah vektor $\mathbf{oh}_{i_j} = [0, \ldots, 0, 1, 0, \ldots, 0] \in \mathbb{R}^{|\mathcal{T}|}$, dengan nilai 1 pada indeks ke j dan nilai 0 pada indeks lainnya. *One-hot encoding* tentunya memiliki kelemahan:

1. Vektor yang dihasilkan adalah *sparse vector*, dan ukuran vektor yang dihasilkan cukup besar, yaitu $|\mathcal{T}|$.

2. Representasi token yang buruk. Operasi vektor yang dilakukan pada *one-hot encoding* tidaklah bermakna. Misalnya, Jarak antar token akan selalu sama pada *one-hot encoding*, yaitu $\sqrt{2}$.

Untuk Mengatasi kekurangan dari representasi *one-hot encoding*, reprentasi yang digunakan adalah vektor padat yang akan dipelajari ketika proses pelatihan. Misalkan $\mathbf{E}_{\mathcal{T}} \in \mathbb{R}^{|\mathcal{T}| \times d_{\text{token}}}$ adalah matriks parameter yang merupakan representasi vektor padat dari

seluruh token ada. Persamaan 3.20 hingga Persamaan 3.22 menunjukkan bagaimana representasi vektor dari barisan suatu token t dihitung.

$$t = (t_{i_1}, t_{i_2}, \ldots, t_{i_L}), \tag{3.20}$$

$$\mathbf{e}_{i_j} = \mathbf{oh}_{i_j}\mathbf{E}_{\mathcal{T}} \in \mathbb{R}^{d_{\text{token}}}, \tag{3.21}$$

$$\text{Embed}(t) = \mathbf{E}_t = \begin{bmatrix} \mathbf{e}_{i_1} \\ \mathbf{e}_{i_2} \\ \vdots \\ \mathbf{e}_{i_L} \end{bmatrix} \in \mathbb{R}^{L \times d_{\text{token}}}. \tag{3.22}$$

3.2.2 *Scaled Dot-Product Attention*

Scaled dot-product attention adalah mekanisme *Attention* parametrik yang digunakan dalam *transformers*. *Scaled dot-product attention* menghitung keserupaan antara vektor kueri $\mathbf{q}$ dan vektor kunci $\mathbf{k}$ pada ruang vektor yang dipelajari (*learned embedding space*) dengan fungsi keserupaan $f_{attn}(\mathbf{q}\mathbf{W}^q, \mathbf{k}\mathbf{W}^k)$ adalah perkalian titik antara $\mathbf{q}\mathbf{W}^q$ dan $\mathbf{k}\mathbf{W}^k$ yang kemudian dibagi dengan $\sqrt{d_{attn}}$, seperti pada Persamaan 3.23.

$$f_{attn}(\mathbf{q}\mathbf{W}^q, \mathbf{k}\mathbf{W}^k) = \frac{\mathbf{q}\mathbf{W}^q(\mathbf{k}\mathbf{W}^k)^\top}{\sqrt{d_{attn}}} \in \mathbb{R}, \tag{3.23}$$

$$\text{dengan } \mathbf{W}^q \in \mathbb{R}^{d_q \times d_{\text{attn}}}, \mathbf{W}^k \in \mathbb{R}^{d_k \times d_{\text{attn}}}. \tag{3.24}$$

pembagian dengan $\sqrt{d_{attn}}$ dilakukan untuk menjaga variansi dari nilai atensi $\mathbf{q}\mathbf{W}^q(\mathbf{k}\mathbf{W}^k)^\top$ tetap serupa dengan variansi $\mathbf{q}\mathbf{W}^q$ dan $\mathbf{k}\mathbf{W}^k$. Tanpa pembagian $\sqrt{d_{attn}}$, variansi dari nilai atensi akan memiliki faktor tambahan $\sigma^2 d_{attn}$, seperti yang ditunjukkan pada Persamaan 3.25 hingga Persamaan 3.26.

$$\mathbf{q}\mathbf{W}^q \sim \mathcal{N}(0, \sigma^2) \text{ dan } \mathbf{k}\mathbf{W}^k \sim \mathcal{N}(0, \sigma^2). \tag{3.25}$$

$$\text{Var}(\mathbf{q}\mathbf{W}^q(\mathbf{k}\mathbf{W}^k)^\top) = \sum_{i=1}^{d_{attn}} \text{Var}\left((\mathbf{q}\mathbf{W}^q)_i((\mathbf{k}\mathbf{W}^k)_i^\top\right) = \sigma^4 d_{attn}. \tag{3.26}$$

Akibatnya, untuk nilai d_{attn} yang cukup besar, akan terdapat satu elemen atensi

acak $(\mathbf{q}\mathbf{W}^q(\mathbf{k}\mathbf{W}^k)^\top)_i$ sehinnga $(\mathbf{q}\mathbf{W}^q(\mathbf{k}\mathbf{W}^k)^\top)_i \gg (\mathbf{q}\mathbf{W}^q(\mathbf{k}\mathbf{W}^k)^\top)_j$ untuk sembarang nilai atensi lainnya. Jika faktor d_{attn} tidak dihilangkan, *softmax* dari nilai atensi akan jenuh ke 1 untuk satu elemen acak tersebut dan 0 untuk elemen lainnya. Akibatnya, gradien pada fungsi *softmax* akan mendekati nol sehingga model tidak dapat belajar parameter dengan baik.

