Fine-Tuning SBERT dengan Multiple Negative Ranking Loss untuk Pencarian Semantik

Adril Putra Merin
Institut Teknologi Bandung
Bekasi, Indonesia
13522068@std.stei.itb.ac.id

Kristo Anugrah
Institut Teknologi Bandung
Surabaya, Indonesia
13522024@std.stei.itb.ac.id

Robert Benyamin
Universitas Indonesia
Bekasi, Indonesia
robert.benyamin@ui.ac.id

***Abstrak*—Salah satu tantangan mesin pencari produk pada *e-commerce* adalah memberikan hasil pencarian yang cepat dan tepat. Hal ini memerlukan suatu pendekatan yang mampu memahami makna dan konteks dari kueri pengguna. Makalah ini mengusulkan penggunaan *Fine-Tuning* SBERT (*Sentence*-BERT) dengan *Multiple Negative Ranking* (MNR) *Loss* untuk meningkatkan performa pencarian semantik. SBERT yang telah terbukti efektif dalam berbagai tugas pencarian, dikombinasikan dengan MNR *Loss*, diharapkan dapat memaksimalkan kualitas *embedding* teks dan relevansi hasil pencarian. *Training* dan evaluasi dilakukan menggunakan *dataset* WANDS untuk mengukur efektivitas pendekatan ini. Hasil penelitian menunjukkan peningkatan akurasi dan relevansi hasil pencarian, memberikan kontribusi signifikan dalam pengembangan teknologi pencarian semantik di *e-commerce*.**

***Kata kunci*—Pencarian Semantik, Pencarian Produk, SBERT, *Multiple Negative Ranking Loss*, *e-commerce*, *text embedding*.**

I. Pendahuluan

Dalam era digital yang terus berkembang, *e-commerce* telah menjadi salah satu sektor industri yang paling dinamis dan kompetitif. Salah satu tantangan utama dalam *e-commerce* adalah menyediakan mesin pencari yang efektif untuk membantu pengguna menemukan produk yang mereka cari dengan cepat dan akurat. Pencarian semantik, yang mampu memahami makna dan konteks dari kueri pengguna, menjadi kunci dalam meningkatkan pengalaman pencarian.

Dalam upaya untuk mengembangkan mesin pencari produk yang lebih efisien dan akurat, kami mengusulkan penggunaan *Fine-Tuning* SBERT (*Sentence*-BERT) dengan *Multiple Negative Ranking* (MNR) *Loss*. SBERT adalah model embedding teks yang telah terbukti efektif dalam berbagai tugas pencarian semantik, termasuk pencarian dokumen dan pencocokan teks. Dengan memanfaatkan MNR *Loss*, model ini dapat meningkatkan kualitas *embedding* teks dengan memaksimalkan jarak antara *embedding* teks relevan dan tidak relevan. Hal ini diharapkan dapat mengurangi kesalahan dalam pemeringkatan hasil pencarian, sehingga hasil pencarian menjadi lebih relevan bagi pengguna.

Tujuan dari makalah ini adalah untuk mengevaluasi efektivitas *Fine-Tuning* SBERT dengan MNR *Loss* dalam konteks pencarian produk *e-commerce*. Kami akan menggunakan *dataset* dari WANDS sebagai sumber data untuk melatih dan menguji model yang diusulkan. Melalui pendekatan ini, diharapkan mesin pencari yang dikembangkan dapat memberikan hasil yang lebih relevan dan memuaskan bagi pengguna.

Makalah ini diharapkan dapat memberikan kontribusi signifikan dalam pengembangan teknologi pencarian semantik, khususnya dalam konteks *e-commerce*. Dengan memahami dan mengimplementasikan teknik-teknik terbaru dalam pembelajaran mesin, kami berharap dapat memberikan solusi yang inovatif dan efektif untuk tantangan pencarian produk yang ada saat ini.

II. Dasar Teori

A. Transformer

Transformers telah merevolusi bidang pemrosesan bahasa alami (NLP) dengan memperkenalkan arsitektur baru yang secara signifikan meningkatkan kinerja model bahasa.

Inovasi utama *transformers* terletak pada penggunaan mekanisme *self-attention*. *Self-attention* memungkinkan model untuk menimbang pentingnya bagian-bagian yang berbeda dari data *input*, sehingga memungkinkan untuk menangkap *long-range dependencies* dan hubungan kontekstual secara lebih efektif daripada model

tradisional [1]. Setiap token input memperhatikan setiap token lainnya, menghasilkan jumlah tertimbang yang merepresentasikan relevansi kontekstualnya. Mekanisme ini memungkinkan *transformers* untuk memahami hubungan dalam seluruh rangkaian data secara bersamaan. *Transformers* memproses beberapa token secara bersamaan, memungkinkan pemrosesan paralel. Pendekatan ini menghasilkan peningkatan kecepatan yang signifikan dan memungkinkan penskalaan ke *datasets* yang lebih besar.

Selain *self-attention*, *transformers* menggabungkan *positional encoding* untuk mempertahankan urutan-urutan *input*. Karena mekanisme *self-attention* memperlakukan *input* sebagai satu *set* tanpa urutan yang melekat, *positional encoding* menyediakan cara untuk menyuntikkan informasi tentang posisi relatif atau absolut token dalam urutan. Hal ini sangat penting untuk tugas-tugas yang bergantung pada sifat sekuensial data, seperti penerjemahan bahasa dan pembuatan teks.

