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.

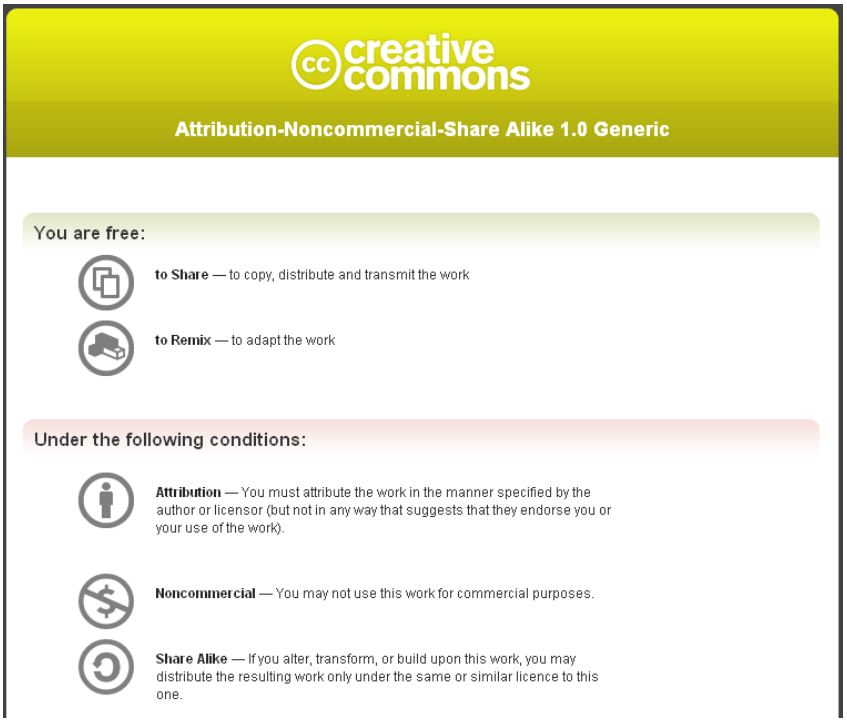
@todo

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`.



Creative Common License 1.0 Generic

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 menggunakan 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

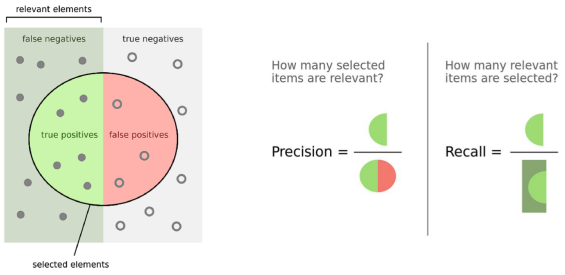
Permasalahan pemeringkatan teks adalah Permasalahan untuk menentukan urutan dokumen yang paling relevan dengan kueri q yang diberikan. Dalam bahasa yang lebih formal, diberikan kueri q dan himpunan dokumen terbatas $\mathcal{D} = \{d_1, d_2, ..., d_n\}$, keluaran yang diinginkan dari permasalahan ini 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. Persamaan **??**.

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: *Creative Common License 1.0 Generic*.

Presisi dan *recall* adalah metrik yang paling sederhana untuk mengukur kemampuan dari suatu model pemeringkatan teks. *Recall* mengukur kemampuan model untuk mengembalikan dokumen yang relevan dengan kueri q dari seluruh dokumen yang relevan dengan kueri q (Lin, Nogueira, & Yates, 2020). Di lain sisi, presisi mengukur kemampuan model dalam mengembalikan dokumen yang relevan dengan kueri q dari seluruh dokumen yang dikembalikan oleh model (Lin et al., 2020). Untuk suatu kueri q, kumpulan dokumen $\mathcal{D} = \{d_1, d_2, ..., d_n\}$, dan barisan k dokumen yang diambil oleh model, $\mathcal{D}_k = (d_{i_1}, d_{i_2}, ..., d_{i_k})$, *recall* dan presisi dapat dihitung dengan Persamaan 2.1 dan Persamaan 2.4.