Dengan *scaled dot product attention*, tidak ada faktor d_{attn} pada variansi dari nilai atensi. faktor σ^4 pada Persamaan 3.27 tidak menjadi masalah karena inisialisasi bobot Kaiming dan *layer normalisasi* yang dijelaskan pada Subbab 2.3.6 dan Subbab 3.2.7 mengakibatkan $\sigma^2 \approx 1$ sehingga $\sigma^4 \approx \sigma^2 \approx 1$.

$$\text{(scaled dot product attention) Var}\left(\frac{\mathbf{q}\mathbf{W}^q(\mathbf{k}\mathbf{W}^k)^\top}{\sqrt{d_{attn}}}\right) = \frac{\sigma^4 d_{attn}}{d_{attn}} = \sigma^4 \tag{3.27}$$

Terakhir, untuk kumpulan vektor kueri $\mathcal{Q} = \{\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_m\}$, dan kumpulan vektor kunci dan nilai $\mathcal{KV} = \{(\mathbf{k}_1, \mathbf{v}_2), (\mathbf{k}_2, \mathbf{v}_2), \ldots, (\mathbf{k}_n, \mathbf{v}_n)\}$, *scaled dot product attention* dapat dihitung secara bersamaan seperti pada Persamaan 3.28 hingga Persamaan 3.31.

$$\text{Tulis Kembali } \mathbf{Q} = \begin{bmatrix} \mathbf{q}_1 \\ \mathbf{q}_2 \\ \vdots \\ \mathbf{q}_m \end{bmatrix} \in \mathbb{R}^{m \times d_q}, \tag{3.28}$$

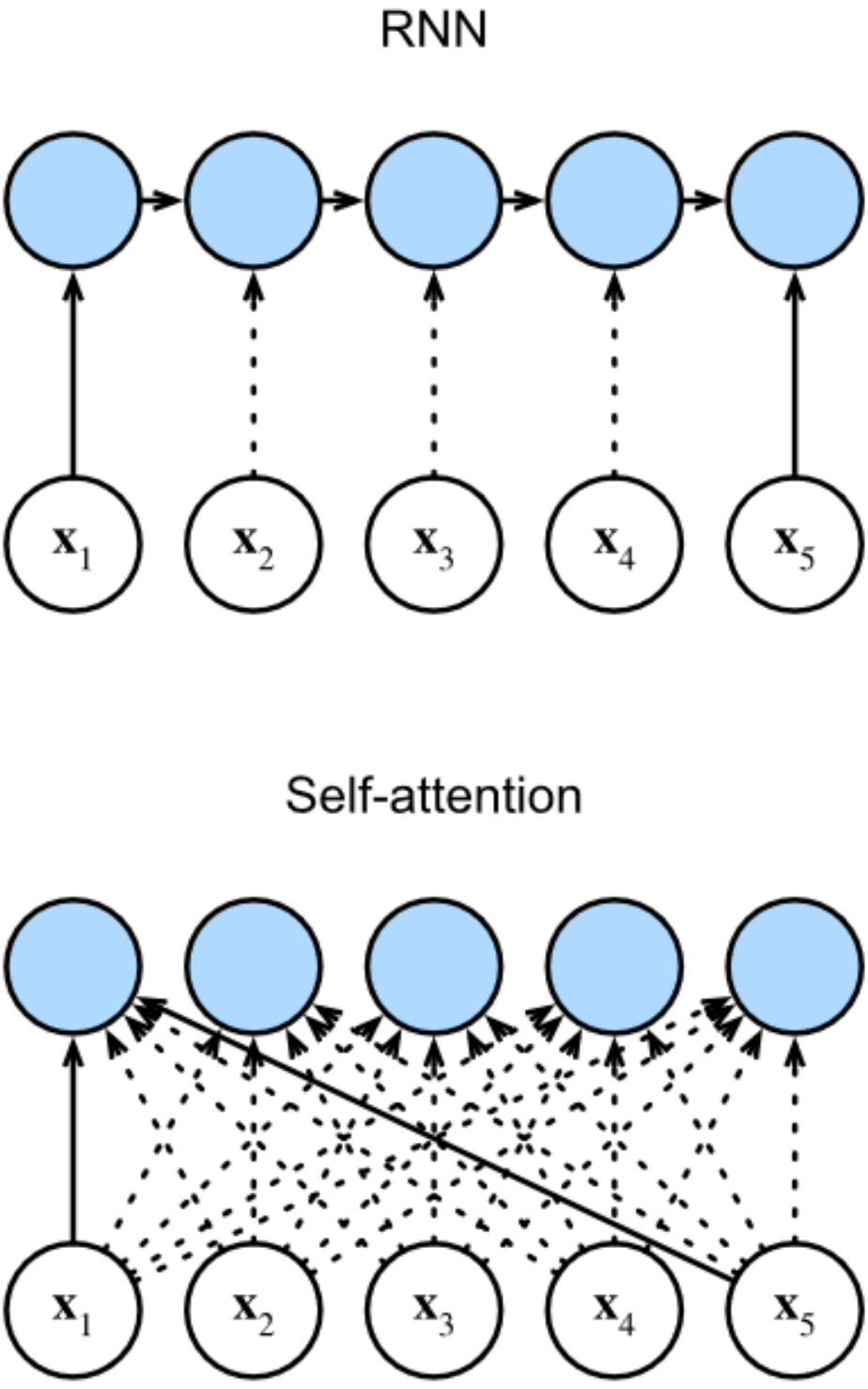
$$\mathbf{K} = \begin{bmatrix} \mathbf{k}_1 \\ \mathbf{k}_2 \\ \vdots \\ \mathbf{k}_n \end{bmatrix} \in \mathbb{R}^{n \times d_k}, \tag{3.29}$$

$$\text{dan } \mathbf{V} = \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_n \end{bmatrix} \in \mathbb{R}^{n \times d_v}, \tag{3.30}$$

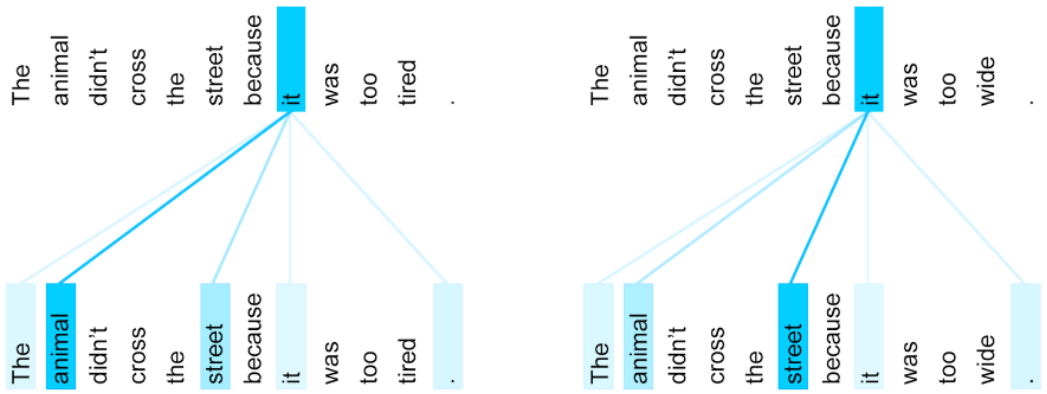
$$\text{Attention}(\mathbf{Q}\mathbf{W}^q, \mathbf{K}\mathbf{W}^k, \mathbf{V}) = \text{Softmax}(\frac{\mathbf{Q}\mathbf{W}^q(\mathbf{K}\mathbf{W}^k)^\top}{\sqrt{d_{attn}}})\mathbf{V} \in \mathbb{R}^{m \times d_v}, \tag{3.31}$$