Dalam NLP, *transformers* telah menjadi tulang punggung model-model mutakhir seperti BERT, GPT, dan T5. Model-model ini unggul dalam tugas-tugas seperti penerjemahan bahasa, peringkasan teks, dan analisis sentimen.

B. BERT (Bidirectional Encoder Representations from Transformers)

BERT [2] adalah model pemrosesan bahasa yang dibangun di atas *neural network* yang berfokus pada pengenalan hubungan kata-ke-kata atau hubungan kalimat-ke-kalimat, dengan menggunakan *semi-supervised learning* dan model representasi bahasa. Ini adalah model berbasis *transformer* dua arah yang secara bersamaan menyesuaikan *transformer* kiri-ke-kanan dan *kanan-ke-kiri*. Pada fase *pretrained*, model ini menggunakan *unsupervised prediction task* yang terdiri dari *Masked Language Model* (MLM) [3]. Kemudian, model menerapkan fase *fine-tuning* pada parameter model, untuk *downstream task*, untuk mencapai kecocokan terbaik.

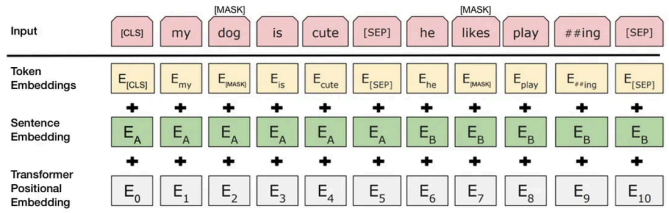
1) Transformer dua arah: Bahasa alami adalah bentuk komunikasi yang telah berevolusi dalam kehidupan manusia, dan sebuah kalimat atau kata biasanya perlu dikontekstualisasikan untuk merefleksikan maknanya. Ini berarti bahwa komputer tidak dapat begitu saja menafsirkan kata-kata dari atas (*sequential parsing*) atau bawah (*inverse parsing*), tetapi membutuhkan pendekatan kontekstual. *Transformator* dua arah dalam BERT merealisasikan ide ini [4].

2) Masked Language Model: Mungkin ada perulangan ketika menggunakan interpretasi dua arah, yang menyebabkan kesalahpahaman tentang pemahaman kata tentang "dirinya sendiri". BERT menggunakan MLM [10] untuk mengatasi kesalahpahaman ini. Model MLM secara acak menutupi kata-kata dalam kalimat input. Sebagai contoh, kalimat berikut ini: "Claire pergi ke Eropa dengan tas kesayangannya." [5]

- Kemungkinan 80% untuk mengganti dengan token "[MASK]" --> "Claire pergi ke [MASK] dengan [MASK] kesayangannya."
- 10% kemungkinan mengganti satu kata dengan kata yang diambil secara acak --> "Claire pergi ke jeruk dengan bulan kesayangannya."
- 10% kemungkinan tidak ada penggantian -> "Claire pergi ke Eropa dengan tas kesayangannya."

3) *Embeddings*: Penyematan kata pada model BERT bukanlah penyandian kata yang sederhana, melainkan sebuah kombinasi penyematan yang mengandung tiga lapisan makna. Lapisan pertama dari penyematan adalah penyandian kata itu sendiri, yang dilakukan dengan menginisialisasi BERT dengan daftar kata masukan eksternal yang berisi semua kata bahasa alami. Lapisan kedua dari penyematan didasarkan pada informasi posisi dari kata yang akan dikodekan. Untuk merefleksikan informasi posisi kata dalam kalimat, BERT akan melakukan penyematan posisi untuk setiap kata dalam setiap kalimat. Lapisan ketiga dari penyematan adalah penyandian tingkat kalimat. Untuk merefleksikan independensi kalimat (BERT menyebutnya sebagai *segment embedding*), BERT menggunakan dua penyambungan kalimat untuk membangun penyandian. Setelah ketiga lapisan *embedding* selesai, BERT akan menggabungkan ketiga penyematan tersebut untuk menentukan vektor kata [6].

Gambar 1 menunjukkan representasi input BERT. Penyematan input merupakan gabungan dari penyematan token, penyematan segmentasi, dan penyematan posisi.



Gambar 1: Representasi *input* BERT

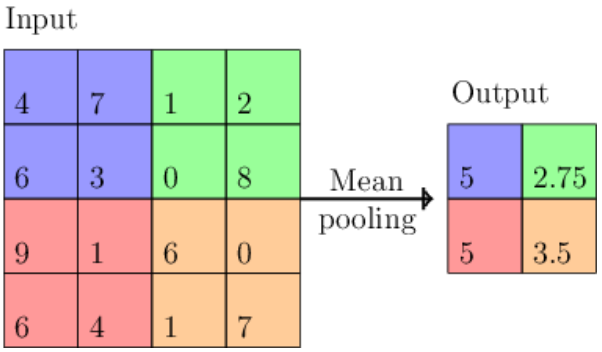
4) *Fine-Tuning* BERT: Parameter model bahasa ditentukan pada saat pelatihan, tetapi tidaklah ilmiah untuk menggunakan himpunan parameter yang sama untuk *downstream tasks* yang berbeda, dan akan memakan waktu untuk melatih ulang. BERT adalah model bahasa *pre-trained* yang mengadopsi *fine-tuning* untuk sedikit menyesuaikan model terlatih untuk *downstream tasks* NLP yang berbeda untuk mencapai hasil pencocokan model terbaik [7].