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

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

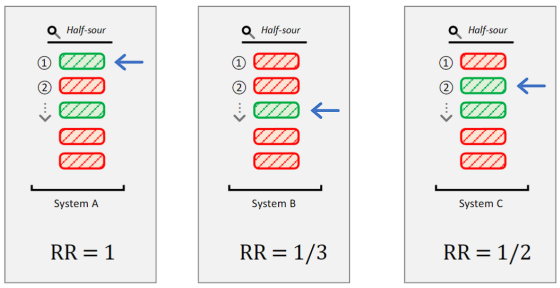
$$\text{recall}(q, D_k)@\text{k} = \frac{\sum_{d \in D_k} \text{rel}(q,d)}{\sum_{d \in \mathcal{D}} \text{rel}(q,d)} \in [0,1] \tag{2.3}$$

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

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

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. Gambar 2.1 mengilustrasikan metrik *recall* dan presisi.

2.1.3.2 *Reciprocal Rank*



Gambar 2.2: idk.

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.

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

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

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)

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.

Biasanya, metrik nDCG digunakan untuk *dataset* dengan *judgements r* yang padat. Selain itu, jika pada *datasets* memiliki *judgements* biner, faktor $2^{\text{rel}(q,d)}-1$ pada Persamaan 2.9 dapat diubah menjadi 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

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

2.2.2 *Best Match 25* (BM25)

2.3 Arsitektur *Deep Learning*

2.3.1 *Multilayer Perceptron* (MLP)

2.3.2 Fungsi Aktivasi

2.3.3 Fungsi *Loss*

2.3.4 *Backpropagation*

2.3.5 Inisialisasi Bobot

2.3.5.1 Inisialisasi Kaiming Menjaga Variansi *ouput* pada *Hidden Layer*

2.3.5.2 Insialisasi Kaiming Menjaga Variansi Gradien

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*

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 serupa dengan vektor kueri $\mathbf{q}$. Persamaan 3.1 hingga Persamaan 3.6 menunjukkan bagaimana mekanisme *attention* dilakukan.

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

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

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

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

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

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

dan $f_{attn}(\mathbf{q}, \mathbf{k})$ adalah fungsi yang menghitung nilai keserupaan 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.

Kasih contoh hard attention.

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, akibatnya *hard attention* tidak dapat dilatih dengan *backpropagation*, *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 dan gambarxx menunjukkan bagaimana *soft attention* dilakukan.

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

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

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

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

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

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

kasih contoh soft attention.

Pada kasus kumpulan kueri $\mathcal{Q} = \{\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_m\}$, Perhitungan atensi untuk setiap triplet $(\mathbf{q}_i, \mathbf{K}, \mathbf{V})$ dapat dihitung secara bersamaan dengan menggunakan operasi matriks. Persamaan 3.7 hingga Persamaan 3.11 yang digunakan untuk kasus 1 kueri dapat ditulis ulang seperti pada Persamaan 3.12 hingga Persamaan 3.15.

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

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

$$\mathbf{A} = \begin{bmatrix}\alpha_1\\ \alpha_2\\ \vdots\\ \alpha_m\end{bmatrix} = \begin{bmatrix}\alpha_{11} & \alpha_{12} & \ldots & \alpha_{1n}\\ \alpha_{21} & \alpha_{22} & \ldots & \alpha_{2n}\\ \vdots & \vdots & \ddots & \vdots\\ \alpha_{m1} & \alpha_{m2} & \ldots & \alpha_{mn}\end{bmatrix} \in \mathbb{R}^{m\times n}, \tag{3.14}$$

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

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

3.1.2 Regresi Kernel Sebagai *Attention* non-parametrik

Salah satu pengunaan mekanisme *attention* terdapat pada regresi kernel, yang merupakan model statistik non-parametrik. *Attention* berubah menjadi regresi kernel dengan memilih fungsi keserupaan $f_{attn}(\mathbf{q},\mathbf{k})$ menjadi fungsi non-parametrik $\mathcal{K}(\mathbf{q},\mathbf{k})$, dan mengganti fungsi softmax menjadi fungsi normalisasi standar, seperti pada Persamaan 3.17.