$$\text{dengan } \mathbf{W}^q \in \mathbb{R}^{d_q \times d_{\text{attn}}}, \mathbf{W}^k \in \mathbb{R}^{d_k \times d_{\text{attn}}}. \tag{3.32}$$

3.2.3 *Self-Attention*



Gambar 3.5: Perbandingan RNN dan *self-attention* dalam menghasilkan representasi vektor kontekstual. Pada RNN, representasi vektor kontekstual setiap token bergantung pada perhitungan token sebelumnya. Pada *self-attention*, representasi vektor kontekstual setiap token dihitung secara independen dan paralel.



Gambar 3.6: Ilustrasi *self-attention* dalam menghasilkan representasi vektor kontekstual dari barisan token. Representasi vektor dari token *it* akan bergantung terhadap barisan token *input*.

self-Attention layer adalah layer yang digunakan *transformer* untuk menghasilkan representasi vektor yang kontekstual dari barisan token input. Berbeda dengan RNN dalam menghasilkan representasi vektor kontekstual, *self-attention* tidak memerlukan ketergantungan sekuensial. Artinya representasi vektor kontekstual setiap tokennya dapat dihitung secara independen dan paralel. Gambar 3.5 mengambarkan perbedaan kedua arsitektur dalam menghasilkan representasi vektor kontekstual. Kemampuan Paralelisme dari *self-attention* membuat proses komputasi menjadi lebih cepat pada *hardware* yang mendukung paralelisme.

Perhitungan *self-attention* pada *transformer* yang digunakan adalah *scaled dot product attention* yang telah dijelaskan pada Subbab 3.2.2. Pada *self-attention*, vektor kueri $\mathbf{q}$, vektor kunci $\mathbf{k}$, dan vektor nilai $\mathbf{v}$ adalah vektor yang sama, yaitu *embedding* token $\mathbf{E}$ yang dijelaskan pada Subbab 3.2.1. Selain itu, Dimensi dari *learned embedding space* d_{attn} yang digunakan untuk perhitungan nilai atensi adalah d_{token} yaitu dimensi dari *token embedding*. Persamaan 3.33 hingga Persamaan 3.35 menunjukkan bagaimana *self-attention* dihitung.

$$\mathbf{E} = \text{Embedding dari barisan token} \in \mathbb{R}^{L \times d_{\text{token}}} \tag{3.33}$$

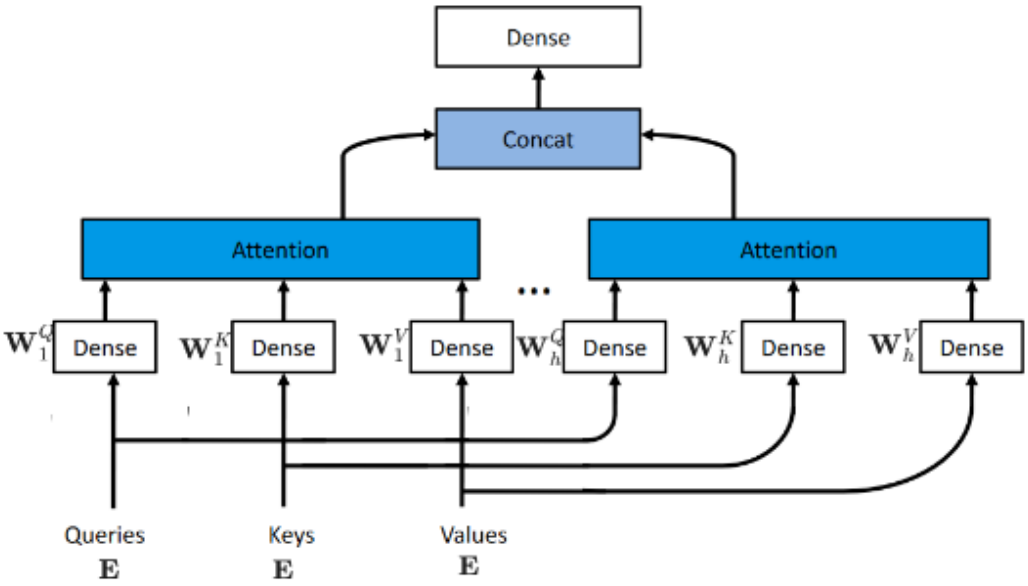
$$\text{Self-Attention}(\mathbf{E}) = \text{Attention}(\mathbf{E}\mathbf{W}^q, \mathbf{E}\mathbf{W}^k, \mathbf{E}\mathbf{W}^v) \tag{3.34}$$

$$= \text{Softmax}(\frac{\mathbf{E}\mathbf{W}^q(\mathbf{E}\mathbf{W}^k)^\top}{\sqrt{d_{attn}}})(\mathbf{E}\mathbf{W}^v) \in \mathbb{R}^{L \times d_{\text{token}}} \tag{3.35}$$

$$\text{dengan } \mathbf{W}^q, \mathbf{W}^k, \in \mathbb{R}^{d_{\text{token}} \times d_{\text{token}}}, \mathbf{W}^v \in \mathbb{R}^{d_{\text{token}} \times d_{\text{token}}} \tag{3.36}$$