C. Mean Pooling Layer

Pooling layer adalah komponen penting dalam *neural network*, khususnya *convolutional neural network* (CNN), yang dirancang untuk mengurangi dimensi spasial dari peta fitur dan dengan demikian mengurangi beban komputasi dengan tetap mempertahankan informasi penting. *Pooling layer* mencapai hal ini dengan meringkas fitur-fitur di dalam wilayah tertentu. Terdapat berbagai jenis *pooling layer*, diantaranya adalah *max pooling*, *mean pooling*, dan *global pooling*.

Mean pooling, juga dikenal sebagai *average pooling*, adalah jenis *pooling layer* tertentu yang menghitung nilai rata-rata elemen dalam setiap wilayah *pooling*. Dalam *mean pooling*, *input* peta fitur dibagi menjadi beberapa wilayah yang tidak tumpang tindih, dan nilai rata-rata piksel di setiap wilayah dihitung untuk membentuk *output* peta fitur. Proses ini mengurangi dimensi *input*, yang pada hasilnya mengurangi jumlah parameter dan komputasi dalam jaringan, dan membantu mengurangi *overfitting* dengan menyediakan bentuk invariansi terjemahan [8].

Mean pooling layer beroperasi dengan menggeser jendela (dengan ukuran yang sudah ditentukan sebelumnya) melintasi *input* peta fitur. Untuk setiap posisi jendela, rata-rata nilai di dalam jendela dihitung dan ditempatkan pada posisi yang sesuai pada *output* peta fitur. Metode ini memastikan bahwa jaringan mempertahankan keseluruhan informasi dan pola yang ada dalam *input* sekaligus mengurangi kompleksitasnya. *Mean pooling* sangat berguna dalam tugas-tugas di mana penting untuk mempertahankan latar belakang informasi dan tren umum dalam data, berlawanan dengan *max pooling*, yang berfokus pada menangkap fitur yang paling menonjol.

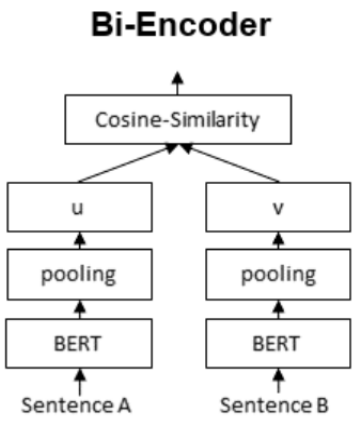


Gambar 2: Ilustrasi dari operasi *Mean Pooling*

D. Bi-Encoder Architecture

Arsitektur *Bi-Encoder* adalah *framework neural network* yang biasa digunakan dalam tugas pemrosesan bahasa alami untuk menangani masalah pengambilan berskala besar secara efisien. Tidak seperti model *Cross-Encoder*, yang memproses urutan *input* berpasangan secara bersama-sama, *Bi-Encoder* menyandikan urutan *input* secara independen ke dalam vektor berukuran tetap. Vektor-vektor ini kemudian dibandingkan menggunakan ukuran similaritas seperti *cosine similarity* atau *dot product* untuk menentukan relevansi.

Arsitektur *Bi-Encoder* terdiri dari dua *encoder* identik, masing-masing bertanggung jawab untuk mengubah urutan *input* menjadi representasi vektor yang padat. Biasanya, *encoder* ini didasarkan pada model *transformer* seperti BERT. Selama pelatihan, model ini belajar untuk memetakan urutan yang seharusnya serupa (misalnya, pasangan pertanyaan-jawaban) ke titik-titik yang dekat dalam ruang vektor sambil menjauhkan pasangan yang tidak serupa [9].



Gambar 3: Arsitektur *Bi-Encoder*

Keuntungan utama dari *Bi-Encoder* terletak pada efisiensi komputasi mereka selama inferensi. Karena urutan *input* dikodekan secara independen, representasi vektor dari korpus yang besar dapat dikomputasi dan disimpan. Ketika sebuah kueri dimasukkan, kueri tersebut dikodekan sekali, dan representasi vektornya dibandingkan dengan vektor yang telah dikomputasi sebelumnya, sehingga memungkinkan pengambilan yang cepat.

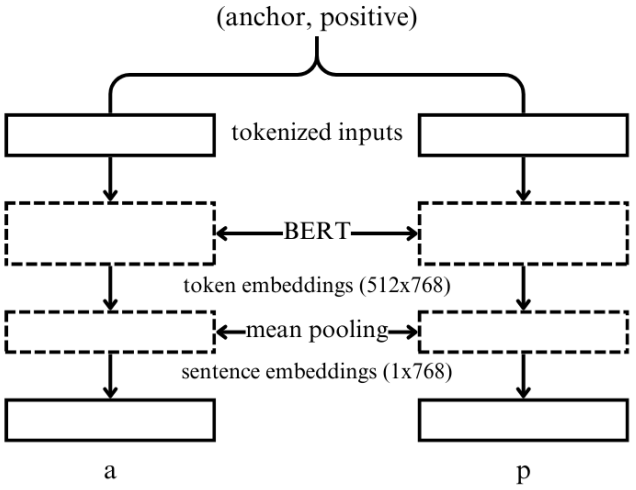
Arsitektur ini sangat efektif untuk tugas-tugas seperti similaritas kalimat, pencarian informasi, dan pemeringkatan dokumen berskala besar, di mana kemampuan untuk melakukan prakomputasi penyematan secara signifikan mengurangi *overhead* komputasi dibandingkan dengan pendekatan pengkodean bersama.