Pada model non-parametrik, model yang dibangun tidak memiliki parameter yang harus dicari atau dipelajari, melainkan model non-parametrik menggunakan seluruh atau sebagian dari *datasets* latih $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$ untuk memberikan prediksi y_* untuk sebuah data uji $\mathbf{x}_*$. Persamaan 3.16 hingga Persamaan 3.17 menunjukkan bagaimana model regresi kernel melakukan prediksi y_* untuk sebuah data uji $\mathbf{x}_*$.

$$y_* = f(\mathbf{x}_*) = \sum_{i=1}^{n} \alpha_i(\mathbf{x}_*, \mathbf{x}_i) y_i, \tag{3.16}$$

$$\text{dengan } \alpha_i(\mathbf{x}_*, \mathbf{x}_i) = \frac{\mathcal{K}(\mathbf{x}_*, \mathbf{x}_i)}{\sum_{j=1}^{n} \mathcal{K}(\mathbf{x}_*, \mathbf{x}_j)} \in [0,1], \tag{3.17}$$

dan $\mathcal{K}(\mathbf{x}_*, \mathbf{x}_i)$ adalah fungsi kernel (non-parametrik) yang menghitung keserupaan antara data uji $\mathbf{x}_*$ dengan data latih $\mathbf{x}_i$.Persamaan 3.17 merupakan bentuk khusus dari mekansime *soft attention* Persamaan 3.7, dengan kueri $\mathbf{q}$ adalah data uji $\mathbf{x}_*$, kunci $\mathbf{k}_i$ adalah data latih $\mathbf{x}_i$, nilai $\mathbf{v}_i$ adalah y_i. Gambarxx dan Persamaan 3.18 menujukkan contoh kernel regresi dengan pemilihan kernel Gaussian $\mathcal{K}(\mathbf{x}_*, \mathbf{x}_i) = \exp(-\frac{||\mathbf{x}_* - \mathbf{x}_i||^2 \beta}{2})$.

$$y_* = f(\mathbf{x}_*) = \sum_{i=1}^{n} \alpha_i(\mathbf{x}_*, \mathbf{x}_i) y_i \tag{3.18}$$

$$= \sum_{i=1}^{n} \frac{\exp\left(-\frac{||\mathbf{x}_* - \mathbf{x}_i||^2 \beta^2}{2}\right)}{\sum_{j=1}^{n} \exp\left(-\frac{||\mathbf{x}_* - \mathbf{x}_j||^2 \beta^2}{2}\right)} y_i \tag{3.19}$$

3.1.3 *Attention* Parametrik

Mekanisme *attention* yang dilakukan oleh Vaswani et al. (2017) merupakan mekanisme *attention* parametrik. Salah satu alasan penggunaan f_{attn} yang parametrik adalah pemilihan fungsi f_{attn} yang non-parametrik seperti pada Subbab 3.1.2 memiliki kelemahan:

1. Relasi antar vektor kueri $\mathbf{q}$ dan vektor kunci $\mathbf{k}$ harus diketahui sebelumnya untuk

memilih fungsi f_{attn} yang tepat.

2. Prediksi y_* memerlukan seluruh data latih $\mathcal{D}$, $O(|\mathcal{D}|)$ komputasi diperlukan untuk melakukan satu prediksi.

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 3.20

$$f_{attn}(\mathbf{q}\mathbf{W}^q, \mathbf{k}\mathbf{W}^k) = (\mathbf{q}\mathbf{W}^q + \mathbf{k}\mathbf{W}^k)\mathbf{W}^{\text{out}} \in \mathbb{R}, \tag{3.20}$$

$$\text{dengan } \mathbf{W}^q \in \mathbb{R}^{d_q \times d_{\text{attn}}}, \mathbf{W}^k \in \mathbb{R}^{d_k \times d_{\text{attn}}}, \mathbf{W}_{\text{out}} \in \mathbb{R}^{d_{\text{attn}} \times 1}, \tag{3.21}$$

Dengan $\mathbf{W}^q$, $\mathbf{W}^k$, dan $\mathbf{W}^{\text{out}}$ adalah matriks parameter bobot yang akan dicari atau dipelajari selama proses pelatihan. Contoh parametrik *attention* 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 3.22 menunjukkan bagaimana *dot-product attention* dihitung.

$$f_{attn}(\mathbf{q}\mathbf{W}^q, \mathbf{k}\mathbf{W}^k) = (\mathbf{q}\mathbf{W}^q)(\mathbf{k}\mathbf{W}^k)^\top \tag{3.22}$$

$$\text{dengan } \mathbf{W}^q \in \mathbb{R}^{d_q \times d_{\text{attn}}}, \mathbf{W}^k \in \mathbb{R}^{d_k \times d_{\text{attn}}}. \tag{3.23}$$

3.2 Transformer

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.