self-attention dapat dikonsepsikan sebagai proses pembentukan represenatsi token yang kontekstual. untuk setiap tokennya, *self-attention* menghitung keserupaan antara token tersebut ($\mathbf{e}_{t_i}\mathbf{W}^q$) dengan seluruh token lainnya ($\mathbf{E}\mathbf{W}^k$) dengan *scaled dot product attention*. Hasil dari *scaled dot product attention* adalah vektor yang menunjukkan bobot atensi dari token tersebut terhadap token lainnya. Bobot atensi tersebut kemudian digunakan untuk menghitung rata-rata terbobot dari seluruh token lainnya ($\mathbf{E}\mathbf{W}^v$). Hasil dari rata-rata terbobot tersebut adalah representasi vektor kontekstual dari token tersebut. Gambar 3.6 adalah contoh dari *self-attention* yang menghasilkan representasi vektor kontekstual pada token *it*. Pada Gambar 3.6 kiri token *it* memiliki bobot atensi yang tinggi terhadap token dan *animal* sehingga representasi vektor kontekstual dari token *it* akan memiliki nilai yang serupa dengan representasi token *animal*. Di lain sisi, token *it* pada Gambar 3.6 memiliki bobot atensi yang tinggi terhadap token *street*.

3.2.4 *Multi-Head Self-Attention*



Gambar 3.7: Ilustrasi *multi-head self-attention* pada *transformer*. *Multi-head self-attention* menghitung *self-attention* sebanyak *h* kali pada subruang yang berbeda.

Multi-Head Self-Attention adalah arsiktetur pada *transformer* untuk melakukan mekanisme *self-attention* beberapa kali pada subruang (*learned embedded space*) yang berbeda. dengan melakukan hal tersebut, diharapkan bahwa model dapat menangkap relasi atau keserupaan antar token dari sudut pandang yang berbeda.

Secara teknis, *embedding* dari barisan token $\mathbf{E}$ akan dipetakan sebanyak h kali dengan *linear layer* yang kemudian hasil *attention* dari setiap *head* akan digabungkan dan dilakukan transformasi sekali lagi dengan*linear layer*. Persamaan 3.37 hingga Persamaan 3.40 menunjukkan bagaimana *multi-head self-attention* dihitung.

$$\text{MHSA}(\mathbf{E}) = \text{Concat}(\text{head}_i, \ldots, \text{head}_h)\mathbf{W}^O \in \mathbb{R}^{L \times d_{\text{token}}} \tag{3.37}$$

$$\text{head}_i = \text{Self-Attention}_i(\mathbf{E}) = \text{Softmax}(\frac{\mathbf{E}\mathbf{W}_i^q(\mathbf{E}\mathbf{W}_i^k)^\top}{\sqrt{d_{\text{token}}/h}})\mathbf{E}\mathbf{W}_i^v \in \mathbb{R}^{L \times \frac{d_{\text{token}}}{h}} \tag{3.38}$$

$$\text{Concat}(\text{head}_1, \ldots, \text{head}_h) = [\text{head}_1 | \ldots | \text{head}_h] \in \mathbb{R}^{L \times d_{\text{token}}} \tag{3.39}$$

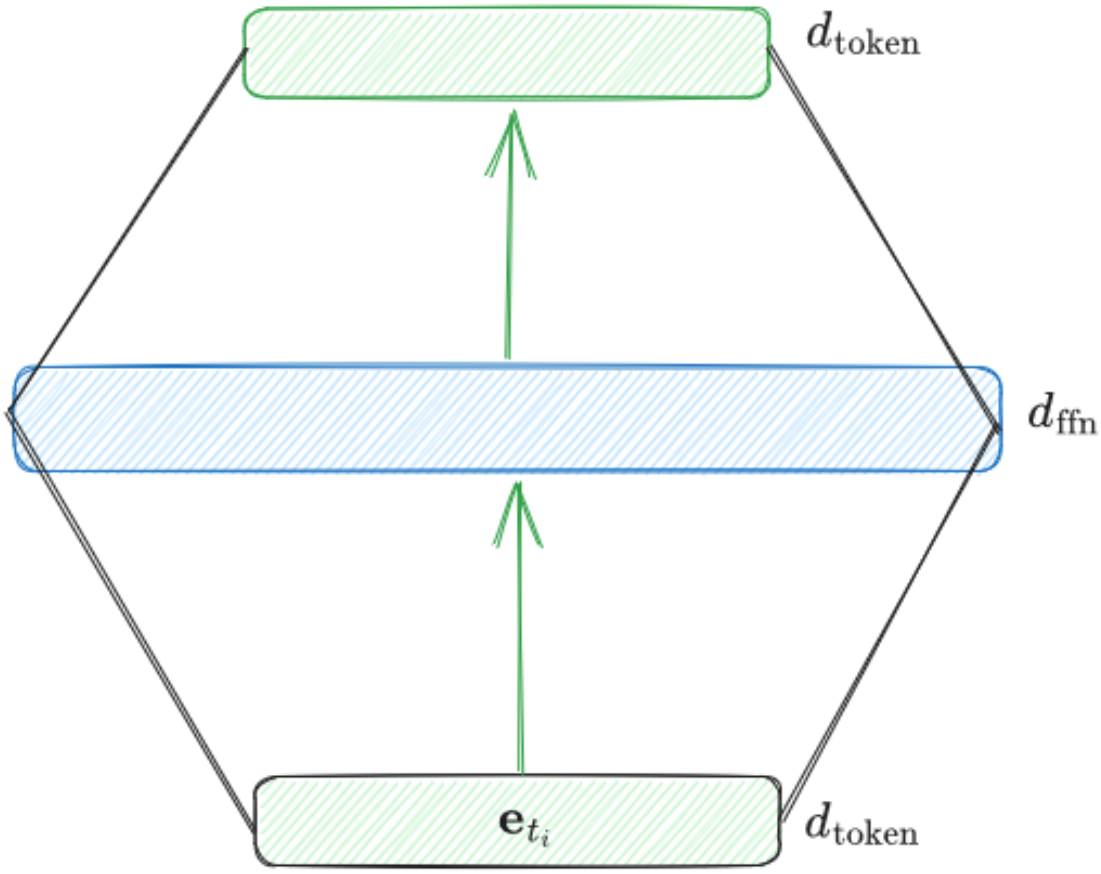
$$\text{dengan } \mathbf{W}_i^q, \mathbf{W}_i^k, \mathbf{W}_i^v, \in \mathbb{R}^{\frac{d_{\text{token}}}{h} \times \frac{d_{\text{token}}}{h}}, \mathbf{W}^O \in \mathbb{R}^{d_{\text{token}} \times d_{\text{token}}} \tag{3.40}$$