E. Multiple Negative Ranking Loss

Multiple Negative Ranking Loss (MNRL) adalah sebuah *loss function* yang dirancang untuk meningkatkan efisiensi pelatihan dalam kerangka kerja pembelajaran kontrastif, khususnya dalam konteks tugas pengambilan dan pembelajaran representasi berskala besar. Metode ini digunakan untuk membedakan antara item yang relevan (positif) dan tidak relevan (negatif) relatif terhadap suatu titik acuan, yang secara efektif meningkatkan kemampuan model untuk membedakan kesamaan dan perbedaan dengan rinci.

Dalam konteks *neural network* dan model berbasis *transformer* seperti BERT, MNRL digunakan untuk meningkatkan kinerja tugas-tugas seperti pencarian semantik, pencarian jawaban, dan sistem rekomendasi. Konsep intinya berkisar pada tiga komponen utama: *anchor*, sampel positif, dan sampel negatif. *Anchor* adalah input awal atau kueri yang digunakan model untuk mencari informasi yang relevan. Sampel positif adalah kecocokan yang benar atau item yang relevan yang sesuai dengan anchor, sedangkan sampel negatif adalah item yang tidak relevan atau tidak cocok.

MNRL bekerja dengan mengkodekan sampel *anchor*, positif, dan negatif ke dalam representasi vektor menggunakan model *neural network*, seperti BERT. Tujuannya adalah untuk meminimalkan jarak antara *anchor* dan representasi positif sambil memaksimalkan jarak antara *anchor* dan representasi negatif. Tidak seperti *triplet loss* tradisional, yang menggunakan satu sampel negatif, MNRL memanfaatkan beberapa sampel negatif secara bersamaan, memberikan sinyal pelatihan yang lebih baik dan mempercepat konvergensi [10].



Gambar 4: Ilustrasi dari MNR *Loss* menggunakan BERT

Pendekatan ini sangat efektif dalam situasi di mana secara komputasi tidak memungkinkan untuk melatih dengan semua sampel negatif yang tersedia. Dengan menggunakan beberapa sampel negatif, model belajar untuk menggeneralisasi dengan lebih baik, sehingga meningkatkan kemampuannya untuk mengambil item yang relevan pada praktiknya. Dengan demikian, MNRL berkontribusi pada sistem pengambilan yang lebih akurat dan efisien, menjadikannya alat yang berharga dalam pemrosesan bahasa alami dan banyak hal lain.

F. Evaluation Metrics

Metrik evaluasi digunakan untuk menilai kinerja sistem pencarian dan rekomendasi. Metrik ini memberikan ukuran kuantitatif yang membantu dalam membandingkan model yang berbeda dan memilih model dengan kinerja terbaik untuk diterapkan. Di antara berbagai metrik yang digunakan, *Mean Reciprocal Rank* (MRR) dan *Normalized Discounted Cumulative Gain* (NDCG) adalah yang paling sering digunakan.

MRR adalah metrik yang digunakan untuk mengevaluasi keefektifan algoritma pencarian berdasarkan posisi peringkat dari hasil pertama yang relevan. Hal ini sangat berguna ketika pengguna terutama tertarik pada jawaban pertama yang benar.

$$MRR = \frac{1}{|U_{all}|} \sum_{u=1}^{|U_{all}|} RR(u)$$

$$RR(u) = \sum_{i \le L} \frac{relevance_i}{rank_i}$$

di mana RR(u) adalah *reciprocal rank* dari pengguna u, yang didefinisikan sebagai jumlah skor relevansi dari L item teratas yang dibobot oleh *reciprocal rank*

MRR bekerja dengan merata-ratakan resiprokal dari peringkat dokumen pertama yang relevan di semua kueri. Metrik ini menekankan pentingnya mengembalikan hasil yang relevan sebagai item dengan peringkat teratas, yang sangat penting dalam banyak aplikasi dunia nyata seperti pencarian *web* dan sistem penjawab pertanyaan [11].

NDCG adalah metrik yang mengevaluasi kualitas hasil pemeringkatan dengan mempertimbangkan posisi dokumen yang relevan. NDCG memberikan nilai yang lebih tinggi pada dokumen yang relevan yang muncul lebih awal dalam daftar hasil, dengan demikian memperhitungkan posisi dokumen yang relevan [12].