3.2.1 *Token Embedding*

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 $\sqrt{2}$.

Vektor yang padat (*dense*) dan memiliki representasi token yang baik adalah vektor yang dinginkan. Representasi vektor yang baik diharapkan dapat dipelajari selama proses pelatihan model. 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 3.24 hingga Persamaan 3.26 menunjukkan bagaimana representasi vektor dari barisan suatu token t dihitung.

$$t = (t_{i_1}, t_{i_2}, \ldots, t_{i_L}), \tag{3.24}$$

$$\mathbf{e}_{i_j} = \mathbf{oh}_{i_j}\mathbf{E}_{\mathcal{T}} \in \mathbb{R}^{d_{\text{token}}}, \tag{3.25}$$

$$\text{Embed}(t) = \mathbf{E}_t = \begin{bmatrix} \mathbf{e}_{i_1} \\ \mathbf{e}_{i_2} \\ \vdots \\ \mathbf{e}_{i_L} \end{bmatrix} \in \mathbb{R}^{L \times d_{\text{token}}}. \tag{3.26}$$

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 3.27.

$$f_{attn}(\mathbf{q}\mathbf{W}^q, \mathbf{k}\mathbf{W}^k) = \frac{\mathbf{q}\mathbf{W}^q(\mathbf{k}\mathbf{W}^k)^\top}{\sqrt{d_{attn}}} \in \mathbb{R}, \tag{3.27}$$

$$\text{dengan } \mathbf{W}^q \in \mathbb{R}^{d_q \times d_{\text{attn}}}, \mathbf{W}^k \in \mathbb{R}^{d_k \times d_{\text{attn}}}. \tag{3.28}$$

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 3.29 hingga Persamaan 3.30.

$$\mathbf{q}\mathbf{W}^q \sim \mathcal{N}(0,\sigma^2) \text{ dan } \mathbf{k}\mathbf{W}^k \sim \mathcal{N}(0,\sigma^2). \tag{3.29}$$

$$\text{Var}(\mathbf{q}\mathbf{W}^q(\mathbf{k}\mathbf{W}^k)^\top) = \sum_{i=1}^{d_{attn}} \text{Var}\left((\mathbf{q}\mathbf{W}^q)_i((\mathbf{k}\mathbf{W}^k)_i^\top\right) = \sigma^4 d_{attn}. \tag{3.30}$$

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 kita tidak menghilangkan faktor d_{attn}, *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 3.31 tidak menjadi masalah karena dengan inisialisasi bobot Kaiming dan *layer normalisasi* yang dijelaskan pada Subbab 2.3.5 dan Subbab 3.2.7, nilai $\sigma^2 \approx 1$ sehingga $\sigma^4 \approx \sigma^2 \approx 1$.

$$\text{(scaled dot product attention) Var}\left(\frac{\mathbf{q}\mathbf{W}^q(\mathbf{k}\mathbf{W}^k)^\top}{\sqrt{d_{attn}}}\right) = \frac{\sigma^4 d_{attn}}{d_{attn}} = \sigma^4 \tag{3.31}$$

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 dengan menggunakan operasi matriks. Persamaan 3.32 hingga Persamaan 3.35 menunjukkan bagaimana *scaled dot product attention* dihitung.

$$\text{Tulis Kembali } \mathbf{Q} = \begin{bmatrix} \mathbf{q}_1 \\ \mathbf{q}_2 \\ \vdots \\ \mathbf{q}_m \end{bmatrix} \in \mathbb{R}^{m \times d_q}, \tag{3.32}$$