$$\tag{3.41}$$

perhatikan bahwa dimensi dari *learned embedding space* menjadi $\frac{d_{\text{token}}}{h}$ untuk setiap *head*-nya. Hal ini dilakukan untuk menjaga dimensi dari *output* terakhir tetap sama dengan dimensi dari *input*, yaitu d_{token}. Selain itu, argumen lainnya yang dapat dibuat adalah setiap *head* hanya perlu menggunakan dimensi yang lebih kecil dari d_{token} untuk menangkap ketergantungan antar-token (pi tau, 2023). Vaswani et al. (2017) menggunakan $h = 8$ *head* pada *transformer* yang digunakan untuk mesin translasi neural, dengan $d_{\text{token}} = 512$, sehingga dimensi dari *learned embedding space* untuk setiap *head* adalah $\frac{512}{8} = 64$.

3.2.5 *Positional Encoding*

3.2.6 *Position-wise Feed-Forward Network*



Gambar 3.8: Ilustrasi *position-wise feed-forward network* pada *transformer*.

Position-wise Feed-Foward Network adalah *feed foward network* dengan dua kali transformasi linear dan sebuah fungsi aktivasi ReLU di antaranya. Gambar 3.8 menunjukkan ilustrasi dari *position-wise feed-forward network* dan Persamaan 3.42 hingga Persamaan 3.43 menunjukkan Transformasi yang dilakukan oleh *position-wise feed-forward network*.

$$\text{FFN}(\mathbf{X}) = \max(0, \mathbf{X}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2 \in \mathbb{R}^{L \times d_{\text{token}}} \tag{3.42}$$

$$\mathbf{W}_1 \in \mathbb{R}^{d_{\text{token}} \times d_{\text{ffn}}}, \mathbf{W}_2 \in \mathbb{R}^{d_{\text{ffn}} \times d_{\text{token}}}, \mathbf{b}_1 \in \mathbb{R}^{d_{\text{ffn}}}, \mathbf{b}_2 \in \mathbb{R}^{d_{\text{token}}}. \tag{3.43}$$

d_{ffn} adalah dimensi dari *feed forward network* yang digunakan. Vaswani et al. (2017) menggunakan $d_{\text{ffn}} = 2048$.

3.2.7 Koneksi Residual dan *Layer Normalization*

Pada tahap pelatihan, pembaruan parameter model dilakukan pada semua *layer* secara serentak setiap iterasi *gradient descent*. Ketika parameter suatu *layer* mengalami pembaruan, distribusi dari *output* yang dihasilkan *layer* tersebut juga akan berubah pada iterasi selanjutnya. *Layer-layer* selanjutnya harus beradaptasi karena distribusi *input* dari *layer* tersebut berubah. Fenomena ini disebut *internal covariate shift* yang mengakibatkan proses pencarian parameter menjadi lebih lambat.

Layer Normalization berfungsi untuk mencegah masalah *internal covariate shift* di atas dengan membatasi distribusi nilai *output* –yang nantinya menjadi *input* pada *layer* selanjutnya– sehingga memiliki variansi 1 dan mean 0. Justifikasi lainnya di balik penggunaan *layer normalization* adalah variansi dari *input* dari *self-attention layer* haruslah 1 (lihat Subbab 3.2.2), sehingga variansi dari bobot atensi $\text{Softmax}(\frac{\mathbf{E}\mathbf{W}^q(\mathbf{E}\mathbf{W}^k)^\top}{\sqrt{d_{\text{token}}}})$ akan 1 juga. Persamaan 3.44 hingga Persamaan 3.51 menunjukkan proses kerja dari *layer normalization*.

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1,d_{\text{token}}} \\ x_{21} & x_{22} & \dots & x_{2,d_{\text{token}}} \\ \vdots & \vdots & \ddots & \vdots \\ x_{L1} & x_{L2} & \dots & x_{L,d_{\text{token}}} \end{bmatrix} \in \mathbb{R}^{L \times d_{\text{token}}} \tag{3.44}$$

$$\text{LayerNorm}(\mathbf{X}) = (\mathbf{X} - \boldsymbol{\mu}) \odot \frac{1}{\boldsymbol{\sigma}} \tag{3.45}$$

$$= \begin{bmatrix} \frac{x_{11}-\mu_1}{\sigma_1} & \frac{x_{12}-\mu_1}{\sigma_1} & \dots & \frac{x_{1,d_{\text{token}}}-\mu_1}{\sigma_1} \\ \frac{x_{21}-\mu_2}{\sigma_2} & \frac{x_{22}-\mu_2}{\sigma_2} & \dots & \frac{x_{2,d_{\text{token}}}-\mu_2}{\sigma_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{x_{L1}-\mu_L}{\sigma_L} & \frac{x_{L2}-\mu_L}{\sigma_L} & \dots & \frac{x_{L,d_{\text{token}}}-\mu_L}{\sigma_L} \end{bmatrix} \in \mathbb{R}^{L \times d_{\text{token}}} \tag{3.46}$$