Untuk memahami NDCG, penting untuk terlebih dahulu memahami *Discounted Cumulative Gain* (DCG). DCG mengukur kegunaan, atau keuntungan, dari sebuah dokumen berdasarkan posisinya dalam daftar hasil. Keuntungan didiskon secara logaritmik sebanding dengan posisi hasil untuk menekankan pentingnya dokumen yang relevan muncul lebih awal dalam daftar. Rumus untuk DCG pada posisi peringkat k tertentu adalah:

$$DCG(k) = \sum_{i=1}^{k} \frac{G_i}{log_2(i+1)}$$

Untuk memberikan perbandingan yang berarti, DCG sering dinormalisasi dengan *Ideal* DCG (IDCG), yang merepresentasikan DCG maksimum yang mungkin untuk sekumpulan dokumen. IDCG dihitung dengan mengurutkan dokumen dalam urutan terbaik berdasarkan relevansi, kemudian menerapkan rumus DCG.

$$IDCG(k) = \sum_{i=1}^{|I(k)|} \frac{G_i}{log_2(i+1)}$$

NDCG kemudian dihitung dengan menormalkan DCG oleh IDCG.

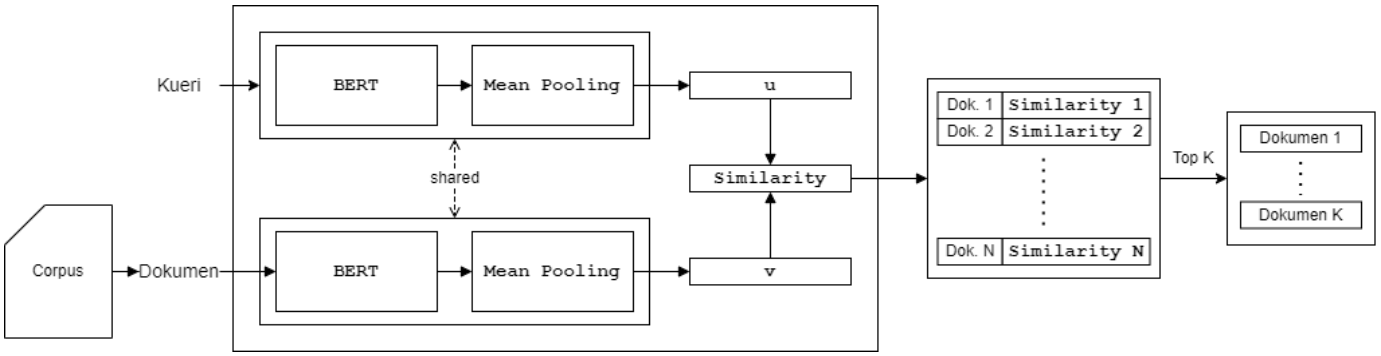
$$NDCG(k) = \frac{DCG(k)}{IDCG(k)}$$

NDCG banyak digunakan dalam sistem rekomendasi dan mesin pencari karena memberikan evaluasi yang komprehensif dengan mempertimbangkan relevansi dan posisi peringkat hasil. Hal ini sangat efektif dalam skenario di mana relevansi hasil menurun secara eksponensial terhadap posisi.

III. Pembersihan Data

Pada makalah ini, kami menggunakan *dataset* WANDS yang dipublikasikan untuk jurnal ECIR 2022. *Dataset* ini terdiri dari 42.994 kandidat produk, 480 kueri, dan 233.448 pasangan kueri dan produk yang telah diberi label dengan salah satu dari tiga nilai: ‘Exact’, ‘Partial’, atau ‘Irrelevant’. *Dataset* ini tersedia dalam tiga berkas csv dengan rincian sebagai berikut:

1. product.csv - menyimpan data kandidat produk dengan kolom-kolom berikut:
    - *product_id*
    - *product_name*
    - *product_class*
    - *category_hierarchy*
    - *product_description*
    - *product_features*
    - *rating_count*
    - *average_rating*
    - *review_count*
2. query.csv - menyimpan data kueri dengan kolom-kolom berikut:



Gambar 5: Ilustrasi kerangka solusi untuk pencarian semantik

- *query_id*
- *query*
- *query_class*

3. label.csv - menyimpan pasangan kueri dan produk yang telah diberi label.
    - *id*
    - *query_id*
    - *product_id*
    - *label*

Proses *text preprocessing* adalah langkah krusial dalam metode *Natural Language Processing* (NLP) untuk mengubah teks kueri dan produk yang tidak terstruktur menjadi format yang siap diolah. Langkah-langkah *text preprocessing* yang kami lakukan meliputi:

1. Penanganan Nilai NaN: Setiap kolom dengan tipe data *string* yang berisi nilai NaN digantikan dengan string kosong untuk menghindari masalah selama pemrosesan data.
2. Konversi ke *Lowercase*: Semua teks dalam kolom dengan tipe data *string* diubah menjadi huruf kecil untuk menjaga konsistensi dalam pemrosesan. Hal ini memungkinkan model untuk memperlakukan kata "Good" dan "good" sebagai kata yang sama sehingga dapat meningkatkan akurasi hasil.
3. Pengubahan Format Teks: Untuk kolom “category_hierarchy” dalam data *product*, karakter '/' diganti dengan ',' untuk meningkatkan pemahaman teks. Misalnya, teks "furniture / bedroom furniture / beds & headboards / beds / twin beds" diubah menjadi "furniture, bedroom furniture, beds & headboards, beds, twin beds".
4. Normalisasi Singkatan dan Karakter: Singkatan umum dalam bahasa Inggris dihilangkan, dan beberapa karakter atau kata yang sering digunakan dalam bahasa Inggris diganti dengan bentuk yang lebih mudah dipahami. Sebagai contoh, tanda petik dua (“) diubah menjadi kata "inch".
5. Penghapusan Tanda Baca dan Normalisasi Karakter: Semua tanda baca dihilangkan kecuali titik (.) dan koma (,). Karakter "&" diganti dengan kata "and" untuk memastikan format teks yang konsisten.