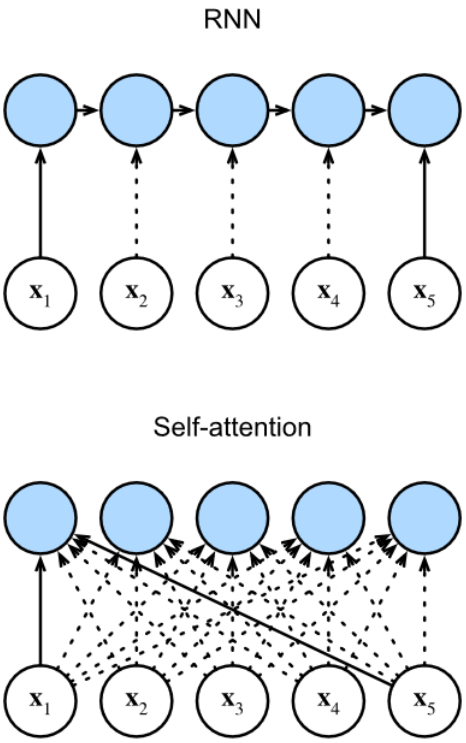
$$\mathbf{K} = \begin{bmatrix} \mathbf{k}_1 \\ \mathbf{k}_2 \\ \vdots \\ \mathbf{k}_n \end{bmatrix} \in \mathbb{R}^{n \times d_k}, \tag{3.33}$$

$$\text{dan } \mathbf{V} = \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_n \end{bmatrix} \in \mathbb{R}^{n \times d_v}, \tag{3.34}$$

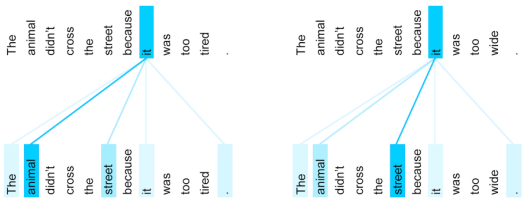
$$\text{Attention}(\mathbf{Q}\mathbf{W}^q, \mathbf{K}\mathbf{W}^k, \mathbf{V}) = \text{Softmax}(\frac{\mathbf{Q}\mathbf{W}^q(\mathbf{K}\mathbf{W}^k)^\top}{\sqrt{d_{attn}}})\mathbf{V} \in \mathbb{R}^{m \times d_v}, \tag{3.35}$$

$$\text{dengan } \mathbf{W}^q \in \mathbb{R}^{d_q \times d_{\text{attn}}}, \mathbf{W}^k \in \mathbb{R}^{d_k \times d_{\text{attn}}}. \tag{3.36}$$

3.2.3 *Self-Attention*



Gambar 3.1: 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 3.2: 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 pada gambarxx 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 3.1 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 **q**, vektor kunci **k**, dan vektor nilai **v** adalah vektor yang sama, yaitu *embedding* token **E** yang dijelaskan pada Subbab 3.2.1. Persamaan 3.37 hingga Persamaan 3.38 menunjukkan bagaimana *self-attention* dihitung.

$$\text{Self-Attention}(\mathbf{E}) = \text{Attention}(\mathbf{E}\mathbf{W}^q, \mathbf{E}\mathbf{W}^k, \mathbf{E}\mathbf{W}^v) \tag{3.37}$$

$$= \text{Softmax}(\frac{\mathbf{E}\mathbf{W}^q(\mathbf{E}\mathbf{W}^k)^\top}{\sqrt{d_{attn}}})(\mathbf{E}\mathbf{W}^v) \in \mathbb{R}^{L \times d_{\text{attn}}} \tag{3.38}$$

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 3.2 adalah contoh dari *self-attention* yang menghasilkan representasi vektor kontekstual pada token *it*. Pada Gambar 3.2 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 3.2 memiliki bobot atensi yang tinggi terhadap token *street*.

3.2.4 *Multi-Head Self-Attention*

Multi-Head Attention adalah arsiktetur *deep learning* yang melakukan mekanisme *attention* sebanyak

3.2.5 *Positional Encoding*

3.2.6 *Position-wise Feed-Forward Network*

3.2.7 Koneksi Residual dan *Layer Normalization*

3.2.8 Transformer Encoder

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.*

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`

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.

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.