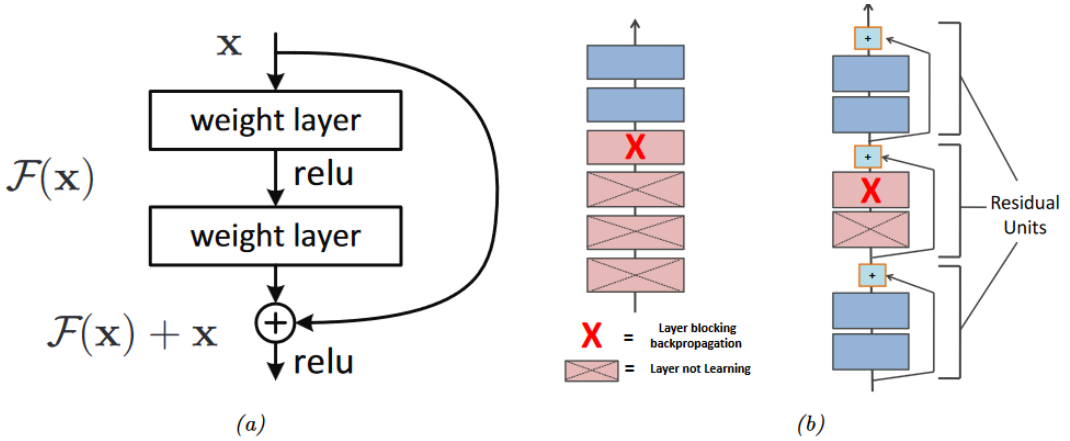
$$\boldsymbol{\mu} = \begin{bmatrix} \mu_1 & \dots & \mu_1 \\ \vdots & \ddots & \vdots \\ \mu_L & \dots & \mu_L \end{bmatrix} \in \mathbb{R}^{L \times d_{\text{token}}}, \tag{3.47}$$

$$\frac{1}{\boldsymbol{\sigma}} = \begin{bmatrix} \frac{1}{\sigma_1} & \dots & \frac{1}{\sigma_1} \\ \vdots & \ddots & \vdots \\ \frac{1}{\sigma_L} & \dots & \frac{1}{\sigma_L} \end{bmatrix} \in \mathbb{R}^{L \times d_{\text{token}}}, \tag{3.48}$$

$$\mu_i = \frac{1}{d_{\text{token}}} \sum_{j=1}^{d_{\text{token}}} x_{ij}, \quad i = 1, \dots, L, \tag{3.49}$$

$$\sigma_i = \sqrt{\frac{1}{d_{\text{token}}} \sum_{j=1}^{d_{\text{token}}} (x_{ij} - \mu_i)^2}, \quad i = 1, \dots, L, \tag{3.50}$$

$$\odot = \text{element-wise product.} \tag{3.51}$$



Gambar 3.9: Ilustrasi *layer normalization* pada *transformer*.

Koneksi Residu adalah koneksi yang menghubungkan *output* dari suatu *layer* dengan *input* dari *layer* selanjutnya. Koneksi residu digunakan untuk mengatasi masalah *vanishing gradient* yang terjadi pada *deep neural network* dengan memperbaiki *flow gradient*

$$f_l'(\mathbf{x}) = f_l(\mathbf{x}) + \mathbf{x} \tag{3.52}$$

$$\text{dengan } f_l(\mathbf{x}) \text{ adalah suatu layer pada } \textit{deep neural network}. \tag{3.53}$$

Pada *transformer*, *residual connection* digunakan sebelum *layer normalization* seperti pada Gambar 3.9. *Residual connection* digunakan untuk mengatasi masalah *vanishing gradient* yang terjadi pada *transformer* yang memiliki banyak *layer* (Vaswani et al., 2017). *Residual connection* juga digunakan pada *transformer decoder* untuk mengatasi masalah *exposure bias* yang terjadi pada *transformer decoder* (**?**).

3.2.8 Transformer Encoder

Dengan menggunakan *multi-head self-attention layer*, *position-wise feed-forward network layer*, dan *layer normalization* dan *residual connection* yang sudah dijelaskan sebelumnya, blok enkoder pada *transformer* enkoder dapat ditulis seperti pada Persamaan **??** hingga Persamaan **??**.

3.3 Bidirectional Encoder Representations from Transformers (BERT)

3.3.1 Representasi Input

3.3.2 Model Pralatih BERT

3.3.2.1 *Masked Language Model*

3.3.2.2 *Next Sentence Prediction*

3.3.3 BERT untuk Bahasa Indonesia (IndoBERT)

3.3.4 Penggunaan BERT untuk Pemeringkatan Teks

3.3.4.1 BERT_{CAT}

3.3.4.2 BERT_{DOT}

BAB 4
HASIL SIMULASI DAN PEMBAHASAN

Bab ini membahas mengenai proses fine tuning model Bidirectional Encoder Representations from Transformers (BERT) untuk mendapatkan model yang dapat digunakan untuk masalah pemeringkatan teks. Subbab4.1 menjelaskan mengenai spesifikasi perangkat keras dan perangkat lunak yang digunakan dalam penelitian. Selanjutnya, Subbab 4.2 menjelaskan mengenai tahapan simulasi yang dilakukan dalam penelitian. Dataset latih (train) dan uji (validation) dijelaskan pada Subbab 4.3. Subbab 4.5 menjelaskan lebih detail mengenai arsitektur model BERT,fungsi loss, serta konfigurasi hyperparameter yang digunakan dalam proses fine tuning model BERT. Subbab 4.4 menjelaskan kembali mengenai metriks evaluasi yang digunakan pada setiap dataset uji yang digunakan. Terakhir, Subbab 4.6 menjelaskan mengenai hasil fine tuning model BERT dan evaluasi dari model-model yang dihasilkan.

4.1 Spesifikasi Mesin dan Perangkat Lunak

@todo
banyak sih :'D, tambahin tabel isi qid, pid, label buat mmarco train tambahin tabel isi qid, pid, label buat miracl test, tunjukkin ini lebih dense dari mrtydi dan mmarco dev/ mrtydi test

Proses fine tuning model BERT untuk pemeringkatan teks dilakukan menggunakan mesin dan perangkat lunak yang tertera pada berikut.

4.2 Tahapan Simulasi

menunjukkan tahapan simulasi yang dilakukan dalam penelitian ini.