IV. Kerangka Solusi

Sebuah kueri pertama-tama akan diubah ke dalam *embedding* 768 dimensi menggunakan model SBERT. Hasil *embedding* tersebut kemudian akan dihitung nilai kemiripannya dengan *embedding* setiap dokumen pada korpus. Dari proses ini akan diperoleh N pasangan dokumen dan nilai kemiripan. Dari sejumlah nilai kemiripan tersebut, akan diambil K-buah dokumen dengan nilai kemiripan terbesar. Dokumen-dokumen tersebut merepresentasikan dokumen dengan makna semantik paling dekat dengan kueri yang diberikan. Proses ini memakan waktu linear, yang artinya proses akan berjalan secara efisien dalam *hardware* yang wajar. Proses *retrieval* dapat dipercepat dengan menyimpan *embedding* setiap dokumen ke dalam sebuah *database*. Proses yang cepat turut menjadi prioritas mengingat proses *retrieval* yang cepat memastikan pengalaman pengguna yang baik. Ilustrasi *framework* untuk sebuah kueri secara lengkap dapat dilihat pada Gambar 5.

A. Model Pre-trained SBERT

Dalam menyelesaikan masalah *semantic search*, digunakan *pretrained model* BERT yang didesain untuk *semantic search*. Model tersebut telah di-*train* menggunakan *dataset* MS MARCO. Model ini akan mentransformasi sebuah kalimat menjadi sebuah *embedding* 768 dimensi, yang kemudian dapat dikalkulasi kemiripannya dengan *dot product* atau *cosine similarity*.

Proses *training* model BERT terdiri dari 2 tahap, yaitu *pre-training* dan *fine-tuning*. Tahap pertama adalah tahap *pre-training* pada *corpus* ukuran besar, yang kemudian diikuti oleh *supervised fine-tuning* untuk *task* NLP *downstream*.

1) Fase *pre-training*: proses *pre-training* dilakukan menggunakan *corpus* berukuran besar. Wikipedia dan BooksCorpus digunakan untuk melakukan *training* pada model BERT orisinal, dengan *corpus* Wikipedia memiliki 13GB data teks dengan 300 juta kata; BooksCorpus memiliki 15GB data teks dengan 400 juta kata.

Model BERT orisinal ini kemudian di-*train* lebih lanjut dengan *dataset* MS MARCO. Dataset tersebut terdiri dari 500 ribu pasangan (kueri, jawaban) yang relevan. Proses *training* lebih lanjut ini bertujuan untuk meningkatkan akurasi model dalam *task* pencarian semantik. Model BERT yang telah di-*train* lebih lanjut ini adalah SBERT (Sentence-BERT).

2) Fase *fine-tuning*: Model SBERT *pre-trained* yang telah dibahas di-*import* dan dilakukan tahap-tahap berikut untuk proses *fine-tuning*:

- **Seleksi *dataset*.** Dalam *dataset* WANDS, setiap produk memiliki fitur nama produk, kelas produk, deskripsi produk, banyak *review,* rata-rata *rating*, dan banyak *rating*. Dalam *task semantic search* ini, fitur yang digunakan adalah fitur nama produk, kelas produk, dan deskripsi produk. Hal ini karena fitur seperti *rating* dan *review* tidak relevan dalam *task semantic search*. Selain itu, karena *loss function* yang dipakai adalah *Multiple Negative Ranking Loss*, data yang akan digunakan adalah data label yang bernilai "Exact".
- **Pembagian *dataset*.** *Dataset* WANDS yang digunakan dipisahkan menjadi *training set* dan *validation set* dengan rasio 4:1. Perhatikan bahwa proses pembagian ini dilakukan berdasarkan kueri, bukan berdasarkan pasangan (kueri, dokumen). Ini artinya pasangan (kueri, dokumen) masuk ke dalam *training set* jika dan hanya jika kueri masuk ke dalam pembagian *training set*.
- **Seleksi parameter.** *Batch size* yang digunakan adalah 96, dengan panjang maksimal token adalah 512. Untuk token dengan panjang lebih 512, dilakukan *truncation* dengan efek korupsi minimal.
- **Definisi *hyperparameter*.** Digunakan *optimizer* AdamW dengan *learning rate* sebesar 2e-5. *Loss function* yang digunakan adalah *Multiple Negative Ranking Loss*. Fungsi *loss* ini memerlukan data *training* dalam bentuk (kueri, dokumen) yang relevan.
- **Komparasi hasil dan evaluasi.** Dalam rangka mendesain model yang cocok, telah dibuat dua model. Model pertama di-*fine-tune* hanya menggunakan data nama produk, sedangkan model kedua telah di-*fine-tune* menggunakan nama dan deskripsi produk. Hal ini dilakukan untuk melihat pengaruh deskripsi produk terhadap performa model.