4.3 Dataset Latih dan Uji

4.3.1 Dataset Latih

4.3.1.1 Mmarco Indonesia Train Set

4.3.2 Dataset Uji

4.3.2.1 Mmarco Indonesia DEV Set

4.3.2.2 Mrtydi Indonesia TEST Set

4.3.2.3 Miracl Indonesia TEST Set

4.4 Metriks Evaluasi

4.5 Fine Tuning BERT

4.5.1 IndoBERT$_{\text{CAT}}$

4.5.2 IndoBERT$_{\text{DOT}}$

4.5.3 IndoBERT$_{\text{DOThardnegs}}$

4.5.4 IndoBERT$_{\text{DOTMargin}}$

4.5.5 IndoBERT$_{\text{KD}}$

4.6 Hasil Fine Tuning dan Evaluasi

4.6.1 Evaluasi BM25

Tabel 4.1: Caption

Model	Mmarco Dev		MrTyDi Test		Miracl Dev	
	MRR@10	R@1000	MRR@10	R@1000	NCDG@10	R@1K
BM25 (Elastic Search)	.114	.642	.279	.858	.391	.971

Tabel 4.2: Caption

Model	Mmarco Dev		MrTyDi Test		Miracl Dev	
	MRR@10	R@1000	MRR@10	R@1000	NCDG@10	R@1K
BM25 (Elastic Search)	.114	.642	.279	.858	.391	.971
$\text{IndoBERT}_{\text{MEAN}}$	.000	.000	.000	.000	.000	.000

4.6.2 Evaluasi $\text{IndoBERT}_{\text{MEAN}}$

4.6.3 Evaluasi $\text{IndoBERT}_{\text{CAT}}$

Tabel 4.3: Caption

Model	Mmarco Dev		MrTyDi Test		Miracl Dev	
	MRR@10	R@1000	MRR@10	R@1000	NCDG@10	R@1K
BM25 (Elastic Search)	.114	.642	.279	.858	.391	.971
$\text{IndoBERT}_{\text{CAT}}$	.181	.642	.447	.858	.455	.971

4.6.4 Evaluasi $\text{IndoBERT}_{\text{DOT}}$

Tabel 4.4: Caption

Model	Mmarco Dev		MrTyDi Test		Miracl Dev	
	MRR@10	R@1000	MRR@10	R@1000	NCDG@10	R@1K
BM25 (Elastic Search)	.114	.642	.279	.858	.391	.971
$\text{IndoBERT}_{\text{DOT}}$	.192	.847	.378	.936	.355	.920

4.6.5 Evaluasi $\text{IndoBERT}_{\text{DOThardnegs}}$

Tabel 4.5: Caption

Model	Mmarco Dev		MrTyDi Test		Miracl Dev	
	MRR@10	R@1000	MRR@10	R@1000	NCDG@10	R@1K
BM25 (Elastic Search)	.114	.642	.279	.858	.391	.971
$\text{IndoBERT}_{\text{DOThardnegs}}$	.232	.847	.471	.921	.397	.898

4.6.6 Evaluasi IndoBERT$_{\text{DOTMargin}}$

Tabel 4.6: Caption

Model	Mmarco Dev		MrTyDi Test		Miracl Dev	
	MRR@10	R@1000	MRR@10	R@1000	NCDG@10	R@1K
BM25 (Elastic Search)	.114	.642	.279	.858	.391	.971
IndoBERT$_{\text{DOTMargin}}$	.207	.799	.446	.929	.387	.899

4.6.7 Evaluasi IndoBERT$_{\text{KD}}$

Tabel 4.7: Caption

Model	Mmarco Dev		MrTyDi Test		Miracl Dev	
	MRR@10	R@1000	MRR@10	R@1000	NCDG@10	R@1K
BM25 (Elastic Search)	.114	.642	.279	.858	.391	.971
IndoBERT$_{\text{KD}}$	-	.803	.300	.761	-	-

4.6.8 Perbandingan Hasil Evaluasi

Tabel 4.8: Caption

Model	Mmarco Dev		MrTyDi Test		Miracl Dev	
	MRR@10	R@1000	MRR@10	R@1000	NCDG@10	R@1K
BM25 (Elastic Search)	.114	.642	.279	.858	.391	.971
IndoBERT$_{\text{MEAN}}$	.000	.000	.000	.000	.000	.000
IndoBERT$_{\text{CAT}}$	.181	.642	.447	.858	.455	.971
IndoBERT$_{\text{DOT}}$	.192	.847	.378	.936	.355	.920
IndoBERT$_{\text{DOTdnegs}}$	.232	.847	.471	.921	.397	.898
IndoBERT$_{\text{DOTMargin}}$	.207	.799	.446	.929	.387	.899
IndoBERT$_{\text{KD}}$	-	.803	.300	.761	-	-

Tabel 4.9: Caption

Model	Latensi (ms)	Memori(MB)
BM25 (Elastic Search)	6.55	800
$\text{IndoBERT}_{\text{DOT}}$	9.9	3072
$\text{IndoBERT}_{\text{CAT}}$	242	800

BAB 5
PENUTUP

Pada bab ini, Penulis akan memaparkan kesimpulan penelitian dan saran untuk penelitian berikutnya.

5.1 Kesimpulan

Berikut ini adalah kesimpulan terkait pekerjaan yang dilakukan dalam penelitian ini:

1. **Poin pertama**
   Penjelasan poin pertama.
2. **Poin kedua**
   Penjelasan poin kedua.

Tulis kalimat penutup di sini.

5.2 Saran

Berdasarkan hasil penelitian ini, berikut ini adalah saran untuk pengembangan penelitian berikutnya:

1. Saran 1.
2. Saran 2.

DAFTAR REFERENSI

Bahdanau, D., Cho, K., & Bengio, Y. (2016). *Neural machine translation by jointly learning to align and translate.*

Geiger, A., Antic, B., & He, H. (2022). *Lecture: Deep learning, university of tübingen.* https://uni-tuebingen.de/fakultaeten/mathematisch-naturwissenschaftliche-fakultaet/fachbereiche/informatik/lehrstuehle/autonomous-vision/lectures/deep-learning/.

Hofstätter, S., Althammer, S., Sertkan, M., & Hanbury, A. (2021). *Advanced information retrieval 2021 & 2022.* Diakses dari https://github.com/sebastian-hofstaetter/teaching

Lin, J., Nogueira, R. F., & Yates, A. (2020). Pretrained transformers for text ranking: BERT and beyond. *CoRR*, *abs/2010.06467*. Diakses dari https://arxiv.org/abs/2010.06467

pi tau. (2023). An even more annotated transformer. *pi-tau.github.io*. Diakses dari https://pi-tau.github.io/posts/transformer/ (Published on July 13, 2023)

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. In *Proceedings of the 31st international conference on neural information processing systems* (p. 6000–6010). Red Hook, NY, USA: Curran Associates Inc.

Weng, L. (2018). Attention? attention! *lilianweng.github.io*. Diakses dari https://lilianweng.github.io/posts/2018-06-24-attention/

Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2023). *Dive into deep learning*. Cambridge University Press. (https://D2L.ai)

LAMPIRAN

LAMPIRAN 1: CHANGELOG

@todo

Silakan hapus lampiran ini ketika Anda mulai menggunakan *template*.

Template versi terbaru bisa didapatkan di `https://gitlab.com/ichlaffterlalu/latex-skripsi-ui-2017`. Daftar perubahan pada *template* hingga versi ini:

- versi 1.0.3 (3 Desember 2010):
  - *Template* Skripsi/Tesis sesuai ketentuan *formatting* tahun 2008.
  - Bisa diakses di `https://github.com/edom/uistyle`.
- versi 2.0.0 (29 Januari 2020):
  - *Template* Skripsi/Tesis sesuai ketentuan *formatting* tahun 2017.
  - Menggunakan BibTeX untuk sitasi, dengan format *default* sitasi IEEE.
  - *Template* kini bisa ditambahkan kode sumber dengan *code highlighting* untuk bahasa pemrograman populer seperti Java atau Python.
- versi 2.0.1 (8 Mei 2020):
  - Menambahkan dan menyesuaikan tutorial dari versi 1.0.3, beserta cara kontribusi ke template.
- versi 2.0.2 (14 September 2020):
  - Versi ini merupakan hasil *feedback* dari peserta skripsi di lab *Reliable Software Engineering* (RSE) Fasilkom UI, semester genap 2019/2020.
  - BibTeX kini menggunakan format sitasi APA secara *default*.
  - Penambahan tutorial untuk `longtable`, agar tabel bisa lebih dari 1 halaman dan header muncul di setiap halaman.
  - Menambahkan tutorial terkait penggunaan BibTeX dan konfigurasi *header/footer* untuk pencetakan bolak-balik.

  - Label ”Universitas Indonesia” kini berhasil muncul di halaman pertama tiap bab dan di bagian abstrak - daftar kode program.
  - *Hyphenation* kini menggunakan `babel` Bahasa Indonesia. Aktivasi dilakukan di `hype-indonesia.tex`.
  - Minor adjustment untuk konsistensi *license* dari template.

- versi 2.0.3 (15 September 2020):
  - Menambahkan kemampuan orientasi *landscape* beserta tutorialnya.
  - `\captionsource` telah diperbaiki agar bisa dipakai untuk `longtable`.
  - Daftar lampiran kini telah tersedia, lampiran sudah tidak masuk daftar isi lagi.
  - Nomor halaman pada lampiran dilanjutkan dari halaman terakhir konten (daftar referensi).
  - Kini sudah bisa menambahkan daftar isi baru untuk jenis objek tertentu (custom), seperti: ”Daftar Aturan Transformasi”. Sudah termasuk mekanisme *captioning* dan tutorialnya.
  - Perbaikan minor pada tutorial.

- versi 2.1.0 (8 September 2021):
  - Versi ini merupakan hasil *feedback* dari peserta skripsi dan tesis di lab *Reliable Software Engineering* (RSE) Fasilkom UI, semester genap 2020/2021.
  - Minor edit: ”Lembar Pengesahan”, dsb. di daftar isi menjadi all caps.
  - Experimental multi-language support (Chinese, Japanese, Korean).
  - Support untuk justifikasi dan word-wrapping pada tabel.
  - Penggunaan suffix ”(sambungan)” untuk tabel lintas halaman. Tambahan support suffix untuk `\captionsource`.

- versi 2.1.1 (7 Februari 2022):
  - Update struktur mengikuti fork template versi 1.0.3 di `https://github.com/rkkautsar/edom/ui-thesis-template`.
  - Support untuk simbol matematis `amsfonts`.

  - Kontribusi komunitas terkait improvement GitLab CI, atribusi, dan format sitasi APA bahasa Indonesia.
  - Perbaikan tutorial berdasarkan perubahan terbaru pada versi 2.1.0 dan 2.1.1.

- versi 2.1.2 (13 Agustus 2022):
  - Modifikasi penamaan beberapa berkas.
  - Perbaikan beberapa halaman depan (halaman persetujuan, halaman orisinalitas, dsb.).
  - Support untuk lembar pengesahan yang berbeda dengan format standar, seperti Laporan Kerja Praktik dan Disertasi.
  - Kontribusi komunitas terkait kesesuaian dengan format Tugas Akhir UI, kelengkapan dokumen, perbaikan format sitasi, dan *quality-of-life*.
  - Perbaikan tutorial.
- versi 2.1.3 (22 Februari 2023):
  - Dukungan untuk format Tugas Akhir Kelompok di Fasilkom UI.
  - Dukungan untuk format laporan Kampus Merdeka Mandiri di Fasilkom UI.
  - Minor bugfix: Perbaikan kapitalisasi variabel.
  - Quality-of-Life: Pengaturan kembali `config/settings.tex`.
  - Tutorial untuk beberapa *use case*.

LAMPIRAN 2: JUDUL LAMPIRAN 2

Lampiran hadir untuk menampung hal-hal yang dapat menunjang pemahaman terkait tugas akhir, namun akan mengganggu *flow* bacaan sekiranya dimasukkan ke dalam bacaan. Lampiran bisa saja berisi data-data tambahan, analisis tambahan, penjelasan istilah, tahapan-tahapan antara yang bukan menjadi fokus utama, atau pranala menuju halaman luar yang penting.

Subbab dari Lampiran 2

@todo

Isi subbab ini sesuai keperluan Anda. Anda bisa membuat lebih dari satu judul lampiran, dan tentunya lebih dari satu subbab.