V. Hasil Simulasi

A. Dataset

1) Dataset pre-processing: setelah proses *data-cleaning* pada bagian III dilakukan, data duplikat dihilangkan supaya data yang dipakai saat *training* unik. Selain itu, dilakukan proses *shuffling* pada data.

B. Parameter Evaluasi

Parameter evaluasi yang digunakan adalah sebagai berikut:

1) Accuracy

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

dimana TP melambangkan *true-positive*, TN melambangkan *true-negative*, FP melambangkan *false-positive,* dan FN melambangkan *false-negative.*

2) Precision

$$Precision = \frac{TP}{TP+FP}$$

dimana TP melambangkan *true-positive*, FP melambangkan *false-positive*.

	Accuracy @3	Precision @3	NDCG @3	MRR @3	Accuracy @5	Precision @5	NDCG @5	MRR @5	MAP @100
Pre-trained model	82,8%	60,0%	67,9%	76,5%	86,8%	54,7%	67,4%	77,3%	54,0%
Model dilatih dengan nama & deskripsi	89,4%	70,6%	77,9%	85,9%	92,1%	63,1%	76,1%	86,4%	68,8%
Model dilatih dengan nama produk	**96,0%**	**77,6%**	**85,3%**	**90,3%**	**96,0%**	**71,0%**	**84,5%**	**90,3%**	**76,6%**

TABEL I
HASIL EVALUASI MODEL

3) *NDCG*

$$NDCG(k) = \frac{DCG(k)}{IDCG(k)}$$

dengan notasi yang telah dijelaskan pada bagian II.

4) *MRR*

$$MRR = \frac{1}{|U_{all}|} \sum_{u=1}^{|U_{all}|} RR(u)$$

dengan notasi yang telah dijelaskan pada bagian II.

5) *MAP (Mean Average Precision)*

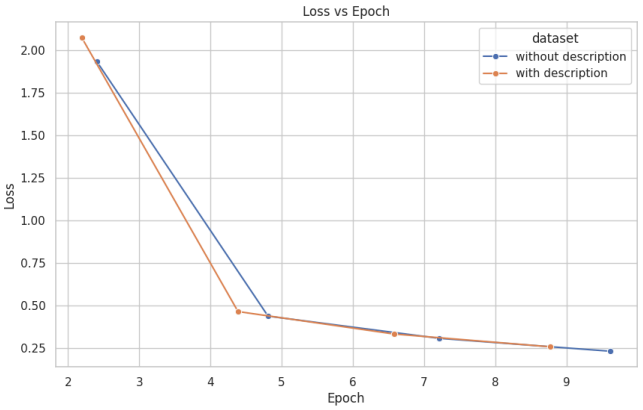
$$MAP = \frac{1}{n} \sum_{k=1}^{n} AP_k$$

dengan n banyak kelas, AP_k melambangkan *average precision* pada kelas k.

C. *Analisis Hasil*

Tabel 1 menunjukkan hasil evaluasi untuk model *pre-trained*, model yang di-*fine-tune* dengan nama produk, serta model yang di-*fine-tune* dengan nama dan deskripsi. Kedua model di-*train* dengan 10 *epochs*. Dapat dilihat bahwa model yang di-*fine-tune* tanpa data deskripsi produk memiliki performa yang lebih baik dalam seluruh metrik evaluasi.

Nilai *loss function* untuk kedua model selama 10 *epoch* dapat dilihat pada Gambar 6. Dapat dilihat bahwa bentuk grafik *loss function* dari kedua model sangat mirip dan menuju 0, yang menandakan proses *fine-tuning* bekerja secara seharusnya. Namun dapat diperhatikan pula bahwa meskipun kedua grafik sangat mirip, hasil evaluasi lebih baik pada model yang dilatih tanpa deskripsi. Hal ini dapat disebabkan karena deskripsi produk mengandung informasi yang tidak relevan, seperti promosi.



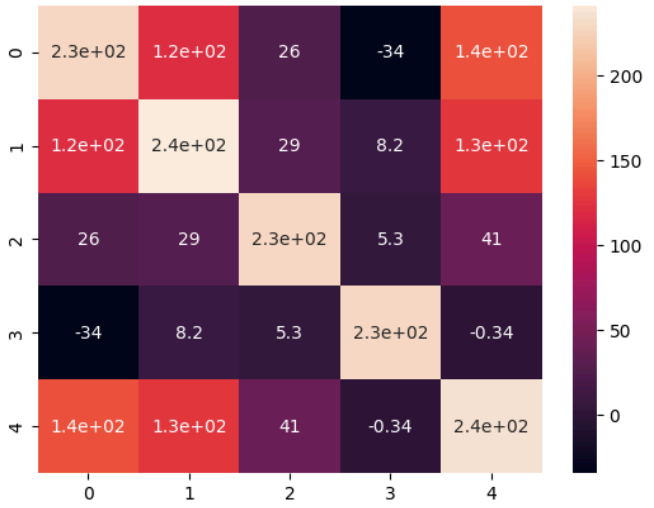
Gambar 6: Grafik *loss* terhadap *epoch* untuk kedua model

Untuk melihat performa model, akan dilihat nilai kemiripan untuk beberapa pasangan kalimat. Akan dibuat 5 kalimat sebagai demonstrasi performa model. Kalimat yang dipakai dapat dilihat pada Gambar 7.

```
sentences = [
  'running shoes',
  'nike air jordan black',
  'spatula stainless steel anti stick',
  'ceiling lamp white',
  'puma shoe 41 white',
]
```

Gambar 7: Kalimat yang dipakai untuk mengetes performa

Untuk setiap pasang kalimat, akan dihitung nilai kemiripannya. Hasil nilai kemiripan dapat dilihat pada Gambar 8.



Gambar 8: Heatmap kalimat dari gambar 7. Semakin tinggi nilai, semakin mirip pasangan kalimat

Dapat dilihat bahwa untuk pasangan kalimat yang memiliki makna yang mirip, nilai kemiripan sangat tinggi. Dapat dilihat bahwa pasangan kalimat yang memiliki nilai tinggi adalah pasangan indeks (0,1), (0,4), dan (1,4). Ketiga pasangan ini memiliki makna sepatu, yang menandakan bahwa model dapat meng-*encode* makna semantik dengan baik. Perlu diperhatikan juga bahwa meskipun indeks 0 dan 1 tidak memiliki kata yang sama, model tetap dapat menyimpulkan makna yang sama.

VI. Kesimpulan

Dalam makalah ini, digunakan model SBERT untuk memenuhi *task* pencarian semantik dalam konteks *e-commerce*. *Framework* yang dipakai akan melakukan *encoding* untuk setiap dokumen dan menghitung nilai kemiripan dengan *embedding* kueri. Data *embedding* setiap dokumen dapat disimpan ke dalam sebuah *database* guna mempercepat proses *retrieval*. Proses *encoding* dari sebuah kalimat menjadi *embedding* menggunakan *pre-trained* model SBERT yang telah di-*fine-tune* dengan data nama produk. Model telah dievaluasi dan menunjukkan hasil yang relatif baik.

Selain itu, telah dibandingkan dua model yang di-*fine-tune* dengan data yang berbeda. Dari hasil evaluasi, dilihat bahwa model yang dilatih dengan nama produk saja memiliki performa yang lebih baik dibandingkan model yang juga dilatih dengan deskripsi produk. Hal ini dapat disebabkan karena deskripsi produk mengandung informasi yang tidak relevan, seperti promosi.

Ucapan Terima Kasih

Penulis mengucapkan terima kasih kepada:

1) Tuhan Yang Maha Esa,
2) orang tua penulis,
3) teman-teman penulis, dan
4) pihak-pihak lain yang telah mendukung pembuatan makalah ini yang tidak dapat disebutkan satu per satu.

Daftar Pustaka

[1] Q. Luo, W. Zeng, M. Chen, G. Peng, X. Yuan, and Q. Yin, "Self-Attention and Transformers: Driving the Evolution of Large Language Models," Jul. 2023, doi: https://doi.org/10.1109/iceict57916.2023.10245906.

[2] Q. Luo, W. Zeng, M. Chen, G. Peng, X. Yuan, and Q. Yin, "Self-Attention and Transformers: Driving the Evolution of Large Language Models," Jul. 2023, doi: https://doi.org/10.1109/iceict57916.2023.10245906.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," Advances in neural information processing systems, vol. 30, 2017.

[4] Y. Guo, Hanane Lamaazi, and Rabeb Mizouni, "Smart Edge-based Fake News Detection using Pre-trained BERT Model," Oct. 2022, doi: https://doi.org/10.1109/wimob55322.2022.9941689.

[5] Rani Horev, "BERT Explained: State of the art language model for NLP," Medium, Nov. 10, 2018. https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270

[6] J. Yadav, D. Kumar, and D. Chauhan, "Cyberbullying Detection using Pre-Trained BERT Model," 2020

International Conference on Electronics and Sustainable Communication Systems (ICESC), Jul. 2020, doi: https://doi.org/10.1109/icesc48915.2020.9155700.

[7] pawangfg, “Explanation of BERT Model - NLP,” GeeksforGeeks, Apr. 30, 2020. www.geeksforgeeks.org/explanation-of-bert-model-nlp/

[8] Rajendran Nirthika, Siyamalan Manivannan, and A. Ramanan, “An experimental study on convolutional neural network-based pooling techniques for the classification of HEp-2 cell images,” Aug. 2021, doi: https://doi.org/10.1109/iciafs52090.2021.9606157.

[9] L. Owen, B. Ahmed, and A. Kumar, “BED: Bi-Encoder-Based Detectors for Out-of-Distribution Detection,” Oct. 2023, doi: https://doi.org/10.1109/icaicta59291.2023.10389907.

[10] “Next-Gen Sentence Embeddings with Multiple Negatives Ranking Loss | Pinecone,” www.pinecone.io. https://www.pinecone.io/learn/series/nlp/fine-tune-sentence-transformers-mnr/

[11] N. Rastogi, P. Verma, and P. Kumar, “Evaluation of Information Retrieval Performance Metrics using Real Estate Ontology,” Aug. 2020, doi: https://doi.org/10.1109/icssit48917.2020.9214285.

[12] B. Wang, “Ranking Evaluation Metrics for Recommender Systems,” Medium, Jan. 18, 2021. https://towardsdatascience.com/ranking-evaluation-metrics-for-recommender-systems-263d0a66ef54

[13] Merin, Adril P. “Semantic Search with SBERT,” www.github.com. www.github.com/ninoaddict/semantic-search-with-sbert